Kingsley Okoye
Samira Hosseini

# R Programming

Statistical Data Analysis in Research

Springer

# R Programming

Kingsley Okoye · Samira Hosseini

# R Programming

## Statistical Data Analysis in Research

Springer

Kingsley Okoye
Department of Computer Science
School of Engineering and Sciences
and Institute for the Future of Education
Tecnológico de Monterrey
Monterrey, Nuevo Leon, Mexico

Samira Hosseini
School of Engineering and Sciences
and Institute for the Future of Education
Tecnológico de Monterrey
Monterrey, Nuevo Leon, Mexico

ISBN 978-981-97-3384-2 ISBN 978-981-97-3385-9 (eBook)
https://doi.org/10.1007/978-981-97-3385-9

# Preface

The goal of scientific research is often to investigate a specific phenomenon or topic and find relationships that exist between the underlying variables or factors within the subject population to be able to draw conclusions. The *statistical analysis* is an important part of the scientific research, particularly in the social sciences. It involves the process of collecting and analyzing different data samples in order to identify patterns or trends that can be used to provide valuable insights into the research or draw conclusions. The procedure for performing the various statistical operations and data scrutiny is called Statistical Data Analysis. The researchers can use this procedure to test different hypotheses and make estimations about the studied populations.

This book is written for statisticians, data analysts, programmers, researchers, teachers, students, professionals, and general consumers on how to perform different types of statistical data analysis for research purposes using the R programming language. R is an open-source software and object-oriented programming language with an integrated development environment (IDE) called RStudio for computing statistics and graphical displays through data manipulation, modeling, and calculation. R packages and supported libraries provide the users with a wide range of functions for programming and analyzing of data. Unlike many of the existing statistical softwares, R has the added benefit of allowing the users to write more efficient codes by using command-line scripting and vectors. It has several built-in functions and libraries that are extensible and allow the users to define their own (customized) functions on how they expect the program to behave while handling the data, which can also be stored in the simple object system.

For all intents and purposes, this book serves as both textbook and manual for R statistics, particularly in academic research, data analytics, and computer programming targeted to help inform and guide the work of the R users or statisticians. It provides information about different types of statistical data analysis and methods, and the best scenarios for use of each case in R. It gives a hands-on step-by-step practical guide on how to identify and conduct the different parametric and non-parametric procedures vastly used in social science research. This includes a description of the different conditions or assumptions that are necessary for performing the various statistical tests, and how to understand the results of the different methods. The book also covers different data formats and sources, and how to test for reliability and validity of the available datasets used for research purposes. Different research experiments, case scenarios, and examples are explained in this book. It is the first book to provide a comprehensive description and step-by-step practical hands-on guide on how to carry out the different types of statistical analysis in R, particularly for research purposes with examples. Ranging from how to import and store datasets in R as objects, how to code and call the R methods and functions for manipulating the datasets and objects, factorization, and vectorization, to better reasoning, interpretation, and storage of the results for future use, and graphical visualizations and representations. Thus, the congruence of Statistics and Computer programming for Scientific Research purposes.

## Structure and Organization

The content of this book is organized into 13 chapters that are subdivided (classified) into two parts: Part I and Part II.

The chapters in Part I—which covers Chaps. 1–6 focus on introducing the readers to the basic concepts of R programming and how to use data in R, particularly for research purposes. This includes introducing the readers to the norm of scientific research and the different elements or components that form a typical research process. This part of the book is especially intended for readers who are new to the topic or require a refresher on their knowledge of the R topic or understanding of the different statistical methods and analysis in scientific research. The chapters in Part I prepare the users for easy and efficient use and application of the different statistical methods and analysis that are written in the remainder of chapters in Part II of the book.

The chapters in Part II—which covers Chaps. 7–13 focus on the practical implementation of the different types of statistical data analysis for research using R. This part is meant for readers who have gained advanced knowledge of the topic, and may have the need for applying the various illustrated methods for their research use or data analysis problems.

The content of the book has been structured in a way that it gives a hands-on step-by-step practical guide to the users on how to identify and conduct the different statistical tests and procedures for their research purpose. The users can make references or choose to skip to the specific methods or chapters of interest based on their expert or individual needs.

The following is a brief description of each of the chapters in this book:

## Part I

Chapter 1 presents an introduction to the R programming language and RStudio software particularly for conducting statistical data analysis, graphical displays, modeling, and calculations. It covers the basic concept of R programming, and how the readers can be able to install and run their first R project.

Chapter 2 explores the basic principles and concepts of data management and manipulation in R by discussing what are R objects, vectors, packages, and libraries, including graphs and data visualization methods using the RStudio IDE. It introduces the users to the different functions and methods for working with data in R.

Chapter 3 introduces the readers to the main tests of data normality and reliability using R. The most commonly used and frequently applied type of methods for research are described and illustrated in detail in this chapter.

Chapter 4 explains the Parametric and Non-Parametric Tests for statistical data analysis, and the best scenarios for the use of each test. The chapter provides a guide for the readers on how to choose which test is most suitable for their specific research, including a description of the differences, advantages, and disadvantages of using the two types of tests.

Chapter 5 describes what are *dependent* and *independent* variables for conducting research experiments. It introduces the readers to the different conditions for the use of the two types of variables in scientific research. The differences between the two variables (independent versus dependent) and examples of each use case scenario are also provided in this chapter.

Chapter 6 provides the readers with information about the various types of statistical data analysis methods used in scientific research and examples of best scenarios for the use of each method.

## Part II

Chapter 7 introduces and explains to the readers how to run a *linear* and *logistic* regression analysis in R using RStudio. This statistical technique helps to estimate the association or dependency of the relationship between two variables.

Chapter 8 provides the readers with guidelines on how to run the T-tests for evaluating the *mean* of one or two groups of variables using R. The Independent samples, Paired sample, and One sample t-tests are explained and practically illustrated in this chapter.

Chapter 9 provides detailed information on how to run the analysis of variance (ANOVA) test in R. This test helps to determine the *mean differences* that may exist in data samples. The One-way and Two-way ANOVA are explained and practically illustrated in this chapter.

Chapter 10 explains and illustrates how to apply a Chi-squared ($X^2$) analysis in R. The test is used to compare how expectations are linked with the actual observed experimental data.

Chapter 11 explains and demonstrates how to analyze the effect of a variable over another using the "Mann-Whitney U" and "Kruskal-Wallis H" tests. These are non-parametric equivalents and alternatives to the Independent t-tests and ANOVA used for non-normally distributed dataset in the nominal or ordinal scales, otherwise referred to as *distribution-free* tests.

Chapter 12 explains and illustrates how to conduct the three primary correlational analyses in R which includes: Pearson cor, Kendall's tau, and Spearman's rho correlation tests.

Chapter 13 explains and demonstrates how to perform the Wilcoxon test in R. This distribution-free test which is of two types (Signed-Rank and Sum-rank) and assumes that the data comes from two matched or dependent populations, are practically illustrated in this chapter.

**Data Availability:** Link to the different example datasets used in the practical illustrations and statistical data analysis and computations in this book have been provided in the individual chapters where each of the specific datasets is used. In addition, the authors have uploaded the list of example datasets to the following repository: https://doi.org/10.6084/m9.figshare.24728073 for easy access and download for use and practice by the readers.

Monterrey, Mexico

Kingsley Okoye
Samira Hosseini

# Acknowledgments

This work would not have become a successful project without the contributions from different people. Writing this book required sacrificing hundreds of hours of both learning and teaching, away from family and friends, and the authors would like to thank each and every person, colleagues, and departments for their wonderful support in the publication of this book.

First and foremost, we would like to thank the Writing Lab, Institute for Future of Education, Tecnologico de Monterrey for the technical and financial support in the publication of this book. Most especially, we would like to thank the editor, Daniel Wetta and the Research Assistants for their professional work in editing the different chapters that are contained in this book and for taking the time to read through the series of chapters and the entire manuscripts for proof-reading, constructive comments, and suggestions.

We would like to also acknowledge the expert contributions from colleagues, mentees, and professors both within and outside the Tecnologico de Monterrey with whom we have discussed some of the draft chapters and series of handouts during the workshops and seminars.

We would also like to express our sincere gratitude to the Institute for Future of Education and the staffs who have provided the authors with the facilities and sophisticated support in both learning and working space and sourcing of some of the materials the authors have used in this book.

The author's gratitude and acknowledgment will not be complete without expressing my sincere thanks for the understanding, support, and encouragement I have received from both family and friends. I would most especially like to thank my wife, Justina for her patience and unwavering assistance during the preparation of this book and the final manuscripts.

# Contents

# Abbreviations

| | |
|---|---|
| ANCOVA | Analysis of Co-variance |
| ANOVA | Analysis of Variance |
| Chr | Character |
| Cor | Correlation |
| CSV | Comma-Separated Values |
| DV | Dependent Variable |
| EFA | Exploratory Factor Analysis |
| FTP | File Transfer Protocol |
| GG | Grammar of Graphics |
| ggplot | Grammar of Graphics Plot |
| GUI | Graphical User Interface |
| IDE | Integrated Development Environment |
| IV | Independent Variable |
| K-S | Kolmogorov-Smirnov |
| MANCOVA | Multivariate Analysis of Co-variance |
| MANOVA | Multivariate Analysis of Variance |
| MSE | Mean Sum of squares due to Error |
| MST | Mean Sum of squares due to Treatment |
| NCA | Necessary Condition Analysis |
| Num | Numeric |
| OLS | Ordinary Least Square |
| OS | Operating System |
| PCA | Principal Component Factor Analysis |
| Q-Q plot | Quantile-Quantile Plot |
| S-W | Shapiro-Wilk |

# Part I
# Fundamental Concepts of R Programming and Statistical Data Analysis in Research

# Chapter 1
# Introduction to R Programming and RStudio Integrated Development Environment (IDE)

## 1.1 What is R Programming Language?

R is an open-source software or programming language for computing statistics and graphical displays through methods such as data manipulation, modeling, and calculation (Ihaka & Gentleman, 1996; Venables et al., 2020). In theory, it is a programming language developed by Ross Ihaka and Robert Gentleman in 1993 (Ihaka & Gentleman, 1996), and is regarded as an implementation of the S and S-Plus language that was originally developed at Bell Laboratories by Rick Becker, John Chambers and Allan Wilks (Becker et al., 1988). Practically, R packages and the several supported methods are available and are implemented using integrated developments environment (IDE) such as the RStudio (see Sect. 1.2). Technically, R provides a wide range of statistical and graphical techniques for programming and modeling: ranging from linear and nonlinear modeling techniques to statistical tests and analysis, predictive modeling such as clustering and classification, and time series analysis, etc.

For research and data analytics purposes (see Fig. 1.1), R programming and statistics (R Project, 2023) are performed and used in a series of steps that includes programming, transforming, discovering, modeling, and communication of the outputs or results (Wickham & Grolemund, 2017).

Whilst many existing statistical software, such as SAS (SAS Institute, 2023) and SPSS (IBM, 2022), provide the researchers or data scientists with a bounteous output from conducting the different statistical analyses or methods, R tends to give the analysts a minimal output by storing the results of the methods in an apt "object" for further functions or interrogations.

Therefore, with R being an "object-oriented programming language" (Mailund, 2017), the intermediate results are stored in *objects* that can be recalled, re-used, or manipulated by running other pre-defined functions or codes by pointing to the "object name". In other terms, R is a functional and object-oriented programming

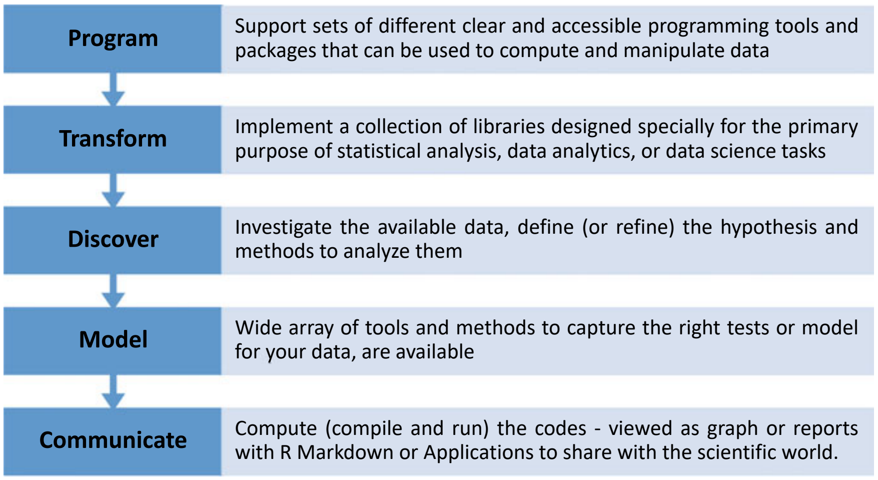

Program
Support sets of different clear and accessible programming tools and packages that can be used to compute and manipulate data
Transform
Implement a collection of libraries designed specially for the primary purpose of statistical analysis, data analytics, or data science tasks
Discover
Investigate the available data, define (or refine) the hypothesis and methods to analyze them
Model
Wide array of tools and methods to capture the right tests or model for your data, are available
Communicate
Compute (compile and run) the codes - viewed as graph or reports with R Markdown or Applications to share with the scientific world.

**Fig. 1.1** Conceptual overview of steps to using R programming for statistical data analysis in research

used to analyze datasets by running mathematical simulations, rearranging complex datasets into simpler and more useful formats and functions, etc. (Matloff, 2011).

Among the many benefits of R in comparison to the other statistical tools and software includes (Douglas et al., 2020; Matloff, 2011):

1. The capacity to write more efficient code using parallel method or vectorization, because of its programmable integrated environment that uses command-line scripting.
2. The capability to define and customize the functions or codes (e.g., how the analysts expect the resultant models to behave) upon handling of the data. R has several built-in functions and libraries that are extendable (extensible) and allow the users to define their own (customized) functions or methods that can be stored in the simple object system.
3. The capability to create artful (illustrative) graphs to visualize or have a conceptual overview of complex data characteristics and functions.
4. The capacity to interface R language with other programs or softwares (e.g., C/C++, Tableau, Python) for improved and better functionality or speed of data analysis.
5. The capacity to find different packages that can be used to perform image manipulation, textual data analysis or natural language processing, machine learning and classifications, etc.
6. Troubleshooting of bugs (code) with an advanced level of debugging performance.

Other advantages of using R particularly as it concerns its technicality or conducting the different statistical data analysis discussed in this book, include (Venables et al., 2020):

- It is built on a well-developed, simple, and effective programming language called "S and S-Plus" that supports user-defined recursive functions and conditionals loops.
- It consists of an effective data handling and storage facility with a wide range of coherent and integrated collections of intermediate tools (packages) and suite operators for statistical data analysis and calculating arrays and matrices.
- It supports different graphical facilities or functions for data manipulation and displays, either directly on the computer screen or storage as soft copy and hard copy on the machine.

However, just like every other programming language, aside from the statistical power of the software, R has its own sets of limitations. R can be daunting for first-time users and people who do not have prior programming knowledge or experience may find it difficult to use the software. Not necessarily because it is more difficult than other programming languages, but because the syntax is different from that of the many other existing languages. Also, R-supported methods or algorithms are spread across different packages, and in consequence, users with no prior knowledge of some of the packages might find it hard to implement the specific methods or algorithms. Thus, the authors has provided in the first part (PART I) of the book, the fundamental concepts of the R programming and statistical data analysis to guide the work of the readers. In addition, R commands give minimal consideration to the computer memory and management and use a lot of the computer physical memory to store the results of the methods as objects, which may be different from some of the other programming languages like Python (Python Software Foundation, 2023). Thus, it uses more computational power and memory. But, R is a continuously evolving language with newer versions and functionalities being developed, and therefore, much of the limitations will eventually fade away with fresher versions and future updates.

## 1.2 RStudio Integrated Development Environment (IDE)

RStudio is a "friendly front end to R language". It allows the researchers and data scientists to practically implement the R packages, methods, and run the several lines of codes. By definition, "RStudio" is an Integrated development environment (IDE) designed to help researchers and analysts to be more efficient and productive with R (Rstudio, 2023). Typically, RStudio consists of a console, syntax-highlighting editor for direct code execution, and several sophisticated tools and functions for viewing the data history, visualization, troubleshooting of codes, and managing of the project workspace as the authors discuss more in detail later in this chapter.

Just like R programming language, RStudio is free and open source (Rstudio, 2023). Its graphical user interface (GUI) is logically systematized in a way that allows the users to clearly view the data tables and graphs, the source codes, and output/results of the codes, simultaneously.

The RStudio IDE offers the users with Import-Wizard features for importing files of different formats into the environment, e.g., comma-separated values (*.csv), Excel (*.xlsl), SAS (*.sas), SPSS (*.sav), and Stata (*.dta) file formats without having to write the codes. Also, just like many of the existing IDEs or GUIs that are used to execute different programming languages, RStudio has windows with multiple tabs, drop-down menus, and many customization options. And, it is available for Windows, Macintosh, and Linux operating systems (OS) (Rstudio, 2023).

Among the many features and functionalities of RStudio IDE includes:

- a window that allows the user to write codes and view the results in real time.
- navigate through the files on the local machine or computer.
- check the details and history of the imported/analyzed data and variables.
- visualization of the results and plots (graphs, models) that are generated.

The RStudio IDE can also be used for developing packages, modeling and writing of executable applications, and natural language and machine learning techniques, etc.

It is important when working with R to know the difference between the "R language" and "RStudio" as covered already in this chapter (see Sects. 1.1 and 1.2). It is noteworthy, to always keep in mind that while "R" is a programming language that can practically be used to statistically compute and manipulate the different variables (data) and models. On the other hand, "RStudio" makes use of R language to develop and show the statistical programs/outputs. Thus, "RStudio allows the users to develop and edit programs using R". Interestingly, R can be used without RStudio, but RStudio cannot be used without R. The user or researcher must first install R before they can install RStudio on their computer, as the authors cover in the next section of this chapter.

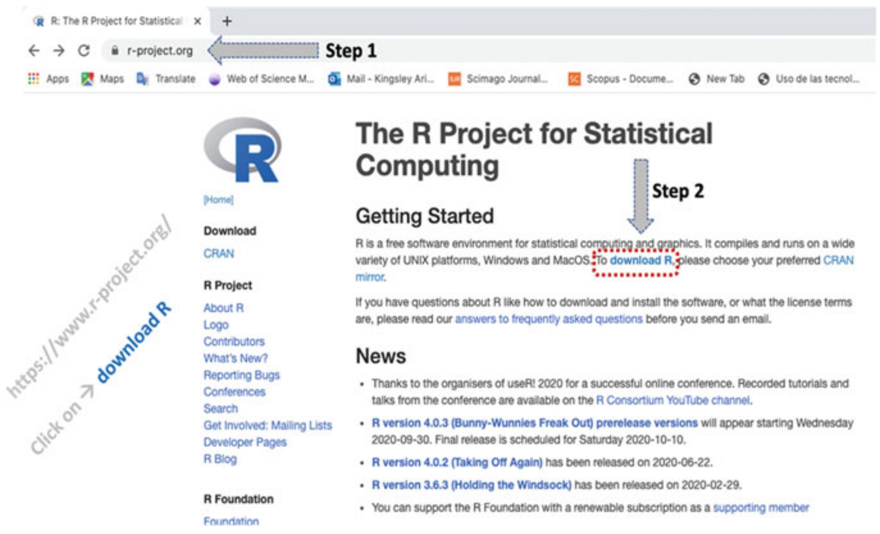
## 1.3 Installing and Configuring R and RStudio Software

This section of the chapter covers the different steps on how to download, install, and configure R and RStudio before using it for statistical data analysis or research purpose.

### *1.3.1 Downloading and Installing R Language*

Installing R on the computer is very simple and easy. All the user need is to know which operating system (OS) they are using so that they can download the right software for installation on the computer system.

The official site for downloading the R free software is via the following link: https://www.r-project.org/.



[Home]

**Download**

CRAN

**R Project**

About R
Logo
Contributors
What's New?
Reporting Bugs
Conferences
Search
Get Involved: Mailing Lists
Developer Pages
R Blog

**R Foundation**

Foundation

**The R Project for Statistical Computing**

**Getting Started**

R is a free software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS. To download R, please choose your preferred CRAN mirror.

If you have questions about R like how to download and install the software, or what the license terms are, please read our answers to frequently asked questions before you send an email.

**News**

- Thanks to the organisers of useR! 2020 for a successful online conference. Recorded tutorials and talks from the conference are available on the R Consortium YouTube channel.
- **R version 4.0.3 (Bunny-Wunnies Freak Out) prerelease versions** will appear starting Wednesday 2020-09-30. Final release is scheduled for Saturday 2020-10-10.
- **R version 4.0.2 (Taking Off Again)** has been released on 2020-06-22.
- **R version 3.6.3 (Holding the Windsock)** has been released on 2020-02-29.
- You can support the R Foundation with a renewable subscription as a supporting member

**Fig. 1.2** Downloading R software

When you visit the site, you will find different binary files for the different types of operating systems (OS) that the R software support, particularly the most common: **Windows**, **Mac OS**, and **Linux**. The latest versions of Linux distributions come with R by default. But for Windows and Mac OS, the user will need to download and install the software as follows:

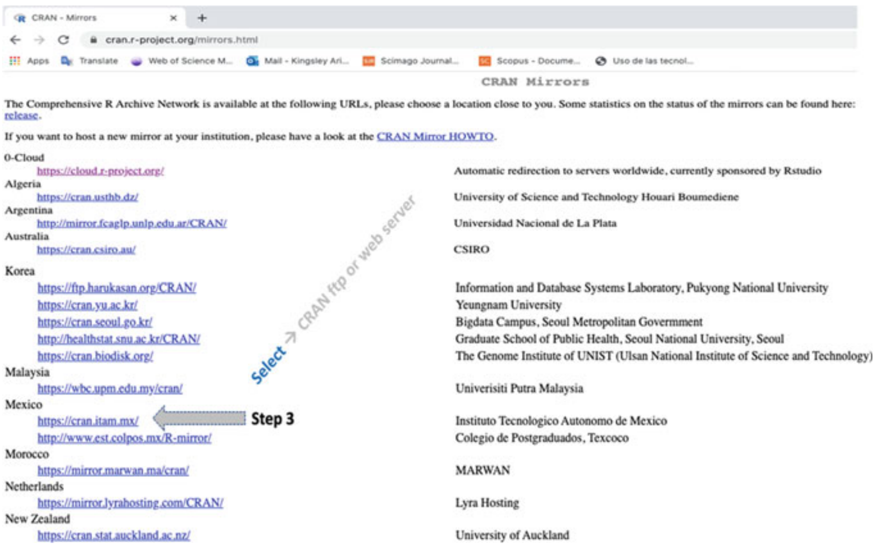
Go to https://www.r-project.org/ by entering the URL on the web browser and click on "**download R**" as shown in Fig. 1.2.

When the user selects **"download R"**, they will be automatically directed to another page where they will be asked to select the **Country** from which they will be using R software, or yet the closest location to you if the country of location is not listed on the site. Navigate to the country of choice and click on any of the **CRAN links** under the country to proceed with the download process. CRAN is a network of *ftp* (file transfer protocol) and web servers around the world that store identical, up-to-date, versions of code and documentation for R.

For example, as shown in Fig. 1.3, the user can navigate to Mexico (e.g., the authors of this book are affiliated with the country at the time of writing this book) and select https://cran.itam.mx/ to proceed with the downloading process. ***Note, it is recommended to always choose the closest CRAN link to you upon downloading the R software.

When the user clicks on the R CRAN binary distribution of their choice, they will be directed to a page where they can download the ***right version of R*** for the specific **operating system (OS)** they are using on the computer (see Fig. 1.4).

For example, as shown in Fig. 1.5 (or step 5), when the users click on **Download R for (Mac) OS X**, they will be directed to where they can then download the **latest**



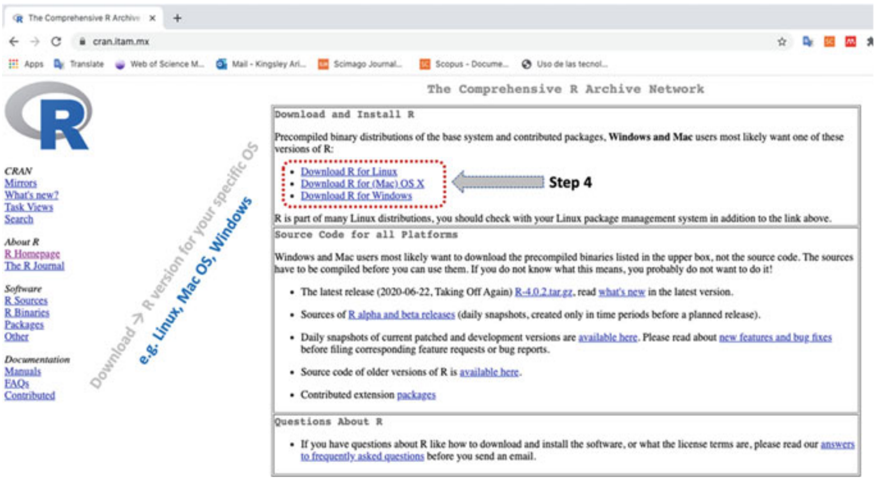
CRAN Mirrors

The Comprehensive R Archive Network is available at the following URLs, please choose a location close to you. Some statistics on the status of the mirrors can be found here: release.

If you want to host a new mirror at your institution, please have a look at the CRAN Mirror HOWTO.

| | |
|---|---|
| 0-Cloud | |
| https://cloud.r-project.org/ | Automatic redirection to servers worldwide, currently sponsored by Rstudio |
| Algeria | |
| https://cran.usthb.dz/ | University of Science and Technology Houari Boumediene |
| Argentina | |
| http://mirror.fcaglp.unlp.edu.ar/CRAN/ | Universidad Nacional de La Plata |
| Australia | |
| https://cran.csiro.au/ | CSIRO |
| Korea | |
| https://ftp.harukasan.org/CRAN/ | Information and Database Systems Laboratory, Pukyong National University |
| https://cran.yu.ac.kr/ | Yeungnam University |
| https://cran.seoul.go.kr/ | Bigdata Campus, Seoul Metropolitan Govermment |
| http://healthstat.snu.ac.kr/CRAN/ | Graduate School of Public Health, Seoul National University, Seoul |
| https://cran.biodisk.org/ | The Genome Institute of UNIST (Ulsan National Institute of Science and Technology) |
| Malaysia | |
| https://wbc.upm.edu.my/cran/ | Univerisiti Putra Malaysia |
| Mexico | |
| https://cran.itam.mx/ | Instituto Tecnologico Autonomo de Mexico |
| http://www.est.colpos.mx/R-mirror/ | Colegio de Postgraduados, Texcoco |
| Morocco | |
| https://mirror.marwan.ma/cran/ | MARWAN |
| Netherlands | |
| https://mirror.lyrahosting.com/CRAN/ | Lyra Hosting |
| New Zealand | |
| https://cran.stat.auckland.ac.nz/ | University of Auckland |

**Fig. 1.3** Country of location and nearest CRAN network for R download



The Comprehensive R Archive Network

Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- Download R for Linux
- Download R for (Mac) OS X
- Download R for Windows

R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

Source Code for all Platforms

Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!

- The latest release (2020-06-22, Taking Off Again) R-4.0.2.tar.gz, read what's new in the latest version.
- Sources of R alpha and beta releases (daily snapshots, created only in time periods before a planned release).
- Daily snapshots of current patched and development versions are available here. Please read about new features and bug fixes before filing corresponding feature requests or bug reports.
- Source code of older versions of R is available here.
- Contributed extension packages

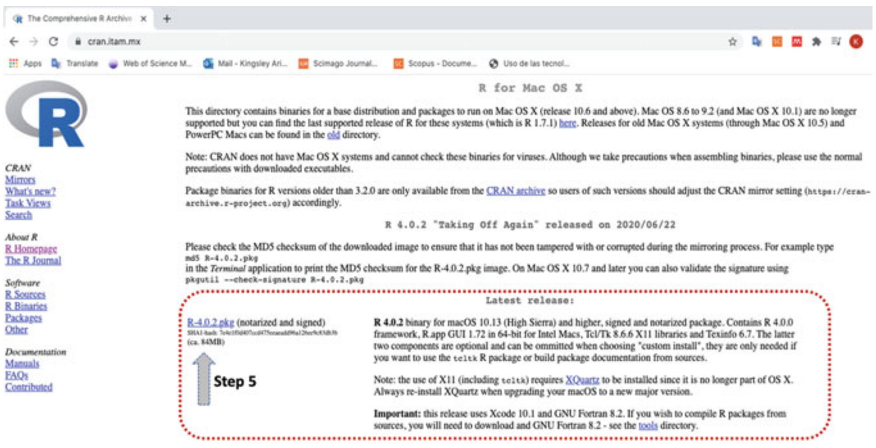
Questions About R

- If you have questions about R like how to download and install the software, or what the license terms are, please read our answers to frequently asked questions before you send an email.

**Fig. 1.4** Downloading the right version of R for your operating system (OS)

**version of R** for Mac OS X. Same applies to the other types of operating systems (OS) such as Windows, if you are using the Windows operating system.

When the user clicks on the download link, the executable program (i.e., installation file) will be automatically downloaded on the computer. Navigate to the location where the downloaded file is stored on the computer and install it as every other application program, e.g., by double-clicking on the downloaded file. When you



R for Mac OS X

This directory contains binaries for a base distribution and packages to run on Mac OS X (release 10.6 and above). Mac OS 8.6 to 9.2 (and Mac OS X 10.1) are no longer supported but you can find the last supported release of R for these systems (which is R 1.7.1) here. Releases for old Mac OS X systems (through Mac OS X 10.5) and PowerPC Macs can be found in the old directory.

Note: CRAN does not have Mac OS X systems and cannot check these binaries for viruses. Although we take precautions when assembling binaries, please use the normal precautions with downloaded executables.

Package binaries for R versions older than 3.2.0 are only available from the CRAN archive so users of such versions should adjust the CRAN mirror setting (`https://cran-archive.r-project.org`) accordingly.

R 4.0.2 "Taking Off Again" released on 2020/06/22

Please check the MD5 checksum of the downloaded image to ensure that it has not been tampered with or corrupted during the mirroring process. For example type
`md5 R-4.0.2.pkg`
in the *Terminal* application to print the MD5 checksum for the R-4.0.2.pkg image. On Mac OS X 10.7 and later you can also validate the signature using
`pkgutil --check-signature R-4.0.2.pkg`

Latest release:

R-4.0.2.pkg (notarized and signed)
SHA1-hash: [illegible]
(ca. 84MB)

**R 4.0.2** binary for macOS 10.13 (High Sierra) and higher, signed and notarized package. Contains R 4.0.0 framework, R.app GUI 1.72 in 64-bit for Intel Macs, Tcl/Tk 8.6.6 X11 libraries and Texinfo 6.7. The latter two components are optional and can be ommitted when choosing "custom install", they are only needed if you want to use the `tcltk` R package or build package documentation from sources.

Note: the use of X11 (including `tcltk`) requires XQuartz to be installed since it is no longer part of OS X. Always re-install XQuartz when upgrading your macOS to a new major version.

**Important:** this release uses Xcode 10.1 and GNU Fortran 8.2. If you wish to compile R packages from sources, you will need to download and GNU Fortran 8.2 - see the tools directory.

**Fig. 1.5** Downloading the latest release of R for your operating system (OS)

double-click on the file, you will get a pop-up window as shown in Fig. 1.6a. Follow the steps illustrated in the figures (Fig. 1.6a, b, and c) by Clicking on **Continue** until you see the window that says you have **successfully installed R software**.

## *1.3.2 Downloading and Installing RStudio Software*

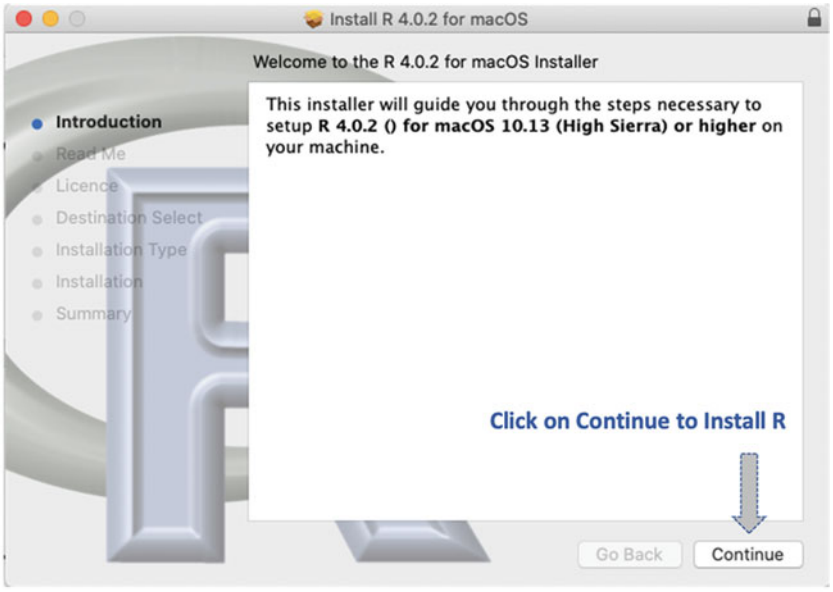
The next step after installing R on your computer, is to download and install the **RStudio IDE** that allows the users to use R.

The official site for downloading the RStudio free software is via the following link: https://rstudio.com/ or https://posit.co/.

As shown in Fig. 1.7, when you visit the RStudio website, you will find the **download** link where the user can download the latest version of RStudio for their computer operating system (OS). ***Note that all the companies update their websites every now and then, and therefore, it may be likely that you find a different front-end display different from the one in Fig. 1.7, which the company uses at the time of writing this book. If you happen to find an updated website depending on when the reader is reading or using this book guide, just simply find where the *download* link is located on the website and follow the same steps or procedure discussed in this current chapter.

Click on the **"DOWNLOAD"** menu (Fig. 1.7), and you will be directed to a page where you can download the RStudio software. Select the "**Download"** link for the "**free version of RStudio Desktop"** as shown in Fig. 1.8. Again, it is important to note that the web display is based on the time of writing this book. As you can see in the figure (Fig. 1.8), there are also paid versions of the software, but those are not covered in this topic.

(a)

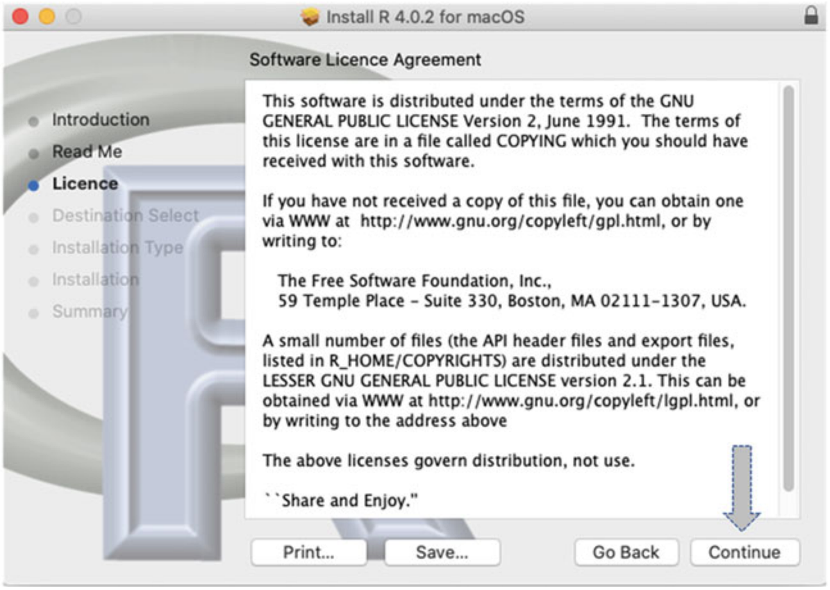


Welcome to the R 4.0.2 for macOS Installer

This installer will guide you through the steps necessary to setup **R 4.0.2 () for macOS 10.13 (High Sierra) or higher** on your machine.

**Click on Continue to Install R**

(b)



Software Licence Agreement

This software is distributed under the terms of the GNU GENERAL PUBLIC LICENSE Version 2, June 1991. The terms of this license are in a file called COPYING which you should have received with this software.

If you have not received a copy of this file, you can obtain one via WWW at http://www.gnu.org/copyleft/gpl.html, or by writing to:

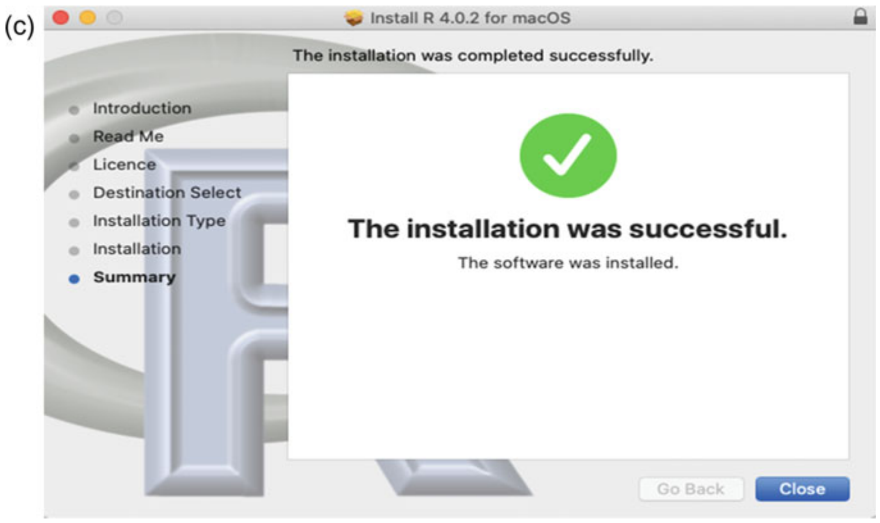
The Free Software Foundation, Inc.,
59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

A small number of files (the API header files and export files, listed in R_HOME/COPYRIGHTS) are distributed under the LESSER GNU GENERAL PUBLIC LICENSE version 2.1. This can be obtained via WWW at http://www.gnu.org/copyleft/lgpl.html, or by writing to the address above

The above licenses govern distribution, not use.

``Share and Enjoy."

**Fig. 1.6** **a** Installing R on the computer or local machine (step 1). **b** Installing R on the computer or local machine (step 2). **c** Successfully installing R on the computer or local machine (step 3)

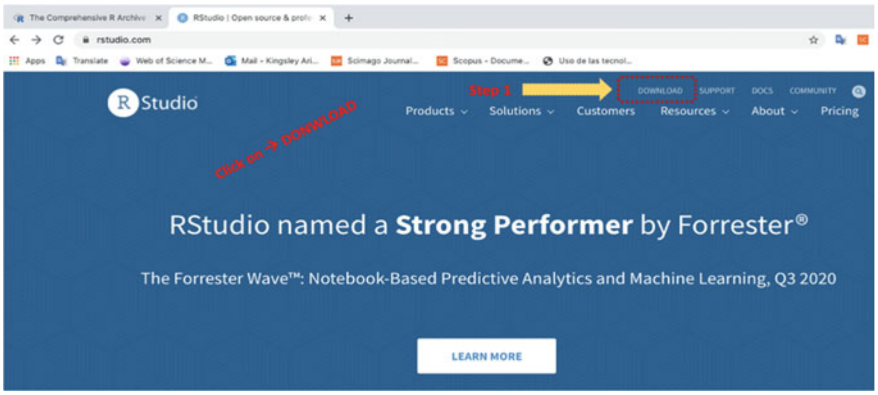


The installation was completed successfully.

The installation was successful.

The software was installed.

**Fig. 1.6** (continued)



RStudio named a **Strong Performer** by Forrester®

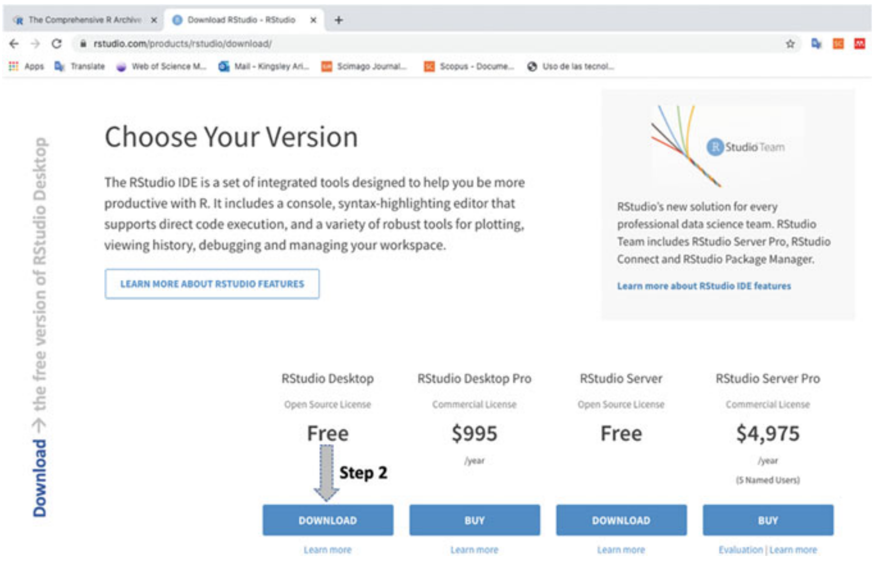
The Forrester Wave™: Notebook-Based Predictive Analytics and Machine Learning, Q3 2020

**Fig. 1.7** Downloading RStudio software

When you have selected the free version of the software, then select the "**right version of the Installer for your Operating System (OS)**" as shown in Fig. 1.9.

For instance, as shown in Fig. 1.9, when the user clicks on the download link for the file "**RStudio.dmg**" (which is for the MacOS latest version at the time of writing this book), the executable program **(Installer)** will be automatically downloaded on the computer system. Navigate to the location where the downloaded (Installer program) file is stored on your computer or local machine, and install it as every other application program, e.g., by double-clicking on the downloaded file.

When you double-click or run the file, you will get a pop-up window as shown in Fig. 1.10.


The Comprehensive R Archive
Download RStudio - RStudio
rstudio.com/products/rstudio/download/
Apps
Translate
Web of Science M...
Mail - Kingsley Ari...
Scimago Journal...
Scopus - Docume...
Uso de las tecnol...
Download → the free version of RStudio Desktop
RStudio Team
LEARN MORE ABOUT RSTUDIO FEATURES
Learn more about RStudio IDE features
Step 2
DOWNLOAD
BUY
DOWNLOAD
BUY
Learn more
Learn more
Learn more
Evaluation | Learn more

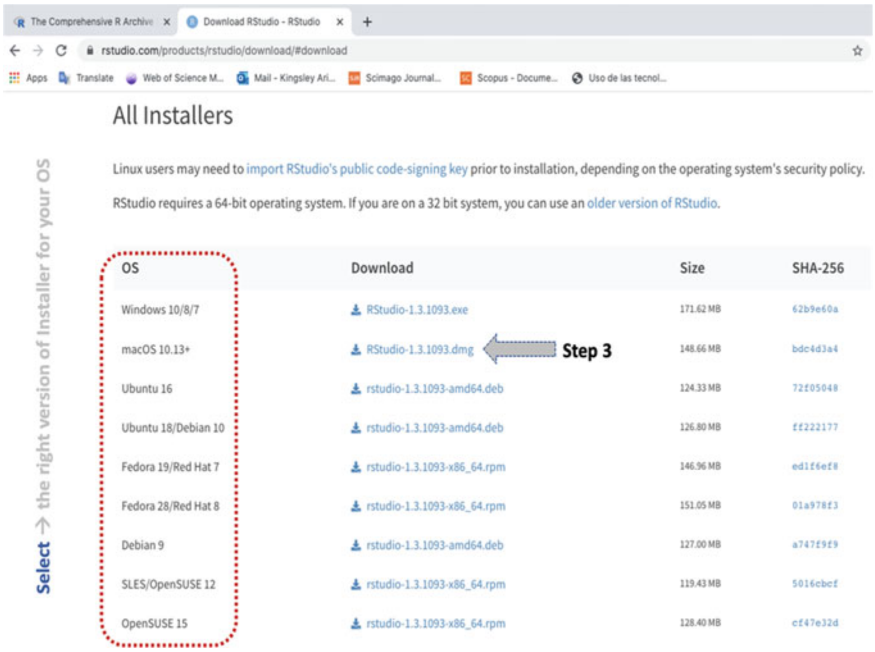
Choose Your Version

The RStudio IDE is a set of integrated tools designed to help you be more productive with R. It includes a console, syntax-highlighting editor that supports direct code execution, and a variety of robust tools for plotting, viewing history, debugging and managing your workspace.

RStudio's new solution for every professional data science team. RStudio Team includes RStudio Server Pro, RStudio Connect and RStudio Package Manager.

| RStudio Desktop | RStudio Desktop Pro | RStudio Server | RStudio Server Pro |
|---|---|---|---|
| Open Source License | Commercial License | Open Source License | Commercial License |
| Free | $995 /year | Free | $4,975 /year (5 Named Users) |

**Fig. 1.8** Downloading free version of RStudio software


The Comprehensive R Archive
Download RStudio - RStudio
rstudio.com/products/rstudio/download/#download
Apps
Translate
Web of Science M...
Mail - Kingsley Ari...
Scimago Journal...
Scopus - Docume...
Uso de las tecnol...
Select → the right version of Installer for your OS
Step 3

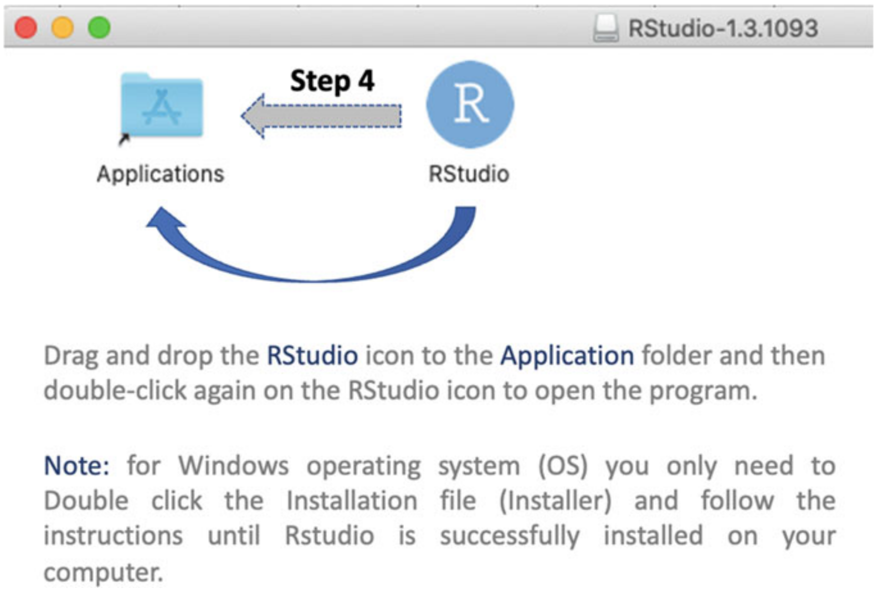
All Installers

Linux users may need to import RStudio's public code-signing key prior to installation, depending on the operating system's security policy.

RStudio requires a 64-bit operating system. If you are on a 32 bit system, you can use an older version of RStudio.

| OS | Download | Size | SHA-256 |
|---|---|---|---|
| Windows 10/8/7 | RStudio-1.3.1093.exe | 171.62 MB | 62b9e60a |
| macOS 10.13+ | RStudio-1.3.1093.dmg | 148.66 MB | bdc4d3a4 |
| Ubuntu 16 | rstudio-1.3.1093-amd64.deb | 124.33 MB | 72f05048 |
| Ubuntu 18/Debian 10 | rstudio-1.3.1093-amd64.deb | 126.80 MB | ff222177 |
| Fedora 19/Red Hat 7 | rstudio-1.3.1093-x86_64.rpm | 146.96 MB | ed1f6ef8 |
| Fedora 28/Red Hat 8 | rstudio-1.3.1093-x86_64.rpm | 151.05 MB | 01a978f3 |
| Debian 9 | rstudio-1.3.1093-amd64.deb | 127.00 MB | a747f9f9 |
| SLES/OpenSUSE 12 | rstudio-1.3.1093-x86_64.rpm | 119.43 MB | 5016cbcf |
| OpenSUSE 15 | rstudio-1.3.1093-x86_64.rpm | 128.40 MB | cf47e32d |

**Fig. 1.9** Downloading the right version of RStudio for your operating system (OS)



Drag and drop the RStudio icon to the Application folder and then double-click again on the RStudio icon to open the program.

Note: for Windows operating system (OS) you only need to Double click the Installation file (Installer) and follow the instructions until Rstudio is successfully installed on your computer.

**Fig. 1.10** Installing RStudio on your computer (e.g., for MacOS)

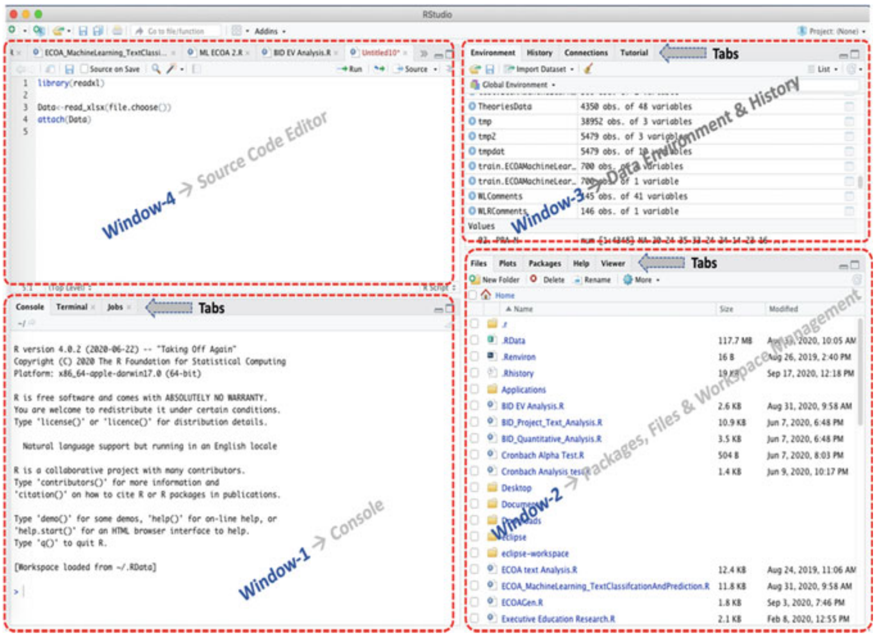
As illustrated in the figure (Fig. 1.10), same installation process applies to other types of OS (operating system) such as Windows if you are using the Windows OS. Double-click the Installation file to start up the executable file (Installer program). Then, click on **Continue** until you see the window that says you have **successfully installed the RStudio software**.

Once you have completed the installation process, **start the RStudio IDE** by either opening the application from the list of programs on your computer or clicking on the desktop shortcut icon. You will be presented with a Window as shown in Fig. 1.11.

Congratulations! You are now set to run and execute your first R project in RStudio. Welcome to using R programming for statistical data analysis in research covered as the main objective of this book.

The first time the users open RStudio, they will be presented with three windows by default, i.e., **Window-1**, **Window-2**, and **Window-3** (see Fig. 1.11). The fourth window (**Window-4**) is hidden by default and is only displayed when the user executes a program or run a command, but the users can also open it by selecting the **"File" drop-down menu**, then **New File**, and then **R Script** or simply by importing a dataset into the environment, which the authors will cover in detail in the next section (Sect. 1.4) and chapter (Chap. 2) of this book.

In Table 1.1, the authors outline the description and functions of the different tabs (component) of the R window or integrated development environment (IDE) (see Windows 1, 2, 3, and 4 in Fig. 1.11).



```
library(readxl)

Data<-read_xlsx(file.choose())
attach(Data)
```

```
R version 4.0.2 (2020-06-22) -- "Taking Off Again"
Copyright (C) 2020 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin17.0 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Workspace loaded from ~/.RData]

> 
```

| | |
|---|---|
| TheoriesData | 4350 obs. of 48 variables |
| tmp | 38952 obs. of 3 variables |
| tmp2 | 5479 obs. of 3 variables |
| tmpdat | 5479 obs. of 10 variables |
| train.ECOAMachineLear… | 700 obs. of 3 variables |
| train.ECOAMachineLear… | 700 obs. of 1 variable |
| WLComments | 145 obs. of 41 variables |
| WLRComments | 146 obs. of 1 variable |
| Values | |
| Q2. PRA-N | num [1:4348] NA 20 24 35 33 24 34 14 23 16 ... |

| Name | Size | Modified |
|---|---|---|
| .. | | |
| .RData | 117.7 MB | Aug 31, 2020, 10:05 AM |
| .Renviron | 16 B | Aug 26, 2019, 2:40 PM |
| .Rhistory | 19 KB | Sep 17, 2020, 12:18 PM |
| Applications | | |
| BID EV Analysis.R | 2.6 KB | Aug 31, 2020, 9:58 AM |
| BID_Project_Text_Analysis.R | 10.9 KB | Jun 7, 2020, 6:48 PM |
| BID_Quantitative_Analysis.R | 3.5 KB | Jun 7, 2020, 6:48 PM |
| Cronbach Alpha Test.R | 504 B | Jun 7, 2020, 8:03 PM |
| Cronbach Analysis test.R | 1.4 KB | Jun 9, 2020, 10:17 PM |
| Desktop | | |
| Documents | | |
| Downloads | | |
| eclipse | | |
| eclipse-workspace | | |
| ECOA text Analysis.R | 12.4 KB | Aug 24, 2019, 11:06 AM |
| ECOA_MachineLearning_TextClassifcationAndPrediction.R | 11.8 KB | Aug 31, 2020, 9:58 AM |
| ECOAGen.R | 1.8 KB | Sep 3, 2020, 7:46 PM |
| Executive Education Research.R | 2.1 KB | Feb 8, 2020, 12:55 PM |

**Fig. 1.11** RStudio integrated development environment (IDE)

## 1.4 Running Your First R Project in R Using RStudio

In this section of the chapter, the authors introduces to the readers steps on how to create or start an R Project in RStudio. RStudio Projects make it easier and straightforward for the users to distribute their work into different categories or contexts, with each having their own working *directory* in the *workspace* including the history and source code documents. It is important to keep in mind that R projects are associated with a "working directory" where the users can save their new or running projects, and also retrieve existing projects.

Users can create an RStudio project in either a (i) **brand-new directory**, (ii) an **existing directory** where they already have R code and data, (iii) or by **cloning a version control** repository, e.g., from Git, GitHub or Subversion (see Fig. 1.13).

To create a **new R Project in RStudio**, start RStudio (see description in the previous section—Sect. 1.3). Once you are logged in and have the RStudio window open (see Fig. 1.11), click on the "**File**" menu at the top left corner of the RStudio window and select the "**New Project**" button as shown in Fig. 1.12. You will be presented with a pop-up window as shown in Fig. 1.13.

Select the "**New Directory**" option and fill in the pop-up with your chosen preferred **project_directory_name** by following the steps illustrated in Figs. 1.14 and 1.15.

**Table 1.1** Description of function of the different tabs (component) of R window (IDE)

| RStudio window | Component/ menu | Description/function |
|---|---|---|
| Window-1 (console) | Console tab | Results/output of the executed codes are displayed (printed) here. Also, further commands can be entered via the window |
| | Terminal tab | Command line system that allows the user to quickly control or access the operating system and make changes |
| | Jobs tab | Contains a list of open job(s) the system has currently running or pending in R |
| Window-2 (file explorer) | Files tab | File Explorer that allows the user to access the different files and folders stored on the hard drive (C: drive) |
| | Plots tab | Plots/graphs visualizations are displayed (outputted) at this location |
| | Packages tab | Contains a list of packages (libraries) that are installed in R on your system. Users can also install new packages or update the existing ones through the tab |
| | Help tab | Search window for help on different R topics, functions, or packages, and output location for the help commands |
| | Viewer tab | Advanced tab for local web content |
| Window-3 (file environment or history tab, e.g., objects, data) | Environment tab | Shows the list of interactive R objects that are loaded in the IDE |
| | History tab | Contains a list of key codes that are entered and executed unto the Console |
| | Connection tab | Tab through which the user can connect R to other external/ existing data sources or database |
| | Tutorial tab | Location where users can access different tutorials and learn about the several R functions, data types/variables, and commands |
| Window-4 (code editor) | Source/code tabs | A built-in text editor where the user enters the codes and commands, and can also find the shortcuts to run and save the commands/codes |

When finished click the "**Create Project**" button and you're done! Congratulations once more! You have created your new project in R.

With the window and a console open, for instance, the authors named our project "**MyFirstR_Project**" in the directory as shown in Fig. 1.16, we are ready to run the R script, therein we can code and run the programs.

To create a "**new R script"** and run your new/written codes, select the "**File**" drop-down menu, then "**New File**", and then "**R Script**" as shown in Fig. 1.17.

You will be presented with a new working window or editor where you can start writing your code (see Fig. 1.18).

Now let's run some simple lines of code. As shown in Fig. 1.19 (see Steps 1 and 2), write the **example codes** from **Line 1** to **Line 4** in the Editor and execute the codes using the "**Run**" button (see Fig. 1.19). Example R code: Line 1: x <- 3 + 5

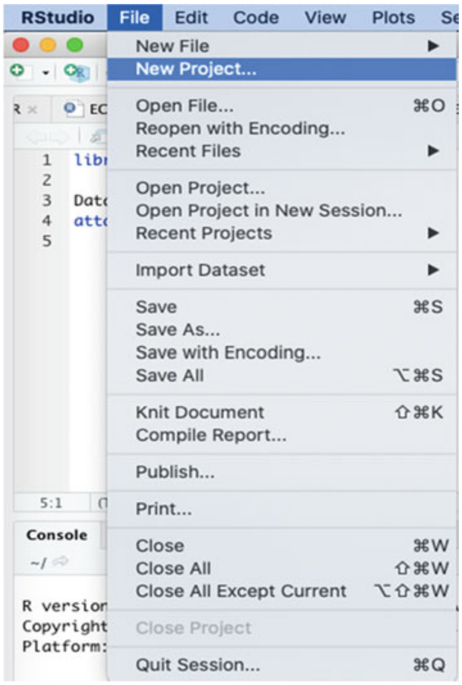


Fig. 1.12 Creating a new project in RStudio

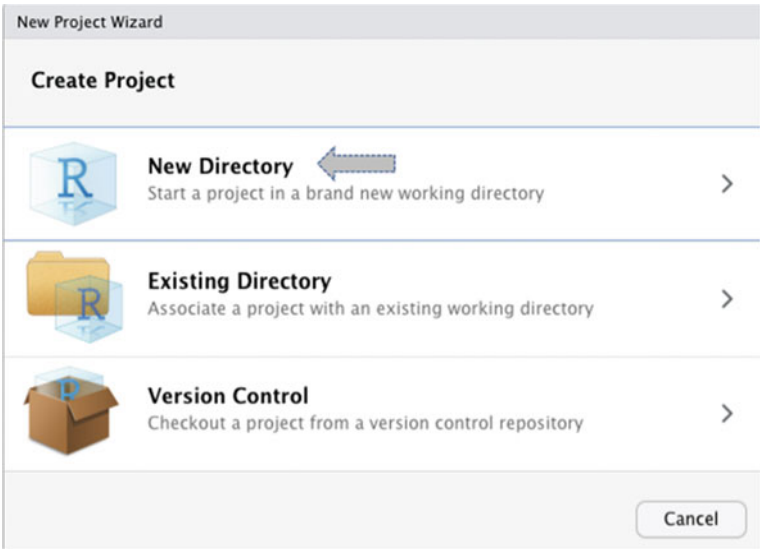

New Project Wizard
Create Project
New Directory
Start a project in a brand new working directory
Existing Directory
Associate a project with an existing working directory
Version Control
Checkout a project from a version control repository
Cancel

Fig. 1.13 New project wizard pop-up window

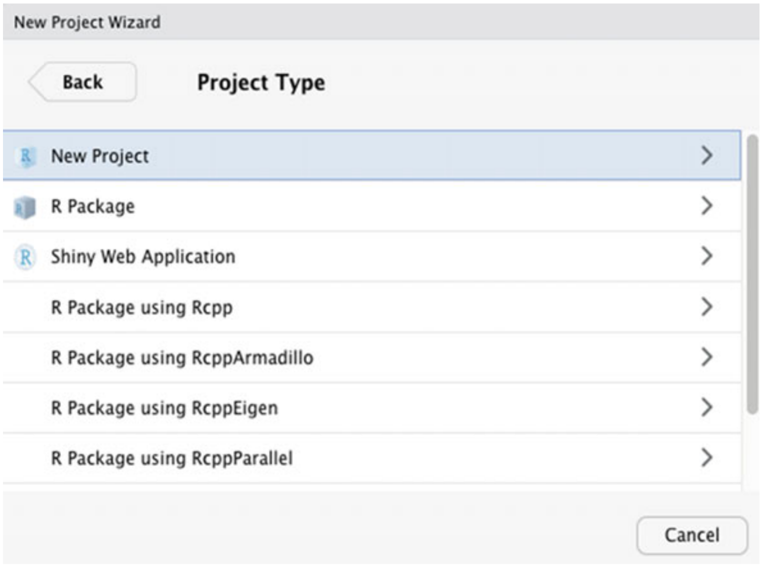
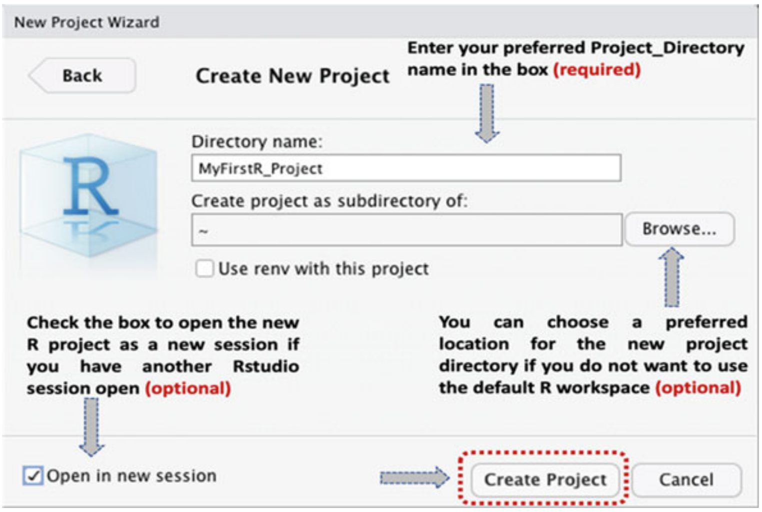


**Fig. 1.14** Selecting the project type



Enter your preferred Project_Directory name in the box **(required)**

Directory name: MyFirstR_Project

Create project as subdirectory of: ~

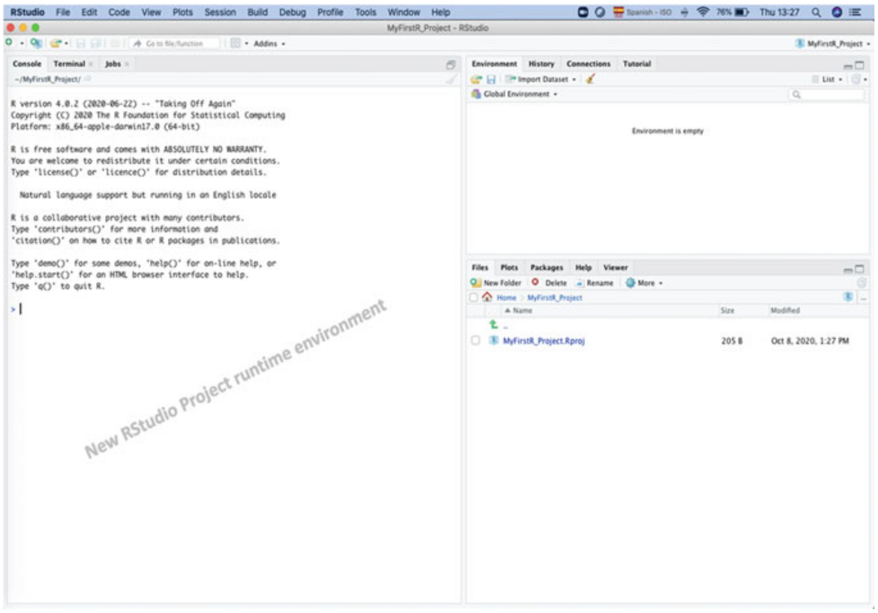
Check the box to open the new R project as a new session if you have another Rstudio session open **(optional)**

You can choose a preferred location for the new project directory if you do not want to use the default R workspace **(optional)**

**Fig. 1.15** Creating a new R project and directory name

Line 2: x Line 3: print(x) Line 4: print("I am ready to work with data in R and start conducting the different statistical analysis for my research")

*Remember, start from Line 1 (e.g., by clicking anywhere in the line) before running the codes, or alternatively, follow the steps illustrated in Fig. 1.20 to run (execute) all the codes at once.


New RStudio Project runtime environment
MyFirstR_Project - RStudio
Environment is empty
MyFirstR_Project.Rproj 205 B Oct 8, 2020, 1:27 PM

```
R version 4.0.2 (2020-06-22) -- "Taking Off Again"
Copyright (C) 2020 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin17.0 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> |
```

**Fig. 1.16** New R project created in RStudio

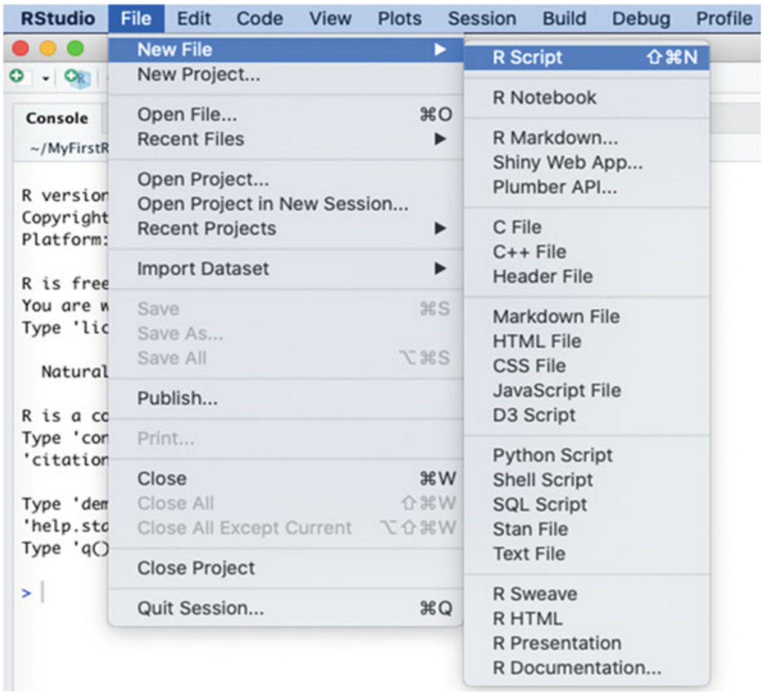
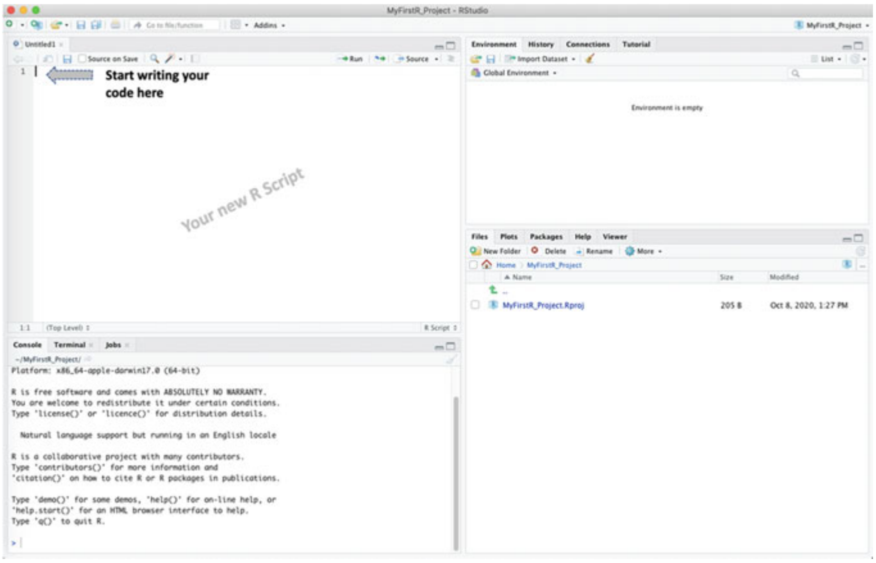
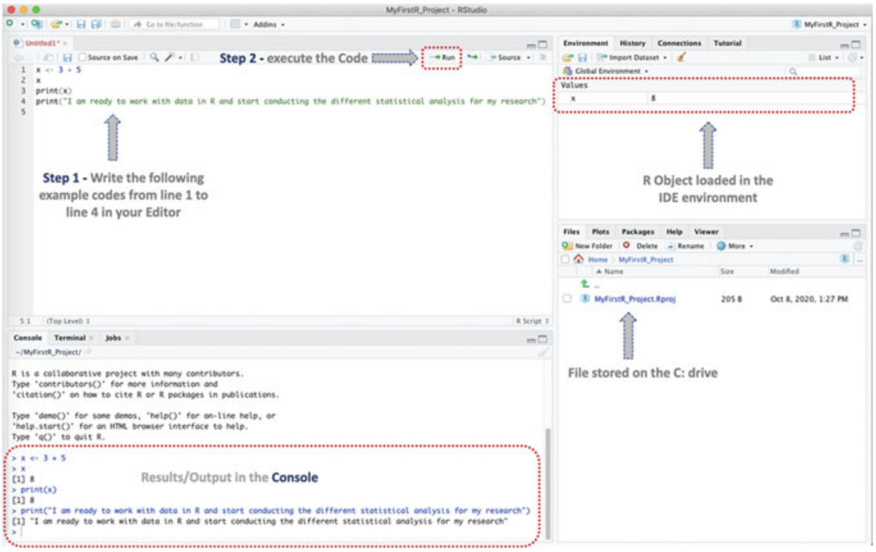


**Fig. 1.17** Creating a new R script



```
Platform: x86_64-apple-darwin17.0 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> 
```

**Fig. 1.18** New R script window with the source/code tab


MyFirstR_Project - RStudio
Untitled1*
Step 2 - execute the Code
Run
Source
Step 1 - Write the following example codes from line 1 to line 4 in your Editor
Environment
History
Connections
Tutorial
Global Environment
R Object loaded in the IDE environment
Files
Plots
Packages
Help
Viewer
File stored on the C: drive
Console
Terminal
Jobs
~/MyFirstR_Project/
Results/Output in the Console

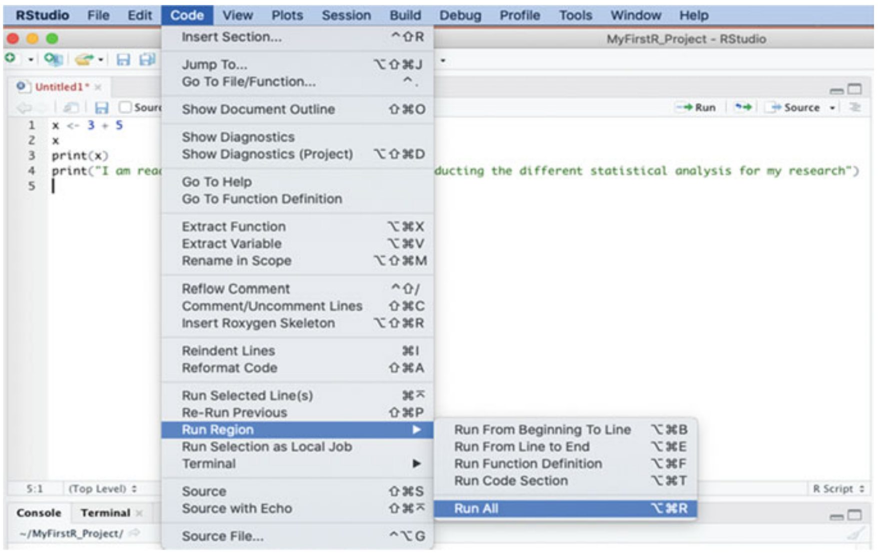
```
x <- 3 + 5
x
print(x)
print("I am ready to work with data in R and start conducting the different statistical analysis for my research")
```

| Values | |
|---|---|
| x | 8 |

| Name | Size | Modified |
|---|---|---|
| .. | | |
| MyFirstR_Project.Rproj | 205 B | Oct 8, 2020, 1:27 PM |

```
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
> x <- 3 + 5
> x
[1] 8
> print(x)
[1] 8
> print("I am ready to work with data in R and start conducting the different statistical analysis for my research")
[1] "I am ready to work with data in R and start conducting the different statistical analysis for my research"
> 
```

**Fig. 1.19** Writing and running R scripts example in RStudio



```
x <- 3 + 5
x
print(x)
print("I am rea
```

```
ducting the different statistical analysis for my research")
```

**Fig. 1.20** Running all the R script source code in RStudio editor

When finished or run all applied, you will be presented with a screen similar to the one in Fig. 1.19. Well done! You have just created and run your first R project and R script in RStudio.

Next, as you can see in Fig. 1.19, the R Script is named "**Untitiled1**" by default. We will save the R script with a name on the computer or workspace so that we can retrieve it anytime or when we want.

As illustrated in Fig. 1.21, click on the "**File**" drop-down menu and choose the "**Save as**" option.

You will be presented with a pop-up window as shown in Fig. 1.22. Enter your chosen or preferred script name, for example, "**MyFirst-RScript**", and click the "**Save**" button to save the Script with the new name.

Now, take another look at the new window or screen (see Fig. 1.23), you will find that the "**Untitiled1**" script has now changed or saved as the new name "**MyFirst-RScript**". Also, you will notice that the updated/saved file has also been included and listed in the File Explorer window (Fig. 1.23).

## 1.5 Tips and Technical Guidelines

Here, the authors provide further tips and other useful information the readers, particularly the first timers to R, can find as a technical guide in their journey with R covered in this book.

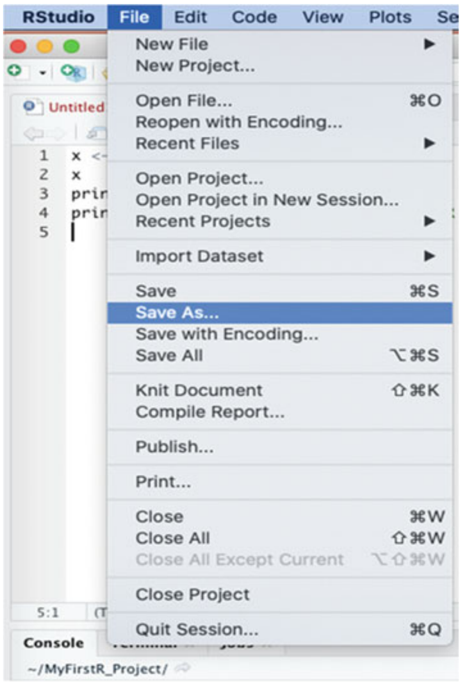
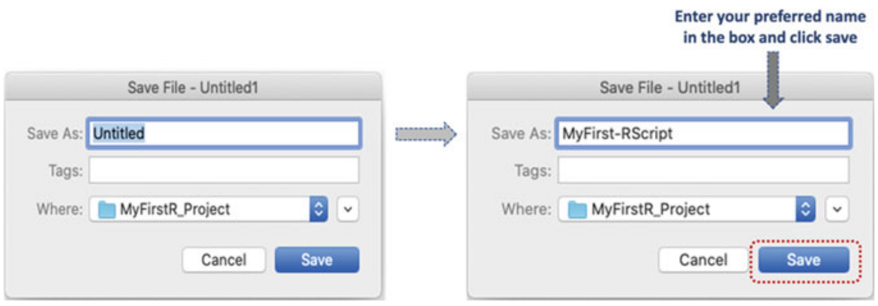


Fig. 1.21 Saving untitled R script



Save As: Untitled
Where: MyFirstR_Project

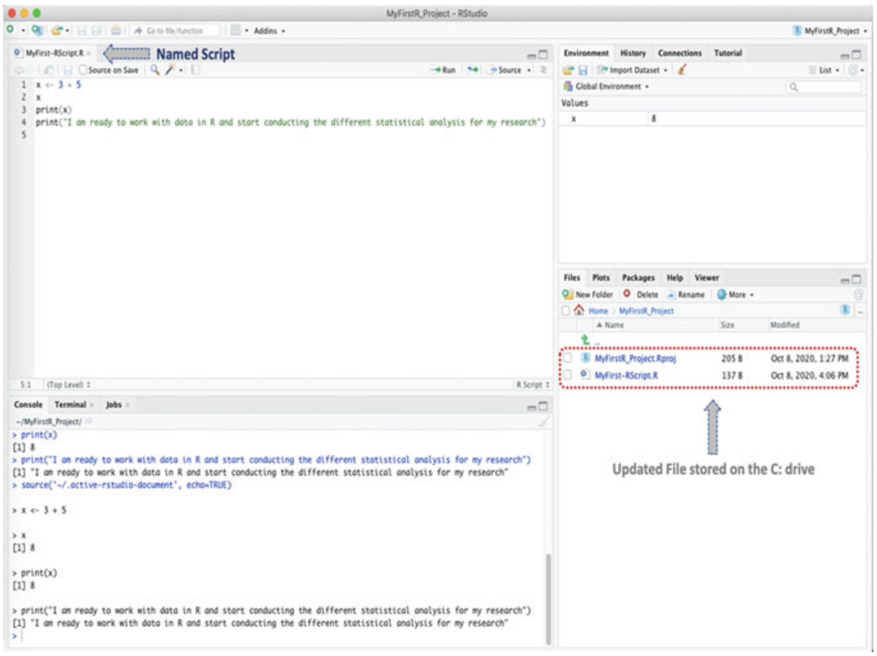
Save As: MyFirst-RScript
Where: MyFirstR_Project

Fig. 1.22 Saving the R script in RStudio

## *1.5.1 Tips About a New R Project*

When the user creates a new project in RStudio:

1. It creates a project file with **.Rproj** extension within the directory. This is used as a reference point to the computer file system or yet shortcut for opening the project directly from the file system.



```
x <- 3 + 5
x
print(x)
print("I am ready to work with data in R and start conducting the different statistical analysis for my research")
```

```
> print(x)
[1] 8
> print("I am ready to work with data in R and start conducting the different statistical analysis for my research")
[1] "I am ready to work with data in R and start conducting the different statistical analysis for my research"
> source('~/.active-rstudio-document', echo=TRUE)

> x <- 3 + 5

> x
[1] 8

> print(x)
[1] 8

> print("I am ready to work with data in R and start conducting the different statistical analysis for my research")
[1] "I am ready to work with data in R and start conducting the different statistical analysis for my research"
>
```

**Fig. 1.23** Saved R script and file explorer

2. It creates another hidden directory named **.Rproj.user** for storing and handling the temporary files, e.g., auto-save or state of the window.
3. It loads the new project unto RStudio and displays the name in the **Projects toolbar** (see top right side of main toolbar in Fig. 1.23).

## *1.5.2 Opening Existing R Projects and R Scripts*

There are various ways to open an existing R project or R Script:

1. By selecting the specific Project from the list of "**Recent Projects**" from the "**File**" drop-down menu. The same procedure applies to opening a specific R script by selecting "**Recent Files**" from the "**File**" menu and selecting the R script name.
2. By using the "**Open Project"** command from the "**File**" drop-down menu, and then browse the working directory and select an existing project file (**.Rproj**). Same procedure applies to opening a specific R script by selecting "**Open File**" from the "**File**" menu, and then browses the working directory and select an existing R file (**.R**).
3. By double-clicking on the specific project/file from the list of files in the File Explorer tab/window.

The following actions are performed when a new or existing project is opened in RStudio:

1. It starts a new R session.
2. It sources the .Rprofile file in the main directory of the project.
3. The .RData file and .Rhistory file in the main directory of the project will be loaded in the Environment or History Tab.
4. The R working (current) directory will be set to the current projects' directory.
5. Existing source codes (if any) will be loaded into the editor window.
6. Other relevant settings and active tabs of the project will be restored to the original form when it was last saved or closed.

### *1.5.3 Working with Multiple R Projects*

RStudio allows the users to simultaneously work with more than one project at once by opening an instance of each project on its own.

1. The users can use the "**Open Project in New Session"** option from the "**File**" drop-down menu to do this.
2. Or by opening multiple project files from the **File Explorer** or system file by double-clicking on the specific folder or file in each setting as required.

### *1.5.4 Closing or Quitting R*

The following actions are performed when the user closes an active project or opens another project concurrently:

- The source codes in the editor window are saved so that you can re-open or restore them when next you open the closed project.
- All available .Rhistory and .RData will be stored on the project directory.
- Other relevant settings and active tabs of the project will be stored in their current form.
- The R session will be terminated.

## 1.6 Summary

In this chapter, the authors provided an introduction to the R programming language and RStudio software or IDE. It covered the basic concept of R programming, and how the readers can install and run their first project using R. The capacity to write more efficient code using parallel method or vectorization is one of the main features of the R software illustrated in this chapter, because of its programmable integrated environment (RStudio) that uses command-line scripting. We also showed the capability to define and customize the R functions or codes. R has several built-in functions and libraries that are extendable (extensible) and allow the users to define their own (customized) functions or methods that are stored in the simple object system. In the next chapter (Chap. 2), the authors focus on introducing the readers to how to effectively work with Data in R. This includes understanding and learning how to create Objects, Vectors, and Factorization in R, to understanding how to install the R Packages and Libraries, and then presents some hands-on examples of Data Visualization methods and practices.

## References

Becker, R. A., Chambers, J. M., & Wilks, A. R. (1988). *The new S language: A programming environment for data analysis and graphics.* Wadsworth and Brooks/Cole Advanced Books & Software.

Douglas, A., Roos, D., Mancini, F., Couto, A., & Lusseau, D. (2020). *An Introduction to R.* bookdown.

IBM. (2022). *SPSS Statistics—Overview|IBM*. https://www.ibm.com/products/spss-statistics

Ihaka, R., & Gentleman, R. (1996). R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics, 5*(3), 299–314. https://doi.org/10.1080/10618600.1996.10474713

Mailund, T. (2017). Advanced object-oriented programming in R. In *Advanced object-oriented programming in R.* Apress. https://doi.org/10.1007/978-1-4842-2919-4

Matloff, N. (2011). *The art of R programming: A tour of statistical software design* (1st ed.).

Python Software Foundation. (2023). *Python Programming Language.* https://www.python.org/

R Project. (2023). R programming language. https://www.r-project.org/

Rstudio. (2023). *RStudio—RStudio.* https://rstudio.com/products/rstudio/

SAS Institute. (2023). *SAS—Analytics & Solutions Software.* https://www.sas.com/es_cl/home.html

Venables, W. N., Smith, D. M., & R Core Team. (2020). *An introduction to R: notes on R—A programming environment for data analysis and graphics.*

Wickham, H., & Grolemund, G. (2017). *R for data science: Import, tidy, transform, visualize, and model data.* O'Reilly Media.

# Chapter 2
# Working with Data in R: Objects, Vectors, Factors, Packages and Libraries, and Data Visualization

## 2.1 Introduction

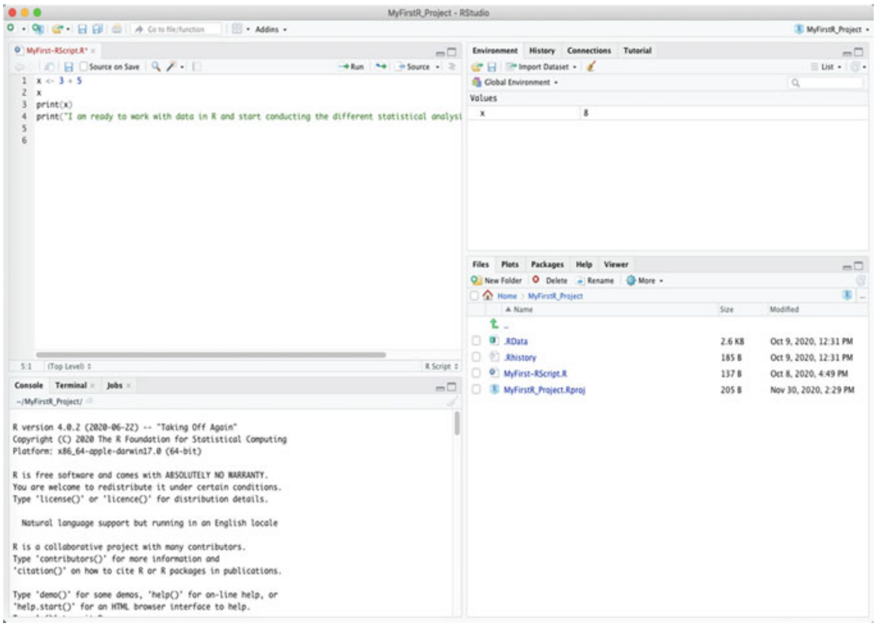
In the previous chapter (Chap. 1), the authors introduced the readers to R programming language and how to successfully install and configure R software and RStudio on the system ready for data analysis. In this chapter, we will illustrate how to start using R to work with data.

R can be used to conduct various statistical data analysis and graphical representations. The R integrated development environment (RStudio) has many built-in functions and packages that allow the users to perform different types of data analysis, and is also *extensible* due to the fact that it allows the users to define their own additional functions or methods. R has a simple *object* system that allows the user to type in data or codes directly into the R's interactive console or source code editor. However, a lot of the time, the researchers or data analysts are more likely to have already an available data that they want to work with, for instance, in a file stored somewhere on their local machine or on the internet.

Therefore, over the next sections of this chapter, the authors will be looking at some of the different methods and ways the users can get data into R for analysis and visualizations.

## 2.2 Preparing RStudio and Script for Working with Data in R

To start working with data in R, first, the user needs to create an "**R Project**" and "**R Script"** or open an existing one, as we have already discussed and demonstrated in Chap. 1 of this book. For the illustrations in this chapter, we will continue working with the "**MyFirstR_Project**" we created earlier in the chapter (see Sect. 1.4, Chap. 1). ***However, the users are also welcome to create a new project with



```
x <- 3 + 5
x
print(x)
print("I am ready to work with data in R and start conducting the different statistical analysi
```

```
R version 4.0.2 (2020-06-22) -- "Taking Off Again"
Copyright (C) 2020 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin17.0 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
```

| Name | Size | Modified |
|---|---|---|
| .RData | 2.6 KB | Oct 9, 2020, 12:31 PM |
| .Rhistory | 185 B | Oct 9, 2020, 12:31 PM |
| MyFirst-RScript.R | 137 B | Oct 8, 2020, 4:49 PM |
| MyFirstR_Project.Rproj | 205 B | Nov 30, 2020, 2:29 PM |

Fig. 2.1 R project in RStudio

a new name if they wish to do so. The "**MyFirstR_Project**" is only for illustration purposes.

Once you have created and/or opened the existing R project, you will be presented with a screen similar to the one in Fig. 2.1. See Sect. 1.3.2 in Chap. 1 for description of the different tabs and functions of the RStudio windows.

We will create a new "**R Script"** named "**WorkingWithDataInR**" for use in working with the different data, R functions, and packages that we will explain and practicalize in this chapter.

As shown in Fig. 2.2, Click on the **File** menu, then select **New File**, and click on **R Script**.

Now, name the new R Script as "**WorkingWithDataInR**" or any name of your choice, using the "**Save As**" option from the **File** menu as described in Fig. 2.3.

Now, we are ready to start working with the different datasets explored in this chapter in R using the new Script.

In the coming subsections, we will cover some of the different ways the users can generate data, upload their own data, and work with data in R.

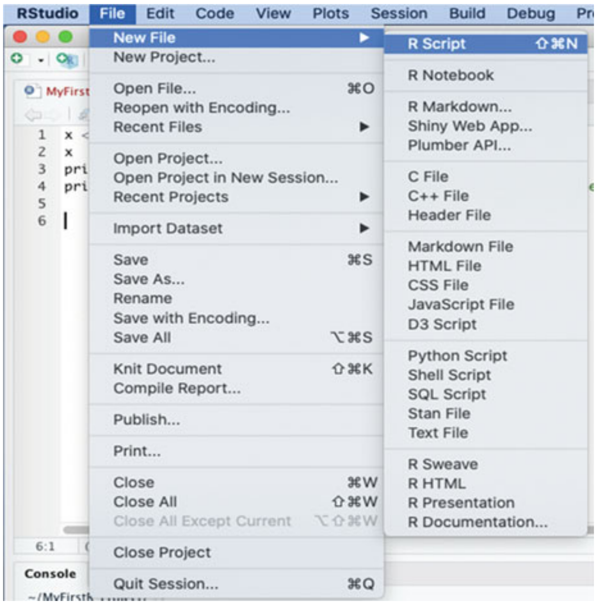
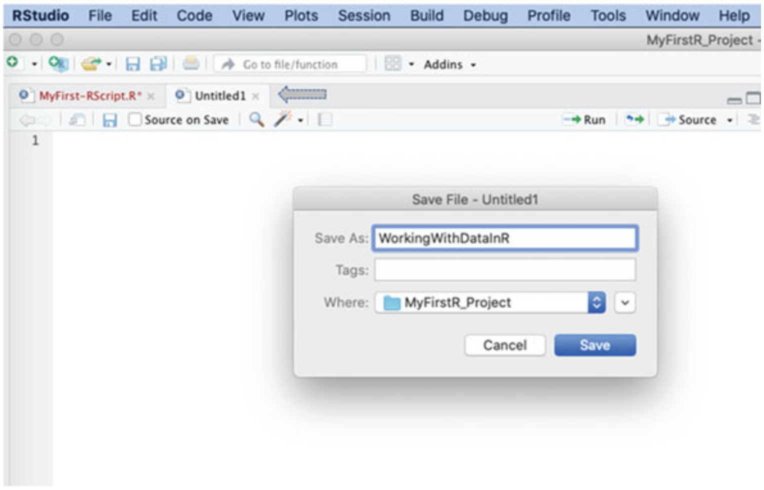


**Fig. 2.2** Creating a new R script



Save As: WorkingWithDataInR
Tags:
Where: MyFirstR_Project

**Fig. 2.3** New R script

## 2.3 Working with Data in R

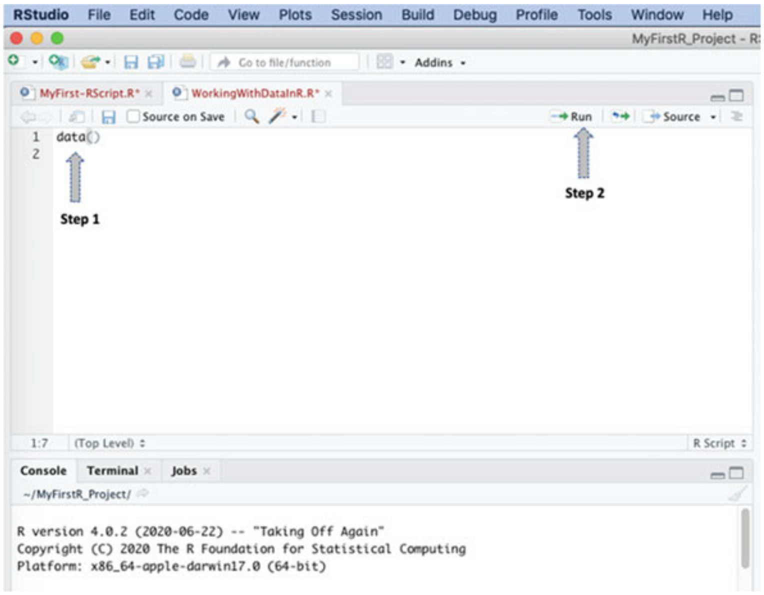
Here, the authors explain and demonstrate how the users can effectively work with data using R with some working examples.

### *2.3.1 Pre-loaded Sample Data in R*

When you install R on the computer, by default, the installation comes with different types of datasets that the users can try out for testing, learning, or even research purposes. The users can explore some of the pre-loaded test data in R to see how the datasets are loaded, the different features, and what basic functions the users can run. Although for research purposes, those pre-defined datasets may not be necessarily useful especially, as in most cases, the researchers already have their own data that they want to analyze which we will cover in more detail in the subsequent sections.

To view the list of the pre-loaded R sample datasets, type the following command: **data( )** in the source code editor window and run the code as shown in Fig. 2.4.

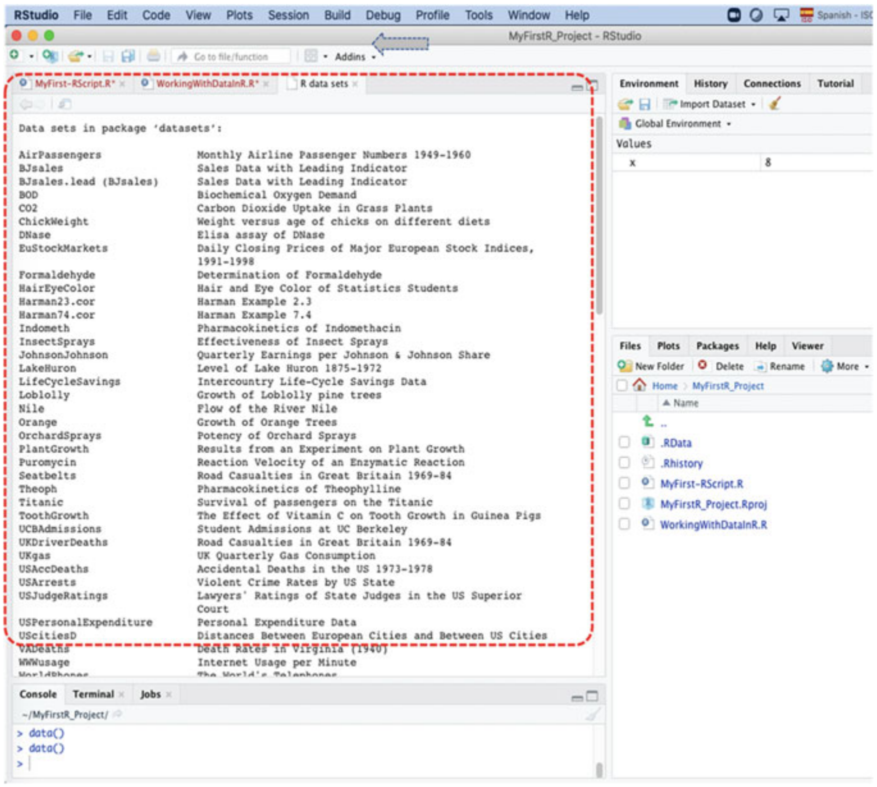
```
data()
```



```
data()
```

```
R version 4.0.2 (2020-06-22) -- "Taking Off Again"
Copyright (C) 2020 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin17.0 (64-bit)
```

Fig. 2.4 Getting the list of pre-loaded sample data in R


RStudio File Edit Code View Plots Session Build Debug Profile Tools Window Help
MyFirstR_Project - RStudio
Go to file/function
Addins
MyFirst-RScript.R*
WorkingWithDataInR.R*
R data sets
Environment History Connections Tutorial
Import Dataset
Global Environment
Files Plots Packages Help Viewer
New Folder Delete Rename More
Home MyFirstR_Project
Name
Console Terminal Jobs
~/MyFirstR_Project/

```
Data sets in package 'datasets':

AirPassengers                Monthly Airline Passenger Numbers 1949-1960
BJsales                      Sales Data with Leading Indicator
BJsales.lead (BJsales)       Sales Data with Leading Indicator
BOD                          Biochemical Oxygen Demand
CO2                          Carbon Dioxide Uptake in Grass Plants
ChickWeight                  Weight versus age of chicks on different diets
DNase                        Elisa assay of DNase
EuStockMarkets               Daily Closing Prices of Major European Stock Indices,
                             1991-1998
Formaldehyde                 Determination of Formaldehyde
HairEyeColor                 Hair and Eye Color of Statistics Students
Harman23.cor                 Harman Example 2.3
Harman74.cor                 Harman Example 7.4
Indometh                     Pharmacokinetics of Indomethacin
InsectSprays                 Effectiveness of Insect Sprays
JohnsonJohnson               Quarterly Earnings per Johnson & Johnson Share
LakeHuron                    Level of Lake Huron 1875-1972
LifeCycleSavings             Intercountry Life-Cycle Savings Data
Loblolly                     Growth of Loblolly pine trees
Nile                         Flow of the River Nile
Orange                       Growth of Orange Trees
OrchardSprays                Potency of Orchard Sprays
PlantGrowth                  Results from an Experiment on Plant Growth
Puromycin                    Reaction Velocity of an Enzymatic Reaction
Seatbelts                    Road Casualties in Great Britain 1969-84
Theoph                       Pharmacokinetics of Theophylline
Titanic                      Survival of passengers on the Titanic
ToothGrowth                  The Effect of Vitamin C on Tooth Growth in Guinea Pigs
UCBAdmissions                Student Admissions at UC Berkeley
UKDriverDeaths               Road Casualties in Great Britain 1969-84
UKgas                        UK Quarterly Gas Consumption
USAccDeaths                  Accidental Deaths in the US 1973-1978
USArrests                    Violent Crime Rates by US State
USJudgeRatings               Lawyers' Ratings of State Judges in the US Superior
                             Court
USPersonalExpenditure        Personal Expenditure Data
UScitiesD                    Distances Between European Cities and Between US Cities
VADeaths                     Death Rates in Virginia (1940)
WWWusage                     Internet Usage per Minute
WorldPhones                  The World's Telephones
```

```
Values
x        8
```

```
..
.RData
.Rhistory
MyFirst-RScript.R
MyFirstR_Project.Rproj
WorkingWithDataInR.R
```

```
> data()
> data()
>
```

Fig. 2.5 List of pre-loaded datasets in R

You will be presented with a list of the pre-loaded datasets in R opened in a new tab as shown in Fig. 2.5.

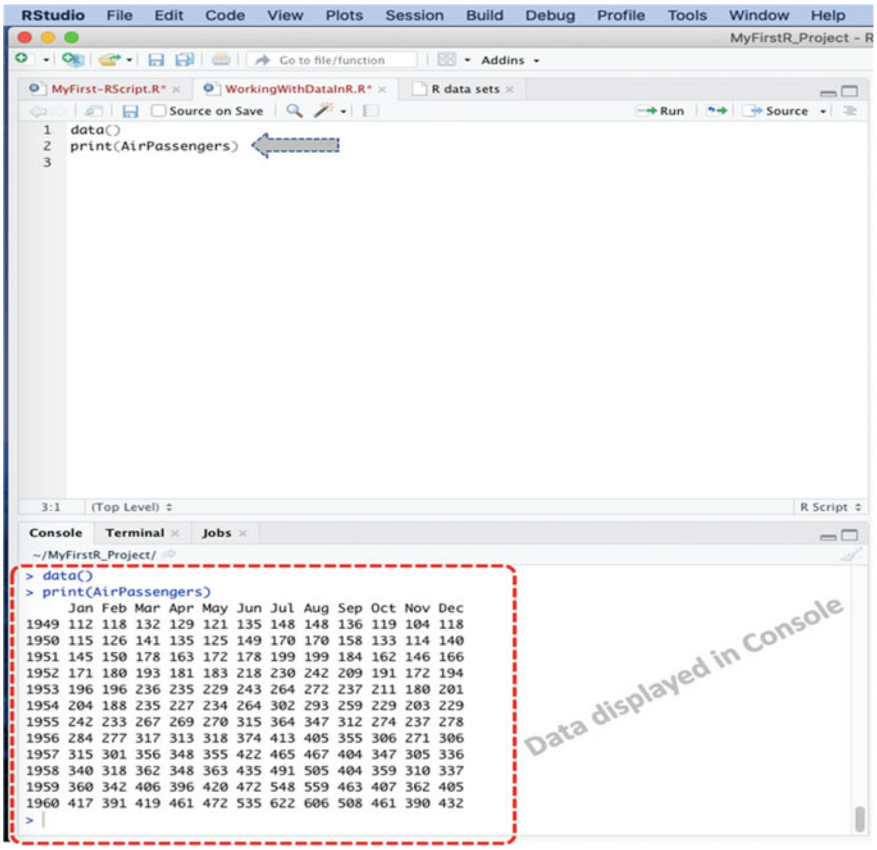
As shown in the figure (Fig. 2.5), we can see that R has a set of **pre-loaded data** in a package called "**datasets**". Each with its own different features or variable types. The users can printout any of these datasets of their choice, e.g., to view the different fields or variables that are contained in them.

For example, let's use the **print()** function to display the first data on that list (see Fig. 2.5) named "**AirPassengers**" as shown in Fig. 2.6.

To do this, go back to the "**WorkingWithDataInR**" script by clicking on the corresponding R Script tab (see Fig. 2.6), and then type and run the following command:

```
print(AirPassengers)
```

As shown in Fig. 2.6, the results of the **print()** command are displayed in the Console, and we can view the features and/or characteristics of the "AirPassengers" data.



```
data()
print(AirPassengers)
```

```
> data()
> print(AirPassengers)
     Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
1949 112 118 132 129 121 135 148 148 136 119 104 118
1950 115 126 141 135 125 149 170 170 158 133 114 140
1951 145 150 178 163 172 178 199 199 184 162 146 166
1952 171 180 193 181 183 218 230 242 209 191 172 194
1953 196 196 236 235 229 243 264 272 237 211 180 201
1954 204 188 235 227 234 264 302 293 259 229 203 229
1955 242 233 267 269 270 315 364 347 312 274 237 278
1956 284 277 317 313 318 374 413 405 355 306 271 306
1957 315 301 356 348 355 422 465 467 404 347 305 336
1958 340 318 362 348 363 435 491 505 404 359 310 337
1959 360 342 406 396 420 472 548 559 463 407 362 405
1960 417 391 419 461 472 535 622 606 508 461 390 432
> 
```

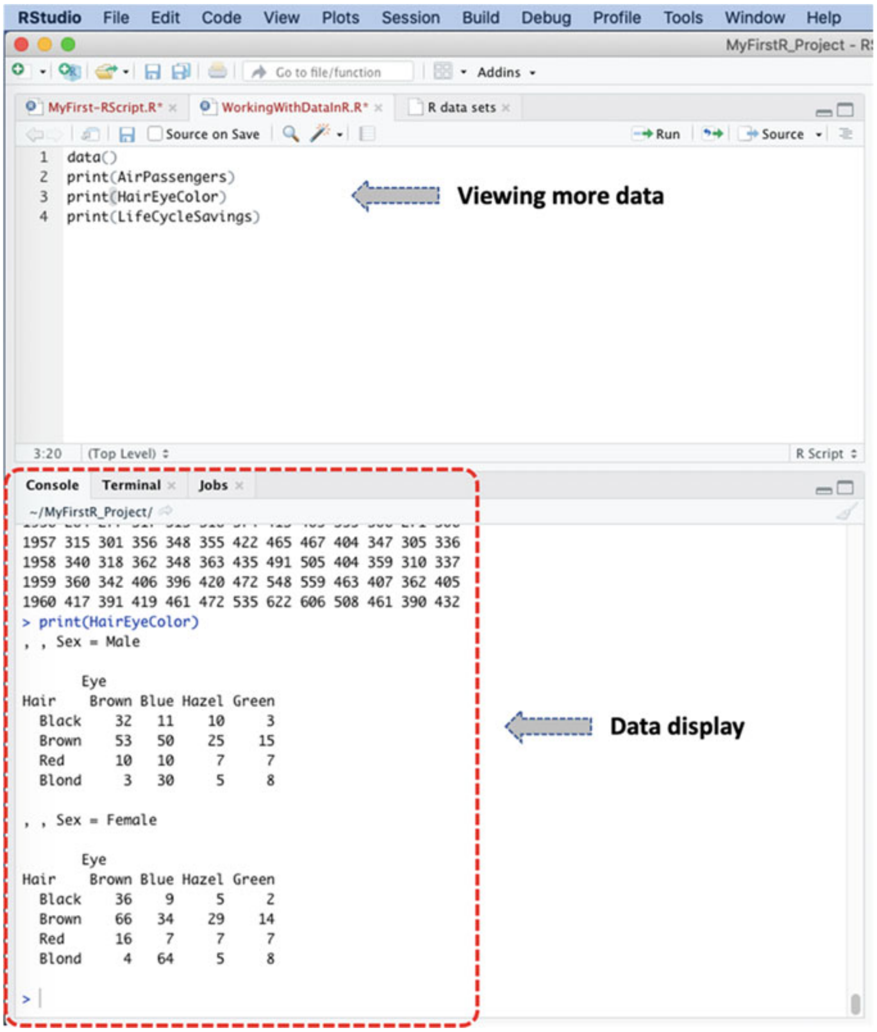
**Fig. 2.6** Exploring the different pre-loaded sample datasets in R

***You can follow the same method to view any of the data of your choice in the list. For instance, "HairEyeColour", "LifeCycleSavings", etc. (see Fig. 2.7).

```
print(HairEyeColor)
print(LifeCycleSavings)
```

## 2.3.2 *Creating Your Own Data in R*

R has a set of functions that allows the users to create their own data (aside from importing their own data which the authors will cover in the next section—Sect. 2.3.3).



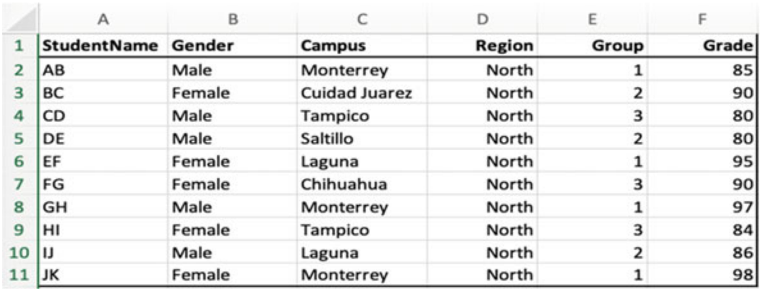
```
data()
print(AirPassengers)
print(HairEyeColor)
print(LifeCycleSavings)
```

```
1957 315 301 356 348 355 422 465 467 404 347 305 336
1958 340 318 362 348 363 435 491 505 404 359 310 337
1959 360 342 406 396 420 472 548 559 463 407 362 405
1960 417 391 419 461 472 535 622 606 508 461 390 432
> print(HairEyeColor)
, , Sex = Male

       Eye
Hair    Brown Blue Hazel Green
  Black    32   11    10     3
  Brown    53   50    25    15
  Red      10   10     7     7
  Blond     3   30     5     8

, , Sex = Female

       Eye
Hair    Brown Blue Hazel Green
  Black    36    9     5     2
  Brown    66   34    29    14
  Red      16    7     7     7
  Blond     4   64     5     8

> 
```

**Fig. 2.7** Viewing more data from the R sample "datasets" package

To illustrate how to create our own data in R, we will use the following R functions, **concatenate: c( )**, **repeat: rep( )**, and **Scan: scan( )** to create a dataset with some variables as shown in Fig. 2.8 (i.e., StudentName, Gender, Campus, Region, Group, and Grade). Then, we will create a "**dataframe"** to hold and display those variables as contained in the data.

**Note**: A "dataframe" is organized with rows and columns, similar to a spreadsheet or database table such as the one represented in Fig. 2.8.

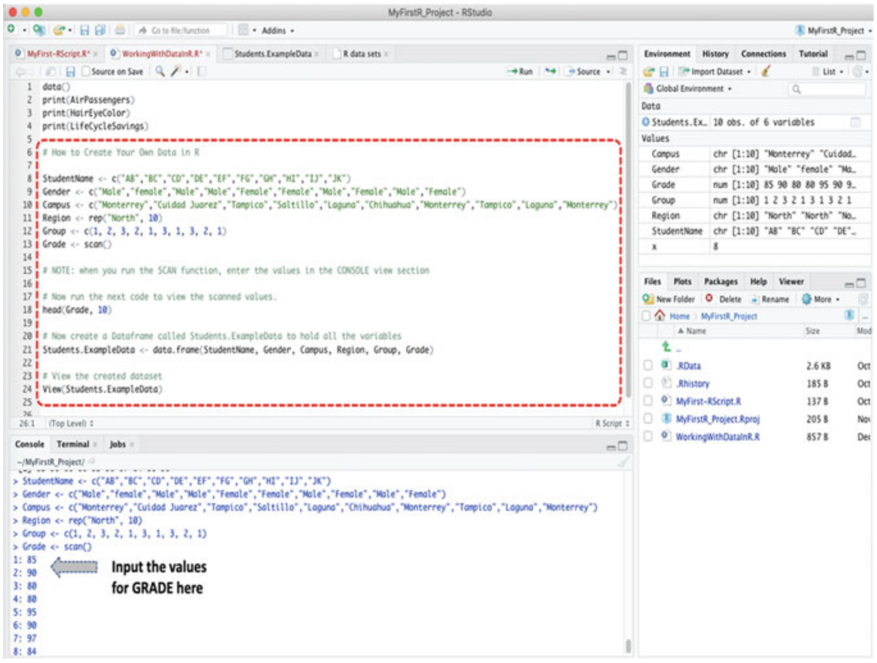


| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | StudentName | Gender | Campus | Region | Group | Grade |
| 2 | AB | Male | Monterrey | North | 1 | 85 |
| 3 | BC | Female | Cuidad Juarez | North | 2 | 90 |
| 4 | CD | Male | Tampico | North | 3 | 80 |
| 5 | DE | Male | Saltillo | North | 2 | 80 |
| 6 | EF | Female | Laguna | North | 1 | 95 |
| 7 | FG | Female | Chihuahua | North | 3 | 90 |
| 8 | GH | Male | Monterrey | North | 1 | 97 |
| 9 | HI | Female | Tampico | North | 3 | 84 |
| 10 | IJ | Male | Laguna | North | 2 | 86 |
| 11 | JK | Female | Monterrey | North | 1 | 98 |

**Fig. 2.8** Example of a typical data and structure format

To create the following data shown in Fig. 2.8 in R, first, the user needs to create a **vector** that holds all the fields or variables in the data (StudentName, Gender, Campus, Region, Group, and Grade).

Now in the "**WorkingWithDataInR**" Script, as shown in Fig. 2.9, create the aforelisted **vectors** (variables are described in the table in Fig. 2.8) using the concatenate: **c( ),** repeat: **rep( ),** and Scan: **scan()** functions.



```
data()
print(AirPassengers)
print(HairEyeColor)
print(LifeCycleSavings)

# How to Create Your Own Data in R

StudentName <- c("AB","BC","CD","DE","EF","FG","GH","HI","IJ","JK")
Gender <- c("Male","female","Male","Male","Female","Female","Male","Female","Male","Female")
Campus <- c("Monterrey","Cuidad Juarez","Tampico","Saltillo","Laguna","Chihuahua","Monterrey","Tampico","Laguna","Monterrey")
Region <- rep("North", 10)
Group <- c(1, 2, 3, 2, 1, 3, 1, 3, 2, 1)
Grade <- scan()

# NOTE: when you run the SCAN function, enter the values in the CONSOLE view section

# Now run the next code to view the scanned values.
head(Grade, 10)

# Now create a Dataframe called Students.ExampleData to hold all the variables
Students.ExampleData <- data.frame(StudentName, Gender, Campus, Region, Group, Grade)

# View the created dataset
View(Students.ExampleData)
```

```
~/MyFirstR_Project/
> StudentName <- c("AB","BC","CD","DE","EF","FG","GH","HI","IJ","JK")
> Gender <- c("Male","female","Male","Male","Female","Female","Male","Female","Male","Female")
> Campus <- c("Monterrey","Cuidad Juarez","Tampico","Saltillo","Laguna","Chihuahua","Monterrey","Tampico","Laguna","Monterrey")
> Region <- rep("North", 10)
> Group <- c(1, 2, 3, 2, 1, 3, 1, 3, 2, 1)
> Grade <- scan()
1: 85
2: 90
3: 80
4: 80
5: 95
6: 90
7: 97
8: 84
```

| | |
|---|---|
| Data | |
| Students.Ex… | 10 obs. of 6 variables |
| Values | |
| Campus | chr [1:10] "Monterrey" "Cuidad… |
| Gender | chr [1:10] "Male" "female" "Ma… |
| Grade | num [1:10] 85 90 80 80 95 90 9… |
| Group | num [1:10] 1 2 3 2 1 3 1 3 2 1 |
| Region | chr [1:10] "North" "North" "No… |
| StudentName | chr [1:10] "AB" "BC" "CD" "DE"… |
| x | 8 |

| Name | Size | Mod |
|---|---|---|
| .RData | 2.6 KB | Oct |
| .Rhistory | 185 B | Oct |
| MyFirst-RScript.R | 137 B | Oct |
| MyFirstR_Project.Rproj | 205 B | Nov |
| WorkingWithDataInR.R | 857 B | Dec |

**Fig. 2.9** How to create your own data in R

The following executed codes are provided in the command: *how to create your own data in R* (see: highlighted part in the Source Code Editor—Fig. 2.9, Lines 6–24).

```
# How to Create Your Own Data in R

StudentName <- c("AB","BC","CD","DE","EF","FG","GH","HI","IJ","JK")
Gender <- c("Male","female","Male","Male","Female","Female","Male","Female","Male","Female")
Campus <- c("Monterrey","Cuidad Juarez","Tampico","Saltillo","Laguna","Chihuahua","Monterrey",
"Tampico","Laguna","Monterrey")
Region <- rep("North", 10)
Group <- c(1, 2, 3, 2, 1, 3, 1, 3, 2, 1)
Grade <- scan()

# NOTE: when you run the SCAN function, enter the values in the CONSOLE view section

# Run the next code to view the scanned values.
head(Grade, 10)

# Create a Dataframe called Students.ExampleData to hold all the variables
Students.ExampleData <- data.frame(StudentName, Gender, Campus, Region, Group, Grade)

# View the created dataset
View(Students.ExampleData)
```

When the user have typed and **run** the lines of code up to the **Grade** vector (Line 13, Fig. 2.9), the different variables as contained in the table described in Fig. 2.8 will be created. Also see Chap. 1, Sect. 1.4 on how to run selected lines of code.

***Remember that you will need to input the values for the "Grade" in the **Console** tab view as pointed out in Fig. 2.9. When you enter the last value in the table, i.e., the 10th value which is 98, press the **Enter key** again to store the values in the "Grade" vector. You will receive a message that says "Read 10 items" as shown in Fig. 2.10.

*****Note**: You can create the "**Grade"** vector using the method we used to create the other vectors, e.g., "**Group"** vector. The **scan( )** method was only used to demonstrate the different functionalities the users can perform in R.

Now, run the rest of the codes, i.e., Lines 18 to 24 (see Fig. 2.9). You will discover that the dataframe named "**Students.ExampleData**" as shown in Fig. 2.11, will be created and displayed in a new tab. The content of this data (**Students.ExampleData)** should be similar to the one represented in Fig. 2.8.

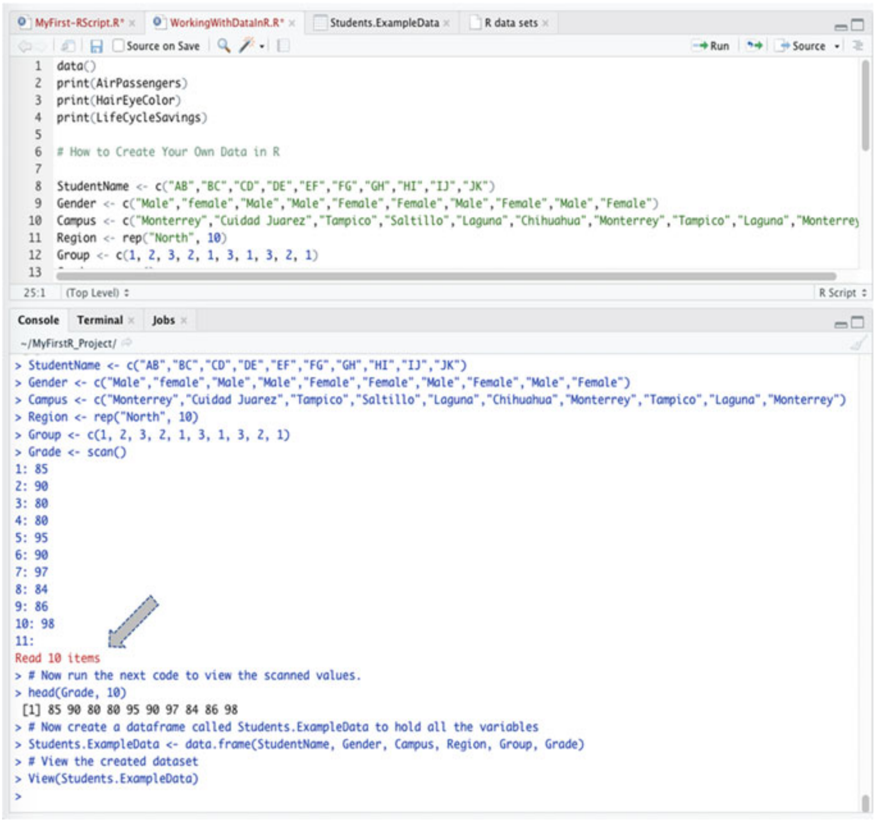
**Useful Tips and Information**:

1. As illustrated in Fig. 2.11 (see: highlighted part in the Environment Tab), we can see that when we created the "**Students.ExampleData"** dataframe; the values are stored as an "R *Object"* with details of the different variables type (e.g., Character, Number) and the number of observations (which the authors will discuss further in detail in the subsequent Sect. 2.4 of this chapter.
2. As shown in the Source Code (see Fig. 2.9), **Character** or **String** values are coded or represented using quotation marks **" ".**

   e.g. `StudentName <- c("AB","BC","CD","DE","EF","FG","GH","HI","IJ","JK")`

3. Whereas, **Number** or **Integer** values are *not* coded using quotes.

   e.g. `Group <- c(1, 2, 3, 2, 1, 3, 1, 3, 2, 1)`



```
data()
print(AirPassengers)
print(HairEyeColor)
print(LifeCycleSavings)

# How to Create Your Own Data in R

StudentName <- c("AB","BC","CD","DE","EF","FG","GH","HI","IJ","JK")
Gender <- c("Male","female","Male","Male","Female","Female","Male","Female","Male","Female")
Campus <- c("Monterrey","Cuidad Juarez","Tampico","Saltillo","Laguna","Chihuahua","Monterrey","Tampico","Laguna","Monterre
Region <- rep("North", 10)
Group <- c(1, 2, 3, 2, 1, 3, 1, 3, 2, 1)
```

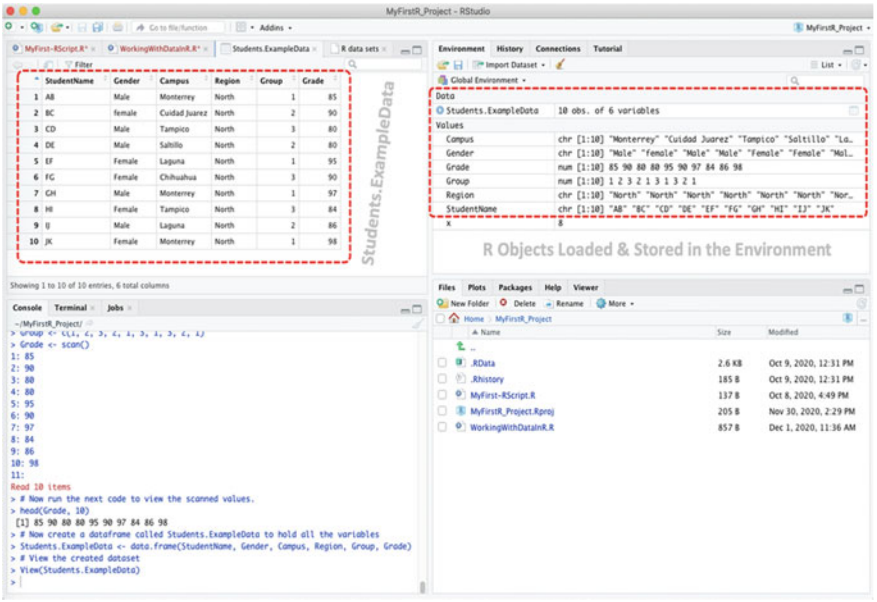
```
> StudentName <- c("AB","BC","CD","DE","EF","FG","GH","HI","IJ","JK")
> Gender <- c("Male","female","Male","Male","Female","Female","Male","Female","Male","Female")
> Campus <- c("Monterrey","Cuidad Juarez","Tampico","Saltillo","Laguna","Chihuahua","Monterrey","Tampico","Laguna","Monterrey")
> Region <- rep("North", 10)
> Group <- c(1, 2, 3, 2, 1, 3, 1, 3, 2, 1)
> Grade <- scan()
1: 85
2: 90
3: 80
4: 80
5: 95
6: 90
7: 97
8: 84
9: 86
10: 98
11:
Read 10 items
> # Now run the next code to view the scanned values.
> head(Grade, 10)
 [1] 85 90 80 80 95 90 97 84 86 98
> # Now create a dataframe called Students.ExampleData to hold all the variables
> Students.ExampleData <- data.frame(StudentName, Gender, Campus, Region, Group, Grade)
> # View the created dataset
> View(Students.ExampleData)
>
```

**Fig. 2.10** Using the scan( ) function in R

4. When using the repetition function: **rep( )**, the user also has to provide the number of times they want R to repeat the coded value. For instance, in the following example for the **Region** vector, the authors have requested the program to repeat “North” ten times.

```
Region <- rep("North", 10)
```

5. When using the **scan( )** function, once the user enters the last value as required, they need to press the **Enter key** again to read the item to the corresponding vector, as pointed in Fig. 2.11.
6. As we can see in Figs. 2.9 and 2.10 (and we will be using in many of the future codes in this book), the *hash sign* (#) is used to provide “**Pseudo Code**”, i.e., codes that **do not count or affect** the different functional lines of code, but are used in the codes to provide comments that are basically used to explain the functions of the different corresponding codes.



|  | StudentName | Gender | Campus | Region | Group | Grade |
|---|---|---|---|---|---|---|
| 1 | AB | Male | Monterrey | North | 1 | 85 |
| 2 | BC | female | Cuidad Juarez | North | 2 | 90 |
| 3 | CD | Male | Tampico | North | 3 | 80 |
| 4 | DE | Male | Saltillo | North | 2 | 80 |
| 5 | EF | Female | Laguna | North | 1 | 95 |
| 6 | FG | Female | Chihuahua | North | 3 | 90 |
| 7 | GH | Male | Monterrey | North | 1 | 97 |
| 8 | HI | Female | Tampico | North | 3 | 84 |
| 9 | IJ | Male | Laguna | North | 2 | 86 |
| 10 | JK | Female | Monterrey | North | 1 | 98 |

Showing 1 to 10 of 10 entries, 6 total columns

```
Data
Students.ExampleData    10 obs. of 6 variables
Values
  Campus                chr [1:10] "Monterrey" "Cuidad Juarez" "Tampico" "Saltillo" "La…
  Gender                chr [1:10] "Male" "female" "Male" "Male" "Female" "Female" "Mal…
  Grade                 num [1:10] 85 90 80 80 95 90 97 84 86 98
  Group                 num [1:10] 1 2 3 2 1 3 1 3 2 1
  Region                chr [1:10] "North" "North" "North" "North" "North" "North" "Nor…
  StudentName           chr [1:10] "AB" "BC" "CD" "DE" "EF" "FG" "GH" "HI" "IJ" "JK"
  x                     8
```

| Name | Size | Modified |
|---|---|---|
| .. | | |
| .RData | 2.6 KB | Oct 9, 2020, 12:31 PM |
| .Rhistory | 185 B | Oct 9, 2020, 12:31 PM |
| MyFirst-RScript.R | 137 B | Oct 8, 2020, 4:49 PM |
| MyFirstR_Project.Rproj | 205 B | Nov 30, 2020, 2:29 PM |
| WorkingWithDataInR.R | 857 B | Dec 1, 2020, 11:36 AM |

```
> Group <- c(1, 2, 3, 2, 1, 3, 1, 3, 2, 1)
> Grade <- scan()
1: 85
2: 90
3: 80
4: 80
5: 95
6: 90
7: 97
8: 84
9: 86
10: 98
11:
Read 10 items
> # Now run the next code to view the scanned values.
> head(Grade, 10)
 [1] 85 90 80 80 95 90 97 84 86 98
> # Now create a dataframe called Students.ExampleData to hold all the variables
> Students.ExampleData <- data.frame(StudentName, Gender, Campus, Region, Group, Grade)
> # View the created dataset
> View(Students.ExampleData)
> |
```

**Fig. 2.11** **"Students.ExampleData"** dataframe created and stored in R

## 2.3.3 *Import and Using External Data in R*

R has different functions and libraries devoted to reading and importing external files and formats into the Run-time environment, e.g., common.txt files (.csv) and Excel (.xlsx) files, STATA (.dta) and SPSS (.sav), etc. In this section, we will look at some of these functions and libraries, and how they are used to import datasets into R, and in the different contexts.

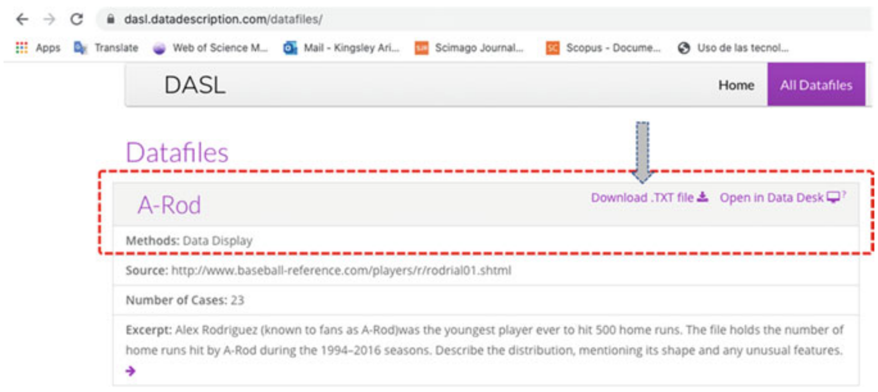
#### 2.3.3.1 Importing and Using .txt and .csv (Comma-Separated) Files in R

For this example, we will import a.txt and.csv files from the local machine or computer, and store the imported datasets in an R object named "**MyData.txt**" and "**MyData.csv**", respectively.

To demonstrate this function in R, visit the following links (see Figs. 2.12 and 2.13) and download the following datasets:

**.txt file** named "A-Rod"—https://dasl.datadescription.com/datafiles/ (Fig. 2.12), and.

**.csv file** named "Import_User_Sample_en.csv"—(https://www.microsoft.com/en-us/download/details.aspx?id=45485 (Fig. 2.13).



DASL

Datafiles

A-Rod

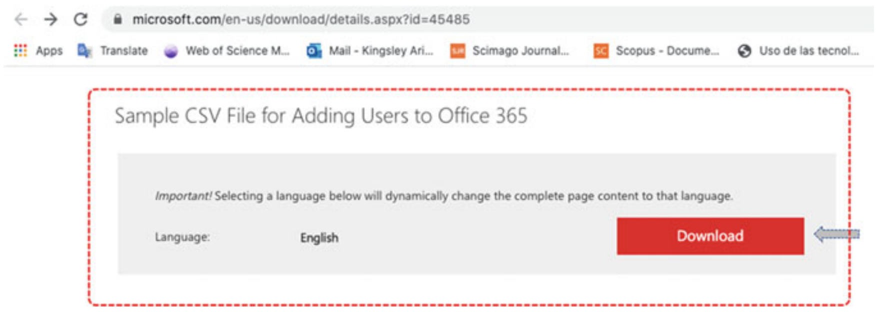
Methods: Data Display

Source: http://www.baseball-reference.com/players/r/rodrial01.shtml

Number of Cases: 23

Excerpt: Alex Rodriguez (known to fans as A-Rod)was the youngest player ever to hit 500 home runs. The file holds the number of home runs hit by A-Rod during the 1994–2016 seasons. Describe the distribution, mentioning its shape and any unusual features.

**Fig. 2.12** Example of.txt file download. (*Source* https://dasl.datadescription.com/datafiles/)



Sample CSV File for Adding Users to Office 365

*Important!* Selecting a language below will dynamically change the complete page content to that language.

Language: English

**Fig. 2.13** Example of.csv file download. (*Source* https://www.microsoft.com/en-us/download/details.aspx?id=45485)

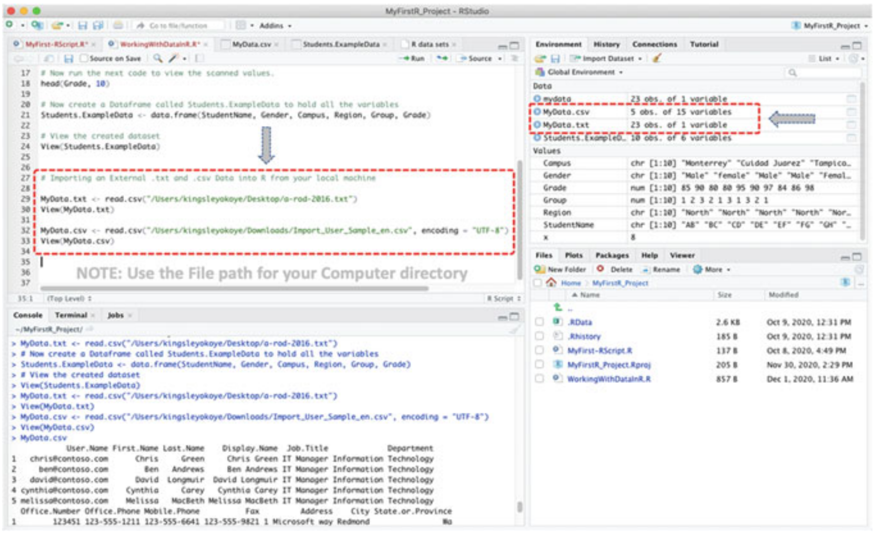
Store the downloaded files on your computer e.g., your **Desktop** location. Note, we will be pointing R to this location where this file is stored on the computer when running the commands. ***The user can store the files in any location on the system, provided they will point R to the correct location in the provided commands. The readers are welcome to use their own files if they wish to do so, or store the files in any location of choice on the computer***. The files and location used in the example described here in this chapter are for illustration purposes only.

Once the user has downloaded and stored the files on the computer **desktop**, we can now import the files into the R environment.

***Note: The example datasets can also be accessed via the following link or repository where the authors have uploaded all the example datasets used in this book: https://doi.org/https://doi.org/10.6084/m9.figshare.24728073.

**Useful Tips**:

- The two files are named “a-rod-2016.txt” and “Import_User_Sample_en.csv” by default when the user download the files.



```
17 # Now run the next code to view the scanned values.
18 head(Grade, 10)
19
20 # Now create a Dataframe called Students.ExampleData to hold all the variables
21 Students.ExampleData <- data.frame(StudentName, Gender, Campus, Region, Group, Grade)
22
23 # View the created dataset
24 View(Students.ExampleData)
25
26
27 # Importing an External .txt and .csv Data into R from your local machine
28
29 MyData.txt <- read.csv("/Users/kingsleyokoye/Desktop/a-rod-2016.txt")
30 View(MyData.txt)
31
32 MyData.csv <- read.csv("/Users/kingsleyokoye/Downloads/Import_User_Sample_en.csv", encoding = "UTF-8")
33 View(MyData.csv)
34
35
36
37
```

NOTE: Use the File path for your Computer directory

```
> MyData.txt <- read.csv("/Users/kingsleyokoye/Desktop/a-rod-2016.txt")
> # Now create a Dataframe called Students.ExampleData to hold all the variables
> Students.ExampleData <- data.frame(StudentName, Gender, Campus, Region, Group, Grade)
> # View the created dataset
> View(Students.ExampleData)
> MyData.txt <- read.csv("/Users/kingsleyokoye/Desktop/a-rod-2016.txt")
> View(MyData.txt)
> MyData.csv <- read.csv("/Users/kingsleyokoye/Downloads/Import_User_Sample_en.csv", encoding = "UTF-8")
> View(MyData.csv)
> MyData.csv
           User.Name First.Name Last.Name    Display.Name  Job.Title            Department
1   chris@contoso.com      Chris     Green     Chris Green IT Manager Information Technology
2     ben@contoso.com        Ben   Andrews     Ben Andrews IT Manager Information Technology
3   david@contoso.com      David  Longmuir  David Longmuir IT Manager Information Technology
4 cynthia@contoso.com    Cynthia     Carey   Cynthia Carey IT Manager Information Technology
5 melissa@contoso.com    Melissa   MacBeth Melissa MacBeth IT Manager Information Technology
  Office.Number Office.Phone Mobile.Phone          Fax         Address    City State.or.Province
1        123451 123-555-1211 123-555-6641 123-555-9821 1 Microsoft way Redmond                 Wa
```

```
Data
mydata                23 obs. of 1 variable
MyData.csv            5 obs. of 15 variables
MyData.txt            23 obs. of 1 variable
Students.ExampleD...  10 obs. of 6 variables
Values
  Campus              chr [1:10] "Monterrey" "Cuidad Juarez" "Tampico...
  Gender              chr [1:10] "Male" "female" "Male" "Male" "Femal...
  Grade               num [1:10] 85 90 80 80 95 90 97 84 86 98
  Group               num [1:10] 1 2 3 2 1 3 1 3 2 1
  Region              chr [1:10] "North" "North" "North" "North" "Nor...
  StudentName         chr [1:10] "AB" "BC" "CD" "DE" "EF" "FG" "GH" "...
  x                   8
```

| Name | Size | Modified |
|---|---|---|
| .RData | 2.6 KB | Oct 9, 2020, 12:31 PM |
| .Rhistory | 185 B | Oct 9, 2020, 12:31 PM |
| MyFirst-RScript.R | 137 B | Oct 8, 2020, 4:49 PM |
| MyFirstR_Project.Rproj | 205 B | Nov 30, 2020, 2:29 PM |
| WorkingWithDataInR.R | 857 B | Dec 1, 2020, 11:36 AM |

**Fig. 2.14** Importing an external .txt or .csv file into R

- The users can rename the files if they want, but should remember to provide the new name and correct file path in the codes or command.
- If the user cannot manually specify the file path for files stored on the computer system, they can follow the following steps to do so; navigate to the folder where the files are stored on your computer, for instance, in our example case the desktop, and then "right click" on the specific file and select "Properties" (for Windows OS) to see the file path details, or select "Copy file path" for Mac OS.

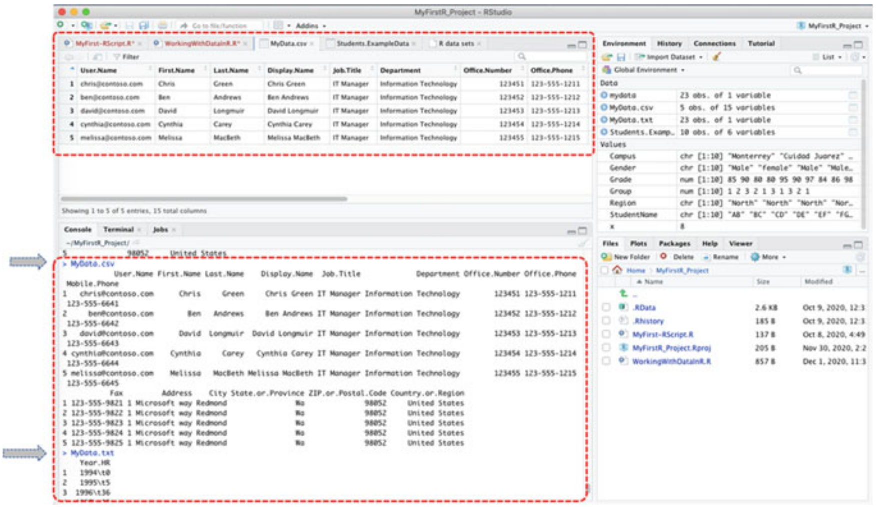
Now, let's go back to the "**WorkingWithDataInR**" Script we created earlier, and input the commands shown in Fig. 2.14 (lines 27–33).

As displayed in Fig. 2.14, the syntax to import the files into R are as follows:

```
R_Object_Name <- read.csv("insert_filepath_for_txt_doc_here")

R_Object_Name <- read.csv("insert_filepath_for_csv_doc_here", encoding =
"UTF-8")
```

For instance, in our example case, the authors have used the following command:

```
# Importing an External .txt and .csv Data into R from the local machine

MyData.txt <- read.csv("/Users/kingsleyokoye/Desktop/a-rod-2016.txt")
View(MyData.txt)

MyData.csv                                                              <-
read.csv("/Users/kingsleyokoye/Desktop/Import_User_Sample_en.csv", encoding
= "UTF-8")
View(MyData.csv)
```



| | User.Name | First.Name | Last.Name | Display.Name | Job.Title | Department | Office.Number | Office.Phone |
|---|---|---|---|---|---|---|---|---|
| 1 | chris@contoso.com | Chris | Green | Chris Green | IT Manager | Information Technology | 123451 | 123-555-1211 |
| 2 | ben@contoso.com | Ben | Andrews | Ben Andrews | IT Manager | Information Technology | 123452 | 123-555-1212 |
| 3 | david@contoso.com | David | Longmuir | David Longmuir | IT Manager | Information Technology | 123453 | 123-555-1213 |
| 4 | cynthia@contoso.com | Cynthia | Carey | Cynthia Carey | IT Manager | Information Technology | 123454 | 123-555-1214 |
| 5 | melissa@contoso.com | Melissa | MacBeth | Melissa MacBeth | IT Manager | Information Technology | 123455 | 123-555-1215 |

Showing 1 to 5 of 5 entries, 15 total columns

```
5                98052      United States
> MyData.csv
            User.Name First.Name Last.Name    Display.Name  Job.Title             Department Office.Number Office.Phone
 Mobile.Phone
1   chris@contoso.com      Chris     Green     Chris Green IT Manager Information Technology        123451 123-555-1211
 123-555-6641
2     ben@contoso.com        Ben   Andrews     Ben Andrews IT Manager Information Technology        123452 123-555-1212
 123-555-6642
3   david@contoso.com      David  Longmuir  David Longmuir IT Manager Information Technology        123453 123-555-1213
 123-555-6643
4 cynthia@contoso.com    Cynthia     Carey   Cynthia Carey IT Manager Information Technology        123454 123-555-1214
 123-555-6644
5 melissa@contoso.com    Melissa   MacBeth Melissa MacBeth IT Manager Information Technology        123455 123-555-1215
 123-555-6645
           Fax         Address    City State.or.Province ZIP.or.Postal.Code Country.or.Region
1 123-555-9821 1 Microsoft way Redmond                Wa              98052     United States
2 123-555-9822 1 Microsoft way Redmond                Wa              98052     United States
3 123-555-9823 1 Microsoft way Redmond                Wa              98052     United States
4 123-555-9824 1 Microsoft way Redmond                Wa              98052     United States
5 123-555-9825 1 Microsoft way Redmond                Wa              98052     United States
> MyData.txt
   Year.HR
1   1994\t0
2   1995\t5
3  1996\t36
```

| | |
|---|---|
| Data | |
| mydata | 23 obs. of 1 variable |
| MyData.csv | 5 obs. of 15 variables |
| MyData.txt | 23 obs. of 1 variable |
| Students.Examp… | 10 obs. of 6 variables |
| Values | |
| Campus | chr [1:10] "Monterrey" "Cuidad Juarez" … |
| Gender | chr [1:10] "Male" "female" "Male" "Male… |
| Grade | num [1:10] 85 90 80 80 95 90 97 84 86 98 |
| Group | num [1:10] 1 2 3 2 1 3 1 3 2 1 |
| Region | chr [1:10] "North" "North" "North" "Nor… |
| StudentName | chr [1:10] "AB" "BC" "CD" "DE" "EF" "FG… |
| x | 8 |

| Name | Size | Modified |
|---|---|---|
| .. | | |
| .RData | 2.6 KB | Oct 9, 2020, 12:3 |
| .Rhistory | 185 B | Oct 9, 2020, 12:3 |
| MyFirst-RScript.R | 137 B | Oct 8, 2020, 4:49 |
| MyFirstR_Project.Rproj | 205 B | Nov 30, 2020, 2:2 |
| WorkingWithDataInR.R | 857 B | Dec 1, 2020, 11:3 |

**Fig. 2.15** Example of .csv and .txt data imported into R

As shown and highlighted in Fig. 2.14, once the user has successfully imported the files, the *R objects* named "**MyData.txt**" and "**MyData.csv**" will be automatically created and used to store the imported files in the **Environment Tab,** with details of the different variable types and number of observations displayed there in R.

Also, the user will be able to view the content of the files in the Editor window when they run the View( ) command, i.e., **View(name_of_data)**—see Fig. 2.14, or Type and Run the name of the data (object), i.e., **"MyData.txt"** or **"MyData.csv"** directly in the **Console tab** (see Fig. 2.15).

**Useful Tips**:

- The read.csv( ) function assumes that your file has a header row, so by default, the first row is taken as the name of each column.
- However, in a situation whereby there is no label for each of the columns (which is often most likely not), the user can add the "header = FALSE" to the command (see code below). In such a situation, R will read the first line as data, not assuming a column header(s), done by automatically assigning a default column header names which the user can change afterward.

```
MyData.txt <- read.csv("filepath_for_txt_doc", header=FALSE)
```

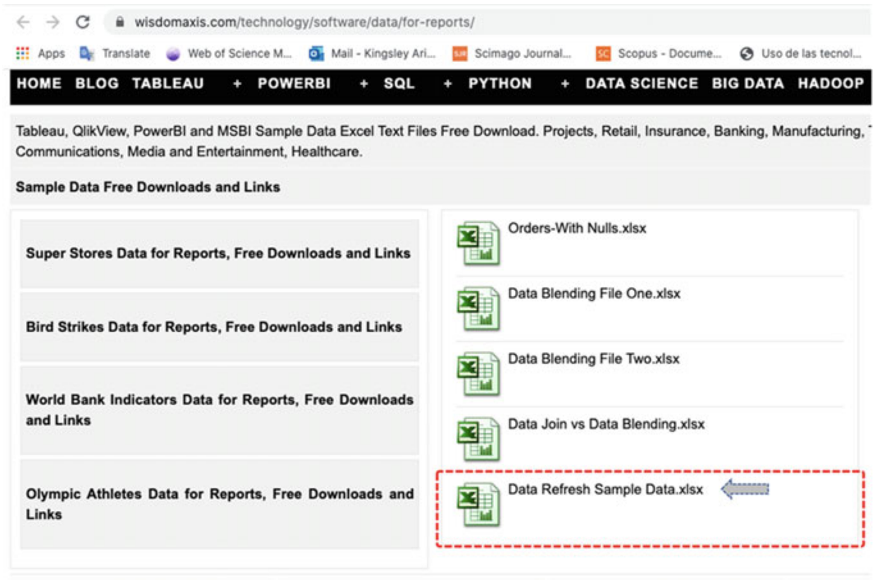
### 2.3.3.2 Importing and Using Excel (.xlsx) File Extension in R

Here, we will look at how to import or read **.xlsx file** (Excel) formats into R. In the previous section, we used a method that points (or references) R to the location where the files are stored on the computer. But for diversification or using different functions or method purposes, we will show another method that can allow the users to navigate (browse) to the location where the files are stored on the computer and ***directly choose the file*** they want, instead of specifying the location of the file in the code or program.

The users can download an .xlsx file example from the following source: https://www.wisdomaxis.com/technology/software/data/for-reports/ (users are welcome to use their own .xlsx file format if they wish to do so). Also, the example file can be accessed via the following repository (https://doi.org/https://doi.org/10.6084/m9.figshare.24728073) where the authors have uploaded all the example datasets used in this book.

As shown in Fig. 2.16, download and save the file named "**Data Refresh Sample Data.xlsx**" on your computer desktop.

Now, we will create an object named "**MyData.xlsx**" to hold and store this file when we import the dataset into R. As shown in Figs. 2.17 and 2.18, the syntax to import the file into R is as follows:


wisdomaxis.com/technology/software/data/for-reports/
Apps
Translate
Web of Science M...
Mail - Kingsley Ari...
Scimago Journal...
Scopus - Docume...
Uso de las tecnol...
HOME BLOG TABLEAU + POWERBI + SQL + PYTHON + DATA SCIENCE BIG DATA HADOOP

Tableau, QlikView, PowerBI and MSBI Sample Data Excel Text Files Free Download. Projects, Retail, Insurance, Banking, Manufacturing, Communications, Media and Entertainment, Healthcare.

Sample Data Free Downloads and Links

Super Stores Data for Reports, Free Downloads and Links

Bird Strikes Data for Reports, Free Downloads and Links

World Bank Indicators Data for Reports, Free Downloads and Links

Olympic Athletes Data for Reports, Free Downloads and Links

Orders-With Nulls.xlsx

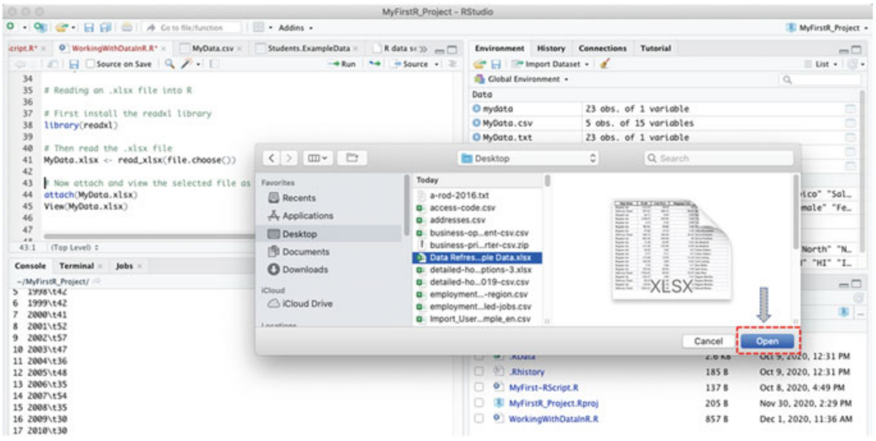
Data Blending File One.xlsx

Data Blending File Two.xlsx

Data Join vs Data Blending.xlsx

Data Refresh Sample Data.xlsx

**Fig. 2.16** .xlsx file download. (*Source* https://www.wisdomaxis.com/technology/software/data/for-reports/)



```
# Reading an .xlsx file into R

# First install the readxl library
library(readxl)

# Then read the .xlsx file
MyData.xlsx <- read_xlsx(file.choose())

# Now attach and view the selected file as
attach(MyData.xlsx)
View(MyData.xlsx)
```

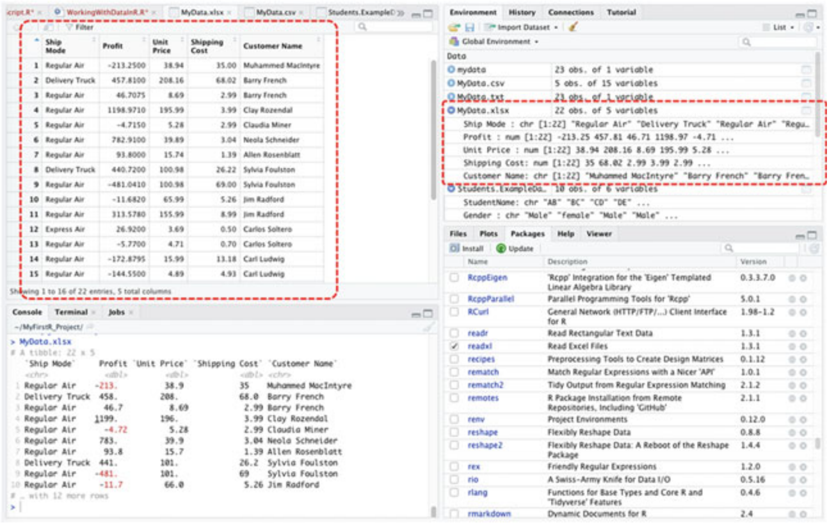
```
5  1998\t42
6  1999\t42
7  2000\t41
8  2001\t52
9  2002\t57
10 2003\t47
11 2004\t36
12 2005\t48
13 2006\t35
14 2007\t54
15 2008\t35
16 2009\t30
17 2010\t30
```

```
Data
mydata        23 obs. of 1 variable
MyData.csv    5 obs. of 15 variables
MyData.txt    23 obs. of 1 variable
```

| | | |
|---|---|---|
| .Rhistory | 185 B | Oct 9, 2020, 12:31 PM |
| MyFirst-RScript.R | 137 B | Oct 8, 2020, 4:49 PM |
| MyFirstR_Project.Rproj | 205 B | Nov 30, 2020, 2:29 PM |
| WorkingWithDataInR.R | 857 B | Dec 1, 2020, 11:36 AM |

MyFirstR_Project - RStudio
Favorites
Recents
Applications
Desktop
Documents
Downloads
iCloud
iCloud Drive
Today
a-rod-2016.txt
access-code.csv
addresses.csv
business-op...ent-csv.csv
business-pri...rter-csv.zip
Data Refres...ple Data.xlsx
detailed-ho...ptions-3.xlsx
detailed-ho...019-csv.csv
employment...-region.csv
employment...led-jobs.csv
Import_User...mple_en.csv
XLSX
Cancel
Open

**Fig. 2.17** Importing .xlsx file example into R



| | Ship Mode | Profit | Unit Price | Shipping Cost | Customer Name |
|---|---|---|---|---|---|
| 1 | Regular Air | -213.2500 | 38.94 | 35.00 | Muhammed MacIntyre |
| 2 | Delivery Truck | 457.8100 | 208.16 | 68.02 | Barry French |
| 3 | Regular Air | 46.7075 | 8.69 | 2.99 | Barry French |
| 4 | Regular Air | 1198.9710 | 195.99 | 3.99 | Clay Rozendal |
| 5 | Regular Air | -4.7150 | 5.28 | 2.99 | Claudia Miner |
| 6 | Regular Air | 782.9100 | 39.89 | 3.04 | Neola Schneider |
| 7 | Regular Air | 93.8000 | 15.74 | 1.39 | Allen Rosenblatt |
| 8 | Delivery Truck | 440.7200 | 100.98 | 26.22 | Sylvia Foulston |
| 9 | Regular Air | -481.0410 | 100.98 | 69.00 | Sylvia Foulston |
| 10 | Regular Air | -11.6820 | 65.99 | 5.26 | Jim Radford |
| 11 | Regular Air | 313.5780 | 155.99 | 8.99 | Jim Radford |
| 12 | Express Air | 26.9200 | 3.69 | 0.50 | Carlos Soltero |
| 13 | Regular Air | -5.7700 | 4.71 | 0.70 | Carlos Soltero |
| 14 | Regular Air | -172.8795 | 15.99 | 13.18 | Carl Ludwig |
| 15 | Regular Air | -144.5500 | 4.89 | 4.93 | Carl Ludwig |

Showing 1 to 16 of 22 entries, 5 total columns

```
> MyData.xlsx
# A tibble: 22 x 5
   `Ship Mode`     Profit `Unit Price` `Shipping Cost` `Customer Name`
   <chr>            <dbl>        <dbl>           <dbl> <chr>
 1 Regular Air     -213.          38.9           35    Muhammed MacIntyre
 2 Delivery Truck   458.         208.            68.0  Barry French
 3 Regular Air       46.7          8.69           2.99 Barry French
 4 Regular Air     1199.         196.             3.99 Clay Rozendal
 5 Regular Air       -4.72         5.28           2.99 Claudia Miner
 6 Regular Air      783.          39.9            3.04 Neola Schneider
 7 Regular Air       93.8         15.7            1.39 Allen Rosenblatt
 8 Delivery Truck   441.         101.            26.2  Sylvia Foulston
 9 Regular Air     -481.         101.            69    Sylvia Foulston
10 Regular Air      -11.7         66.0            5.26 Jim Radford
# … with 12 more rows
>
```

```
Data
mydata            23 obs. of 1 variable
MyData.csv        5 obs. of 15 variables
MyData.txt        23 obs. of 1 variable
MyData.xlsx       22 obs. of 5 variables
  Ship Mode : chr [1:22] "Regular Air" "Delivery Truck" "Regular Air" "Regu…
  Profit : num [1:22] -213.25 457.81 46.71 1198.97 -4.71 ...
  Unit Price : num [1:22] 38.94 208.16 8.69 195.99 5.28 ...
  Shipping Cost: num [1:22] 35 68.02 2.99 3.99 2.99 ...
  Customer Name: chr [1:22] "Muhammed MacIntyre" "Barry French" "Barry Fren…
Students.ExampleDa…  10 obs. of 6 variables
  StudentName: chr "AB" "BC" "CD" "DE" ...
  Gender : chr "Male" "female" "Male" "Male" ...
```

| | Name | Description | Version |
|---|---|---|---|
| | RcppEigen | 'Rcpp' Integration for the 'Eigen' Templated Linear Algebra Library | 0.3.3.7.0 |
| | RcppParallel | Parallel Programming Tools for 'Rcpp' | 5.0.1 |
| | RCurl | General Network (HTTP/FTP/...) Client Interface for R | 1.98-1.2 |
| | readr | Read Rectangular Text Data | 1.3.1 |
| ✓ | readxl | Read Excel Files | 1.3.1 |
| | recipes | Preprocessing Tools to Create Design Matrices | 0.1.12 |
| | rematch | Match Regular Expressions with a Nicer 'API' | 1.0.1 |
| | rematch2 | Tidy Output from Regular Expression Matching | 2.1.2 |
| | remotes | R Package Installation from Remote Repositories, Including 'GitHub' | 2.1.1 |
| | renv | Project Environments | 0.12.0 |
| | reshape | Flexibly Reshape Data | 0.8.8 |
| | reshape2 | Flexibly Reshape Data: A Reboot of the Reshape Package | 1.4.4 |
| | rex | Friendly Regular Expressions | 1.2.0 |
| | rio | A Swiss-Army Knife for Data I/O | 0.5.16 |
| | rlang | Functions for Base Types and Core R and 'Tidyverse' Features | 0.4.6 |
| | rmarkdown | Dynamic Documents for R | 2.4 |

**Fig. 2.18** Example of .xlsx data imported and stored as object in R

```
library(readxl)

MyData.xlsx <- read_xlsx(file.choose())

attach(MyData.xlsx)
```

If you notice, this time we used a *library* named “**readxl**” that allows us to implement the method or function to read the **.xlsx** file, which the authors will cover in detail in the subsequent section (Sect. 2.4) on how to install and use R packages and libraries.

Type the above lines of code (see Lines 35–45, Fig. 2.17) in the "**WorkingWithDataInR**" Script and Run the commands.

As illustrated in Fig. 2.17, when you have initiated the "**readxl"** library (Line 38), and run the next line of code to read the file (Line 41); you will be prompted with a pop-up window that allows the user to browse (navigate) to where the file is kept or stored on the computer, e.g., in our example case, the **desktop**.

Once you locate the file "**Data Refresh Sample Data.xlsx**" where it is stored on your computer, click on **Open** (see Fig. 2.17) to import the file, and then **attach** the file to the object called "**MyData.xlsx**" by running the rest of the code (Line 44 attaches the file to the object name, and Line 45 allows the user to view the imported file).

Now you have successfully read and attached the **.xlsx** file in R, you will be presented with a screen similar to the one in Fig. 2.18. You can also Type and Run the name of the data object (MyData.xlsx) directly in the Console tab to view the file details (Fig. 2.18), as we have covered earlier in the previous section.

#### 2.3.3.3 Importing and Using STATA (.dta) and SPSS (.sav) Files in R

In this section, we will look at how to import other file formats such as **STATA (.dta)** and **SPSS (.sav)** file into R environment. In R, such files are termed as foreign or distant files, and R has a library called "**foreign**" which can be used to read such files.

We will use another method of reading files (data) into R to do this, in order to continue to demonstrate the different ways the users can import datasets into R for analysis.

For this example, we will point R to the "**Folder"** where the files are stored by setting the folder as the **working directory**, and then read the data we want by just specifying the file name.

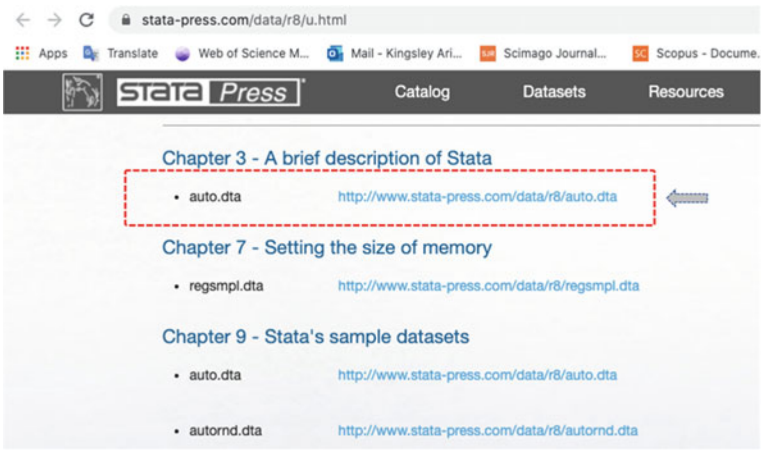
But first, let's download an example of **.dta** (Stata) and**.sav** (SPSS) files. As shown in Figs. 2.19 and 2.20, go to the following URLs and download the files named:

"**auto.dat**"—https://www.stata-press.com/data/r8/u.html (Fig. 2.19).

"**HLTH1025_2016.sav"—**https://lo.unisa.edu.au/mod/book/view.php?id=646443&chapterid=106604 (Fig. 2.20).

***Note: the users can also directly access and download the example files from the following repository where the authors have uploaded all the example datasets used in this book: https://doi.org/https://doi.org/10.6084/m9.figshare.24728073.

Once, the user has downloaded the **.dta** (Stata) and **.sav** (SPSS) files and saved them on the **Desktop,** go back to the "**WorkingWithDataInR**" Script, and enter the following codes (see highlighted part in Fig. 2.21, Lines 47–58).



Chapter 3 - A brief description of Stata

- auto.dta http://www.stata-press.com/data/r8/auto.dta

Chapter 7 - Setting the size of memory

- regsmpl.dta http://www.stata-press.com/data/r8/regsmpl.dta

Chapter 9 - Stata's sample datasets

- auto.dta http://www.stata-press.com/data/r8/auto.dta
- autornd.dta http://www.stata-press.com/data/r8/autornd.dta

**Fig. 2.19** Stata (.dta) file download. (*Source* https://www.stata-press.com/data/r8/u.html)

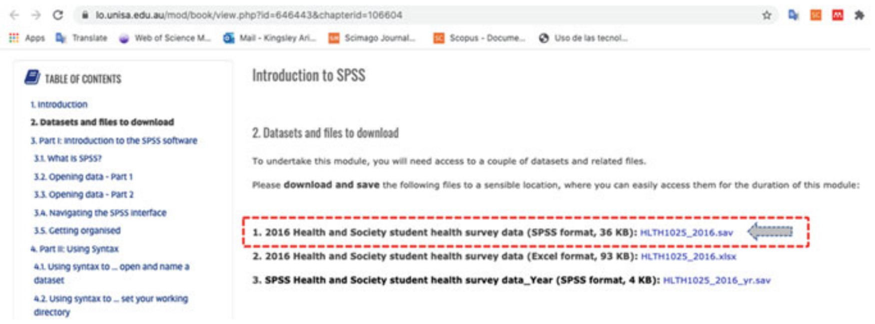


TABLE OF CONTENTS

1. Introduction
2. Datasets and files to download
3. Part I: Introduction to the SPSS software
3.1. What is SPSS?
3.2. Opening data - Part 1
3.3. Opening data - Part 2
3.4. Navigating the SPSS interface
3.5. Getting organised
4. Part II: Using Syntax
4.1. Using syntax to ... open and name a dataset
4.2. Using syntax to ... set your working directory

Introduction to SPSS

2. Datasets and files to download

To undertake this module, you will need access to a couple of datasets and related files.

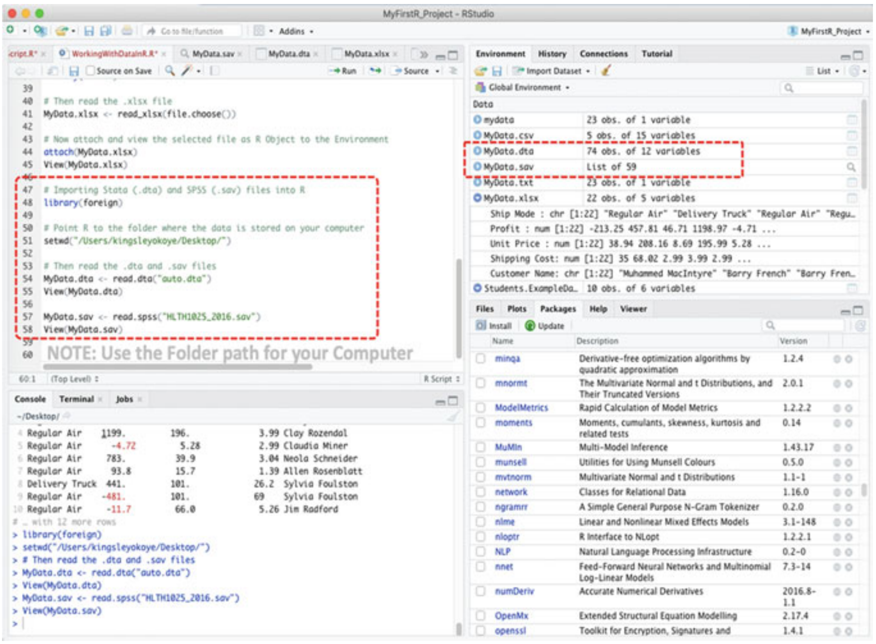
Please **download and save** the following files to a sensible location, where you can easily access them for the duration of this module:

**1. 2016 Health and Society student health survey data (SPSS format, 36 KB):** HLTH1025_2016.sav
**2. 2016 Health and Society student health survey data (Excel format, 93 KB):** HLTH1025_2016.xlsx
**3. SPSS Health and Society student health survey data_Year (SPSS format, 4 KB):** HLTH1025_2016_yr.sav

**Fig. 2.20** SPSS (.sav) file download. (*Source* https://lo.unisa.edu.au/mod/book/view.php?id=646443&chapterid=106604)

```
library(foreign)

setwd("Path_to_Folder")

MyData.dta <- read.dta("auto.dta")
View(MyData.dta)

MyData.sav <- read.spss("HLTH1025_2016.sav")
View(MyData.sav)
```

As illustrated in Fig. 2.21, we installed a library called "**foreign**" and then used the set working directory function, **setwd( )** to tell R where the files we will be importing are stored (i.e., which folder the user will be working from), in this authors' example case, the computer **desktop.**



```
39
40  # Then read the .xlsx file
41  MyData.xlsx <- read_xlsx(file.choose())
42
43  # Now attach and view the selected file as R Object to the Environment
44  attach(MyData.xlsx)
45  View(MyData.xlsx)
46
47  # Importing Stata (.dta) and SPSS (.sav) files into R
48  library(foreign)
49
50  # Point R to the folder where the data is stored on your computer
51  setwd("/Users/kingsleyokoye/Desktop/")
52
53  # Then read the .dta and .sav files
54  MyData.dta <- read.dta("auto.dta")
55  View(MyData.dta)
56
57  MyData.sav <- read.spss("HLTH1025_2016.sav")
58  View(MyData.sav)
59
60  NOTE: Use the Folder path for your Computer
```

```
~/Desktop/
4  Regular Air     1199.      196.       3.99 Clay Rozendal
5  Regular Air       -4.72      5.28     2.99 Claudia Miner
6  Regular Air      783.       39.9      3.04 Neola Schneider
7  Regular Air       93.8      15.7      1.39 Allen Rosenblatt
8  Delivery Truck   441.      101.      26.2  Sylvia Foulston
9  Regular Air     -481.      101.      69    Sylvia Foulston
10 Regular Air      -11.7      66.0      5.26 Jim Radford
# … with 12 more rows
> library(foreign)
> setwd("/Users/kingsleyokoye/Desktop/")
> # Then read the .dta and .sav files
> MyData.dta <- read.dta("auto.dta")
> View(MyData.dta)
> MyData.sav <- read.spss("HLTH1025_2016.sav")
> View(MyData.sav)
>
```

```
Data
mydata               23 obs. of 1 variable
MyData.csv           5 obs. of 15 variables
MyData.dta           74 obs. of 12 variables
MyData.sav           List of 59
MyData.txt           23 obs. of 1 variable
MyData.xlsx          22 obs. of 5 variables
   Ship Mode : chr [1:22] "Regular Air" "Delivery Truck" "Regular Air" "Regu…
   Profit : num [1:22] -213.25 457.81 46.71 1198.97 -4.71 ...
   Unit Price : num [1:22] 38.94 208.16 8.69 195.99 5.28 ...
   Shipping Cost: num [1:22] 35 68.02 2.99 3.99 2.99 ...
   Customer Name: chr [1:22] "Muhammed MacIntyre" "Barry French" "Barry Fren…
Students.ExampleDa…  10 obs. of 6 variables
```

| Name | Description | Version |
|---|---|---|
| minqa | Derivative-free optimization algorithms by quadratic approximation | 1.2.4 |
| mnormt | The Multivariate Normal and t Distributions, and Their Truncated Versions | 2.0.1 |
| ModelMetrics | Rapid Calculation of Model Metrics | 1.2.2.2 |
| moments | Moments, cumulants, skewness, kurtosis and related tests | 0.14 |
| MuMIn | Multi-Model Inference | 1.43.17 |
| munsell | Utilities for Using Munsell Colours | 0.5.0 |
| mvtnorm | Multivariate Normal and t Distributions | 1.1-1 |
| network | Classes for Relational Data | 1.16.0 |
| ngramrr | A Simple General Purpose N-Gram Tokenizer | 0.2.0 |
| nlme | Linear and Nonlinear Mixed Effects Models | 3.1-148 |
| nloptr | R Interface to NLopt | 1.2.2.1 |
| NLP | Natural Language Processing Infrastructure | 0.2-0 |
| nnet | Feed-Forward Neural Networks and Multinomial Log-Linear Models | 7.3-14 |
| numDeriv | Accurate Numerical Derivatives | 2016.8-1.1 |
| OpenMx | Extended Structural Equation Modelling | 2.17.4 |
| openssl | Toolkit for Encryption, Signatures and | 1.4.1 |

**Fig. 2.21** Importing .dta (Stata) and .sav (SPSS) file formats in R

Once the folder has been set (see Line 51, Fig. 2.21); for the rest of the code, you can see that we only need to specify the **file_name** as contained in the folder (in this case the desktop folder).

When you successfully run the codes (Lines 54–58), the datasets "**auto.dta**" and "**HLTH1025_2016.sav**" will be imported and stored as object in R (see Fig. 2.21), and the user will be presented with a screen similar to the one displayed in Fig. 2.22, where they can explore the datasets and details, i.e., "**MyData.dta**" and "**MyData.sav**".

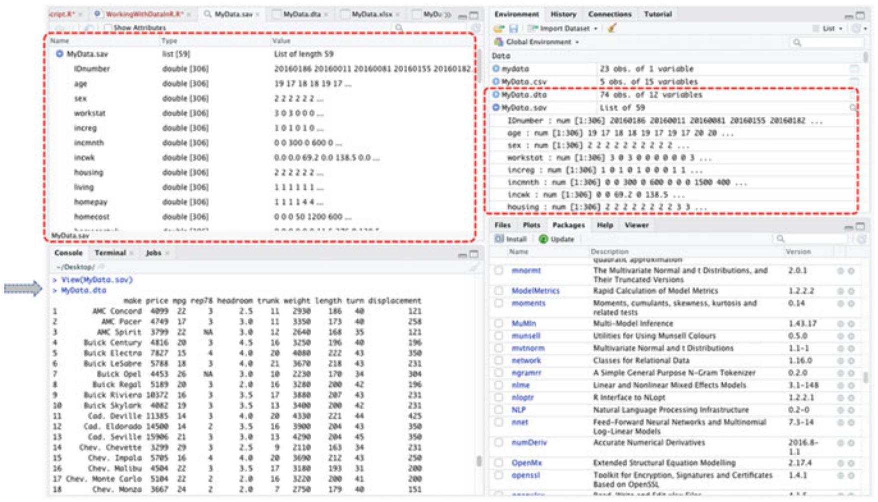
Finally, the users can also export any of the data types we have created so far to the local computer by using the **write.table( )** function.

For example, to export the "**MyData.dta**" or "**MyData.csv**" data we have created to the "**Desktop**", the authors have written the following command.

```
write.table(MyData.dta, file = "/Users/kingsleyokoye/Desktop/MyData.dta")

write.table(MyData.csv,  file  =  "/Users/kingsleyokoye/Desktop/MyData.csv",
sep = ",", row.names = FALSE)
```

*****Note**: Remember to write your specific own "file path" (e.g., to the "Desktop" used in our example) when writing the code, i.e., write.table(DataName, file = file_path_to_folder).

Over the next sections, the authors will explain some of the functionalities and methods we have implemented so far in R in detail, including the many other useful



| Name | Type | Value |
|---|---|---|
| MyData.sav | list [59] | List of length 59 |
| IDnumber | double [306] | 20160186 20160011 20160081 20160155 20160182 … |
| age | double [306] | 19 17 18 18 19 17 … |
| sex | double [306] | 2 2 2 2 2 2 … |
| workstat | double [306] | 3 0 3 0 0 0 … |
| increg | double [306] | 1 0 1 0 1 0 … |
| incmnth | double [306] | 0 0 300 0 600 0 … |
| incwk | double [306] | 0.0 0.0 69.2 0.0 138.5 0.0 … |
| housing | double [306] | 2 2 2 2 2 2 … |
| living | double [306] | 1 1 1 1 1 1 … |
| homepay | double [306] | 1 1 1 1 4 4 … |
| homecost | double [306] | 0 0 0 50 1200 600 … |

```
> View(MyData.sav)
> MyData.dta
                make price mpg rep78 headroom trunk weight length turn displacement
1        AMC Concord  4099  22     3      2.5    11   2930    186   40          121
2          AMC Pacer  4749  17     3      3.0    11   3350    173   40          258
3         AMC Spirit  3799  22    NA      3.0    12   2640    168   35          121
4      Buick Century  4816  20     3      4.5    16   3250    196   40          196
5      Buick Electra  7827  15     4      4.0    20   4080    222   43          350
6      Buick LeSabre  5788  18     3      4.0    21   3670    218   43          231
7         Buick Opel  4453  26    NA      3.0    10   2230    170   34          304
8        Buick Regal  5189  20     3      2.0    16   3280    200   42          196
9      Buick Riviera 10372  16     3      3.5    17   3880    207   43          231
10     Buick Skylark  4082  19     3      3.5    13   3400    200   42          231
11      Cad. Deville 11385  14     3      4.0    20   4330    221   44          425
12     Cad. Eldorado 14500  14     2      3.5    16   3900    204   43          350
13      Cad. Seville 15906  21     3      3.0    13   4290    204   45          350
14    Chev. Chevette  3299  29     3      2.5     9   2110    163   34          231
15      Chev. Impala  5705  16     4      4.0    20   3690    212   43          250
16      Chev. Malibu  4504  22     3      3.5    17   3180    193   31          200
17 Chev. Monte Carlo  5104  22     2      2.0    16   3220    200   41          200
18       Chev. Monza  3667  24     2      2.0     7   2750    179   40          151
```

| Data | |
|---|---|
| mydata | 23 obs. of 1 variable |
| MyData.csv | 5 obs. of 15 variables |
| MyData.dta | 74 obs. of 12 variables |
| MyData.sav | List of 59 |

```
IDnumber : num [1:306] 20160186 20160011 20160081 20160155 20160182 ...
age : num [1:306] 19 17 18 18 19 17 19 17 20 20 ...
sex : num [1:306] 2 2 2 2 2 2 2 2 2 2 ...
workstat : num [1:306] 3 0 3 0 0 0 0 0 0 3 ...
increg : num [1:306] 1 0 1 0 1 0 0 0 1 1 ...
incmnth : num [1:306] 0 0 300 0 600 0 0 0 1500 400 ...
incwk : num [1:306] 0 0 69.2 0 138.5 ...
housing : num [1:306] 2 2 2 2 2 2 2 2 3 3 ...
```

| Name | Description | Version |
|---|---|---|
| mnormt | The Multivariate Normal and t Distributions, and Their Truncated Versions | 2.0.1 |
| ModelMetrics | Rapid Calculation of Model Metrics | 1.2.2.2 |
| moments | Moments, cumulants, skewness, kurtosis and related tests | 0.14 |
| MuMIn | Multi-Model Inference | 1.43.17 |
| munsell | Utilities for Using Munsell Colours | 0.5.0 |
| mvtnorm | Multivariate Normal and t Distributions | 1.1-1 |
| network | Classes for Relational Data | 1.16.0 |
| ngramrr | A Simple General Purpose N-Gram Tokenizer | 0.2.0 |
| nlme | Linear and Nonlinear Mixed Effects Models | 3.1-148 |
| nloptr | R Interface to NLopt | 1.2.2.1 |
| NLP | Natural Language Processing Infrastructure | 0.2-0 |
| nnet | Feed-Forward Neural Networks and Multinomial Log-Linear Models | 7.3-14 |
| numDeriv | Accurate Numerical Derivatives | 2016.8-1.1 |
| OpenMx | Extended Structural Equation Modelling | 2.17.4 |
| openssl | Toolkit for Encryption, Signatures and Certificates Based on OpenSSL | 1.4.1 |

**Fig. 2.22** Example of .dta (Stata) and .sav (SPSS) data imported in R

functions and components that we will be exploring or come across in the future chapters, particularly when conducting the different statistical data analysis in R in PART II of this book.

## 2.4 R Objects

Understanding how ***R objects*** are created, and *values* assigned to the objects is key to using R for programming or statistical data analysis. R is an "object-oriented programming language" (Mailund, 2017; Matloff, 2011) and so everything or tasks you perform in R is all about *objects.*

To create an "**object"** in R, the user first needs to give the *object* a name so that R will know about this and then the user can use or manipulate this object anytime they want in the program using that name. Second, the user subsequently needs to assign a "**value**" or "**values**" to this object using the **assignment operator** "<–"**.** As illustrated in the example below, the assignment operator consists of a less than (<) and a minus or dash (–) sign typed together.

For instance, a typical R *object* will be created as follows:

```
my_object1 <- 234
my_object2 <- c("east", "west", "north", "south")
my_object3 <- read.csv("/Users/kingsleyokoye/Desktop/a-rod-2016.txt") etc
```

As shown in the codes above, the authors created an object called "**my_object1**" and assigned it a value of the number "**234**" using the assignment operator (**<**–). In a natural language, we will read this command as "my_object1 gets 234 as value".

Same principle applies to **my_object2** and **my_object3**, we can read this as "my_object2 gets a combination of east, west, north, and south as values". Whereas, "my_object3 reads a file named a-rod-2016.txt from my computer desktop and assigns the values of this file to the my_object3".

In order to view the values of the R objects, the user simply needs to type the name of the object they want to view in the **Console** and run it, or use the **View( )** function to execute this in the Editor window as the authors have used and illustrated in most of the example cases and tasks performed in the previous sections (see Section 2.3) of this chapter. For example, see the illustration in Fig. 2.23.

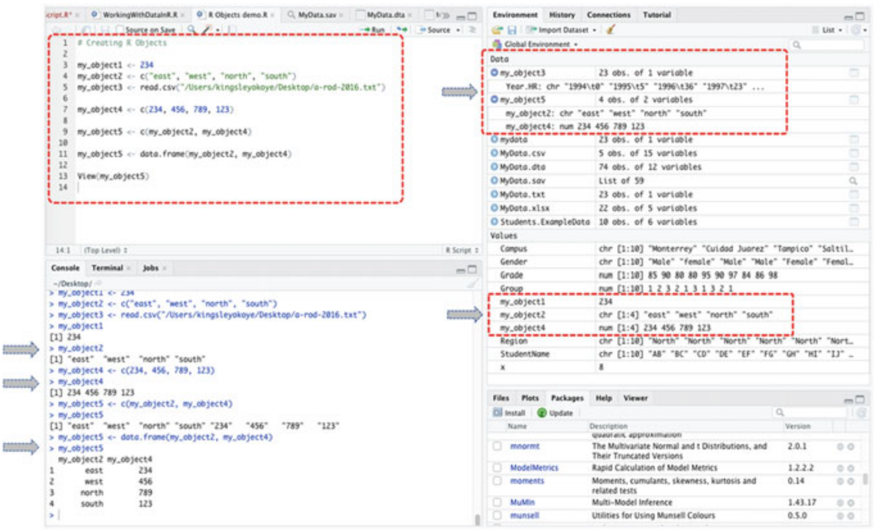
Now that we are familiar with the concept of R objects, and have also practically created the following objects (my_object1, my_object2, my_object3); R will know about these objects and what information is contained in them (remember, you can view all the details of the different objects in the workbench through the **Environment Tab**—see Fig. 2.23).

So, in practice, one can use those objects we have created anytime they want or further manipulate/extend the objects in the different tasks or analyses, as we will illustrate further in the coming sections.

For example, as shown in Fig. 2.23 (Line 7), we create a fourth object called "**my_object4**" that will hold the following numbers: 234, 456, 789, 123, and then merge this object (my_object4) with the existing "my_object2" to create a new set of information or data called "**my_object5**" (see Fig. 2.23, Line 9).

To do this, we first need to create the **my_object4** as follows:

```
my_object4 <- c(234, 456, 789, 123)
```



```
1  # Creating R Objects
2
3  my_object1 <- 234
4  my_object2 <- c("east", "west", "north", "south")
5  my_object3 <- read.csv("/Users/kingsleyokoye/Desktop/a-rod-2016.txt")
6
7  my_object4 <- c(234, 456, 789, 123)
8
9  my_object5 <- c(my_object2, my_object4)
10
11 my_object5 <- data.frame(my_object2, my_object4)
12
13 View(my_object5)
14
```

```
> my_object1 <- 234
> my_object2 <- c("east", "west", "north", "south")
> my_object3 <- read.csv("/Users/kingsleyokoye/Desktop/a-rod-2016.txt")
> my_object1
[1] 234
> my_object2
[1] "east"  "west"  "north" "south"
> my_object4 <- c(234, 456, 789, 123)
> my_object4
[1] 234 456 789 123
> my_object5 <- c(my_object2, my_object4)
> my_object5
[1] "east"  "west"  "north" "south" "234"   "456"   "789"   "123"
> my_object5 <- data.frame(my_object2, my_object4)
> my_object5
  my_object2 my_object4
1       east        234
2       west        456
3      north        789
4      south        123
> 
```

```
Data
my_object3              23 obs. of 1 variable
  Year.HR: chr "1994\t0" "1995\t5" "1996\t36" "1997\t23" ...
my_object5              4 obs. of 2 variables
  my_object2: chr "east" "west" "north" "south"
  my_object4: num 234 456 789 123
mydata                  23 obs. of 1 variable
MyData.csv              5 obs. of 15 variables
MyData.dta              74 obs. of 12 variables
MyData.sav              List of 59
MyData.txt              23 obs. of 1 variable
MyData.xlsx             22 obs. of 5 variables
Students.ExampleData    10 obs. of 6 variables
Values
  Campus                chr [1:10] "Monterrey" "Cuidad Juarez" "Tampico" "Saltil…
  Gender                chr [1:10] "Male" "female" "Male" "Male" "Female" "Femal…
  Grade                 num [1:10] 85 90 80 80 95 90 97 84 86 98
  Group                 num [1:10] 1 2 3 2 1 3 1 3 2 1
  my_object1            234
  my_object2            chr [1:4] "east" "west" "north" "south"
  my_object4            num [1:4] 234 456 789 123
  Region                chr [1:10] "North" "North" "North" "North" "North" "Nort…
  StudentName           chr [1:10] "AB" "BC" "CD" "DE" "EF" "FG" "GH" "HI" "IJ" …
  x                     8
```

| Name | Description | Version |
|---|---|---|
| mnormt | The Multivariate Normal and t Distributions, and Their Truncated Versions | 2.0.1 |
| ModelMetrics | Rapid Calculation of Model Metrics | 1.2.2.2 |
| moments | Moments, cumulants, skewness, kurtosis and related tests | 0.14 |
| MuMIn | Multi-Model Inference | 1.43.17 |
| munsell | Utilities for Using Munsell Colours | 0.5.0 |

**Fig. 2.23** Creating R objects

Then we will create "**my_object5**" by merging "**my_object2**" and "**my_object4**".

```
my_object5 <- c(my_object2, my_object4)
```

Now, type "**my_object5"** in the **Console** to see the new information or data we have created.

```
Output:
> my_object5
[1] "east"  "west"  "north" "south" "234"   "456"   "789"   "123"
```

Furthermore, instead of merging the following two objects (my_object2, my_object4) with different **Character (chr)** and **Numerical** (num) values; let's create a **dataframe** that will assign each of the values (num) in **my_object4** (i.e., 234, 456, 789, 123) to the corresponding values (chr) in **my_object2** (i.e., "east", "west", "north", "south") as a table using dataframe.

To do this, we can use the following syntax (see Fig. 2.23, Line 11):

```
my_object5 <- data.frame(my_object2, my_object4)
```

Now, type "**my_object5"** in the **Console** to see the **dataframe** we have created and how we have structured and changed the characteristics of the resultant or corresponding object (Fig. 2.23).

```
> my_object5
  my_object2 my_object4
1       east        234
2       west        456
3      north        789
4      south        123
```

The user can also use the **view( )** function to display the data through the "Source Code Editor" window.

As illustrated in this section and examples so far, this is how R keeps track of all the **objects** the users create during the running or respective R sessions.

## 2.5 R Vectors: Vectorization and Factorization

Creating and working with "Vectors" in R can serve as a more organized way of handling too many or complicated R objects. As we demonstrated in the previous section (Sect. 2.4) e.g., with regards to how we created the "**my_object5**" in R. A lot of the time, when using R, the users will want to progress further or create more complicated R objects. Fortunately, R has a vast range of **functions** that can help the users do this.

Basically, an R *function* or *method* can be referred to as an *object* that contains a series of instructions defined to perform a specific job.

By default, R comes with over thousands of *packages* already defined to help the users in performing the different tasks, functions, or data analyses ( these are covered in the next Sect. 2.6 of this book). As soon as the users get conversant with R or the data analysis tasks in R become more advanced, they will find that in some cases they may or will need to define their own functions in order to perform some customized or additional/specific tasks that are tailored to their needs, goals or objectives.

### *2.5.1 Creating and Working with Vectors in R*

Here, the authors will demonstrate what a simple function in R is; and how to create what is called the ***vectors***, which is simply a type of R object but **with same type (kind) or sequence of elements (components)** contained in it. For illustration purpose, we can define a "vector" as a single column in an Excel spreadsheet in which the values are usually of the same type, e.g., either *numbers* or *characters* but not a mixture of both. We have covered some of these in the previous section (Sect. 2.3), but we will explain this in detail here.

**Note**:

1. A function in R is syntactically (always) followed by a pair of round brackets ( ), i.e., **functionName( )** even when there is nothing defined in between the brackets.
2. The argument of a function are always placed inside the **brackets ( ),** and if there are more than one argument, are separated using commas.

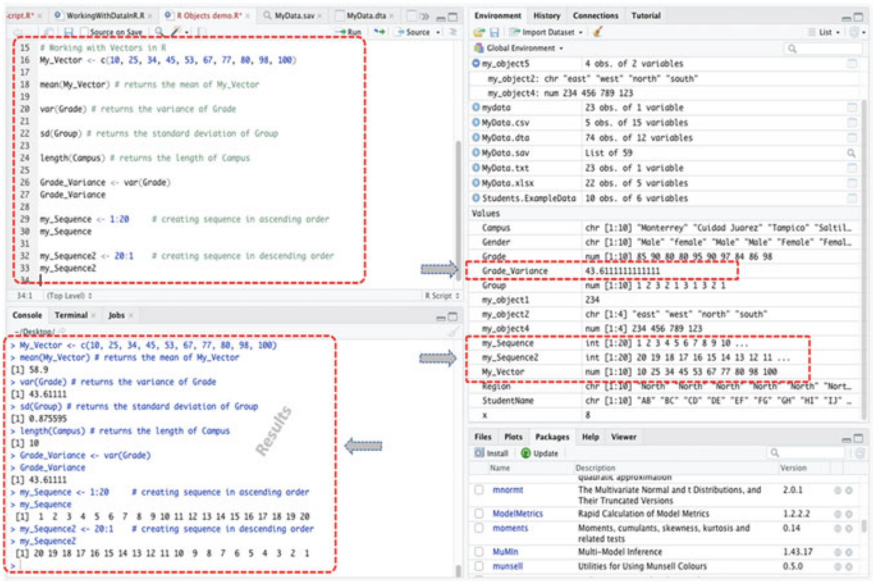
Keeping these views in mind, in order to demonstrate what a **vector** is, and how we can use the different *functions* in R to manipulate this. The first function we will look at is the ***concatenate*** function, i.e., **c( ).** This function is very useful when joining together a series or sequence of values and then storing the values (elements) in a data structure called the **vector.**

To do this, let's create a new Script or go to the "WorkingWithDataInR" script we have created in the previous examples, and type the following command as shown in Fig. 2.24.

```
My_Vector <- c(10, 25, 34, 45, 53, 67, 77, 80, 98, 100)
```

In the example code above, the authors created an object called "**My_Vector**" and assigned it a value or some elements (i.e., **10, 25, 34, 45, 53, 67, 77, 80, 98, 100**) using the concatenate function **c( ).**

As the reader can see, the above vector is similar to the R objects (i.e., StudentName, Gender, Campus, Region, Group, and Grade) we have created when creating the **Students.ExampleData** in Fig. 2.11 (see Sect. 2.3.2). We can also use those objects define there in Sect. 2.3.2 to illustrate the further example functions to work with our vectors.



```
15  # Working with Vectors in R
16  My_Vector <- c(10, 25, 34, 45, 53, 67, 77, 80, 98, 100)
17
18  mean(My_Vector) # returns the mean of My_Vector
19
20  var(Grade) # returns the variance of Grade
21
22  sd(Group) # returns the standard deviation of Group
23
24  length(Campus) # returns the length of Campus
25
26  Grade_Variance <- var(Grade)
27  Grade_Variance
28
29  my_Sequence <- 1:20      # creating sequence in ascending order
30  my_Sequence
31
32  my_Sequence2 <- 20:1     # creating sequence in descending order
33  my_Sequence2
34
```

```
> My_Vector <- c(10, 25, 34, 45, 53, 67, 77, 80, 98, 100)
> mean(My_Vector) # returns the mean of My_Vector
[1] 58.9
> var(Grade) # returns the variance of Grade
[1] 43.61111
> sd(Group) # returns the standard deviation of Group
[1] 0.875595
> length(Campus) # returns the length of Campus
[1] 10
> Grade_Variance <- var(Grade)
> Grade_Variance
[1] 43.61111
> my_Sequence <- 1:20      # creating sequence in ascending order
> my_Sequence
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
> my_Sequence2 <- 20:1     # creating sequence in descending order
> my_Sequence2
 [1] 20 19 18 17 16 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1
>
```

```
my_object5              4 obs. of 2 variables
   my_object2: chr "east" "west" "north" "south"
   my_object4: num 234 456 789 123
mydata                  23 obs. of 1 variable
MyData.csv              5 obs. of 15 variables
MyData.dta              74 obs. of 12 variables
MyData.sav              List of 59
MyData.txt              23 obs. of 1 variable
MyData.xlsx             22 obs. of 5 variables
Students.ExampleData    10 obs. of 6 variables
Values
  Campus                chr [1:10] "Monterrey" "Cuidad Juarez" "Tampico" "Saltil…
  Gender                chr [1:10] "Male" "female" "Male" "Male" "Female" "Femal…
  Grade                 num [1:10] 85 90 80 80 95 90 97 84 86 98
  Grade_Variance        43.6111111111111
  Group                 num [1:10] 1 2 3 2 1 3 1 3 2 1
  my_object1            234
  my_object2            chr [1:4] "east" "west" "north" "south"
  my_object4            num [1:4] 234 456 789 123
  my_Sequence           int [1:20] 1 2 3 4 5 6 7 8 9 10 ...
  my_Sequence2          int [1:20] 20 19 18 17 16 15 14 13 12 11 ...
  My_Vector             num [1:10] 10 25 34 45 53 67 77 80 98 100
  Region                chr [1:10] "North" "North" "North" "North" "North" "Nort…
  StudentName           chr [1:10] "AB" "BC" "CD" "DE" "EF" "FG" "GH" "HI" "IJ" …
  x                     8
```

| | Name | Description | Version |
|---|---|---|---|
| | | quadratic approximation | |
| ☐ | mnormt | The Multivariate Normal and t Distributions, and Their Truncated Versions | 2.0.1 |
| ☐ | ModelMetrics | Rapid Calculation of Model Metrics | 1.2.2.2 |
| ☐ | moments | Moments, cumulants, skewness, kurtosis and related tests | 0.14 |
| ☐ | MuMIn | Multi-Model Inference | 1.43.17 |
| ☐ | munsell | Utilities for Using Munsell Colours | 0.5.0 |

**Fig. 2.24** Creating and working with vectors in R

To illustrate the use of vectors in R, for example, we can calculate the **mean, variance, standard deviation,** and **length** (number of elements) in any of the listed objects (Grade, Group, Campus, My_Vector) using the **mean( ), var( ), sd( ),** and **length()** functions as shown in the following codes (see Fig. 2.24).

```
mean(My_Vector)        # returns the mean of My_Vector object

var(Grade)             # returns the variance of Grade

sd(Group)              # returns the standard deviation of Group

length(Campus)         # returns the length of Campus
```

It is also important to mention that, in case the user wants to use any of the resulting values or output (e.g., the variance of the Grades) later in the individual or further analysis, then they can decide to assign this result or value to another object, e.g., called "**Grade_variance**" (see Code below, Fig. 2.24).

```
Grade_Variance <- var(Grade)

Grade_Variance
```

Now, when the user type and run the "Grade_variance" object, as shown in Line 27 in Fig. 2.24, same value or result for the command or function var(Grade) is returned.

### 2.5.2 *Understanding Sequence in Vectors*

R users can create a vector that contains a “**regular sequence of values”,** e.g., in ascending or descending order using the colon symbol “**:**” (see Lines 29–33, Fig. 2.24).

```
my_Sequence <- 1:20     # creating sequence in ascending order
my_Sequence

my_Sequence2 <- 20:1    # creating sequence in descending order
my_Sequence2
```

The users can also use or experiment with other useful functions used to generate vectors, e.g., using the **seq( )** and **rep()** functions.

For example (see Fig. 2.25):

One can use the **seq()** function to generate a series or order of numbers from 10 to 20 in steps of 0.5,

```
my_Sequence3 <- seq(from = 10, to = 20, by = 0.5)  # create sequence in Steps
my_Sequence3
```

or

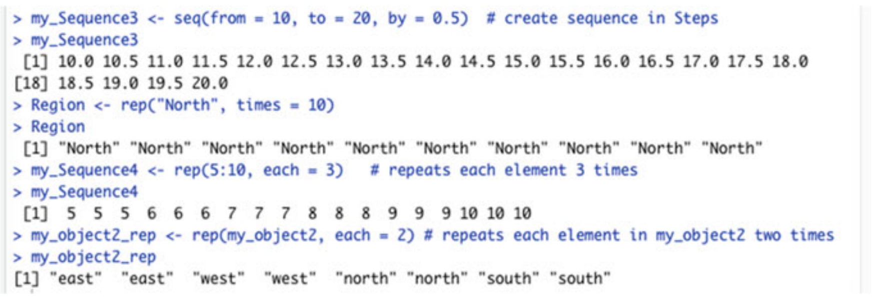
Use the **rep()** to replicate (repeat) a String (chr) value for a specified number of times, e.g., repeat the value “North” ten (10) times, as shown in the following command

```
Region <- rep(“North”, times = 10)
Region
```

Likewise, the users can uniformly repeat elements of a series (see Fig. 2.25):

```
my_Sequence4 <- rep(5:10, each = 3)   # repeats each element 3 times
my_Sequence4
```

or



```
> my_Sequence3 <- seq(from = 10, to = 20, by = 0.5)  # create sequence in Steps
> my_Sequence3
 [1] 10.0 10.5 11.0 11.5 12.0 12.5 13.0 13.5 14.0 14.5 15.0 15.5 16.0 16.5 17.0 17.5 18.0
[18] 18.5 19.0 19.5 20.0
> Region <- rep("North", times = 10)
> Region
 [1] "North" "North" "North" "North" "North" "North" "North" "North" "North" "North"
> my_Sequence4 <- rep(5:10, each = 3)  # repeats each element 3 times
> my_Sequence4
 [1]  5  5  5  6  6  6  7  7  7  8  8  8  9  9  9 10 10 10
> my_object2_rep <- rep(my_object2, each = 2) # repeats each element in my_object2 two times
> my_object2_rep
[1] "east"  "east"  "west"  "west"  "north" "north" "south" "south"
```

**Fig. 2.25** Sequence and repeating of elements in a vector

Repeat a non-sequential series e.g., using the previously defined “my_object2” object (see Sect. 2.3).

```
 my_object2_rep <- rep(my_object2, each = 2)     # repeats each element in
                                                          my_object2 two times
my_object2_rep
```

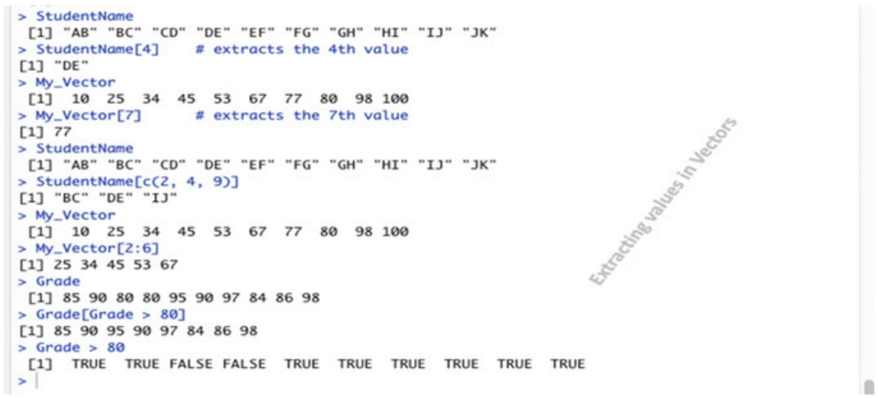
### 2.5.3 *Extracting and Replacing Elements in Vectors*

In R, the users can use the **square bracket [ ]** notation to extract elements or values from a vector, or use a combination of the **square bracket [ ]** notation along with the assignment operator <– to replace elements or values in a vector. This is known as “indexing” or “subscripting”.

#### 2.5.3.1 Extracting Elements (Values) in Vectors

To extract values at a particular position in a vector, the user simply will type the position of the specified value inside the square bracket [ ]. For example, let’s use the “**StudentName**” and “**My_Vector**” objects we created earlier in the previous examples to do this (see Sect. 2.3.2 and 2.5.1).

As shown in the commands below (see Fig. 2.26), we are going to extract the 4th value of the R object “StudentName” and the 7th value of the “My_Vector” object.



```
> StudentName
 [1] "AB" "BC" "CD" "DE" "EF" "FG" "GH" "HI" "IJ" "JK"
> StudentName[4]     # extracts the 4th value
[1] "DE"
> My_Vector
 [1]  10  25  34  45  53  67  77  80  98 100
> My_Vector[7]       # extracts the 7th value
[1] 77
> StudentName
 [1] "AB" "BC" "CD" "DE" "EF" "FG" "GH" "HI" "IJ" "JK"
> StudentName[c(2, 4, 9)]
[1] "BC" "DE" "IJ"
> My_Vector
 [1]  10  25  34  45  53  67  77  80  98 100
> My_Vector[2:6]
[1] 25 34 45 53 67
> Grade
 [1] 85 90 80 80 95 90 97 84 86 98
> Grade[Grade > 80]
[1] 85 90 95 90 97 84 86 98
> Grade > 80
 [1]  TRUE  TRUE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
> |
```

Fig. 2.26 Extracting values in vectors in R

```
StudentName[4]     # extracts the 4th value from StudentName

My_Vector[7]       # extracts the 7th value from My_Vector
```

Furthermore, the users can extract more than one value from a vector using the concatenation function **c( )**, e.g., in the code below the authors illustrates that we can extract the 2nd, 4th, and 9th values from the "**StudentName**" object (see Fig. 2.26).

```
StudentName[c(2, 4, 9)]   # extracts the 2nd, 4th, and 9th values in StudentName
```

or,

Extract a range of values using the colon (:) notation, e.g., we can extract the range of values from the 2nd to the 6th value in the "**My_Vector**" object (see Fig. 2.26).

```
My_Vector[2:6]    # extracts the 2nd to 6th values in My_Vector
```

Interestingly, the user can also use a **logical expression** to extract values from vectors. For instance, in the "**Grade**" object we defined earlier (see Fig. 2.11, Sect. 2.3.2), we can check and extract the values which are **greater (>) than 80** by using the following command (see Fig. 2.26).

```
Grade[Grade > 80] # extracts values greater than 80 from the Grade
```

Here, what R has done is to return the values that fulfill or meet the given logical condition or expression. In this case, for the ten values contained in the Grade object (i.e., 85, 90, 80, 80, 95, 90, 97, 84, 86, 98) it uses the TRUE or FALSE bolean condition to calculate this.

Type the following command to see how R does this:

```
Grade > 80
#  R will assign the TRUE or FALSE values to the numbers in Grade and returns only
the values that are TRUE

[1]   85     90    80     80      95     90     97     84     86     98
[1]  TRUE   TRUE  FALSE  FALSE   TRUE   TRUE   TRUE   TRUE   TRUE   TRUE
```

***For further practice and hands-on illustrations, the readers can also experiment with the:

- **less than <** sign
- **greater or equal to >=** sign
- **less or equal to <=** sign

### 2.5.3.2 Replacing Elements (Values) in Vectors

Now, let's look at how to replace a value(s) in a vector in R. Users can replace (or change) the values or elements contained in a vector using the **square bracket [ ]** notation along with the **assignment operator <–** in R.

For illustration (see Fig. 2.27); let's replace the 5th value in the "**My_Vector**" object (Sect. 2.5.1) from number "53" to number "100".

To do this, type the following command.

```
My_Vector                    # displays the current My_Vector object

My_Vector [5] <- 100         # replaces the 5th value with 100

My_Vector    # displays the current My_Vector after replacing the 5th value
```

Interestingly, the users can even replace more than one value or replace based on some logical expression.

For instance:

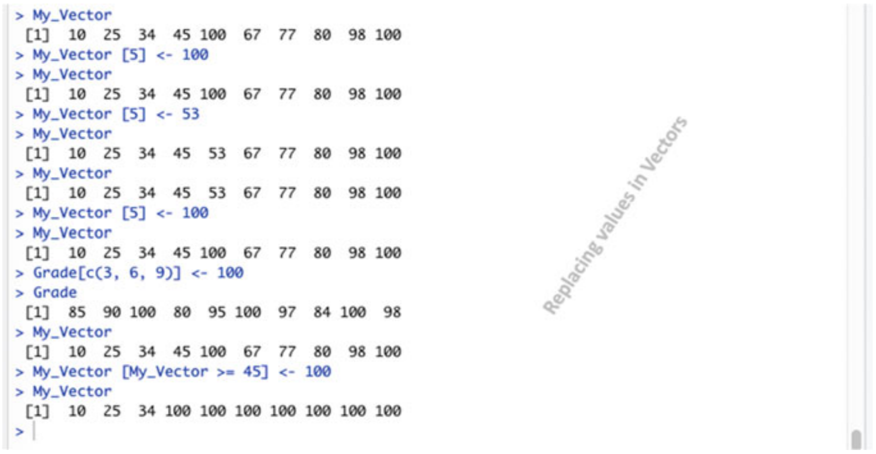
Let's replace the 3rd, 6th, and 9th values in the "**Grade**" object with 100.

```
Grade[c(3, 6, 9)] <- 100       # replaces the 3rd, 6th, and 9th values in Grade
                                  with 100
```

or,

Replace the values in "**My_Vector**" object that are greater than or equals to 45 with 100.

```
My_Vector [My_Vector >= 45] <- 100      # replaces values in My_Vector that are
                                           greater than or equal to 45 with 100
```



```
> My_Vector
 [1]  10  25  34  45 100  67  77  80  98 100
> My_Vector [5] <- 100
> My_Vector
 [1]  10  25  34  45 100  67  77  80  98 100
> My_Vector [5] <- 53
> My_Vector
 [1]  10  25  34  45  53  67  77  80  98 100
> My_Vector
 [1]  10  25  34  45  53  67  77  80  98 100
> My_Vector [5] <- 100
> My_Vector
 [1]  10  25  34  45 100  67  77  80  98 100
> Grade[c(3, 6, 9)] <- 100
> Grade
 [1]  85  90 100  80  95 100  97  84 100  98
> My_Vector
 [1]  10  25  34  45 100  67  77  80  98 100
> My_Vector [My_Vector >= 45] <- 100
> My_Vector
 [1]  10  25  34 100 100 100 100 100 100 100
> 
```

**Fig. 2.27** Replacing values in vectors in R

## 2.5.4 Vectorization in R

As we can see and practiced in the previous section (Sects. 2.5.1–2.5.3), *vectorization* is an important method in R as this means that any referenced or defined function(s) will apply to all the values (elements) contained in the vector without having to apply the function(s) individually or separately on each element in the vector.

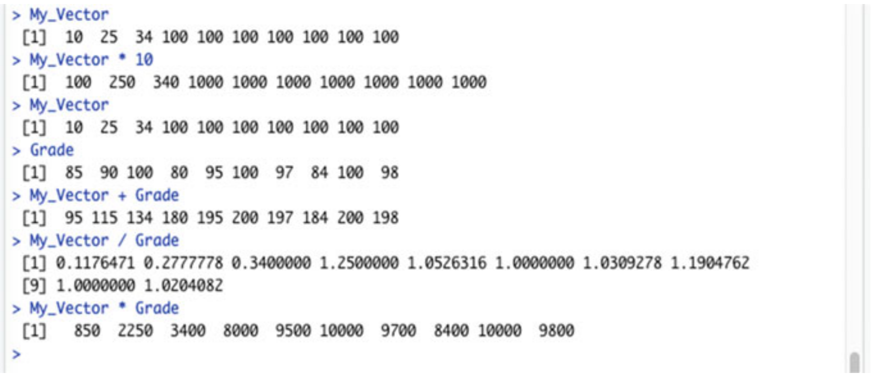
Let's take a look at the following example shown in Fig. 2.28; if we want to multiply each value in the "**My_Vector**" object by 10 (see Sect. 2.5.1), we will simply use the following command:

```
My_Vector            # displays the current My_Vector object

My_Vector * 10       # multiplies each value in My_Vector by 10
```

For further illustrations, we can **add, divide**, or **multiply** the values contained in two different (separate) vectors.

For instance, let's use the "**My_Vector**" and "**Grade**" objects (see Sects. 2.3.2 and 2.5.1) to demonstrate this. Type and execute the following command, and inspect the results as shown in Fig. 2.28.

```
My_Vector + Grade    # add the values in My_Vector with the corresponding values
                                                                        in Grade

My_Vector / Grade       # divides the values in My_Vector with the corresponding
                                                                  values in Grade

My_Vector * Grade    # multiplies the values in My_Vector with the corresponding
                                                                  values in Grade
```



```
> My_Vector
 [1]  10  25  34 100 100 100 100 100 100 100
> My_Vector * 10
 [1]  100  250  340 1000 1000 1000 1000 1000 1000 1000
> My_Vector
 [1]  10  25  34 100 100 100 100 100 100 100
> Grade
 [1]  85  90 100  80  95 100  97  84 100  98
> My_Vector + Grade
 [1]  95 115 134 180 195 200 197 184 200 198
> My_Vector / Grade
 [1] 0.1176471 0.2777778 0.3400000 1.2500000 1.0526316 1.0000000 1.0309278 1.1904762
 [9] 1.0000000 1.0204082
> My_Vector * Grade
 [1]   850  2250  3400  8000  9500 10000  9700  8400 10000  9800
>
```

**Fig. 2.28** Vectorization in R

### 2.5.5 Factorization in R

*Factors* in R can be defined as a set(s) of data structure used for elements (or values) that take pre-defined states or a finite number of values, e.g., categorical, nominal, or continuous datasets.

For example, let's look at the "**Gender**" data field or variable in the earlier example **Students.ExampleData** dataset (Sect. 2.3.2, Fig. 2.11).

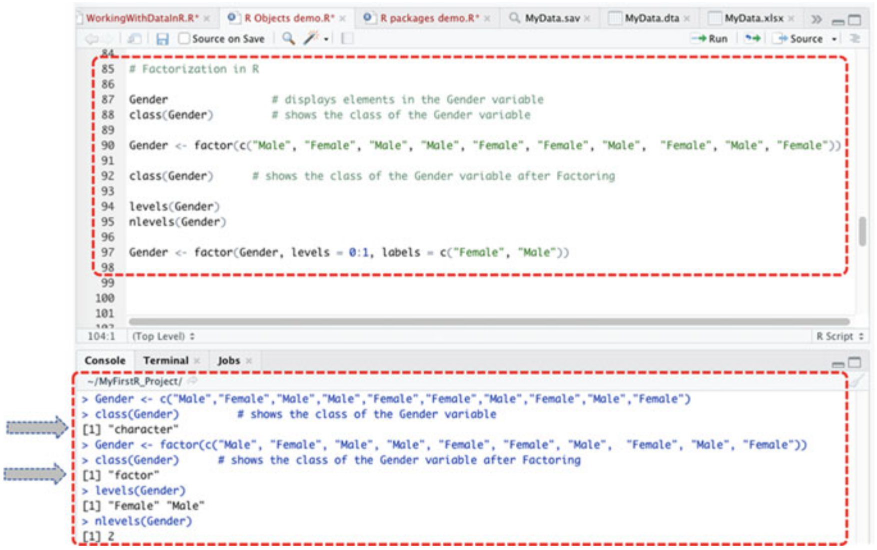
```
> Gender
 [1] "Male" "Female" "Male" "Male" "Female" "Female" "Male" "Female" "Male" "Female"
```

The key point to note here is; while the values of the "Gender" variable are defined as Characters (categorical variable), i.e., "Male" and "Female". However in many cases, these values are required to be stored as integers for R to be able to handle or compute them when conducting the different statistical data analysis or data visualizations, by associating them with unique integer values called "Factors".

The **factor( )** command is used to create and modify factors in R.

Now, let's look at an example of how to do this. As shown in Fig. 2.29, particularly Line 90, type the following command to factor the "Gender" variable.

```
class(Gender)          # shows the class of the Gender variable

Gender <- factor(c("Male", "Female", "Male", "Male", "Female", "Female",
"Male", "Female", "Male", "Female"))

class(Gender)        # displays the class of the Gender variable after
Factoring
```



```
84
85  # Factorization in R
86
87  Gender                  # displays elements in the Gender variable
88  class(Gender)           # shows the class of the Gender variable
89
90  Gender <- factor(c("Male", "Female", "Male", "Male", "Female", "Female", "Male",  "Female", "Male", "Female"))
91
92  class(Gender)      # shows the class of the Gender variable after Factoring
93
94  levels(Gender)
95  nlevels(Gender)
96
97  Gender <- factor(Gender, levels = 0:1, labels = c("Female", "Male"))
98
99
100
101
```

```
~/MyFirstR_Project/
> Gender <- c("Male","Female","Male","Male","Female","Female","Male","Female","Male","Female")
> class(Gender)        # shows the class of the Gender variable
[1] "character"
> Gender <- factor(c("Male", "Female", "Male", "Male", "Female", "Female", "Male",  "Female", "Male", "Female"))
> class(Gender)      # shows the class of the Gender variable after Factoring
[1] "factor"
> levels(Gender)
[1] "Female" "Male"
> nlevels(Gender)
[1] 2
```

**Fig. 2.29** Factoring in R (factorization)

When the above command is executed, **by default**, R will assign a number value of "1" to the level "Female" and "2" to the level "Male". This is because the letter "F" comes before "M", despite the fact that in the data or variable (Gender) the first value in the list is male. We can check these details using the **levels ( )** function, or check the number of levels using the function **nlevels()** as shown below (see Fig. 2.29, Lines 94 and 95).

```
levels(Gender)

nlevels(Gender)
```

Now, let's try another method in factoring; the users can decide to set their own levels, rather than using the default assignment by R. To do this, type the following command:

```
Gender <- factor(Gender, levels = 0:1, labels = c("Female", "Male"))
```

In the above command, what the authors have done is to assign the level "0" to "Female" and "1" to "Male".

**Note:** Even though, R after factoring recognizes the "Gender" variable as a factor by assigning the levels 0 and 1 to the "Female" and "Male" values; the values still exist as String, except we decide to convert them into another form (e.g., numeric) which is completely another different task from the way R recognizes the values, as we will illustrate in the next section.

#### 2.5.5.1 Converting Factors into Other Data Types

The users can convert a Numeric (num) or Character (chr) value to Factor, Factors to Numeric or Character values, vice and versa. The functions used to do this are as follows: **as.numeric( )**, **as.factor( ), and as.character( ),** respectively.

To illustrate this, let's take a look at the **Students.ExampleData** dataset (see Fig. 2.30). Type the following command to display the data:

```
str(Students.ExampleData) # display class or R object before converting
```

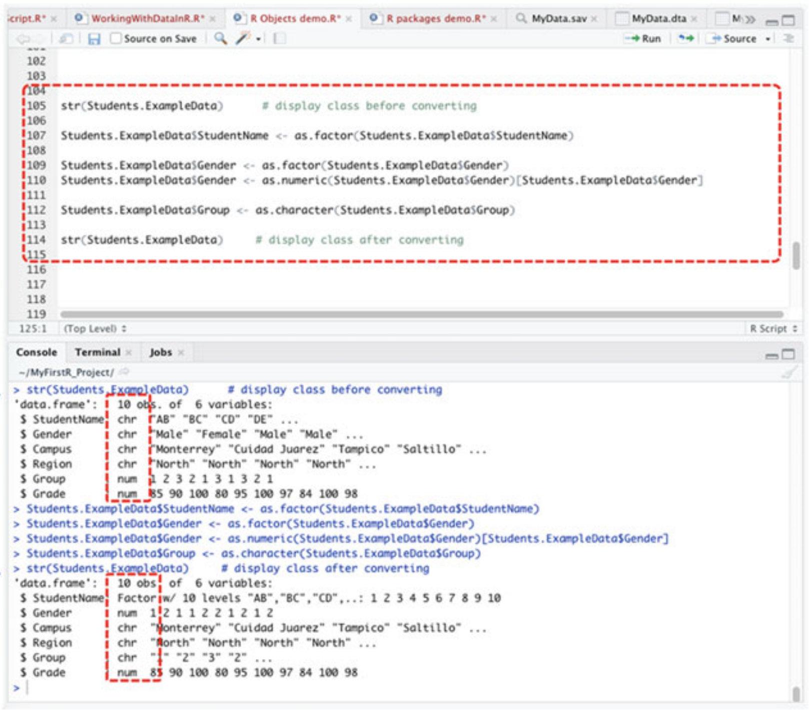
Now, we will convert the following variables;

"StudentName" (**chr**) variable - to **Factor**

"Gender" (**chr**) variable - to **Factor**, and then - to Numeric (**num**)

"Group" (num) variable - to Character (**chr**)

Type the following command, as shown in Fig. 2.30.



```
str(Students.ExampleData)      # display class before converting

Students.ExampleData$StudentName <- as.factor(Students.ExampleData$StudentName)

Students.ExampleData$Gender <- as.factor(Students.ExampleData$Gender)
Students.ExampleData$Gender <- as.numeric(Students.ExampleData$Gender)[Students.ExampleData$Gender]

Students.ExampleData$Group <- as.character(Students.ExampleData$Group)

str(Students.ExampleData)     # display class after converting
```

```
~/MyFirstR_Project/
> str(Students.ExampleData)      # display class before converting
'data.frame':   10 obs. of  6 variables:
 $ StudentName: chr  "AB" "BC" "CD" "DE" ...
 $ Gender     : chr  "Male" "Female" "Male" "Male" ...
 $ Campus     : chr  "Monterrey" "Cuidad Juarez" "Tampico" "Saltillo" ...
 $ Region     : chr  "North" "North" "North" "North" ...
 $ Group      : num  1 2 3 2 1 3 1 3 2 1
 $ Grade      : num  85 90 100 80 95 100 97 84 100 98
> Students.ExampleData$StudentName <- as.factor(Students.ExampleData$StudentName)
> Students.ExampleData$Gender <- as.factor(Students.ExampleData$Gender)
> Students.ExampleData$Gender <- as.numeric(Students.ExampleData$Gender)[Students.ExampleData$Gender]
> Students.ExampleData$Group <- as.character(Students.ExampleData$Group)
> str(Students.ExampleData)     # display class after converting
'data.frame':   10 obs.  of  6 variables:
 $ StudentName: Factor w/ 10 levels "AB","BC","CD",..: 1 2 3 4 5 6 7 8 9 10
 $ Gender     : num  1 2 1 1 2 2 1 2 1 2
 $ Campus     : chr  "Monterrey" "Cuidad Juarez" "Tampico" "Saltillo" ...
 $ Region     : chr  "North" "North" "North" "North" ...
 $ Group      : chr  "1" "2" "3" "2" ...
 $ Grade      : num  85 90 100 80 95 100 97 84 100 98
>
```

**Fig. 2.30** Converting factors to other data types

```
Students.ExampleData$StudentName <- as.factor(Students.ExampleData$StudentName)

Students.ExampleData$Gender <- as.factor(Students.ExampleData$Gender)
Students.ExampleData$Gender <-
as.numeric(Students.ExampleData$Gender)[Students.ExampleData$Gender]

Students.ExampleData$Group <- as.character(Students.ExampleData$Group)

str(Students.ExampleData)      # display class after converting
```

***Users can view the new updated dataset or information via the **Environment Tab** (upper right window) in R. Feel free to experiment and practice with the different datasets we have covered and currently have stored there as R objects.

## 2.6 R Packages and Libraries

By definition, "*R packages*" are a collection of functions, datasets, and other help and support files that are codified into a well-defined structure, so that the users can be able to download and install them in R for different data analysis, modeling, or visualization purposes (Douglas et al., 2020). When you first install R on your computer; by default, it comes with many standard packages that you will normally use to perform the different tasks or data analysis in R. Although, the more you become familiar and use R to perform more customized projects or applications, you will notice at some point that you may require or need to extend the capabilities of the R functions. This is where the users need to install some extra packages that are not already in R by default.

The official repository to download R packages is through the CRAN repository (https://cran.r-project.org/web/packages/) which can also be accessed via the RStudio environment. Although other sources exist such as GitHub: https://github.com/ (a website that hosts git repositories for all sorts of software and projects including R), and Bioconductor: https://www.bioconductor.org/ (that hosts a number of R packages oriented toward bioinformatics).

There are two ways the users can install R **packages** from CRAN,

1. Either, by installing the package(s) you want to install directly in R through the **Console** using the **install.packages( )** function.
2. Or, by directly visiting the CRAN website (https://cran.r-project.org/web/packages/), browse and download the specific package(s) you want to install, and then upload the downloaded file to RStudio or environment.

Either way, we will demonstrate how the users can do this in R.

For the first method, the users can make use of the **install.packages( )** function to install the packages directly through the Console. For instance, if you want to install the packages named "**abind**" and "**cluster**", you will need to type the following command as shown in Fig. 2.31 (make sure you have an internet connection to your system as the files will be downloaded and installed from the web):

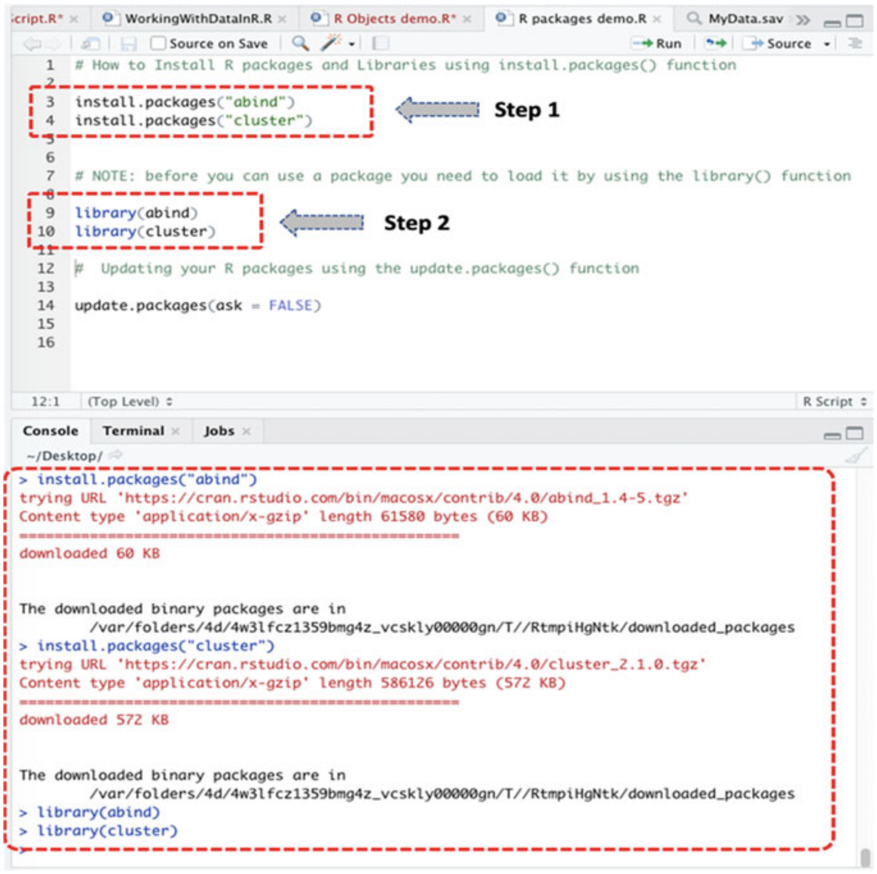
```
install.packages("abind")

install.packages("cluster")
```

Additionally, but not necessarily in all cases, the users can confirm that any other additional packages (dependencies) that may be needed in order to install the selected package are also installed by using, for instance, the following code:

```
install.packages("abind", dependencies = TRUE)
```

Furthermore, once the user has installed any of the R package(s) on the computer system; by default the installed package(s) are not available to use immediately. Therefore, as shown in the 2nd step in Fig. 2.31, in order to use an installed package, the user will also need to run or load the package by using the **library( )** function. For example, to use the installed packages "**abind**" and "**cluster**", the user will need to run the following command:

```
library(abind)

library(cluster)
```



```
# How to Install R packages and Libraries using install.packages() function

install.packages("abind")
install.packages("cluster")


# NOTE: before you can use a package you need to load it by using the library() function

library(abind)
library(cluster)

#  Updating your R packages using the update.packages() function

update.packages(ask = FALSE)
```

```
~/Desktop/
> install.packages("abind")
trying URL 'https://cran.rstudio.com/bin/macosx/contrib/4.0/abind_1.4-5.tgz'
Content type 'application/x-gzip' length 61580 bytes (60 KB)
==================================================
downloaded 60 KB


The downloaded binary packages are in
	/var/folders/4d/4w3lfcz1359bmg4z_vcskly00000gn/T//RtmpiHgNtk/downloaded_packages
> install.packages("cluster")
trying URL 'https://cran.rstudio.com/bin/macosx/contrib/4.0/cluster_2.1.0.tgz'
Content type 'application/x-gzip' length 586126 bytes (572 KB)
==================================================
downloaded 572 KB


The downloaded binary packages are in
	/var/folders/4d/4w3lfcz1359bmg4z_vcskly00000gn/T//RtmpiHgNtk/downloaded_packages
> library(abind)
> library(cluster)
>
```

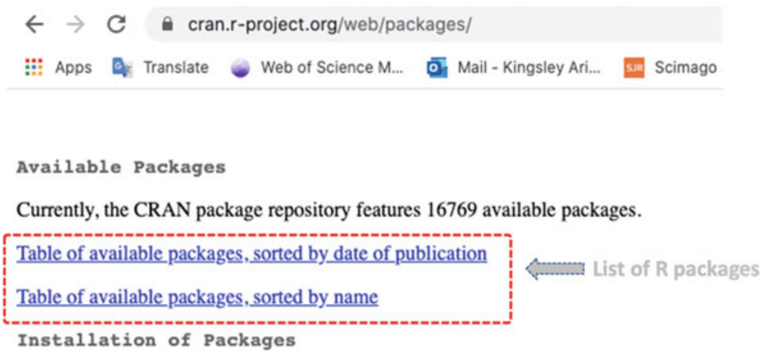
**Fig. 2.31** Installing R packages through the console

Now, let’s look at the second option on how to directly install R package(s) on your system by visiting the CRAN website (https://cran.r-project.org/web/packages/).

There are two ways the users can do this. For the first method as shown in Fig. 2.32, you can visit the CRAN website via the browser, then browse and download the specific package(s) you want to install, and then upload the downloaded file to RStudio.

The second method, as shown in Fig. 2.33, is to search for the R package(s) from the **CRAN** website via the **Files/Packages Tab** in RStudio.

As demonstrated in Fig. 2.33, when you click on the “**Install**” tab or button (see: Step 1), you will be prompted with a pop-up window where you can search for the specific package(s) you want to install on your system. Search and find the package(s) you want to install (see example in Fig. 2.34), then click on the “**Install**” button to install it. Also, remember to check the “**Install dependencies**” box (see Fig. 2.33) in



**Available Packages**

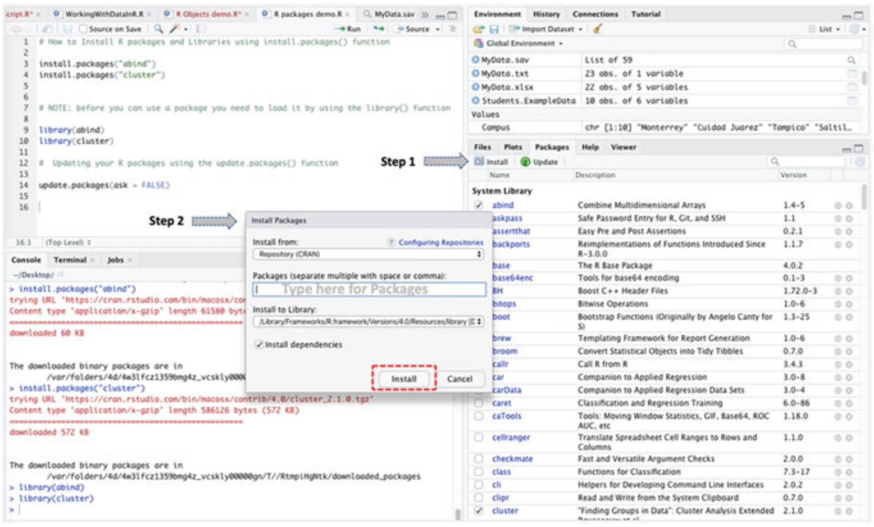
Currently, the CRAN package repository features 16769 available packages.

Table of available packages, sorted by date of publication

Table of available packages, sorted by name

**Installation of Packages**

Fig. 2.32 The CRAN website to download R packages



```
1   # How to Install R packages and Libraries using install.packages() function
2
3   install.packages("abind")
4   install.packages("cluster")
5
6
7   # NOTE: before you can use a package you need to load it by using the library() function
8
9   library(abind)
10  library(cluster)
11
12  #  Updating your R packages using the update.packages() function
13
14  update.packages(ask = FALSE)
15
16
```

```
> install.packages("abind")
trying URL 'https://cran.rstudio.com/bin/macosx/co
Content type 'application/x-gzip' length 61580 byte
==================================================
downloaded 60 KB


The downloaded binary packages are in
        /var/folders/4d/4w31fcz1359bmg4z_vcskly000
> install.packages("cluster")
trying URL 'https://cran.rstudio.com/bin/macosx/contrib/4.0/cluster_2.1.0.tgz'
Content type 'application/x-gzip' length 586126 bytes (572 KB)
==================================================
downloaded 572 KB


The downloaded binary packages are in
        /var/folders/4d/4w31fcz1359bmg4z_vcskly00000gn/T//RtmpiHgNtk/downloaded_packages
> library(abind)
> library(cluster)
>
```

Install from: Repository (CRAN)

Packages (separate multiple with space or comma): Type here for Packages

Install to Library: /Library/Frameworks/R.framework/Versions/4.0/Resources/library

[✓] Install dependencies

| | |
|---|---|
| MyData.sav | List of 59 |
| MyData.txt | 23 obs. of 1 variable |
| MyData.xlsx | 22 obs. of 5 variables |
| Students.ExampleData | 10 obs. of 6 variables |
| Values | |
| Campus | chr [1:10] "Monterrey" "Cuidad Juarez" "Tampico" "Saltil… |

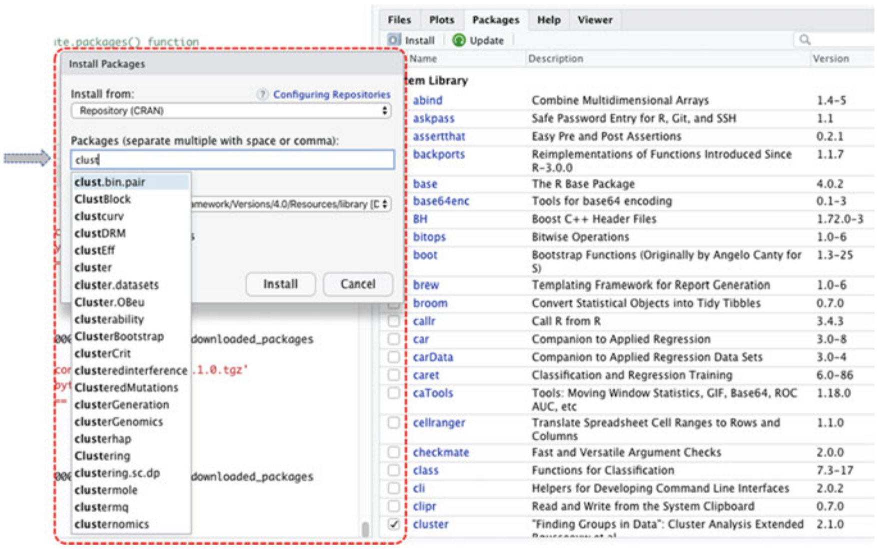
| | Name | Description | Version |
|---|---|---|---|
| | **System Library** | | |
| ✓ | abind | Combine Multidimensional Arrays | 1.4-5 |
| | askpass | Safe Password Entry for R, Git, and SSH | 1.1 |
| | assertthat | Easy Pre and Post Assertions | 0.2.1 |
| | backports | Reimplementations of Functions Introduced Since R-3.0.0 | 1.1.7 |
| | base | The R Base Package | 4.0.2 |
| | base64enc | Tools for base64 encoding | 0.1-3 |
| | BH | Boost C++ Header Files | 1.72.0-3 |
| | bitops | Bitwise Operations | 1.0-6 |
| | boot | Bootstrap Functions (Originally by Angelo Canty for S) | 1.3-25 |
| | brew | Templating Framework for Report Generation | 1.0-6 |
| | broom | Convert Statistical Objects into Tidy Tibbles | 0.7.0 |
| | callr | Call R from R | 3.4.3 |
| | car | Companion to Applied Regression | 3.0-8 |
| | carData | Companion to Applied Regression Data Sets | 3.0-4 |
| | caret | Classification and Regression Training | 6.0-86 |
| | caTools | Tools: Moving Window Statistics, GIF, Base64, ROC AUC, etc | 1.18.0 |
| | cellranger | Translate Spreadsheet Cell Ranges to Rows and Columns | 1.1.0 |
| | checkmate | Fast and Versatile Argument Checks | 2.0.0 |
| | class | Functions for Classification | 7.3-17 |
| | cli | Helpers for Developing Command Line Interfaces | 2.0.2 |
| | clipr | Read and Write from the System Clipboard | 0.7.0 |
| ✓ | cluster | "Finding Groups in Data": Cluster Analysis Extended | 2.1.0 |

Fig. 2.33 Installing R packages through the file/packages tab

order to confirm any other additional packages (dependencies) that may be required in order to fully install the selected R package.

Lastly, just like every other program or software, occasionally the users may be prompted or will want to update previously installed package(s) on the system in order to continue to get access to new functionalities or fixing of bugs. Users can use the **update.packages( )** function to update (CRAN) packages already installed or are existing on the system library (see Line 14, Fig. 2.33). Alternatively, you can click the "**Update**" button on the File/Packages Tab (see Fig. 2.34) to perform this task.

```
update.packages(ask = FALSE)
```



```
clust.bin.pair
ClustBlock
clustcurv
clustDRM
clustEff
cluster
cluster.datasets
Cluster.OBeu
clusterability
ClusterBootstrap
clusterCrit
clusteredinterference
ClusteredMutations
clusterGeneration
clusterGenomics
clusterhap
Clustering
clustering.sc.dp
clustermole
clustermq
clusternomics
```

| Name | Description | Version |
|---|---|---|
| abind | Combine Multidimensional Arrays | 1.4-5 |
| askpass | Safe Password Entry for R, Git, and SSH | 1.1 |
| assertthat | Easy Pre and Post Assertions | 0.2.1 |
| backports | Reimplementations of Functions Introduced Since R-3.0.0 | 1.1.7 |
| base | The R Base Package | 4.0.2 |
| base64enc | Tools for base64 encoding | 0.1-3 |
| BH | Boost C++ Header Files | 1.72.0-3 |
| bitops | Bitwise Operations | 1.0-6 |
| boot | Bootstrap Functions (Originally by Angelo Canty for S) | 1.3-25 |
| brew | Templating Framework for Report Generation | 1.0-6 |
| broom | Convert Statistical Objects into Tidy Tibbles | 0.7.0 |
| callr | Call R from R | 3.4.3 |
| car | Companion to Applied Regression | 3.0-8 |
| carData | Companion to Applied Regression Data Sets | 3.0-4 |
| caret | Classification and Regression Training | 6.0-86 |
| caTools | Tools: Moving Window Statistics, GIF, Base64, ROC AUC, etc | 1.18.0 |
| cellranger | Translate Spreadsheet Cell Ranges to Rows and Columns | 1.1.0 |
| checkmate | Fast and Versatile Argument Checks | 2.0.0 |
| class | Functions for Classification | 7.3-17 |
| cli | Helpers for Developing Command Line Interfaces | 2.0.2 |
| clipr | Read and Write from the System Clipboard | 0.7.0 |
| cluster | "Finding Groups in Data": Cluster Analysis Extended | 2.1.0 |

**Fig. 2.34** Searching for R packages in RStudio

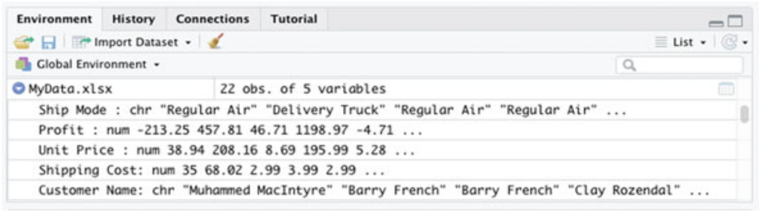
***Users can include the "ask = FALSE" argument to avoid being prompted to confirm each time, every package(s) that the system will be updating***.

## 2.7 Plots and Data Visualization

Apart from analyzing, modeling, and exploring datasets in R; it is also a great tool for creating almost any type of visualization or graph. Here, the authors take a look at some simple ways to visualize graphs or plot data in R.

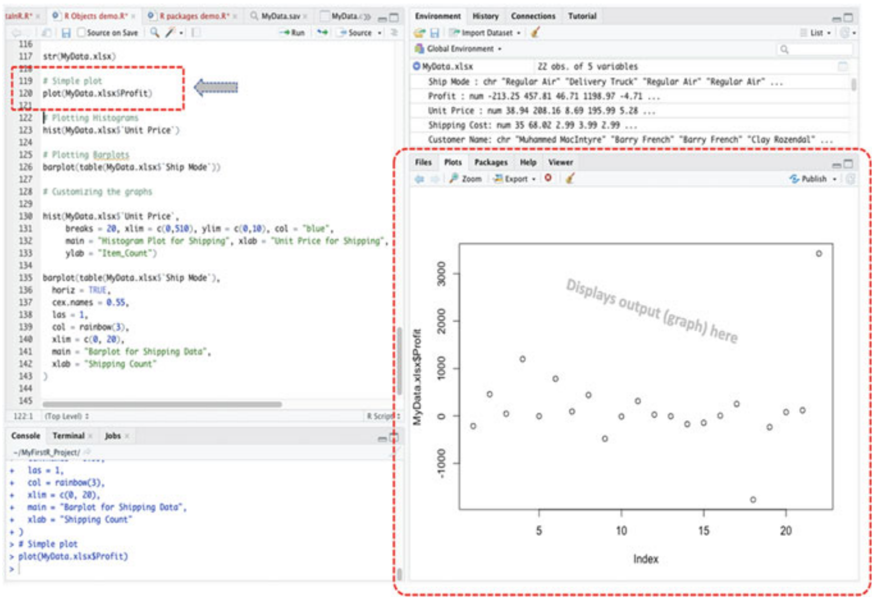
For instance, let's explore the dataset we imported earlier in R called "**MyData.xlsx**" (see Sect. 2.3.3.2 and Fig. 2.35).

```
str(MyData.xlsx)       # view data properties
```



```
MyData.xlsx          22 obs. of 5 variables
  Ship Mode : chr "Regular Air" "Delivery Truck" "Regular Air" "Regular Air" ...
  Profit : num -213.25 457.81 46.71 1198.97 -4.71 ...
  Unit Price : num 38.94 208.16 8.69 195.99 5.28 ...
  Shipping Cost: num 35 68.02 2.99 3.99 2.99 ...
  Customer Name: chr "Muhammed MacIntyre" "Barry French" "Barry French" "Clay Rozendal" ...
```

**Fig. 2.35** Example dataset (MyData.xlsx) for plotting



```
str(MyData.xlsx)

# Simple plot
plot(MyData.xlsx$Profit)

# Plotting Histograms
hist(MyData.xlsx$`Unit Price`)

# Plotting Barplots
barplot(table(MyData.xlsx$`Ship Mode`))

# Customizing the graphs

hist(MyData.xlsx$`Unit Price`,
     breaks = 20, xlim = c(0,510), ylim = c(0,10), col = "blue",
     main = "Histogram Plot for Shipping", xlab = "Unit Price for Shipping",
     ylab = "Item_Count")

barplot(table(MyData.xlsx$`Ship Mode`),
  horiz = TRUE,
  cex.names = 0.55,
  las = 1,
  col = rainbow(3),
  xlim = c(0, 20),
  main = "Barplot for Shipping Data",
  xlab = "Shipping Count"
)
```

```
~/MyFirstR_Project/
+   las = 1,
+   col = rainbow(3),
+   xlim = c(0, 20),
+   main = "Barplot for Shipping Data",
+   xlab = "Shipping Count"
+ )
> # Simple plot
> plot(MyData.xlsx$Profit)
>
```

```
MyData.xlsx          22 obs. of 5 variables
  Ship Mode : chr "Regular Air" "Delivery Truck" "Regular Air" "Regular Air" ...
  Profit : num -213.25 457.81 46.71 1198.97 -4.71 ...
  Unit Price : num 38.94 208.16 8.69 195.99 5.28 ...
  Shipping Cost: num 35 68.02 2.99 3.99 2.99 ...
  Customer Name: chr "Muhammed MacIntyre" "Barry French" "Barry French" "Clay Rozendal" ...
```

Fig. 2.36 Simple plot in R

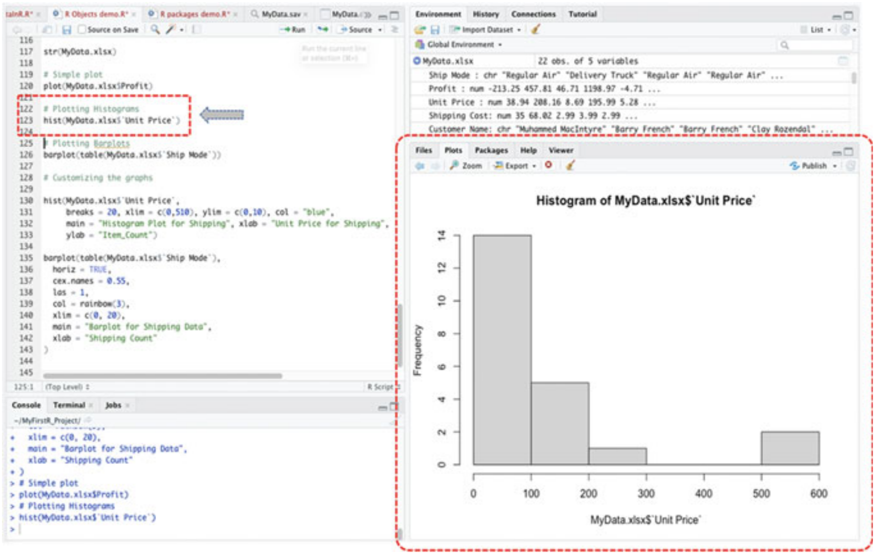
Let's plot some of the variables in this data using different R functions that can be used to visualize the dataset. This includes; **plot( ), hist( ), and barplot( ),** etc.

Now, type and run the following codes in the source code editor and view the results as shown in Figs. 2.36, 2.37, and 2.38, respectively.

```
# Simple plot
plot(MyData.xlsx$Profit)

# Plotting Histograms
hist(MyData.xlsx$`Unit Price`)

# Plotting Barplots
barplot(table(MyData.xlsx$`Ship Mode`))
```

**Other Useful Tips**:

1. When you use the **plot( )** function, R usually uses its own default style to plot the graph, which a lot of time may be plotted as a Scatter plot.
2. For **hist( )** function, before you can plot other types of variable, e.g., Character (chr), you will need to convert the vector or factor to numeric (num) value as we have covered earlier in the previous section (see: Sect. 2.5.5) of this chapter.
3. For **barplot( )** function, if you want to plot a single variable, e.g., the "Shipping Mode" variable in the "**MyData.xlsx**" dataset, you will need to create a table to hold the values in the *vector* using the **table( )** function.

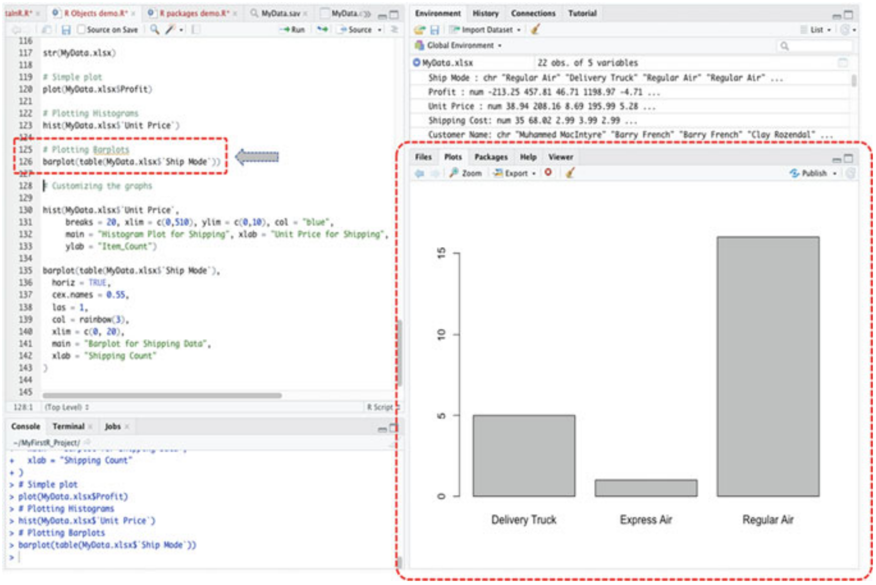


```
str(MyData.xlsx)

# Simple plot
plot(MyData.xlsx$Profit)

# Plotting Histograms
hist(MyData.xlsx$`Unit Price`)

# Plotting Barplots
barplot(table(MyData.xlsx$`Ship Mode`))

# Customizing the graphs

hist(MyData.xlsx$`Unit Price`,
     breaks = 20, xlim = c(0,510), ylim = c(0,10), col = "blue",
     main = "Histogram Plot for Shipping", xlab = "Unit Price for Shipping",
     ylab = "Item_Count")

barplot(table(MyData.xlsx$`Ship Mode`),
  horiz = TRUE,
  cex.names = 0.55,
  las = 1,
  col = rainbow(3),
  xlim = c(0, 20),
  main = "Barplot for Shipping Data",
  xlab = "Shipping Count"
)
```

```
~/MyFirstR_Project/
+   xlim = c(0, 20),
+   main = "Barplot for Shipping Data",
+   xlab = "Shipping Count"
+ )
> # Simple plot
> plot(MyData.xlsx$Profit)
> # Plotting Histograms
> hist(MyData.xlsx$`Unit Price`)
>
```

```
MyData.xlsx        22 obs. of 5 variables
  Ship Mode : chr "Regular Air" "Delivery Truck" "Regular Air" "Regular Air" ...
  Profit : num -213.25 457.81 46.71 1198.97 -4.71 ...
  Unit Price : num 38.94 208.16 8.69 195.99 5.28 ...
  Shipping Cost: num 35 68.02 2.99 3.99 2.99 ...
  Customer Name: chr "Muhammed MacIntyre" "Barry French" "Barry French" "Clay Rozendal" ...
```

**Fig. 2.37** Plotting histograms in R



```
str(MyData.xlsx)

# Simple plot
plot(MyData.xlsx$Profit)

# Plotting Histograms
hist(MyData.xlsx$`Unit Price`)

# Plotting Barplots
barplot(table(MyData.xlsx$`Ship Mode`))

# Customizing the graphs

hist(MyData.xlsx$`Unit Price`,
     breaks = 20, xlim = c(0,510), ylim = c(0,10), col = "blue",
     main = "Histogram Plot for Shipping", xlab = "Unit Price for Shipping",
     ylab = "Item_Count")

barplot(table(MyData.xlsx$`Ship Mode`),
  horiz = TRUE,
  cex.names = 0.55,
  las = 1,
  col = rainbow(3),
  xlim = c(0, 20),
  main = "Barplot for Shipping Data",
  xlab = "Shipping Count"
)
```

```
~/MyFirstR_Project/
+   xlab = "Shipping Count"
+ )
> # Simple plot
> plot(MyData.xlsx$Profit)
> # Plotting Histograms
> hist(MyData.xlsx$`Unit Price`)
> # Plotting Barplots
> barplot(table(MyData.xlsx$`Ship Mode`))
>
```

```
MyData.xlsx        22 obs. of 5 variables
  Ship Mode : chr "Regular Air" "Delivery Truck" "Regular Air" "Regular Air" ...
  Profit : num -213.25 457.81 46.71 1198.97 -4.71 ...
  Unit Price : num 38.94 208.16 8.69 195.99 5.28 ...
  Shipping Cost: num 35 68.02 2.99 3.99 2.99 ...
  Customer Name: chr "Muhammed MacIntyre" "Barry French" "Barry French" "Clay Rozendal" ...
```

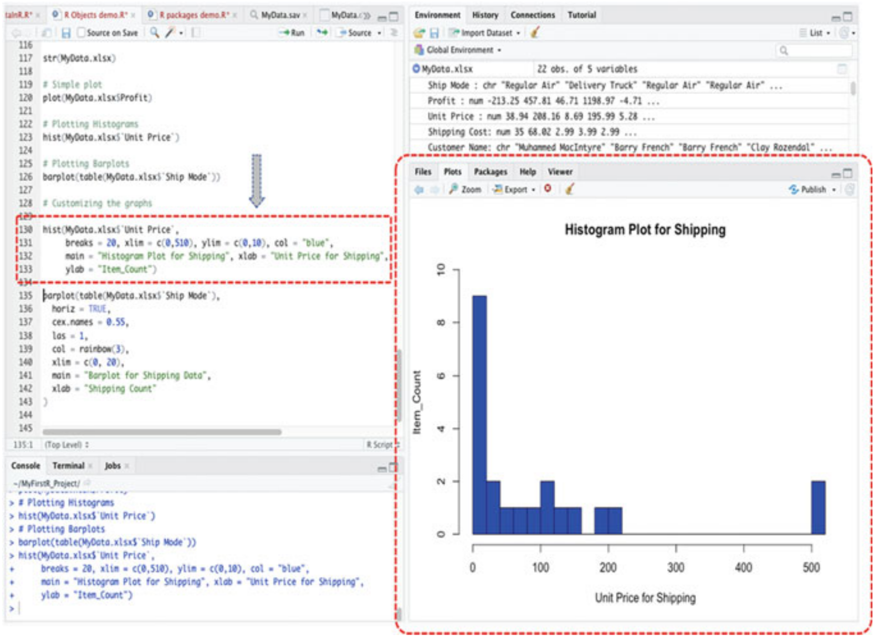
**Fig. 2.38** Plotting BarPlots in R

4. The “$” sign, for instance, in plot(MyData.xlsx$Profit)is used to tell the program which variable (value) to select from the specified data.

Furthermore, one good thing about R is that it allows the users to define (customize) how they want to display the graphs or plots.

For instance, as illustrated in the following figures (Figs. 2.39 and 2.40), the authors have used the following code to customize the histogram and barplot (see Figs. 2.37 and 2.38) to produce the information we want as more clearly and specific as defined by ourselves.

```
# Customizing graphs in R

hist(MyData.xlsx$`Unit Price`,
     breaks = 20, xlim = c(0,510), ylim = c(0,10), col = "blue",
     main = "Histogram Plot for Shipping", xlab = "Unit Price for Shipping",
     ylab = "Item_Count")


barplot(table(MyData.xlsx$`Ship Mode`),
  horiz = TRUE,
  cex.names = 0.55,
  las = 1,
  col = rainbow(3),
  xlim = c(0, 20),
  main = "Barplot for Shipping Data",
  xlab = "Shipping Count"
)
```

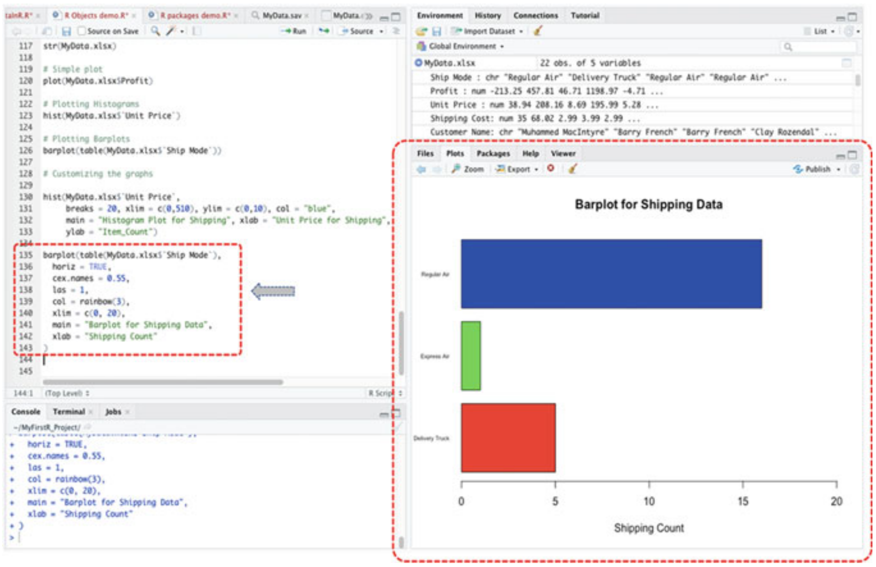


```

str(MyData.xlsx)

# Simple plot
plot(MyData.xlsx$Profit)

# Plotting Histograms
hist(MyData.xlsx$`Unit Price`)

# Plotting Barplots
barplot(table(MyData.xlsx$`Ship Mode`))

# Customizing the graphs

hist(MyData.xlsx$`Unit Price`,
     breaks = 20, xlim = c(0,510), ylim = c(0,10), col = "blue",
     main = "Histogram Plot for Shipping", xlab = "Unit Price for Shipping",
     ylab = "Item_Count")

barplot(table(MyData.xlsx$`Ship Mode`),
  horiz = TRUE,
  cex.names = 0.55,
  las = 1,
  col = rainbow(3),
  xlim = c(0, 20),
  main = "Barplot for Shipping Data",
  xlab = "Shipping Count"
)


```

```
> # Plotting Histograms
> hist(MyData.xlsx$`Unit Price`)
> # Plotting Barplots
> barplot(table(MyData.xlsx$`Ship Mode`))
> hist(MyData.xlsx$`Unit Price`,
+      breaks = 20, xlim = c(0,510), ylim = c(0,10), col = "blue",
+      main = "Histogram Plot for Shipping", xlab = "Unit Price for Shipping",
+      ylab = "Item_Count")
> |
```

```
MyData.xlsx        22 obs. of 5 variables
  Ship Mode : chr "Regular Air" "Delivery Truck" "Regular Air" "Regular Air" ...
  Profit : num -213.25 457.81 46.71 1198.97 -4.71 ...
  Unit Price : num 38.94 208.16 8.69 195.99 5.28 ...
  Shipping Cost: num 35 68.02 2.99 3.99 2.99 ...
  Customer Name: chr "Muhammed MacIntyre" "Barry French" "Barry French" "Clay Rozendal" ...
```

**Fig. 2.39** Customizing histogram in R



```
str(MyData.xlsx)

# Simple plot
plot(MyData.xlsx$Profit)

# Plotting Histograms
hist(MyData.xlsx$`Unit Price`)

# Plotting Barplots
barplot(table(MyData.xlsx$`Ship Mode`))

# Customizing the graphs

hist(MyData.xlsx$`Unit Price`,
     breaks = 20, xlim = c(0,510), ylim = c(0,10), col = "blue",
     main = "Histogram Plot for Shipping", xlab = "Unit Price for Shipping",
     ylab = "Item_Count")

barplot(table(MyData.xlsx$`Ship Mode`),
  horiz = TRUE,
  cex.names = 0.55,
  las = 1,
  col = rainbow(3),
  xlim = c(0, 20),
  main = "Barplot for Shipping Data",
  xlab = "Shipping Count"
)
```

```
+   horiz = TRUE,
+   cex.names = 0.55,
+   las = 1,
+   col = rainbow(3),
+   xlim = c(0, 20),
+   main = "Barplot for Shipping Data",
+   xlab = "Shipping Count"
+ )
>
```

```
MyData.xlsx          22 obs. of 5 variables
  Ship Mode : chr "Regular Air" "Delivery Truck" "Regular Air" "Regular Air" ...
  Profit : num -213.25 457.81 46.71 1198.97 -4.71 ...
  Unit Price : num 38.94 208.16 8.69 195.99 5.28 ...
  Shipping Cost: num 35 68.02 2.99 3.99 2.99 ...
  Customer Name: chr "Muhammed MacIntyre" "Barry French" "Barry French" "Clay Rozendal" ...
```

**Fig. 2.40** Customizing barplot in R

Lastly, we will look at how to plot and compare relationships between variables (i.e., one variable vs another variable), particularly common among the researchers or data analysts when conducting different statistical analyses for research purposes or technical reports, for example.

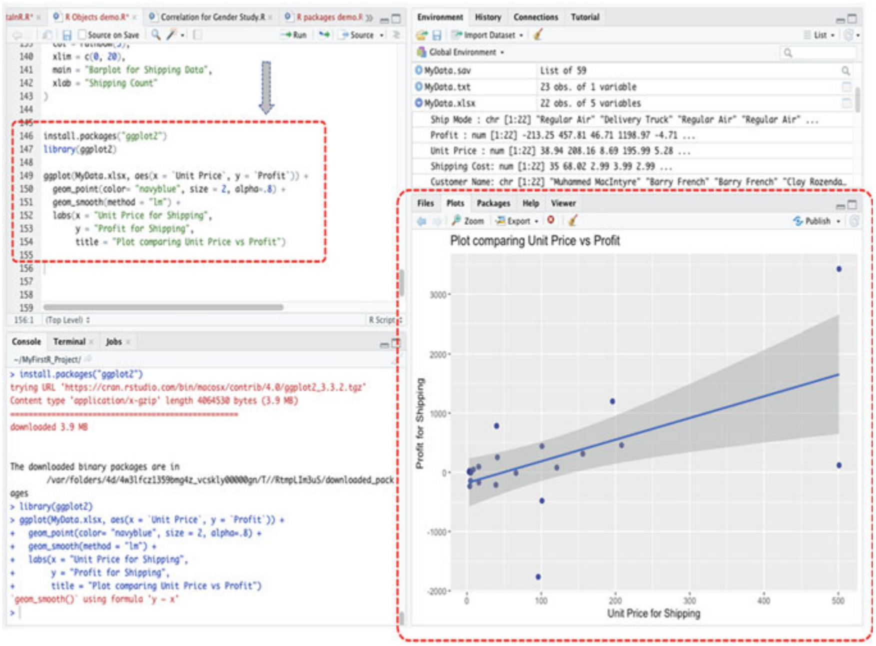
Let’s illustrate this using a package called “**ggplot2**” which is the latest upgraded version of the **ggplot** package and library at the time of writing this book, there may be other newer version in the subsequent future. The **ggplot2** is an open-source data visualization package for the R statistical programming dedicated to the visualization of data. It impressively improves the aesthetics and quality of the graphs by declaratively creating graphics, based on the Grammar of Graphics (GG).

As an example, we will plot the “**Unit Price**” versus “**Profit**” variables in the “**MyData.xlsx**” data (see Sect. 2.3.3.2) using the **ggplot2** package and library.

First, download and install the “**ggplot2**” package (as explained earlier in the steps described in Sect. 2.6) (see Fig. 2.41).

```
install.packages("ggplot2")
library(ggplot2)
```

Once you have installed and run the library for the ggplot2, type and run the following command as shown in Fig. 2.41 (Lines 149–154).



```
  xlim = c(0, 20),
  main = "Barplot for Shipping Data",
  xlab = "Shipping Count"
)


install.packages("ggplot2")
library(ggplot2)

ggplot(MyData.xlsx, aes(x = `Unit Price`, y = `Profit`)) +
  geom_point(color= "navyblue", size = 2, alpha=.8) +
  geom_smooth(method = "lm") +
  labs(x = "Unit Price for Shipping",
       y = "Profit for Shipping",
       title = "Plot comparing Unit Price vs Profit")
```

```
MyData.sav     List of 59
MyData.txt     23 obs. of 1 variable
MyData.xlsx    22 obs. of 5 variables
  Ship Mode : chr [1:22] "Regular Air" "Delivery Truck" "Regular Air" "Regular Air" ...
  Profit : num [1:22] -213.25 457.81 46.71 1198.97 -4.71 ...
  Unit Price : num [1:22] 38.94 208.16 8.69 195.99 5.28 ...
  Shipping Cost: num [1:22] 35 68.02 2.99 3.99 2.99 ...
  Customer Name: chr [1:22] "Muhammed MacIntyre" "Barry French" "Barry French" "Clay Rozenda…
```

```
> install.packages("ggplot2")
trying URL 'https://cran.rstudio.com/bin/macosx/contrib/4.0/ggplot2_3.3.2.tgz'
Content type 'application/x-gzip' length 4064530 bytes (3.9 MB)
==================================================
downloaded 3.9 MB


The downloaded binary packages are in
	/var/folders/4d/4w3lfcz1359bmg4z_vcskly00000gn/T//RtmpLIm3uS/downloaded_pack
ages
> library(ggplot2)
> ggplot(MyData.xlsx, aes(x = `Unit Price`, y = `Profit`)) +
+   geom_point(color= "navyblue", size = 2, alpha=.8) +
+   geom_smooth(method = "lm") +
+   labs(x = "Unit Price for Shipping",
+        y = "Profit for Shipping",
+        title = "Plot comparing Unit Price vs Profit")
`geom_smooth()` using formula 'y ~ x'
> 
```

**Fig. 2.41** Data visualization using the ggplot2 package and library

```
ggplot(MyData.xlsx, aes(x = `Unit Price`, y = `Profit`)) +
  geom_point(color= "navyblue", size = 2, alpha=.8) +
  geom_smooth(method = "lm") +
  labs(x = "Unit Price for Shipping",
       y = "Profit for Shipping",
       title = "Plot comparing Unit Price vs Profit")
```

***Now, the authors invite the readers and are confident that the readers can work with data in R and experiment with and plot the different data and variables we have covered so far in this chapter.

## 2.8 Summary

To summarize the content of this chapter; the authors covered how to work with data in R, including the different functions that R users can use to create or import data in RStudio for further analysis and visualizations. We looked at what R Vectors and Factors are. Then we covered how to install R packages and libraries for data analysis and manipulations or calculations. Lastly, we looked at different ways the users can plot or visualize the data and graphs in R. In the next remaining chapters in this Part I of the book, the authors will introduce the users to How to conduct the test of normality and reliability of data in R (Chap. 3), Choosing between parametric and non-parametric tests (Chap. 4), Understanding the difference between dependent and independent variables in research experiments and hypothesis testing (Chap. 5), and an Introduction to understanding the different types of statistical data analysis and methods (Chap. 6), as a foundation to introducing the readers to the topic of different types of statistical data analysis for research using R, which the authors presents in the second (Part II) of the book.

## References

Comma-separated (.csv) data format sample. Available at: https://www.microsoft.com/en-us/download/details.aspx?id=45485

CRAN Repository for R packages. Available at: https://cran.r-project.org/web/packages/

Douglas, A., Roos, D., Mancini, F., Couto, A., & Lusseau, D. (2020). *An introduction to R*. bookdown. https://intro2r.com/

Example .txt data format. Available at: https://dasl.datadescription.com/datafiles/

Excel (.xlsx) Data format sample. Available at: https://www.wisdomaxis.com/technology/software/data/for-reports/

Mailund, T. (2017). Advanced object-oriented programming in R. In *Advanced object-oriented programming in R*. Apress. https://doi.org/10.1007/978-1-4842-2919-4

Matloff, N. (2011). *The art of R programming: A tour of statistical software design* (1st ed.). https://freecomputerbooks.com/The-Art-of-R-Programming-Matloff-Norman.html

Stata (.data) Data format sample. Available at: https://www.stata-press.com/data/r8/u.html

SPSS (.sav) Data format sample. Available at: https://lo.unisa.edu.au/mod/book/view.php?id=646443&chapterid=106604

# Chapter 3
# Test of Normality and Reliability of Data in R

## 3.1 Introduction

In scientific research and statistics, assessment of *normality* and *reliability* of data is commonly a good practice, and a lot of the time, a necessary test performed by the users before conducting the different statistical analysis and procedures.

In theory, *data normality tests* are performed to assess if a data sample is well modeled by a normal distribution. In other words, the probability bell-shaped density curve described by its *mean* and *standard deviation* with the extreme values assumed to have no significant impact on the mean value (Mishra et al., 2019; Vaclavik et al., 2020). The importance and relevance of these measurements and data distribution in determining the direction or type of research carried out by the researchers are discussed in detail in Chap. 4 of this book. The test for normality of data for research is a fundamental assumption and has a significant role in statistics (particularly in parametric tests), especially for the purpose of experimentation and testing, for instance, the researchers or analysts can test whether the hypothesis developed by them fulfills the underlying assumptions or not (Vaclavik et al., 2020).

Some example of the most commonly used types of test for "normality" of data in the available literature include but to mention a few: Kolmogorov–Smirnov (K-S) and Shapiro–Wilk (S-W) tests, Lilliefors corrected K-S test, Anderson–Darling test, Cramer–von Mises test, D'Agostino skewness test, Anscombe–Glynn kurtosis test, D'Agostino–Pearson omnibus test, and Jarque–Bera test, (Ghasemi & Zahediasl, 2012; ÖZTUNA et al., 2006; Peat & Barton, 2005; Thode, 2019), etc.

On the other hand, the *test for reliability* of data or research instruments determines the extent (consistency of measures) to which the scales or variables (items) in the available data are capable of producing a reliable (coherent) result (de Barros Ahrens et al., 2020; Taber, 2018). The reliability and validity of data samples and size is a test mainly carried out by the researchers to demonstrate that the scales that have been construed or adopted in the questionnaires or research instrument are fit for the purpose of experimentation and analysis (Taber, 2018). Some example of the commonly used types of tests for the "reliability" of data samples in the literature includes: Cronbach's alpha test (Taber, 2018; Tavakol & Dennick, 2011), Cohen kappa coefficient test (Carpentier et al., 2017), Exploratory Factor Analysis (EFA) (de Barros Ahrens et al., 2020; Goni et al., 2020), Principal Component Factor Analysis (PCA) (Isnainiyah et al., 2019), etc.

It is noteworthy to mention that with large sample sizes ($n > 30$ or 40), the violation of the data normality or assumption should not necessarily be a concrete factor or problem when conducting statistical analysis for research especially the parametric methods (Elliott & Woodward, 2007; Ghasemi & Zahediasl, 2012; Mishra et al., 2019; Pallant, 2007). In other words, the researchers can still use any of the *parametric* methods or procedures that they find fitting for their experiments or data analysis even when the data are not normally distributed (which a lot of the time are attributed to the non-parametric procedures) for the large enough sample sizes ($n > 30$ or 40) (Taber, 2018).

Over the next sections of this chapter, the authors will introduce the readers to how to conduct the most frequently used types of test for data normality and reliability in the literature; Kolmogorov–Smirnov (K-S) and Shapiro–Wilk (S-W) tests for normality (Sect. 3.2), and Cronbach's alpha test for reliability (Sect. 3.3) in R statistics (Rstudio, 2023).

## 3.2 Test of Data Normality in R: *Kolmogorov–Smirnov (K-S) and Shapiro–Wilk (S-W) Test*

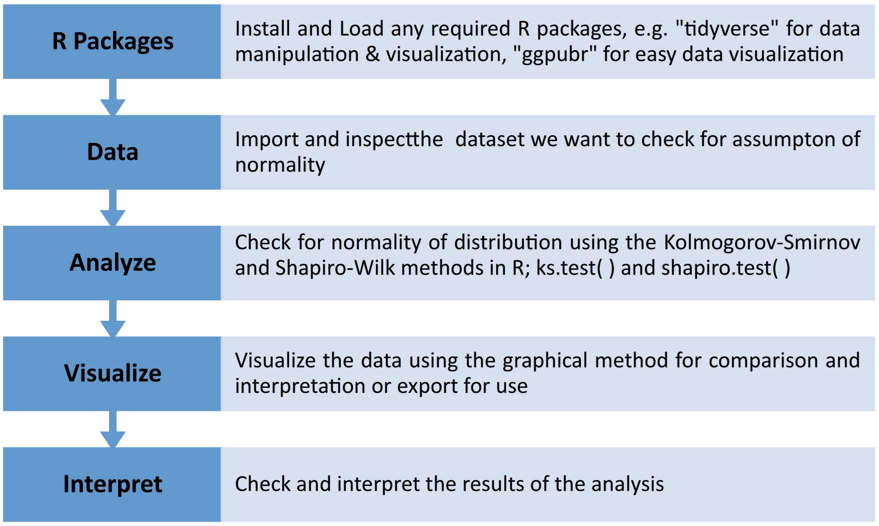
Kolmogorov–Smirnov (K-S) and Shapiro–Wilk (S-W) tests are a great way to determine if variable(s) in a dataset are normally distributed. The default hypothesis for testing whether a given data is normally distributed is *IF* the p-value is greater than 0.05 ($p > 0.05$) and the test statistics above 0.5, *THEN* normality is assumed, *ELSE IF* the p-value is less than or equal to 0.05 ($p \leq 0.05$) THEN normality is not assumed.

We will demonstrate how to conduct the data normality tests using the Kolmogorov–Smirnov (K-S) and Shapiro–Wilk (S-W) methods in R. We will do this in five steps as shown in Fig. 3.1.

To start, **Open RStudio** and **Create a new project** or you can open the existing example project we have created in the previous chapter (Chap. 1) called "**MyFirstR_Project.Rproj**" which the authors will be using to illustrate the examples in this chapter. Once the user have the RStudio and R Project opened, **Create a new RScript** and name it "**NormalityTestDemo**" or any name of your choice (the readers can refer to Chap. 1 on how to do these steps).

Now let's download an example file that we will be using to demonstrate the Kolmogorov–Smirnov (K-S) and Shapiro–Wilk normality tests in R (***the readers are also welcome to use any pre-existing data or file format of their choice***).

Visit the following link as shown in Fig. 3.2 and download the.csv file named "**Sample CVS Files**" and save this on your computer desktop: https://www.learningcontainer.com/sample-excel-data-for-analysis/#Sample_CSV_file_download.



**R Packages**: Install and Load any required R packages, e.g. "tidyverse" for data manipulation & visualization, "ggpubr" for easy data visualization

**Data**: Import and inspectthe  dataset we want to check for assumpton of normality

**Analyze**: Check for normality of distribution using the Kolmogorov-Smirnov and Shapiro-Wilk methods in R; ks.test( ) and shapiro.test( )

**Visualize**: Visualize the data using the graphical method for comparison and interpretation or export for use

**Interpret**: Check and interpret the results of the analysis

**Fig. 3.1** Steps to conducting Kolmogorov–Smirnov and Shapiro–Wilk tests in R

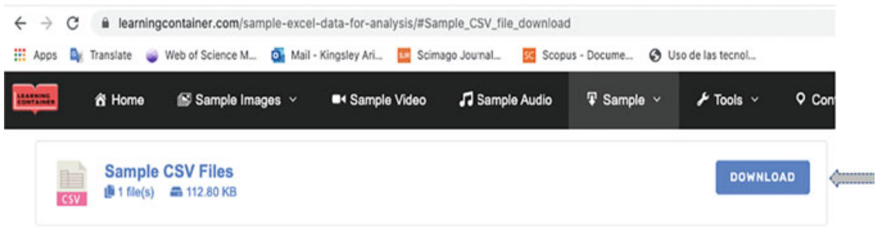


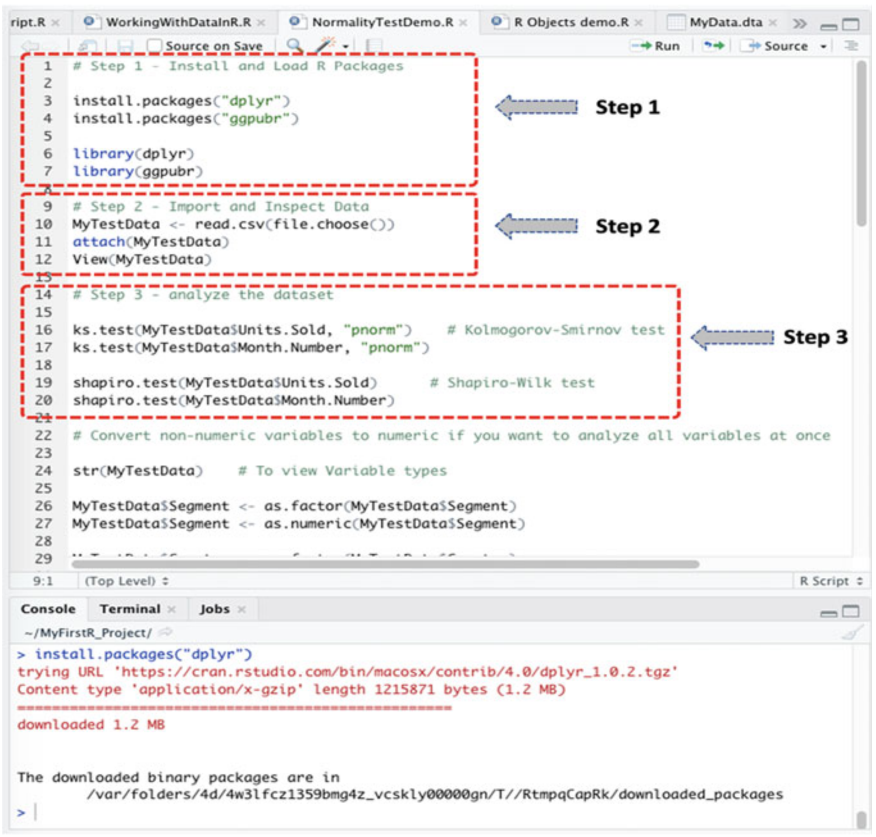
**Fig. 3.2** Example of CSV file format download. (*Source* https://www.learningcontainer.com/sample-excel-data-for-analysis/#Sample_CSV_file_download)

***Note: The example file is also available at the following repository where the authors have uploaded all the example files used in this book: https://doi.org/10.6084/m9.figshare.24728073

Once the user has downloaded and saved the file named “sample-csv-file-for-testing” on the local computer or desktop, we can proceed to conduct the analysis: Kolmogorov–Smirnov (K-S) and Shapiro–Wilk (S-W) tests for normality.

**Install and load the following R packages** (“**dplyr**” and “**ggpubr**”) that we will be using to define or call the functions for the test.

The syntax to install and load the “dplyr” and “ggpubr” packages are as follows (see Fig. 3.3).



```
# Step 1 - Install and Load R Packages

install.packages("dplyr")
install.packages("ggpubr")

library(dplyr)
library(ggpubr)

# Step 2 - Import and Inspect Data
MyTestData <- read.csv(file.choose())
attach(MyTestData)
View(MyTestData)

# Step 3 - analyze the dataset

ks.test(MyTestData$Units.Sold, "pnorm")    # Kolmogorov-Smirnov test
ks.test(MyTestData$Month.Number, "pnorm")

shapiro.test(MyTestData$Units.Sold)      # Shapiro-Wilk test
shapiro.test(MyTestData$Month.Number)

# Convert non-numeric variables to numeric if you want to analyze all variables at once

str(MyTestData)    # To view Variable types

MyTestData$Segment <- as.factor(MyTestData$Segment)
MyTestData$Segment <- as.numeric(MyTestData$Segment)
```

```
> install.packages("dplyr")
trying URL 'https://cran.rstudio.com/bin/macosx/contrib/4.0/dplyr_1.0.2.tgz'
Content type 'application/x-gzip' length 1215871 bytes (1.2 MB)
==================================================
downloaded 1.2 MB


The downloaded binary packages are in
	/var/folders/4d/4w3lfcz1359bmg4z_vcskly00000gn/T//RtmpqCapRk/downloaded_packages
> 
```

**Fig. 3.3** Conducting Kolmogorov–Smirnov and Shapiro–Wilk tests in R

## # Step 1—Install and Load R Packages

```
install.packages("dplyr")
install.packages("ggpubr")

library(dplyr)
library(ggpubr)
```

As demonstrated in Fig. 3.3, once you have installed and loaded the required R packages (see Step 1, Fig. 3.3); the next step is to import the dataset for analysis (Step 2—see command below). See also Chap. 2 for more detailed description on how to import datasets into RStudio environment.

# **Step 2—Import and Inspect Data for** conducting the **Normality Test**

```
MyTestData <- read.csv(file.choose())
attach(MyTestData)
View(MyTestData)
```

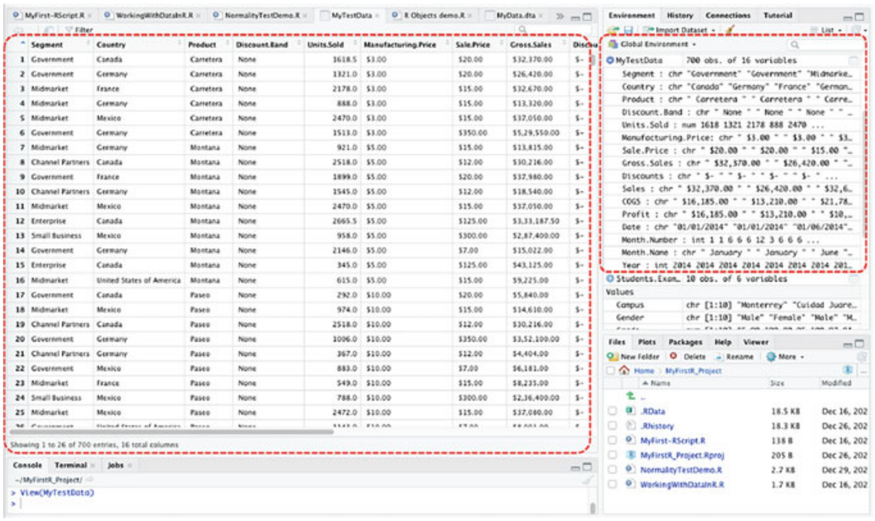
As defined in Step 2 (see Fig. 3.3), when you run the above codes (see Lines 9–12) the user will be presented with a window through which they can navigate and choose the.csv file named "**Sample CSV files**" (Fig. 3.2) which we have downloaded earlier and stored on the desktop.

Once completed, the data will be stored in an R object named "**MyTestData**" and the user will be able to see the details of the dataset (as shown in Fig. 3.4) with 700 observations and 16 variables contained in the data sample. (*** remember you can use any name of your choice, but for learning purposes and examples described in this book, the authors recommend practicing with the example names and objects created/provided in this book***).

Next, the imported dataset is ready to be analyzed. As defined in Fig. 3.3 (see Step 3, Lines 14–20), we can now perform the Kolmogorov–Smirnov and Shapiro–Wilk normality tests.

The syntax for performing the two tests in R is by using the following functions: **ks.test( )** for Kolmogorov–Smirnov, and **shapiro.test( )** for Shapiro–Wilk tests, respectively.

As shown in the codes provided below, the authors used the methods, i.e., **ks.test( )** and **shapiro.test( )** to check for normality of the distribution for the variables named "**Units.Sold**" and "**Month.Number**" in the "**MyTestData"** dataset.



| | Segment | Country | Product | Discount.Band | Units.Sold | Manufacturing.Price | Sale.Price | Gross.Sales | Discou |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Government | Canada | Carretera | None | 1618.5 | $3.00 | $20.00 | $32,370.00 | $- |
| 2 | Government | Germany | Carretera | None | 1321.0 | $3.00 | $20.00 | $26,420.00 | $- |
| 3 | Midmarket | France | Carretera | None | 2178.0 | $3.00 | $15.00 | $32,670.00 | $- |
| 4 | Midmarket | Germany | Carretera | None | 888.0 | $3.00 | $15.00 | $13,320.00 | $- |
| 5 | Midmarket | Mexico | Carretera | None | 2470.0 | $3.00 | $15.00 | $37,050.00 | $- |
| 6 | Government | Germany | Carretera | None | 1513.0 | $3.00 | $350.00 | $5,29,550.00 | $- |
| 7 | Midmarket | Germany | Montana | None | 921.0 | $5.00 | $15.00 | $13,815.00 | $- |
| 8 | Channel Partners | Canada | Montana | None | 2518.0 | $5.00 | $12.00 | $30,216.00 | $- |
| 9 | Government | France | Montana | None | 1899.0 | $5.00 | $20.00 | $37,980.00 | $- |
| 10 | Channel Partners | Germany | Montana | None | 1545.0 | $5.00 | $12.00 | $18,540.00 | $- |
| 11 | Midmarket | Mexico | Montana | None | 2470.0 | $5.00 | $15.00 | $37,050.00 | $- |
| 12 | Enterprise | Canada | Montana | None | 2665.5 | $5.00 | $125.00 | $3,33,187.50 | $- |
| 13 | Small Business | Mexico | Montana | None | 958.0 | $5.00 | $300.00 | $2,87,400.00 | $- |
| 14 | Government | Germany | Montana | None | 2146.0 | $5.00 | $7.00 | $15,022.00 | $- |
| 15 | Enterprise | Canada | Montana | None | 345.0 | $5.00 | $125.00 | $43,125.00 | $- |
| 16 | Midmarket | United States of America | Montana | None | 615.0 | $5.00 | $15.00 | $9,225.00 | $- |
| 17 | Government | Canada | Paseo | None | 292.0 | $10.00 | $20.00 | $5,840.00 | $- |
| 18 | Midmarket | Mexico | Paseo | None | 974.0 | $10.00 | $15.00 | $14,610.00 | $- |
| 19 | Channel Partners | Canada | Paseo | None | 2518.0 | $10.00 | $12.00 | $30,216.00 | $- |
| 20 | Government | Germany | Paseo | None | 1006.0 | $10.00 | $350.00 | $3,52,100.00 | $- |
| 21 | Channel Partners | Germany | Paseo | None | 367.0 | $10.00 | $12.00 | $4,404.00 | $- |
| 22 | Government | Mexico | Paseo | None | 883.0 | $10.00 | $7.00 | $6,181.00 | $- |
| 23 | Midmarket | France | Paseo | None | 549.0 | $10.00 | $15.00 | $8,235.00 | $- |
| 24 | Small Business | Mexico | Paseo | None | 788.0 | $10.00 | $300.00 | $2,36,400.00 | $- |
| 25 | Midmarket | Mexico | Paseo | None | 2472.0 | $10.00 | $15.00 | $37,080.00 | $- |

Showing 1 to 26 of 700 entries, 16 total columns

```
~/MyFirstR_Project/
> View(MyTestData)
>
```

```
MyTestData       700 obs. of 16 variables
  Segment : chr "Government" "Government" "Midmarke…
  Country : chr "Canada" "Germany" "France" "German…
  Product : chr " Carretera " " Carretera " " Carre…
  Discount.Band : chr " None " " None " " None " " …
  Units.Sold : num 1618 1321 2178 888 2470 ...
  Manufacturing.Price: chr " $3.00 " " $3.00 " " $3…
  Sale.Price : chr " $20.00 " " $20.00 " " $15.00 "…
  Gross.Sales : chr " $32,370.00 " " $26,420.00 " "…
  Discounts : chr " $- " " $- " " $- " " $- " ...
  Sales : chr " $32,370.00 " " $26,420.00 " " $32,6…
  COGS : chr " $16,185.00 " " $13,210.00 " " $21,78…
  Profit : chr " $16,185.00 " " $13,210.00 " " $10,…
  Date : chr "01/01/2014" "01/01/2014" "01/06/2014"…
  Month.Number : int 1 1 6 6 6 12 3 6 6 6 ...
  Month.Name : chr " January " " January " " June "…
  Year : int 2014 2014 2014 2014 2014 2014 2014 201…
Students.Exam… 10 obs. of 6 variables
Values
  Campus         chr [1:10] "Monterrey" "Cuidad Juare…
  Gender         chr [1:10] "Male" "Female" "Male" "M…
```

Home › MyFirstR_Project

| Name | Size | Modified |
|---|---|---|
| .RData | 18.5 KB | Dec 16, 202 |
| .Rhistory | 18.3 KB | Dec 26, 202 |
| MyFirst-RScript.R | 138 B | Dec 16, 202 |
| MyFirstR_Project.Rproj | 205 B | Dec 26, 202 |
| NormalityTestDemo.R | 2.7 KB | Dec 29, 202 |
| WorkingWithDataInR.R | 1.7 KB | Dec 16, 202 |

**Fig. 3.4** Example of file or dataset imported in R

## # Step 3—Analyze the Dataset

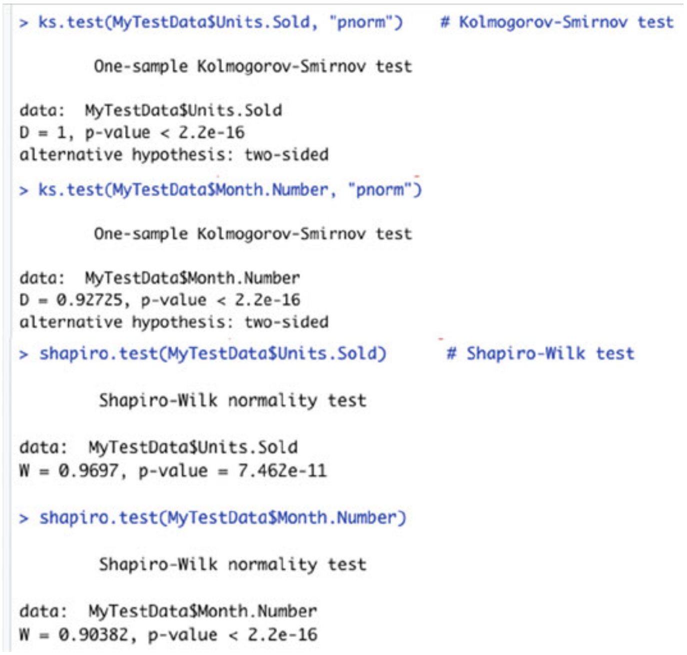
```
ks.test(MyTestData$Units.Sold, "pnorm")    # Kolmogorov-Smirnov test
ks.test(MyTestData$Month.Number, "pnorm")

shapiro.test(MyTestData$Units.Sold)       # Shapiro-Wilk test
shapiro.test(MyTestData$Month.Number)
```

When the user has successfully run the tests (see Lines 14–20, Fig. 3.3), you will be presented with the results in the Console tab, similar to the one shown in Fig. 3.5.

Furthermore, depending on the type of research or variables that are being considered or analyzed by the researchers or analysts (which the book will cover in Part II), the users can decide to check for the normality of distribution of all the variables in the dataset (e.g., MyTestData) all at once.

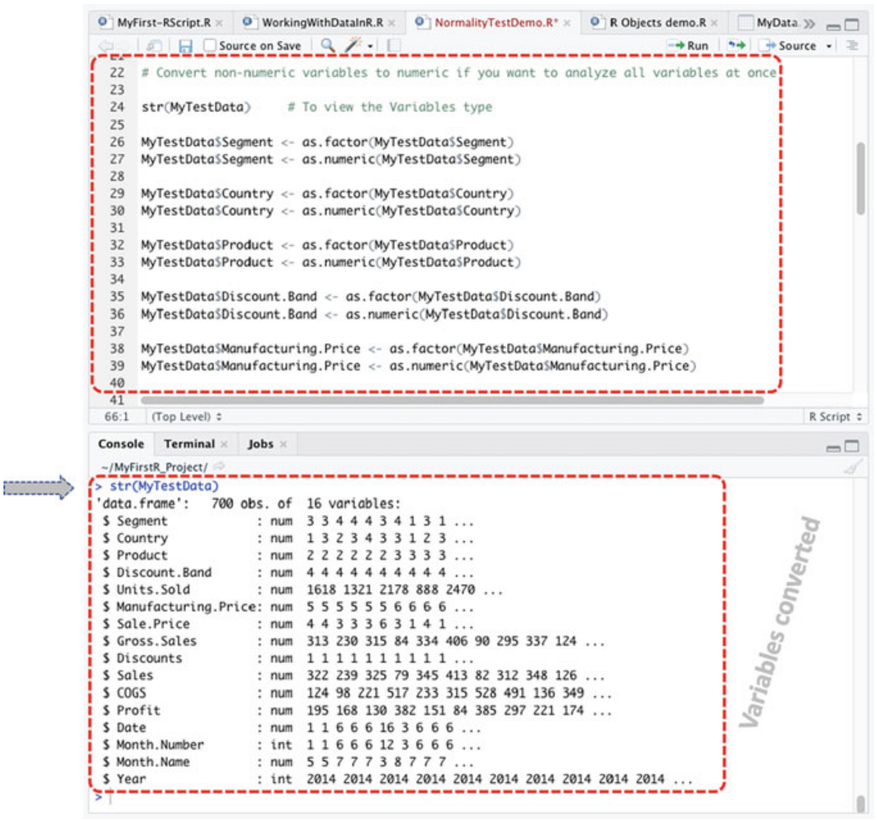
***To do this, we must carry out an important step which is to inspect the dataset and ensure to convert the *non-numeric* variables to *numeric* form.

As illustrated in Fig. 3.6, the users can use the **str( )** function to view a full list of the different variable(s) names and types, including the total number of observations and the total number of variables (note: this information can also be viewed through the Environment Tab).



```
> ks.test(MyTestData$Units.Sold, "pnorm")    # Kolmogorov-Smirnov test

        One-sample Kolmogorov-Smirnov test

data:  MyTestData$Units.Sold
D = 1, p-value < 2.2e-16
alternative hypothesis: two-sided

> ks.test(MyTestData$Month.Number, "pnorm")

        One-sample Kolmogorov-Smirnov test

data:  MyTestData$Month.Number
D = 0.92725, p-value < 2.2e-16
alternative hypothesis: two-sided
> shapiro.test(MyTestData$Units.Sold)      # Shapiro-Wilk test

        Shapiro-Wilk normality test

data:  MyTestData$Units.Sold
W = 0.9697, p-value = 7.462e-11

> shapiro.test(MyTestData$Month.Number)

        Shapiro-Wilk normality test

data:  MyTestData$Month.Number
W = 0.90382, p-value < 2.2e-16
```

**Fig. 3.5** Results of the Kolmogorov–Smirnov and Shapiro–Wilk tests for data normality displayed in the Console tab in R



```
# Convert non-numeric variables to numeric if you want to analyze all variables at once

str(MyTestData)      # To view the Variables type

MyTestData$Segment <- as.factor(MyTestData$Segment)
MyTestData$Segment <- as.numeric(MyTestData$Segment)

MyTestData$Country <- as.factor(MyTestData$Country)
MyTestData$Country <- as.numeric(MyTestData$Country)

MyTestData$Product <- as.factor(MyTestData$Product)
MyTestData$Product <- as.numeric(MyTestData$Product)

MyTestData$Discount.Band <- as.factor(MyTestData$Discount.Band)
MyTestData$Discount.Band <- as.numeric(MyTestData$Discount.Band)

MyTestData$Manufacturing.Price <- as.factor(MyTestData$Manufacturing.Price)
MyTestData$Manufacturing.Price <- as.numeric(MyTestData$Manufacturing.Price)
```

```
> str(MyTestData)
'data.frame':   700 obs. of  16 variables:
 $ Segment            : num  3 3 4 4 4 3 4 1 3 1 ...
 $ Country            : num  1 3 2 3 4 3 3 1 2 3 ...
 $ Product            : num  2 2 2 2 2 2 3 3 3 3 ...
 $ Discount.Band      : num  4 4 4 4 4 4 4 4 4 4 ...
 $ Units.Sold         : num  1618 1321 2178 888 2470 ...
 $ Manufacturing.Price: num  5 5 5 5 5 5 6 6 6 6 ...
 $ Sale.Price         : num  4 4 3 3 3 6 3 1 4 1 ...
 $ Gross.Sales        : num  313 230 315 84 334 406 90 295 337 124 ...
 $ Discounts          : num  1 1 1 1 1 1 1 1 1 1 ...
 $ Sales              : num  322 239 325 79 345 413 82 312 348 126 ...
 $ COGS               : num  124 98 221 517 233 315 528 491 136 349 ...
 $ Profit             : num  195 168 130 382 151 84 385 297 221 174 ...
 $ Date               : num  1 1 6 6 6 16 3 6 6 6 ...
 $ Month.Number       : int  1 1 6 6 6 12 3 6 6 6 ...
 $ Month.Name         : num  5 5 7 7 7 3 8 7 7 7 ...
 $ Year               : int  2014 2014 2014 2014 2014 2014 2014 2014 2014 2014 ...
>
```

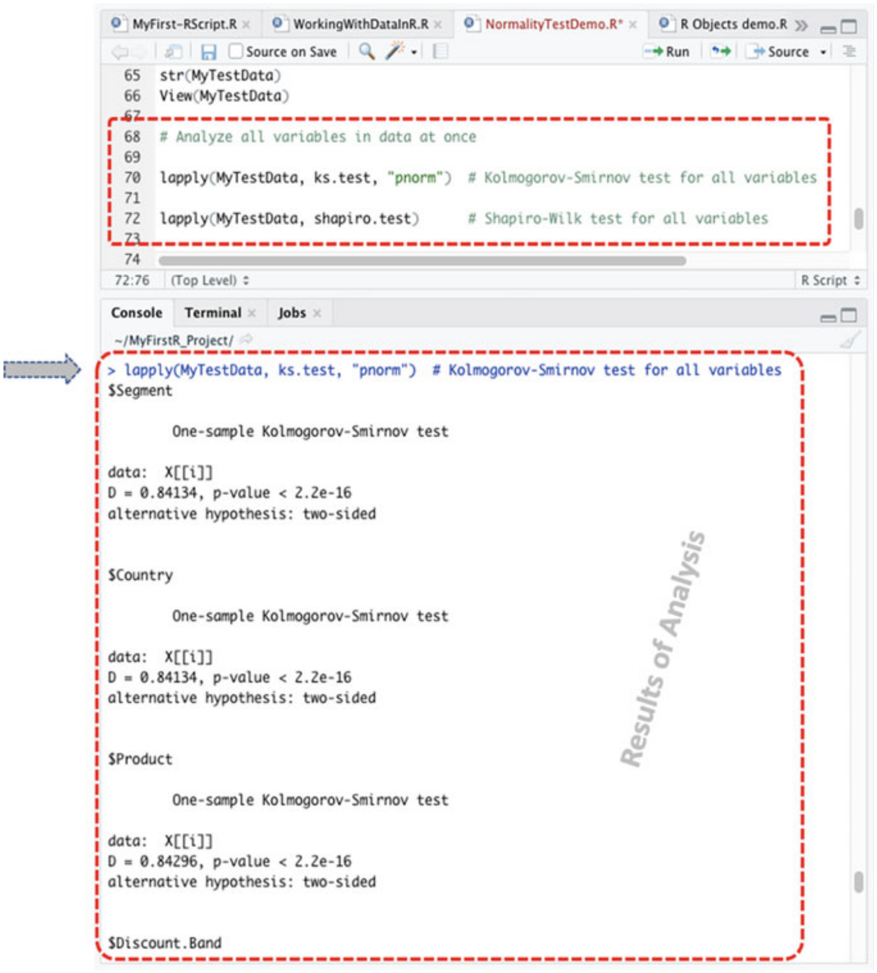
**Fig. 3.6** Converting non-numeric variables or factors to numeric form

Based on the example data that we are working with (see: Figs. 3.2 and 3.4), the authors have implemented the following codes (Fig. 3.6) to convert the “non-numeric (chr)” variables to “numeric (num)” form. (***Note: the readers can refer to Sect. 2.5.5 in Chap. 2 for more detailed description on how to create and perform the *factorization* procedures in R).

**# Converting non-numeric variables or factor to numeric form**

```
str(MyTestData)     # To view the Variables type

MyTestData$Segment <- as.factor(MyTestData$Segment)
MyTestData$Segment <- as.numeric(MyTestData$Segment)

MyTestData$Country <- as.factor(MyTestData$Country)
MyTestData$Country <- as.numeric(MyTestData$Country)

MyTestData$Product <- as.factor(MyTestData$Product)
MyTestData$Product <- as.numeric(MyTestData$Product)

MyTestData$Discount.Band <- as.factor(MyTestData$Discount.Band)
MyTestData$Discount.Band <- as.numeric(MyTestData$Discount.Band)

MyTestData$Manufacturing.Price <- as.factor(MyTestData$Manufacturing.Price)
MyTestData$Manufacturing.Price <-as.numeric(MyTestData$Manufacturing.Price)

MyTestData$Sale.Price <- as.factor(MyTestData$Sale.Price)
MyTestData$Sale.Price <- as.numeric(MyTestData$Sale.Price)

MyTestData$Gross.Sales <- as.factor(MyTestData$Gross.Sales)
MyTestData$Gross.Sales <- as.numeric(MyTestData$Gross.Sales)

MyTestData$Discounts <- as.factor(MyTestData$Discounts)
MyTestData$Discounts <- as.numeric(MyTestData$Discounts)

MyTestData$Sales <- as.factor(MyTestData$Sales)
MyTestData$Sales <- as.numeric(MyTestData$Sales)

MyTestData$COGS <- as.factor(MyTestData$COGS)
MyTestData$COGS <- as.numeric(MyTestData$COGS)

MyTestData$Profit <- as.factor(MyTestData$Profit)
MyTestData$Profit <- as.numeric(MyTestData$Profit)

MyTestData$Date <- as.factor(MyTestData$Date)
MyTestData$Date <- as.numeric(MyTestData$Date)

MyTestData$Month.Name <- as.factor(MyTestData$Month.Name)
MyTestData$Month.Name <- as.numeric(MyTestData$Month.Name)

str(MyTestData)       # To view the new and converted variables
View(MyTestData)
)
```

Once the user has successfully converted the “**chr**” (character) variable(s) to “**num**” (number) (see Fig. 3.6), we can now run the Kolmogorov–Smirnov and Shapiro–Wilk normality tests for all the elements in the data by using, for instance, the **lapply( )** command as demonstrated code below and results in Fig. 3.7.



```
str(MyTestData)
View(MyTestData)

# Analyze all variables in data at once

lapply(MyTestData, ks.test, "pnorm")  # Kolmogorov-Smirnov test for all variables

lapply(MyTestData, shapiro.test)      # Shapiro-Wilk test for all variables
```

```
> lapply(MyTestData, ks.test, "pnorm")  # Kolmogorov-Smirnov test for all variables
$Segment

	One-sample Kolmogorov-Smirnov test

data:  X[[i]]
D = 0.84134, p-value < 2.2e-16
alternative hypothesis: two-sided


$Country

	One-sample Kolmogorov-Smirnov test

data:  X[[i]]
D = 0.84134, p-value < 2.2e-16
alternative hypothesis: two-sided


$Product

	One-sample Kolmogorov-Smirnov test

data:  X[[i]]
D = 0.84296, p-value < 2.2e-16
alternative hypothesis: two-sided


$Discount.Band
```

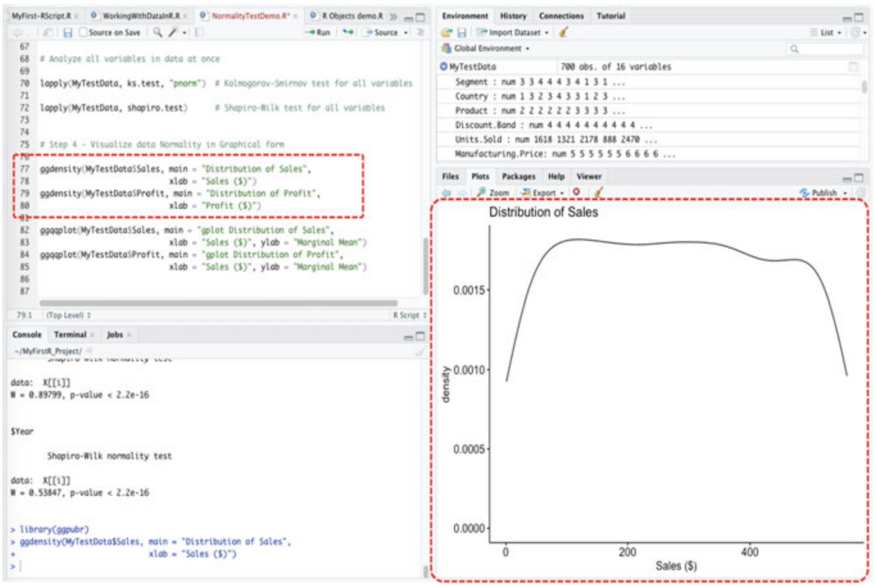
**Fig. 3.7** Conducting Kolmogorov–Smirnov and Shapiro–Wilk normality tests for all elements (variables) in the data

## # Analyze all the variables in the dataset all at once

```
lapply(MyTestData, ks.test, "pnorm") # Kolmogorov-Smirnov test for all variables

lapply(MyTestData, shapiro.test)     # Shapiro-Wilk test for all variables
```

## # Step 4—Visualize Data Normality as a Plot (Graphical Representation)

Another important and useful way to visually check the normality of data in R is by plotting the distribution of the different variables by using, for instance, the "density plot" and "quantile–quantile plot" functions.



```
# Analyze all variables in data at once

lapply(MyTestData, ks.test, "pnorm")  # Kolmogorov-Smirnov test for all variables

lapply(MyTestData, shapiro.test)      # Shapiro-Wilk test for all variables


# Step 4 - Visualize data Normality in Graphical form

ggdensity(MyTestData$Sales, main = "Distribution of Sales",
                            xlab = "Sales ($)")
ggdensity(MyTestData$Profit, main = "Distribution of Profit",
                             xlab = "Profit ($)")

ggqqplot(MyTestData$Sales, main = "gplot Distribution of Sales",
                           xlab = "Sales ($)", ylab = "Marginal Mean")
ggqqplot(MyTestData$Profit, main = "gplot Distribution of Profit",
                            xlab = "Sales ($)", ylab = "Marginal Mean")
```

```
data:  X[[i]]
W = 0.89799, p-value < 2.2e-16


$Year

        Shapiro-Wilk normality test

data:  X[[i]]
W = 0.53847, p-value < 2.2e-16


> library(ggpubr)
> ggdensity(MyTestData$Sales, main = "Distribution of Sales",
+                              xlab = "Sales ($)")
> 
```

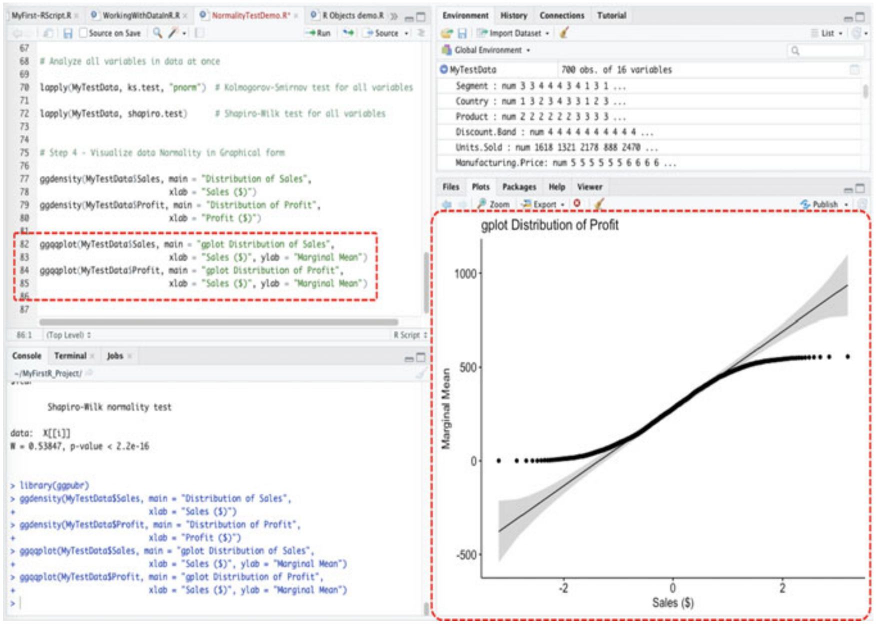
```
MyTestData              700 obs. of 16 variables
  Segment : num 3 3 4 4 4 3 4 1 3 1 ...
  Country : num 1 3 2 3 4 3 3 1 2 3 ...
  Product : num 2 2 2 2 2 2 3 3 3 3 ...
  Discount.Band : num 4 4 4 4 4 4 4 4 4 4 ...
  Units.Sold : num 1618 1321 2178 888 2470 ...
  Manufacturing.Price: num 5 5 5 5 5 5 6 6 6 6 ...
```

**Fig. 3.8** Density plot of the example dataset

To describe the different functions, the *density plot* (see Fig. 3.8) gives a visual judgment with respect to whether the distribution of the data is "bell-shaped" or "skewed". The *quantile–quantile plot* (Q-Q plot) (Fig. 3.9) tends to draw a correlation between the specified sample and the normality of the distribution. The Q-Q plot also includes a reference line that is usually plotted or mapped at 45 degrees. Thus, each observation is plotted as a single dot, and the dots should form a straight line if the data is normal.

To demonstrate to the readers how to do this, we will use the **ggdensity( )** and **ggqqplot( )** functions, which are supported by the "**ggpubr**" package (see Step 1), to visualize the normality of the example data (MyTestData), respectively. As shown in Figs. 3.8 and 3.9, we will be using the "**Sales**" and "**Profit**" variables in the example data (MyTestData) to demonstrate this. ***Also, feel free to try and plot the other variable(s) in the data using this particular example.

The syntax and code for the two example plots using the ggdensity( ) and ggqqplot( ) functions are as shown below, and the results (distribution graph) are as represented in Figs. 3.8 and 3.9.



```
# Analyze all variables in data at once

lapply(MyTestData, ks.test, "pnorm")  # Kolmogorov-Smirnov test for all variables

lapply(MyTestData, shapiro.test)      # Shapiro-Wilk test for all variables


# Step 4 - Visualize data Normality in Graphical form

ggdensity(MyTestData$Sales, main = "Distribution of Sales",
                            xlab = "Sales ($)")
ggdensity(MyTestData$Profit, main = "Distribution of Profit",
                            xlab = "Profit ($)")

ggqqplot(MyTestData$Sales, main = "gplot Distribution of Sales",
                            xlab = "Sales ($)", ylab = "Marginal Mean")
ggqqplot(MyTestData$Profit, main = "gplot Distribution of Profit",
                            xlab = "Sales ($)", ylab = "Marginal Mean")
```

```
	Shapiro-Wilk normality test

data:  X[[i]]
W = 0.53847, p-value < 2.2e-16


> library(ggpubr)
> ggdensity(MyTestData$Sales, main = "Distribution of Sales",
+                             xlab = "Sales ($)")
> ggdensity(MyTestData$Profit, main = "Distribution of Profit",
+                             xlab = "Profit ($)")
> ggqqplot(MyTestData$Sales, main = "gplot Distribution of Sales",
+                             xlab = "Sales ($)", ylab = "Marginal Mean")
> ggqqplot(MyTestData$Profit, main = "gplot Distribution of Profit",
+                             xlab = "Sales ($)", ylab = "Marginal Mean")
> |
```

```
MyTestData          700 obs. of 16 variables
  Segment : num 3 3 4 4 4 3 4 1 3 1 ...
  Country : num 1 3 2 3 4 3 3 1 2 3 ...
  Product : num 2 2 2 2 2 2 3 3 3 3 ...
  Discount.Band : num 4 4 4 4 4 4 4 4 4 4 ...
  Units.Sold : num 1618 1321 2178 888 2470 ...
  Manufacturing.Price: num 5 5 5 5 5 5 6 6 6 6 ...
```

**Fig. 3.9** Quantile–quantile plot (Q-Q plot) of the example data

## # Visualizing the normality of data in Graphical form

```
ggdensity(MyTestData$Sales, main = "Distribution of Sales",
                            xlab = "Sales ($)")
ggdensity(MyTestData$Profit, main = "Distribution of Profit",
                            xlab = "Profit ($)")



ggqqplot(MyTestData$Sales, main = "gplot Distribution of Sales",
                            xlab = "Sales ($)", ylab = "Marginal Mean")
ggqqplot(MyTestData$Profit, main = "gplot Distribution of Profit",
                            xlab = "Sales ($)", ylab = "Marginal Mean")
```

## Useful Tips and Information:

1. The users can use the "**Export**" menu found there on the same **Plot Tab** (see: Figs. 3.8 and 3.9) to save the plot (graphics) as image or pdf file on the local computer for later or further use in presentations or write-up.
2. Use the histogram function, **hist( )** to experiment and view the distribution of the different variables. For example; `hist(MyTestData$Sales)`.

**# Step 5—Results Interpretation for the Data Normality Test**

The last step in the data normality test or analysis is to interpret or understand the results of the test. By default, the null hypothesis for testing whether a dataset is normally distributed is IF the test statistics or value is greater than 0.5, and the p-value greater than the 0.05 threshold (i.e, $p > 0.05$), THEN normality is assumed, ELSE IF $p \leq 0.05$ THEN normality is not assumed.

As shown in the example and results described below by using the variable called "**Segment**" in the example data (MyTestData); the following outputs `D = 0.84134` and `W = 0.89536` represents the value of the statistics for the Kolmogorov–Smirnov and Shapiro–Wilk tests, respectively. While the `p-value < 2.2e-16` (i.e., $p = 0.00$) which was found for both test, represents the significant level (referred to as p-values) of the corresponding tests.

```
> lapply(MyTestData, ks.test, "pnorm")
$Segment
     One-sample Kolmogorov-Smirnov test
data:  X[[i]]
D = 0.84134, p-value < 2.2e-16
alternative hypothesis: two-sided
```

```
> lapply(MyTestData, shapiro.test)
$Segment
     Shapiro-Wilk normality test
data:  X[[i]]
W = 0.89536, p-value < 2.2e-16
```

As reported in the second part of the highlighting in Fig. 3.7 and in the above results, we can see that the **p-values** for each of the variables or test (Kolmogorov: D = 0.84134; Shapiro: W = 0.89536) are less than 0.05 ($p < 0.05$) with the majority of them showing a *p-value of* $p <$ 2.2e-16 (scientifically interpreted as $p = 0.00$). The number or value: 2.2e-16 is the scientific notation of 0.00000000000000022 which means that the values are fundamentally very close to zero.

Therefore, with the above results, we can reject the null hypothesis (i.e., $p > 0.05$), and assume that the example dataset is not normally distributed.

Also, it is important to mention to the readers that datasets which are normally distributed are expected to be "bell-shaped" when graphically represented, whereas non-normally distributed datasets are "skewed". For example, as seen in Figs. 3.8 and 3.9, we can see that the plots are not represented to be bell-shaped (skewed) (Fig. 3.8) or not on a 45 degrees straight line (Fig. 3.9), thus, affirming the statistical results the authors have interpreted earlier (see Fig. 3.7) which shows that the example dataset is not normally distributed.

**Useful Tips and Information:**

1. The Shapiro–Wilk normality test is sensitive and are typically appropriate for small sample sizes.
2. Kolmogorov–Smirnov test is recommended for large sample sizes, for instance, samples greater than 100 ($n > 100$).
3. Shapiro–Wilk test is widely recommended for data normality tests and tends to provide better statistical power than Kolmogorov–Smirnov test. It is based on the correlation between the data and the corresponding normal scores (Ghasemi & Zahediasl, 2012).
4. Parametric procedures or analysis (which the authors will cover in the next chapter—Chap. 4) can still be conducted on large datasets (e.g., $n > 30$ or 40), regardless of whether the specified data violates the assumption of normality (Roscoe, 1975). However, for small sample sizes ($n < 30$) that appear to be non-normally distributed, the *non-parametric* procedures are scientifically recommended.
5. In some situations, the users may get a warning message such as "ties should not be present for the Kolmogorov–Smirnov test" when conducting the Kolmogorov–Smirnov (K-S) test depending on the type of variable(s) being analyzed. K-S test assumes that the datasets are *continuous* (which in some cases are likely not) and therefore tends to generate a warning when it finds the presence of ties. Nonetheless, the diffident sums of parsing or rounding on the variables are still significantly effective on the calculated statistics, and as such are still theoretically estimated to be valid.

## 3.3 Test of Data Reliability in R: Cronbach's Alpha Test

The Cronbach's alpha, α (or coefficient alpha), test is one of the most commonly used method to determine if a dataset or instrument is *reliable*, for instance, for research purposes or the many other types of data analysis and computation. The test (α) is used by researchers to ascertain how closely related a set of item(s) in a dataset are as a group.

By default, the general rule of thumb is that a Cronbach's alpha (α) test result of 0.70 and above is good and acceptable. Meaning that the results or conclusions drawn from analyzing the available data are assumed to be reliable for further analysis and/ or drawing scientific conclusions.

Here, we will demonstrate how to conduct Cronbach's alpha test in R. As described in Fig. 3.10, the authors will show two ways or methods of how to perform this test (Cronbach's alpha) using the four defined steps in R:

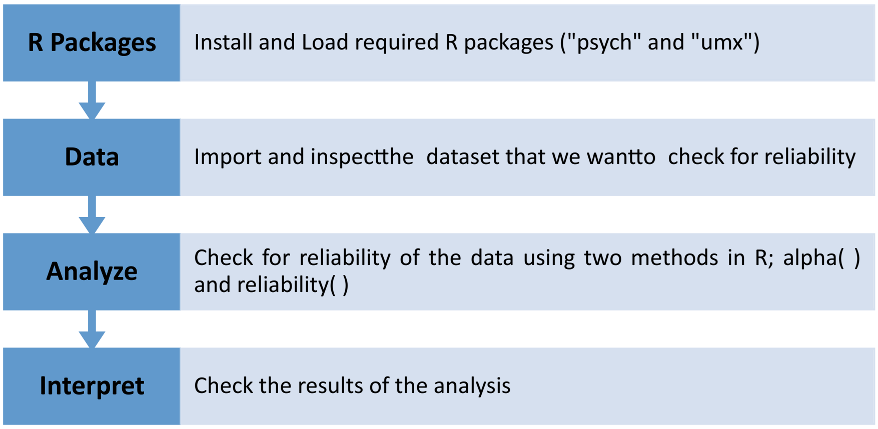
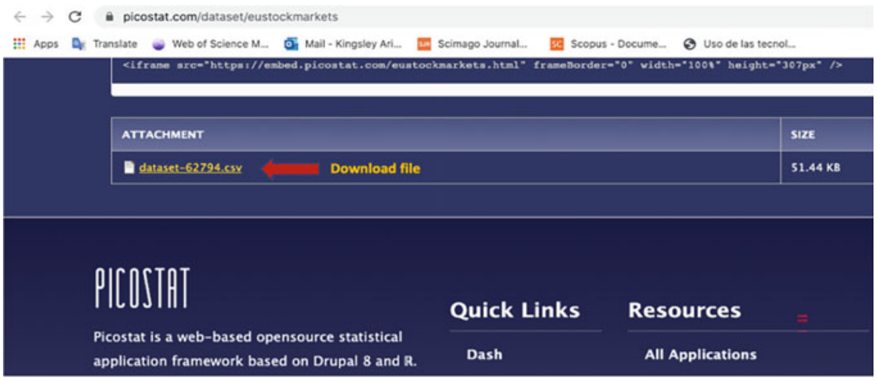

R Packages
Install and Load required R packages ("psych" and "umx")
Data
Import and inspectthe  dataset that we wantto  check for reliability
Analyze
Check for reliability of the data using two methods in R; alpha( ) and reliability( )
Interpret
Check the results of the analysis

**Fig. 3.10** Steps for conducting Cronbach's alpha test in R



```
<iframe src="https://embed.picostat.com/eustockmarkets.html" frameBorder="0" width="100%" height="307px" />
```

| ATTACHMENT | SIZE |
|---|---|
| dataset-62794.csv Download file | 51.44 KB |

PICOSTAT

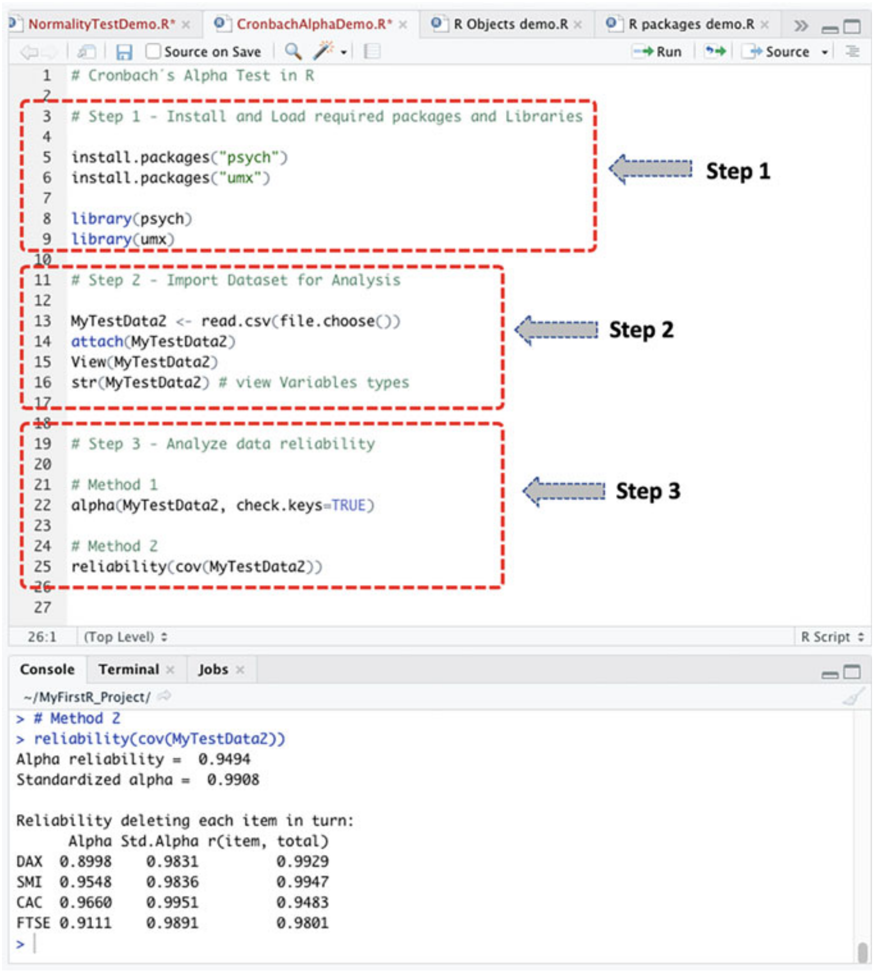
Picostat is a web-based opensource statistical application framework based on Drupal 8 and R.

**Fig. 3.11** Example of CVS file download. *Source* https://www.picostat.com/dataset/eustockmarkets)

Now, let's start by creating a new R Script and name it "**CronbachAlphaDemo**" or any name the reader chooses.

Once you have created the R Script, download the example file that we will use to demonstrate the Cronbach's reliability test in R.

As shown in Fig. 3.11, visit the following link (https://www.picostat.com/dataset/eustockmarkets) and download the CVS file named "**dataset-62794.csv**" and save this on your computer desktop. (***the readers are welcome to use any file or format of choice, but for this example, the authors will be using the example CSV file***). The example dataset is also available for download at the following data repository: https://doi.org/https://doi.org/10.6084/m9.figshare.24728073.



```
 1  # Cronbach's Alpha Test in R
 2
 3  # Step 1 - Install and Load required packages and Libraries
 4
 5  install.packages("psych")
 6  install.packages("umx")
 7
 8  library(psych)
 9  library(umx)
10
11  # Step 2 - Import Dataset for Analysis
12
13  MyTestData2 <- read.csv(file.choose())
14  attach(MyTestData2)
15  View(MyTestData2)
16  str(MyTestData2) # view Variables types
17
18
19  # Step 3 - Analyze data reliability
20
21  # Method 1
22  alpha(MyTestData2, check.keys=TRUE)
23
24  # Method 2
25  reliability(cov(MyTestData2))
26
27
```

```
> # Method 2
> reliability(cov(MyTestData2))
Alpha reliability =  0.9494
Standardized alpha =  0.9908

Reliability deleting each item in turn:
      Alpha Std.Alpha r(item, total)
DAX  0.8998    0.9831         0.9929
SMI  0.9548    0.9836         0.9947
CAC  0.9660    0.9951         0.9483
FTSE 0.9111    0.9891         0.9801
> 
```

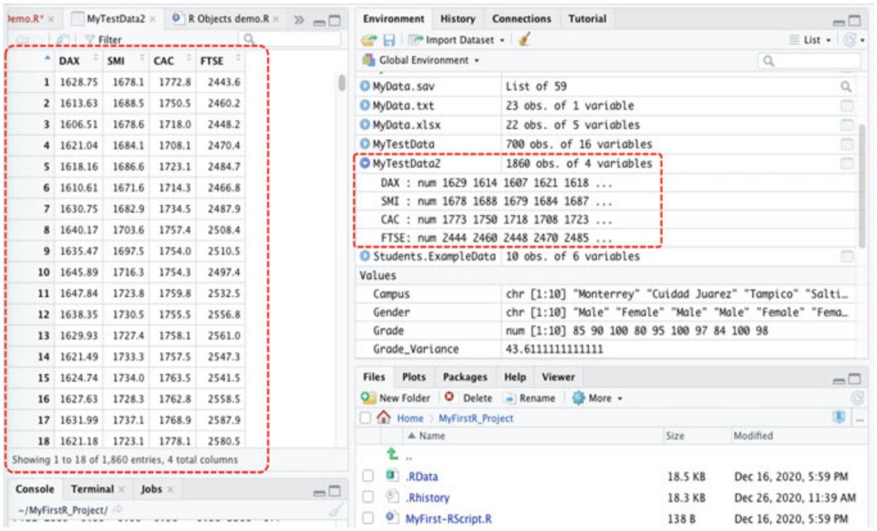
**Fig. 3.12** Cronbach's alpha reliability test in R

Once the file download is completed and the CSV file saved on the local machine or system, e.g., the user's desktop, we can proceed to conduct Cronbach's alpha reliability test by following the steps we have defined in Fig. 3.10.

## # Step 1—Install and Load the Required R Packages

As shown in Fig. 3.12 (see Lines 3–9) **Install and Load** the necessary **R packages** ("**psych**" and "**umx**") that we will be using to conduct Cronbach's alpha test.

The syntax to install and load the "psych" and "umx" packages are as follows:



| | DAX | SMI | CAC | FTSE |
|---|---|---|---|---|
| 1 | 1628.75 | 1678.1 | 1772.8 | 2443.6 |
| 2 | 1613.63 | 1688.5 | 1750.5 | 2460.2 |
| 3 | 1606.51 | 1678.6 | 1718.0 | 2448.2 |
| 4 | 1621.04 | 1684.1 | 1708.1 | 2470.4 |
| 5 | 1618.16 | 1686.6 | 1723.1 | 2484.7 |
| 6 | 1610.61 | 1671.6 | 1714.3 | 2466.8 |
| 7 | 1630.75 | 1682.9 | 1734.5 | 2487.9 |
| 8 | 1640.17 | 1703.6 | 1757.4 | 2508.4 |
| 9 | 1635.47 | 1697.5 | 1754.0 | 2510.5 |
| 10 | 1645.89 | 1716.3 | 1754.3 | 2497.4 |
| 11 | 1647.84 | 1723.8 | 1759.8 | 2532.5 |
| 12 | 1638.35 | 1730.5 | 1755.5 | 2556.8 |
| 13 | 1629.93 | 1727.4 | 1758.1 | 2561.0 |
| 14 | 1621.49 | 1733.3 | 1757.5 | 2547.3 |
| 15 | 1624.74 | 1734.0 | 1763.5 | 2541.5 |
| 16 | 1627.63 | 1728.3 | 1762.8 | 2558.5 |
| 17 | 1631.99 | 1737.1 | 1768.9 | 2587.9 |
| 18 | 1621.18 | 1723.1 | 1778.1 | 2580.5 |

Showing 1 to 18 of 1,860 entries, 4 total columns

```
MyData.sav            List of 59
MyData.txt            23 obs. of 1 variable
MyData.xlsx           22 obs. of 5 variables
MyTestData            700 obs. of 16 variables
MyTestData2           1860 obs. of 4 variables
  DAX : num 1629 1614 1607 1621 1618 ...
  SMI : num 1678 1688 1679 1684 1687 ...
  CAC : num 1773 1750 1718 1708 1723 ...
  FTSE: num 2444 2460 2448 2470 2485 ...
Students.ExampleData  10 obs. of 6 variables
Values
  Campus              chr [1:10] "Monterrey" "Cuidad Juarez" "Tampico" "Salti…
  Gender              chr [1:10] "Male" "Female" "Male" "Male" "Female" "Fema…
  Grade               num [1:10] 85 90 100 80 95 100 97 84 100 98
  Grade_Variance      43.6111111111111
```

Home › MyFirstR_Project

| Name | Size | Modified |
|---|---|---|
| .. | | |
| .RData | 18.5 KB | Dec 16, 2020, 5:59 PM |
| .Rhistory | 18.3 KB | Dec 26, 2020, 11:39 AM |
| MyFirst-RScript.R | 138 B | Dec 16, 2020, 5:59 PM |

**Fig. 3.13** Example of CSV dataset imported in R

```
install.packages("psych")
install.packages("umx")

library(psych)
library(umx)
```

As shown in Fig. 3.12 (Step 1—Lines 5–9), once the user has the necessary packages installed and loaded, the next step is to import the dataset for analysis (Step 2: see code below).

## # Step 2—Import and Inspect Data

```
MyTestData2 <- read.csv(file.choose())
attach(MyTestData2)
View(MyTestData2)
str(MyTestData2)      # view Variables types
```

As demonstrated in Fig. 3.12 (Step 2), when you run the commands (see Lines 11–16) you will be presented with a window through which you can navigate and choose the.csv file named "**dataset-62794**" that we downloaded earlier and stored on the computer desktop.

Once selected and successful, the data will be imported and stored in an R object defined as "**MyTestData2**" (***remember you can always use any name of your choice***). The users can see the details of the dataset highlighted in Fig. 3.13, with 1860 observations and 4 variables in the data sample.

***Note**: Remember to Convert *non-numeric* variables to *numeric* if using a dataset that contains Character or Categorical values. Users can refer to Sect. 3.2 on how to do this task or Chap. 2 “Working with Data in R (see Sect 2.5.5)” for more details.

**# Step 3—Analyze the Data Reliability using Cronbach’s Alpha Methods**

Now we can proceed to analyze the imported dataset (stored as R object we named or defined as—MyTestData2) using the **alpha( )** and **reliability( )** functions in R.

The syntax for performing the reliability tests in R using the two methods is as follows:

**# Method 1**

```
alpha(MyTestData2, check.keys=TRUE)
```

**# Method 2**

```
reliability(cov(MyTestData2))
```

Once the user have successfully run the codes (Lines 19–25), the user will be presented with the results of the tests in the Console, similar to the one illustrated in Fig. 3.14.

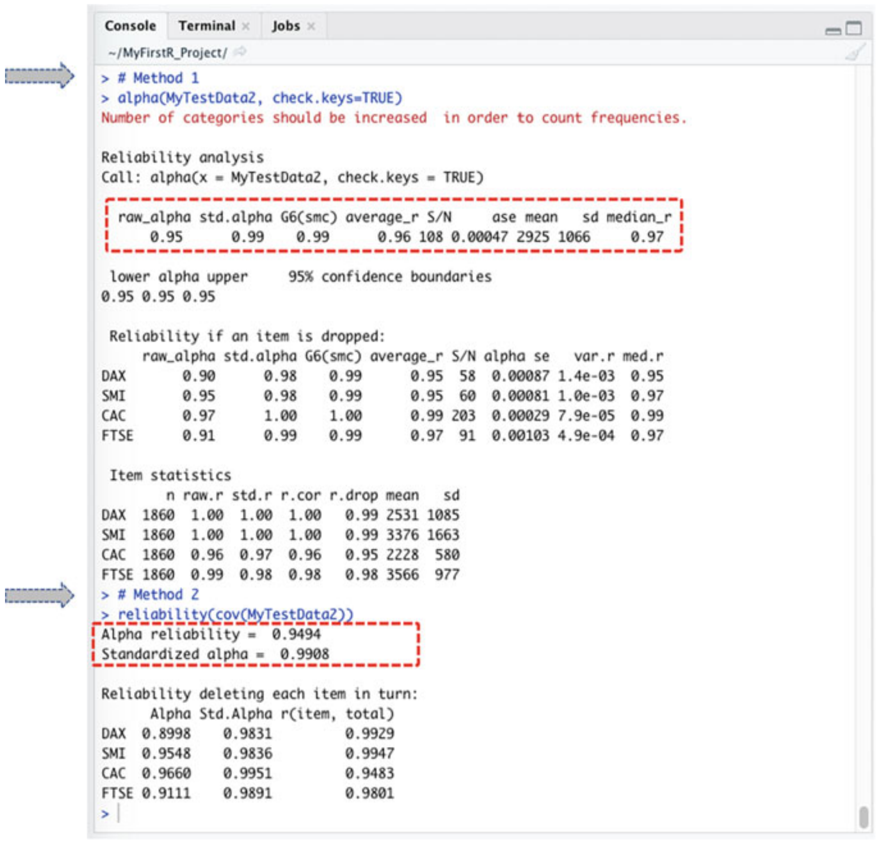
**# Step 4–Results Interpretation for Cronbach’s Alpha Test**

The final step in the reliability of data analysis is to interpret the results of Cronbach’s alpha test. By default, the null hypothesis for testing whether a given dataset or instrument is reliable for research purposes or data analysis is *IF* α (coefficient alpha) is greater than or equal to 0.70 ($\alpha \geq 0.70$), *THEN* reliability of data is statistically good and scientifically acceptable, *ELSE IF* α is less than 0.70 ($\alpha < 0.70$), *THEN* data reliable is questionable. It is also important to mention that these measures can vary based on different context scenarios or research experimental settings.

As highlighted in the results we derived by using the example dataset (defined as MyTestData2) (see: Fig. 3.14), the statistics (reliability result) of the data is $\alpha = 0.95$ (for Method 1) and $\alpha = 0.9494$ (for Method 2) with std. alpha of 0.99 (Method 1) and 0.9908 (Method 2), respectively. Therefore, we can accept the null hypothesis, i.e., $\alpha \geq 0.70$, and assume that the tested or analyzed dataset is reliable for any statistical analysis or research experiments.

## 3.4 Summary

In this chapter, the authors looked at the preliminary, yet, important tests that are conducted by the researchers when carrying out the experimentations or statistical data analysis. This consists of the process of testing the available datasets for



```
> # Method 1
> alpha(MyTestData2, check.keys=TRUE)
Number of categories should be increased  in order to count frequencies.

Reliability analysis
Call: alpha(x = MyTestData2, check.keys = TRUE)

  raw_alpha std.alpha G6(smc) average_r S/N     ase mean   sd median_r
      0.95      0.99    0.99      0.96 108 0.00047 2925 1066     0.97

 lower alpha upper     95% confidence boundaries
0.95 0.95 0.95

 Reliability if an item is dropped:
     raw_alpha std.alpha G6(smc) average_r S/N alpha se   var.r med.r
DAX       0.90      0.98    0.99      0.95  58  0.00087 1.4e-03  0.95
SMI       0.95      0.98    0.99      0.95  60  0.00081 1.0e-03  0.97
CAC       0.97      1.00    1.00      0.99 203  0.00029 7.9e-05  0.99
FTSE      0.91      0.99    0.99      0.97  91  0.00103 4.9e-04  0.97

 Item statistics
        n raw.r std.r r.cor r.drop mean   sd
DAX  1860  1.00  1.00  1.00   0.99 2531 1085
SMI  1860  1.00  1.00  1.00   0.99 3376 1663
CAC  1860  0.96  0.97  0.96   0.95 2228  580
FTSE 1860  0.99  0.98  0.98   0.98 3566  977
> # Method 2
> reliability(cov(MyTestData2))
Alpha reliability =  0.9494
Standardized alpha =  0.9908

Reliability deleting each item in turn:
      Alpha Std.Alpha r(item, total)
DAX  0.8998    0.9831         0.9929
SMI  0.9548    0.9836         0.9947
CAC  0.9660    0.9951         0.9483
FTSE 0.9111    0.9891         0.9801
>
```

**Fig. 3.14** Results of Cronbach's alpha tests displayed in the Console in R

*normality of distribution* and *reliability of the data sample* for scientific research purposes. In theory, these tests (i.e., normality and reliability) are done to assess the extent to which the data is capable of producing coherent and assertive research results or conclusions. In Sect. 3.2, the authors demonstrated how to conduct the *Kolmogorov–Smirnov* and *Shapiro–Wilk* normality tests in R, which in comparison to the other mentioned types of tests, is the most commonly used method by the researchers to check the distribution of the data. In Sect. 3.3, we illustrated how to conduct *Cronbach's alpha* reliability test using two different methods in R. In each case (Sect. 3.2 and 3.3), we discussed the meaning of the test statistics and how to interpret the results of the different tests (Kolmogorov–Smirnov, Shapiro–Wilk, and Cronbach's alpha) in R.

## References

Carpentier, M., Combescure, C., Merlini, L., & Perneger, T. V. (2017). Kappa statistic to measure agreement beyond chance in free-response assessments. *BMC Medical Research Methodology, 17*(1), 1–8. https://doi.org/10.1186/S12874-017-0340-6

de Barros Ahrens, R., da Silva Lirani, L., & de Francisco, A. C. (2020). Construct validity and reliability of the work environment assessment instrument WE-10. *International Journal of Environmental Research and Public Health, 17*(20), 7364. https://doi.org/10.3390/ijerph17207364

Elliott, A. C., & Woodward, W. A. (2007). *Statistical analysis quick reference guidebook with SPSS examples* (1st ed.). Sage Publications.

Ghasemi, A., & Zahediasl, S. (2012). Normality tests for statistical analysis: A guide for non-statisticians. *International Journal of Endocrinology and Metabolism, 10*(2), 486–489. https://doi.org/10.5812/ijem.3505

Goni, M. D., Naing, N. N., Hasan, H., Wan-Arfah, N., Deris, Z. Z., Arifin, W. N., Hussin, T. M. A. R., Abdulrahman, A. S., Baaba, A. A., & Arshad, M. R. (2020). Development and validation of knowledge, attitude and practice questionnaire for prevention of respiratory tract infections among Malaysian Hajj pilgrims. *BMC Public Health, 20*(1), 1–10. https://doi.org/10.1186/s12889-020-8269-9

Isnainiyah, I. N., Yulnelly, Y., & Balqis, A. N. (2019). Usability analysis using principal component analysis (PCA) method for online fish auction application. In *Proceedings–1st international conference on informatics, multimedia, cyber and information system, ICIMCIS 2019* (pp. 231–236). https://doi.org/10.1109/ICIMCIS48181.2019.8985225

Mishra, P., Pandey, C. M., Singh, U., Gupta, A., Sahu, C., & Keshri, A. (2019). Descriptive statistics and normality tests for statistical data. *Annals of Cardiac Anaesthesia, 22*(1), 67–72. https://doi.org/10.4103/aca.ACA_157_18

Öztuna, D., Elhan, A. H., & Tüccar, E. (2006). Investigation of four different normality tests in terms of type 1 error rate and power under different distributions. *Turkish Journal of Medical Sciences, 36*(3). The Scientific and Technological Research Council of Turkey.

Pallant, J. (2007). *SPSS survival manual, a step by step guide to data analysis using SPSS for windows* (3rd ed.). McGraw Hill.

Peat, J., & Barton, B. (2005). *Medical statistics: A guide to data analysis and critical appraisal*. Wiley.

Roscoe, J. T. (1975). *Fundamental research statistics for the behavioral sciences* (2nd ed.). Holt, Rinehart, and Winston.

Rstudio. (2023). *RStudio–Statistics*. https://rstudio.com/products/rstudio/

Taber, K. S. (2018). The use of Cronbach's Alpha when developing and reporting research instruments in science education. *Research in Science Education, 48*(6), 1273–1296. https://doi.org/10.1007/s11165-016-9602-2

Tavakol, M., & Dennick, R. (2011). Making sense of Cronbach's alpha. *International journal of medical education, 2*, 53–55. https://doi.org/10.5116/ijme.4dfb.8dfd

Thode, H. C. (2019). *Testing for normality*. CRC Press, Taylor and Francis group.

Vaclavik, M., Sikorova, Z., & Barot, T. (2020). *Skewness in applied analysis of normality* (pp. 927–937). Springer, Cham. https://doi.org/10.1007/978-3-030-63319-6_86

# Chapter 4
# Choosing Between Parametric and Non-parametric Tests in Statistical Data Analysis

## 4.1 Introduction

Statistical data analysis and hypothesis testing in any type of research or data analysis usually fall under two likely categories: *parametric* and *non-parametric* tests. In scientific research, statistical methods are mainly used to conduct quantitative research or analysis. Thus, the *hypothesis* testing (be it parametric or non-parametric) is primarily designed and used to describe and interpret the "assumptions" that underlie the adopted statistical methods. In Chap. 6, the authors will expand on the different types of Statistical Methods and Analysis the researchers can perform particularly taking into account the parametric and non-parametric tests discussed in this current chapter.

The decision to conduct parametric or non-parametric analysis largely depends on the type of data (see Chap. 5) the researchers intend to investigate or analyze (Stojanović et al., 2018). The different statistical approaches or procedures are followed based on the type of the available dataset (nominal, ordinal, continuous, discrete, number of independent versus dependent variables, etc.). Hopkins et al. (2018) note that when researchers collect data for hypotheses testing and the investigations to follow; they often refer to the underlying proportions, means, medians, or standard deviations, etc. as "parameters" to describe the general population in question. As a result, the researchers tend to predetermine those "parameters" from the collected sample (i.e., drawn from the population) by calculating the sample estimates or quantification, otherwise referred to as "statistics". Thus, statistical methods or analysis are applied to estimate the parameters (Hopkins et al., 2018). Likewise, Stojanović et al. (2018) note that making the right choice as to which statistical method or test to apply for the research strongly influences the extent and level of the data interpretation and impact.

## 4.2 Parametric Versus Non-parametric Tests

In this section, the authors focus on describing in detail what parametric and non-parametric tests are, and when to choose between the two types of test in research experiments or hypothesis testing.

### *4.2.1 Parametric Test*

Parametric tests are fundamentally used to make assumptions based on the *distributions* that underlie the available data the researchers or analysts want to analyze (Hopkins et al., 2018; Turner et al., 2020). With the parametric tests, it is assumed that several conditions which are used to measure the validity and reliability of the test and results must be met. According to Turner et al. (2020), the term "parametric" refers to parameters of the resultant data (distribution) which assumes that the sample (mean, standard deviations, etc.) is normally distributed as illustrated in Fig. 4.1a.

In the other setting (non-parametric), which the authors discuss in detail in the next section of this chapter (Sect. 4.2.2)—the data samples or population can appear to be not normally distributed (see: Fig. 4.1b and c), which in turn, implies the

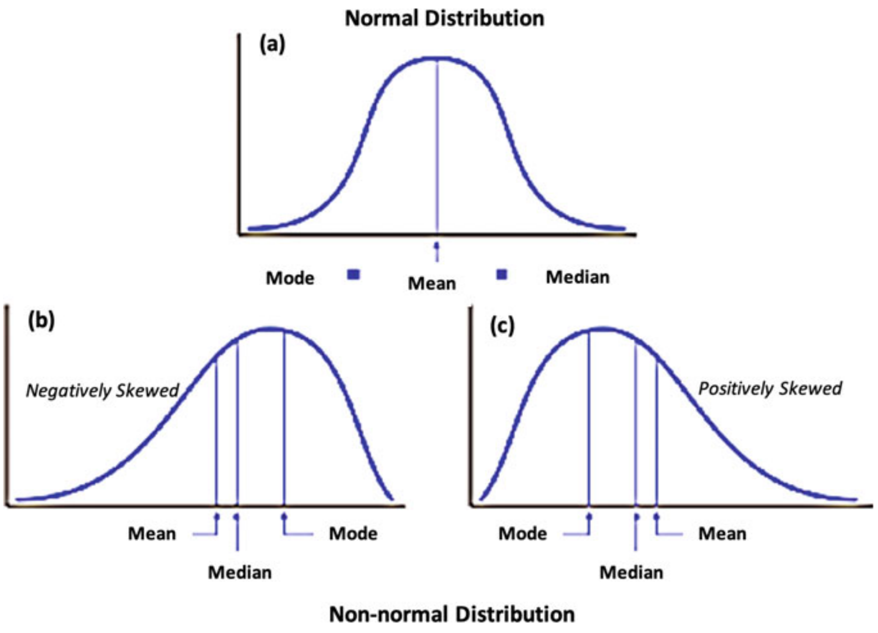


**Fig. 4.1** Normal versus non-normal data distribution, adapted from Antonopoulos and Kakisis (2019)

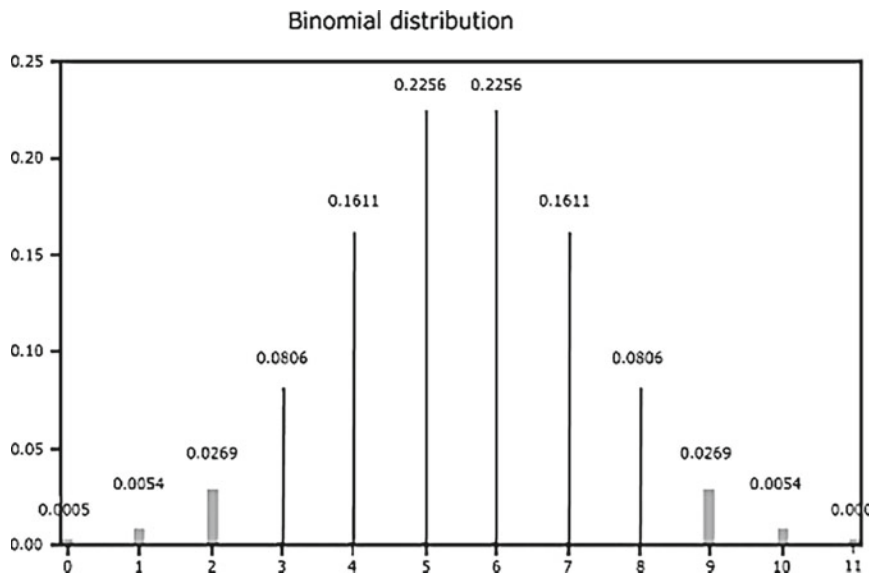


**Fig. 4.2** Binomial distribution sample example, adapted from ScienceDirect (2019)

application of statistical methods that support those types of data (otherwise referred to as non-parametric procedures).

By definition, the term parametric can be referred to methods or procedures that assume specific types of distributions such as: the Binomial or Poisson distributions (Turner et al., 2020), as shown in Figs. 4.1a and 4.2.

In theory, as gathered in Figs. 4.1a and 4.2; *parametric tests* are based on the presupposition that the analyzed or investigated datasets follow a normal "bell-shaped curve" distribution of values (Antonopoulos & Kakisis, 2019; ScienceDirect, 2019) often allied to the *central limit theorem* (Stojanović et al., 2018). According to the ScienceDirect (2019) source or topic on the parametric tests, a graphical representation of data drawn from a particular group of population (i.e., studied phenomenon) that appears to be closely or normally distributed will always come out as a typical bell-shaped curve (referred to as Gaussian distribution) in contrast to the *non-parametric* datasets that tend to be skewed, lumpy, with gaps scattered about, or having a few warts and outliers.

In a nutshell, the standard conditions (or assumptions) that underlie the parametric tests are outlined as follows (Rana et al., 2016):

- The observations must be independent. The sampled data should not be associated to any factor that can potentially affect the outcome of the analysis.
- The observations should (necessarily) be drawn from a normally distributed population.
- The sample data is better represented by the *mean* or standard deviation.

- The captured datasets should essentially be measured or represented on an interval or ratio scale.

### 4.2.2 Non-parametric Tests

Non-parametric tests are referred to as "distribution-free" statistical tests given the fact that the supporting methods assume that the readily available datasets follow a certain but not specified distribution (De Canditiis, 2019; Minitab, 2015; Rana et al., 2016; Turner et al., 2020). The non-parametric tests are mostly applied to samples which are represented as nominal or ordinal data (Rana et al., 2016). Although, the test can also be conducted on interval and ratio datasets provided the data sample(s) in question do not follow a normal distribution. Therefore, a majority of the non-parametric tests or methods can handle ordinal data, ranked data, etc., without being utterly affected by outliers (Derrick et al., 2020; Winthrop, 2019). The tests (non-parametric) are applied with the emphasis that it does not require or demand for any given (specified) condition(s) to be met particularly with regards to the parameters of the population from which the sample is drawn.

As illustrated earlier in Figs. 4.1b and c part, unlike the parametric tests (that are better represented in mean or standard deviations—see Fig. 4.1a), the non-parametric tests are most adequate or suitable when the said datasets in question are better represented by median (see: Fig. 4.1b and c).

Furthermore, as gathered in Figs. 4.1b and c, non-parametric tests are grounded on the presupposition that the investigated or scrutinized datasets show a non-normal "skewed or uneven curve" distribution of values (Antonopoulos & Kakisis, 2019; ScienceDirect, 2019).

In short definition, the authors outline the various common conditions (assumptions) that constitute the non-parametric tests as follows:

- When the analysis does not require any rigorous (stringent) assumptions to be met for the test or hypothesis testing process to follow.
- The observations should necessarily be drawn from a *non-normally distributed population*.
- The sampled data is better represented by the *median.*
- The captured datasets should essentially be measured on a *nominal* or *ordinal* scale. Although, in some settings, the methods (non-parametric) support the interval and ratio data provided they are non-normally distributed.

In summary, the *non-parametric* tests are alternative to the *parametric* tests, especially for small sample sizes, whereby there is the presence of extreme asymmetries, skewness, and/or multimodality (Stojanović et al., 2018; Woodrow, 2014).

## 4.3 Choosing Between Parametric and Non-parametric Test

"…the type of dataset one has ready for the research or hypothesis testing will essentially determine the type of statistical method or data analysis that is applicable or fitting for the research…" (see Fig. 4.3).

When statistically comparing the causes (based on the input variables) and effects (output variables) between different groups (sample composition) in a dataset, researchers have to choose whether to use the parametric or non-parametric statistical methods (le Cessie et al., 2020). For example, the suitability of a particular statistical method when investigating the difference(s) between variables (e.g., nominal, ordinal, continuous, discrete, etc.) among different groups (e.g., independent or dependent) would necessarily depend on the distribution of the variables (e.g., normal versus non-normal) (see: Table 4.1).

In summary, when choosing the type of statistical method to apply for the research investigation (parametric versus non-parametric), one should consider among the many factors the following elements (Rana et al., 2016):

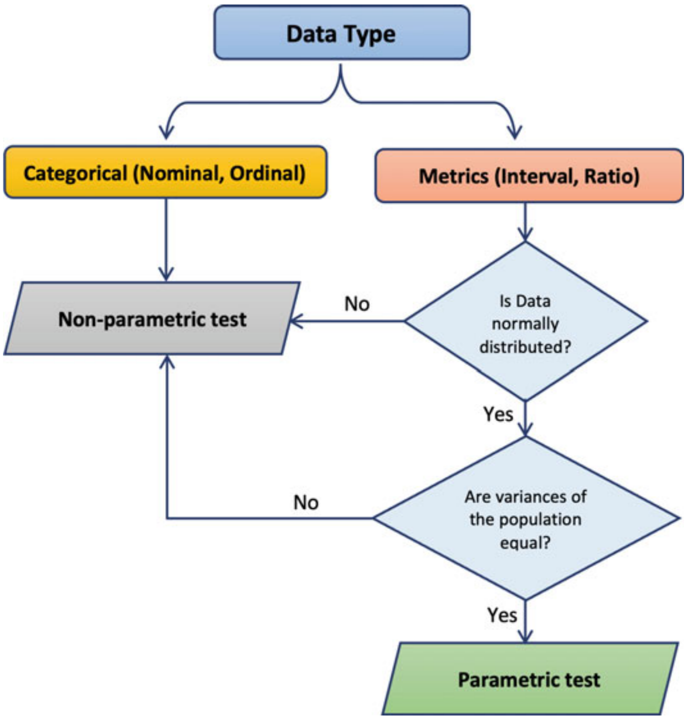

Data Type
Categorical (Nominal, Ordinal)
Metrics (Interval, Ratio)
Non-parametric test
No
Is Data normally distributed?
Yes
No
Are variances of the population equal?
Yes
Parametric test

**Fig. 4.3** Choosing between parametric and non-parametric tests

**Table 4.1** Parametric tests versus the non-parametric equivalents table based on the data distribution, type of variable and correlation, number of groups, and independent versus dependent variables

| Statistical test | Data distribution | Level of measurement | Correlation | Sample composition (groups and independence between groups) | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | 1 group | 2 groups | | K groups (>2) | |
| | | | | | Independent | Dependent | Independent | Dependent |
| Parametric | Normal | Interval or ratio | Pearson’s cor test | *t*-test or Z-test | Independent sample *t*-test | Paired sample *t*-test | One-way ANOVA | Repeated measure ANOVA |
| Non-parametric | Non-normal | Nominal (categorical) | | Chi-squared test ($X^2$) | Chi-squared test ($X^2$) | McNemar´s test | Chi-squared test ($X^2$) | Cochran´s Q test |
| | | Ordinal or rank | Kendall’s tau or Spearman´s rho test | Chi-squared test ($X^2$) | Mann–Whitney Utest | Wilcoxon signed-rank test | Kruskal–Wallis H test | Friedman´s ANOVA |

- Scale of measurement of the data, e.g., nominal, ordinal, interval, ratio, continuous, discrete, etc.
- Distribution of the population, e.g., normal versus non-normal
- Homogeneity of variances, e.g., equal versus unequal variances
- Independence versus dependency of the drawn samples considering the variables.

### 4.3.1 Types of Parametric Versus Non-parametric Tests in Statistical Analysis

It is paramount to decide between the *parametric* and *non-parametric* tests, including all the assumptions associated with the different methods (as outlined in the previous section–Sect. 4.2 and Table 4.1) when choosing one above the other.

Here in this section of the chapter, the authors outline and illustrate some of the most commonly used parametric tests in the available or current literature and their non-parametric equivalents (Table 4.1), with some examples of use case scenarios of each test presented in Sect. 4.3.2.

In Table 4.1, the authors provided a list of the different parametric and non-parametric tests, and the necessary conditions under which one should perform each test.

### 4.3.2 Examples and Use Case Scenarios: Parametric Versus Non-parametric Tests

Following the guideline listed in Table 4.1, the authors explain the different types of statistical tests using examples of some typical research scenarios:

**Parametric test (for Normally distributed data,** Interval or Ratio, Large data sample size)**:**

- **Pearson's Cor test** (*Correlation*): determine if student's grades (dependent variable) are increased or tend to increase in proportion to their study time (independent variable).
- ***t*-test** (*One sample group*): determine the *Mean* of the age of all students within a particular school or department.
- **Independent sample *t*-test** (*Two groups of Independent variables):* determine the *Mean* of all students within a particular school or department who are undergoing a specific teaching methodology (e.g., Hybrid model versus Traditional model of teaching).
- **Paired sample *t*-test** (*Two groups of Dependent variables*): determine the difference in the grades of students *before* (pre) versus *after* (post) undergoing the hybrid model of teaching.

- **One-way ANOVA** (*More than 2 groups (i.e., K > 2) of Independent variables*): determine the *Mean* of all students within a particular school or department who are undergoing either the Hybrid model versus Traditional model versus Both (Hybrid and Traditional).
- **Repeated measure ANOVA** (*More than 2 groups (i.e., K > 2) of Dependent variables*): determine the Difference in the grades of students before versus after 1st semester versus after 2nd semester of undergoing the Hybrid model of teaching.

**Non-parametric test (for Non-normally or distribution-free data, Nominal or Ordinal data, Small data sample size):**

- **Kendall´s tau or Spearman´s rho test** (*Correlation*): determine if student's grades (dependent variable) are increased in proportion to the teaching methodology adopted by the teachers.
- **Chi-squared test ($X^2$)** (*One sample group, 2 groups, and K groups (>2) of Independent variables*): determine the Differences between the observed and expected values or scores in male versus female students´ grade undergoing either the Hybrid model versus Traditional teaching model.
- **Mann–Whitney U test** (*Two groups of Independent variables, ordinal data*): determine the *Median* of all students within a particular school or department who are either undergoing the Hybrid model versus Traditional model of teaching.
- **Wilcoxon signed-rank test** (*Two groups of Dependent ranked variables*): determine the Difference in the grades of students before versus after undergoing the Hybrid model of teaching.
- **Mc-Nemar´s test** (*Two groups of Dependent nominal or categorical variables*): determine the Difference in number of students with good grades versus poor grades after 1st semester versus after 2nd semester of undergoing the hybrid model of teaching.
- **Kruskal–Wallis H test** (*More than 2 groups (i.e., K > 2) of Independent ordinal variables*): determine the *Median* of all students within a particular school or department who are undergoing the hybrid model versus Traditional model versus Both (Hybrid and Traditional).
- **Friedman´s ANOVA** (*K groups i.e., > 2 of Dependent variables*): determine the Difference in the grades of students before versus after 1st semester versus after 2nd semester of undergoing the Hybrid model of teaching.

***Note: the authors have provided a detailed/in-depth description of the different types of statistical data analysis and methods in Chap. 6 "Understanding the Different Types of Statistical Data Analyzes and Methods" (see Chap. 6).

**Table 4.2** Difference between parametric versus non-parametric test

| Parametric | Non-parametric |
|---|---|
| Existence of specific assumption(s) being made about the population in question | There is no specific assumption(s) with regard to the population in question |
| Information about the studied population is known | There is no available information about the studied population |
| Null hypotheses are developed based on the parameters or distribution of the population | The Null hypotheses are developed independent of the parameters or distribution of the population |
| The tests (parametric) apply to measuring just the variables and not their attributes | The tests (non-parametric) are applicable to both the variables and their attributes, e.g., gender, marital status, etc |
| The outcomes of the tests are more powerful and convincing when applicable | The outcomes of the test are less powerful than the parametric tests |
| The statistical tests and methods are grounded on the distribution of the available datasets | The statistical tests and methods are arbitrary |
| Cannot be applied for nominal data types, only interval or ratio | Can be applied for nominal and ordinal data types |
| Mostly used to measure Mean and Standard deviations | Mostly used to measure MEDIANS |
| Sampled datasets show a "bell-shaped" curve when graphically represented | Sampled datasets show a "skewed or lumpy" curve when graphically represented |

## 4.4 Differences Between Parametric Versus Non-parametric Tests

In this section, the authors outline some of the main differences between the Parametric and Non-parametric tests (Table 4.2):

## 4.5 Advantages and Disadvantages of Parametric Versus Non-parametric Tests

Some advantages and disadvantages of adopting or applying each of the tests (parametric versus non-parametric) are provided in Table 4.3 below.

## 4.6 Summary

As a general rule of thumb in statistics, in situations whereby the *variables* are represented or measured on a continuous (or metric) scale, the "parametric tests" should be applied or conducted.

**Table 4.3** Advantages and disadvantages of using the parametric versus non-parametric tests

| | Advantages (pro) | Disadvantages (cons) |
|---|---|---|
| Parametric | Ensures that all components (population, parameters, assumptions) are compatible with one another<br>Most useful when determining variations between groups of variables<br>Can be easily applied and less complicated than the non-parametric methods<br>Given that real information regarding the population is known, the confidence intervals are guaranteed<br>Has more statistical power than other tests such as the non-parametric | Mostly used only for quantitative datasets<br>Data has to follow an approximate interval (normal) for the test to be applied<br>Results may not be valid when it comes to small datasets or sample size<br>They are not effective for ranked data or samples with outliers<br>Since parametric tests are conducted based on pre-defined assumptions, it consequentially, allows one to make generalizations from a sample to the studied population |
| Non-parametric | Simple and easy to understand<br>Methods are applicable for datasets with attributes, e.g., gender, marital status, etc<br>Complicated sampling theory is not a problem<br>No assumptions are made about the population<br>Supports nominal data scales | In statistical settings where parametric alternative is applicable, the non-parametric tests are less powerful<br>Supported methods are less effective than the parametric tests in drawing reliable conclusions<br>No current method for analyzing the association of variances in the underlying models |

On the other hand, if the variable(s) are represented in nominal or ordinal (categorical) scale of measurement, the "non-parametric tests" should be considered or applied.

Furthermore, the parametric tests assume that the analyzed data or sample distribution is normally distributed (i.e., bell-shaped) and consists of related parameters in the population distribution by *mean* or *standard deviations*.

On the other hand, the non-parametric tests make no assumption(s) about the shape (which are often skewed) or parameters of the population distribution by the *median* (Hopkins et al., 2018).

Statistically, the parametric tests are generally more powerful than the non-parametric methods.

Different methods can be used to determine if the sampled datasets for the research are derivative from a normally distributed population. The most frequently used methods in the current literature are the Kolmogorov–Smirnov test, the Anderson–Darling test, and the Shapiro–Wilk test (Antonopoulos & Kakisis, 2019; LaMorte, 2017; Stojanović et al., 2018), as discussed earlier by the authors in Chap. 3. According to LaMorte (2017), each test for normality of a data or sample is essentially a goodness of fit test and consequentially compares the said dataset to quantiles of the normal and/or specified distribution. Thus, the default (null) hypothesis for each test is represented as $H_0$; whereby the data is assumed to follow a normal distribution in comparison to the alternative hypothesis ($H_1$) in which the data are assumed or may in reality not follow a normal distribution. In return, if the test turns out to be statistically significant, i.e., $p < 0.05$; with the value (statistics) of the normality test less than 0.5, then the readily available data is said to not follow a normal distribution (and as described earlier in this chapter, a non-parametric test would be best decided) (LaMorte, 2017; Stojanović et al., 2018). Thus, with the normality results (test statistics):

- When $p < 0.5$, then non-parametric test is best conducted.
- When $p > 0.5$, then parametric test is best conducted.

Technically, there are many existing statistical tools that can be used to determine if the researchers' datasets or samples are normally distributed, and consequently, perform either the parametric or the non-parametric procedures. Among the many existing tools includes statistical packages such as: R (Rstudio, 2023), SPSS Statistics (IBM, 2023), STATA (Stata.com ©, 2023), SAS (Sas.com ©, 2023), Minitab (Minitab.com ©, 2023), MATLAB (MATLAB, 2023), Python (Python, 2023), etc.

## References

Antonopoulos, C., & Kakisis, J. (2019). Applied statistics in vascular surgery Part 1: Choosing between parametric and non-parametric tests. https://www.heljves.com

De Canditiis, D. (2019). Statistical inference techniques. In *Encyclopedia of bioinformatics and computational biology: ABC of bioinformatics* (vols 1–3, pp. 698–705). Elsevier. https://doi.org/10.1016/B978-0-12-809633-8.20357-9

Derrick, B., White, P., & Toher, D. (2020). Parametric and non-parametric tests for the comparison of two samples which both include paired and unpaired observations. *Journal of Modern Applied Statistical Journal of Modern Applied Statistical Methods Methods*, *18*. https://doi.org/10.22237/jmasm/1556669520

Hopkins, S., Dettori, J. R., & Chapman, J. R. (2018). Parametric and nonparametric tests in spine research: Why do they matter? In *Global spine journal* (vol. 8, Issue 6, pp. 652–654). SAGE Publications Ltd. https://doi.org/10.1177/2192568218782679

IBM. (2023). *SPSS statistics–overview*. IBM. https://www.ibm.com/products/spss-statistics

LaMorte, W. W. (2017). *When to use a nonparametric test*. Boston University School of Public Health. http://sphweb.bumc.bu.edu/otlt/MPH-Modules/BS/BS704_Nonparametric/BS704_Nonparametric2.html

le Cessie, S., Goeman, J. J., & Dekkers, O. M. (2020). Who is afraid of non-normal data? Choosing between parametric and non-parametric tests. *European Journal of Endocrinology, 182*(2), E1–E3. NLM (Medline). https://doi.org/10.1530/EJE-19-0922

MATLAB. (2023). MATLAB. https://www.mathworks.com/

Minitab. (2015). *Choosing between a nonparametric test and a parametric test*. https://blog.minitab.com/blog/adventures-in-statistics-2/choosing-between-a-nonparametric-test-and-a-parametric-test

Minitab.com ©. (2023). Statistical & data analysis software package. Minitab. https://www.minitab.com/en-us/products/minitab/

Python. (2023). *Python*. https://www.python.org/

Rana, R., Singhal, R., & Dua, P. (2016). Deciphering the dilemma of parametric and nonparametric tests. *Journal of the Practice of Cardiovascular Sciences, 2*(2), 95. https://doi.org/10.4103/2395-5414.191521

Rstudio (2023). *RStudio. RStudio.* https://rstudio.com/products/rstudio/

Sas.com ©. (2023). *SAS: Analytics, artificial intelligence and data management*. SAS. https://www.sas.com/en_us/home.html

ScienceDirect. (2019). Parametric test–an overview. In *Encyclopedia of bioinformatics and computational biology*. ScienceDirect Topics. https://www.sciencedirect.com/topics/medicine-and-dentistry/parametric-test

Stata.com ©. (2023). *Stata: Software for statistics and data science*. https://www.stata.com/

Stojanović, M., Andjelković-Apostolović, M., Milošević, Z., Ignjatović, A. (2018). Parametric versus nonparametric tests in biomedical research. *Acta Medica, 75*. https://doi.org/10.5633/amm.2018.0212

Turner, R., Samaranayaka, A., & Cameron, C. (2020). Parametric versus nonparametric statistical methods: which is better, and why? In *New Zealand Medical*. https://nzmsj.scholasticahq.com/article/12577.pdf

Winthrop. (2019). *Parametric versus non-parametric statistical tests.* Department of Biostatistics, Winthrop University Hospital. https://nyuwinthrop.org/wp-content/uploads/2019/08/Parametric-non-parametric-tests.pdf

Woodrow, L. (2014). Writing about quantitative research in applied linguistics. In *Writing about quantitative research in applied linguistics*. Palgrave Macmillan. https://doi.org/10.1057/9780230369955

# Chapter 5
# Understanding Dependent and Independent Variables in Research Experiments and Hypothesis Testing

## 5.1 What Are Variables in Scientific Research?

*Variables* represent any quantifiable or measurable attributes in a given dataset. In theory, variable can be used to represent anything; ranging from some kind of phenomenon or entity one is trying to measure, to empirical study of events, ideas, subjects and objects or even time (Sarikas, 2020).

There are different ways in which variables are defined depending on the context it is used or applied.

In scientific research, "variable" can be defined as a measurable property or traits that changes (varies) or are affected as a result of the change across the experiment or hypothesis when testing. Perhaps, such properties or changes are captured regardless of whether the scientists or researchers are comparing the outcomes or relationships that exist among multiple groups of objects, multiple persons, or a single entity in an experimentation, e.g., performed over a period of time (Agravante, 2018).

In mathematics, the opposite of "variable" is referred to as "constant". Therefore, we assume that a variable is an entity or quantifiable object with an *unknown value*, thus, the concept for which its application for scientific research is drawn.

Researchers often choose some kind of letters (annotations) to remind them of the quantities or parameters that are being measured or they are measuring (statistically represented). For example, in an equation, graph or linear regression model in which the "y"-axis or parameters can be defined as a function of "x", i.e., f(x). It means that the value of y is dependent upon the value of x. Thus "x" and "y" values are described as the "variables". In short definition, the term *variable* implies that the represented values or parameters can change over the course of a scientific experiment or hypothesis testing.

Logically, variables and the associated attributes can also be allied to the concept of necessary condition analysis (NCA) (Dul, 2016). Dul (2016) emphasized on the logic and methodology of "necessary-but-not-sufficient" conditions to define the

K. Okoye and S. Hosseini, *R Programming*,
https://doi.org/10.1007/978-981-97-3385-9_5

various processes or factors (e.g., measures of occurrences, features or characteristics of objects, etc.) that can be used for hypothesis testing, thus, leading to some sort of expected outcomes (conclusions). In essence, when defining the concept of NCA in relation to scientific research and experiments, the authors note that a necessary determinant (i.e., variables) must be present for achieving an outcome (or drawing conclusions). Although Dul (2016) states that the presence of the determinants (variables) is not always sufficient to prove that outcome, but instead to suggest or support the conclusions. In other words, variables for research purposes can only be used to draw causal inferences. More importantly, the validity of the resultant inferences or conclusion is grounded on the level of adequacy of the proposed or supported theories, quality of the measurement and analysis process, the statistical data analysis method or hypothesis testing, and SMART research design adopted by the researchers or scientists (Dul, 2016; Stone-Romero, 2002; Tynan et al., 2020).

Despite the fact that the necessary condition analysis (Tynan et al, 2020) is a theoretical method of its own in literature, it not only allows us to relatedly perform hypothesis testing through the underlying logic (indication) of *necessity of X for Y* in any statistical or rule-based analysis (e.g., IF – THEN statement or Association Rule Learning) (Okoye et al., 2014), but also the logic (NCA) can be grasped as an effective and efficient way to find relevant variables that must be leveraged or analyzed to support the occurring conclusions or hypothesis acceptance or rejection. Thus, variables are normally, in theory, represented as a group of items with several attributes or characteristics that can be combined (e.g., by adding or averaging) in order to obtain a concluding score for the said variables of interest or scrutiny (Dul, 2016).

Likewise, Tynan et al. (2020) note that although the NCA logic (in carrying out research experiments and hypothesis testing) enables the researchers to determine if the observed relations or association between the variables (e.g., independent versus dependent) (see: Sect. 5.1.1.1 and 5.1.1.2) are consistent with the "necessary-but-not-sufficient relation or condition" rule. It states that such condition suggests that the results of the experimentations or hypothesis testing are only probable—but not guaranteed—given the high levels or reliability of the considered variables (Tynan et al., 2020). Moreover, the outcomes may sometimes consequently lead to re-consideration (re-examination) of other variables that may have been apriori dismissed or considered as being irrelevant in the analysis process.

Theoretically, there are different types of variables, considering the diverse settings or context in which it is being used or applied. We note some of the examples to include: independent and dependent variables, intervening and moderator variables, constant or controllable variables, extraneous or predictor variables, etc. (Agravante, 2018). For all intents and purposes, this book focuses on the most frequently named or used variable when conducting the scientific research and data analytics as follows:

- Independent variables, and
- Dependent variables.

### *5.1.1 Types of Variables in Scientific Research*

Deciding if a variable influences other variable(s) is one of the main challenges or components of conducting some scientific research particularly the social science. This is owing to the fact that by establishing the effects or assertions from the examined variable(s), the investigators are able to accept or reject a hypothesis, and, in turn, create a new knowledge about the phenomenon that is being studied.

In theory, *variable*s can be either dependent or independent based on how the researcher(s) design or sets up the experiments or hypothesis. It is important to note that the research design (whether qualitative or quantitative) resolves to or leads to what variables are manipulated (independent) or are consequently measured/affected (dependent) as a result of that manipulation. The authors will discuss this in detail in the following subsections.

#### 5.1.1.1 Independent Variable

*Independent variable,* otherwise referred to as the *manipulated variable(s)* represents the items or parameters being changed or manipulated by the researcher (Boyd, 2008; Shuttleworth & Wilson, 2008a, 2008b). According to Boyd (2008), an *independent variable* is expected to influence or at least be correlated with another variable (i.e., the dependent variable—see Sect. 5.1.1.2) in a data sample or scientific experiment. A lot of the time, the independent variable(s) are similarly referred to as the *controlled variable* as this is the variable(s) that are being decided (selected) or manipulated by the researchers in the experiment. In fact, an independent variable is considered “independent” because its distinctive attributes do not depend on the variation of other variables in a specific experiment or setup (Shuttleworth & Wilson, 2008a, 2008b). Hence, an “independent variable” is the variable that its value or change is not affected by other variables in an experiment.

Technically, as gathered in Fig. 5.1 (see more detail in Chap. 6); in linear regression models, *x-axis* (horizontal axis) is normally used to represent the *independent variable* in a graph or equation.

In principle, when we consider the graph (Fig. 5.1) which shows a linear relationship between x and y, the value of y is represented as a function of x “f(x)”. Meaning that y, the dependent variable (see: Sect. 5.1.1.2) is dependent upon the value of x. Consequentially, the final outcome or result of the formula can be interpreted as: y depends on the x value (i.e., the independent variable) which can be changed or changes on its own. According to Sarikas (2020), a typical example of independent variables is studies or samples representing age and time. Basically, there is nothing the researcher or any likely factors could neither do to slow or speed up the time, nor decrease or increase the age. Thus, it can be said that the variables (x), e.g., age and time, are often independent of every other variable in scientific experiments.

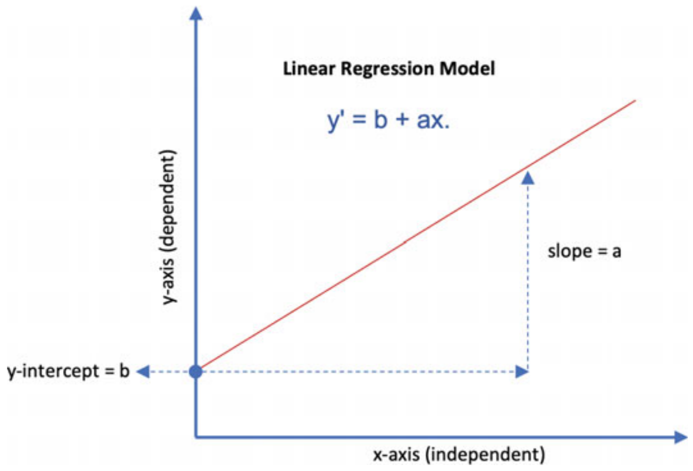


**Fig. 5.1** Graph representing the independent (x) and dependent (y) variables

### 5.1.1.2 Dependent Variables

As illustrated earlier in Fig. 5.1, *y* (i.e., vertical axis) is normally used to represent the *dependent variable* in a graph or equation. The authors highlighted in the figure (Fig. 5.1), how the independent variables (x) relates to the dependent variable (y). This is given that quite often either of the variables cannot be defined or discussed without referring to the other. Here, we explain the dependent variable in detail. Earlier on, we noted that in a typical experiment or hypothesis testing, the independent variables represent the items or parameters that are being manipulated, and, in turn, (the effects) subsequently observed or recorded. By definition, those observed or recorded effects are referred to as the *dependent variables* (Cao, 2008; Shuttleworth & Wilson, 2008a, 2008b). According to Shuttleworth and Wilson (2008a, 2008b), the *dependent variables* are often the *hypothesized consequences* (effects) as a result of manipulating the independent variable(s). Therefore, in a typical experiment, the dependent variable is assumed to respond to the independent variable. Thus, in theory, the dependent variables are also referred to as the "response variables". Cao (2008) states that the judgment to treat a variable as a "dependent" might not only mean that an independent variable predicts the said variable but also happens to cause (or effects) the dependent variable.

### 5.1.1.3 Independent Versus Dependent Variables

In the following figure (Fig. 5.2), the authors show that the *independent variable* is the variable the researchers change (or controls), and it is expected that it will have a direct effect on the dependent variable. Thus, the *dependent variable* is the variable

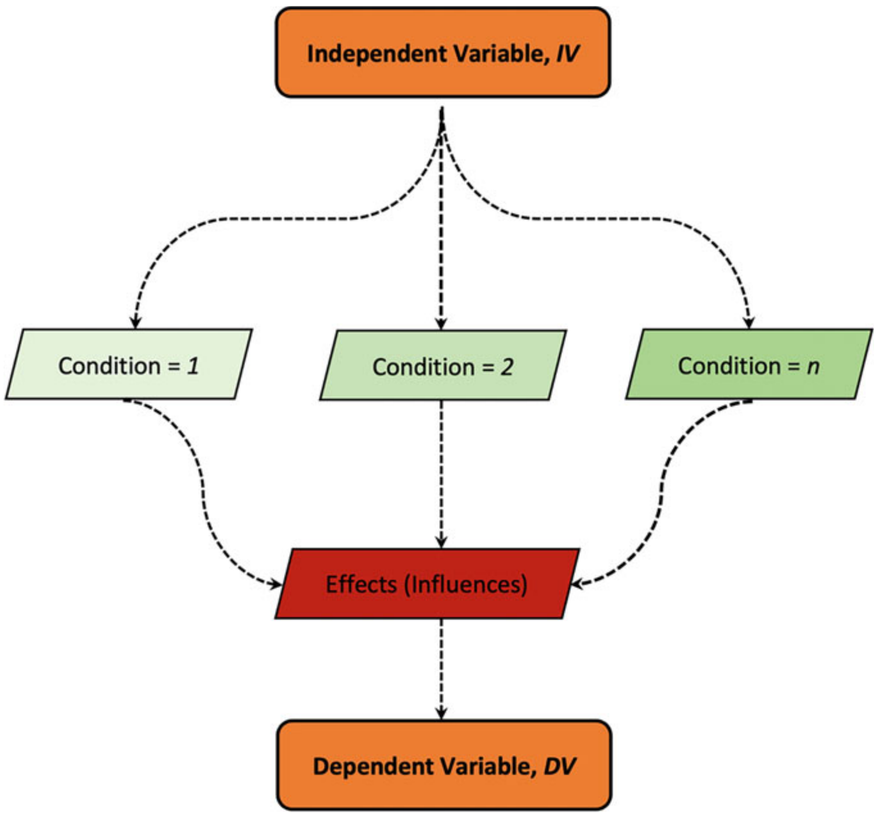


**Fig. 5.2** Relationship and condition between independent versus dependent variables

being tested (or measured) during the experimentation, hence, as the name implies is "dependent" on the independent variable (McLeod, 2019).

The illustration in Fig. 5.2 shows that the purpose of any typical research experimentation or hypothesis testing should be focused on determining possible effects (influence) that leads to the *dependent variable* (DV) which may be caused by changing or altering (conditions) the *independent variables* (IV).

Furthermore, the authors provides in Table 5.1 some of the distinctive features of the independent (IV) versus the dependent variables (DV) that may guide the work of the researchers in determining the different types or categories of variables when conducting their experiments.

In summary, the independent variable (IV) provides the "input" to the statistical test or hypothesis which is modified by the model or the adopted method(s) by the researchers to change the "output" (dependent variable, DV). Interestingly, Cao (2008) notes that to conclude whether the dependent variable (DV) is caused by the independent variable (IV); that it is important to establish the relationship between

**Table 5.1** Independent versus dependent variable

| Independent (IV) | Dependent (DV) |
|---|---|
| Manipulated variable | Response variable |
| X or horizontal axis | Y or vertical axis |
| What the researchers change or what changes on its own | What is being studied or is being measured or determined |
| Input of an experiment | Output of an experiment |
| Causes the variation | Depends on the variation |

the two variables based on some pre-defined criteria. For example, as the authors outline below:

- The two variables (independent versus dependent) must be correlated, thus, a change in one variable (mainly the independent variable) must be accompanied by a change in the other (dependent variable).
- The observed correlation between the variables (independent versus dependent) must be genuine and conform to the measures by validity. In other words, the resultant relationship cannot be explained by other variables. Although the independent variable can be one of many other factors that could influence the dependent variable.
- Causal relationships between the independent and dependent variables are typically probabilistic in nature rather than deterministic; meaning that such association will not always necessarily be true for every run test or case scenario.
- When considering the order or timing of occurrence, the dependent variable (DV) must follow the independent variable (IV). For instance, a researcher who seeks to determine how ones' level of education influences ones' level of academic performance or work production would necessarily show that changes in the latter (academic or work production) occurred after changes in the former (education level).

### *5.1.2 Examples and Use Case Scenarios of Independent Versus Dependent Variables*

In the preceding section (Sect. 5.1.1), we described the different types of variables (independent and dependent) including the distinctive features or how to identify each type in a dataset or experiments. Here, in Table 5.2, the authors look at some examples or use case scenarios that can guide the researchers or statisticians in defining or establishing the differences between the two variables (IV versus DV) especially in the different research settings or domains that they (researchers, statisticians, data analyst) undergo.

**Table 5.2** Use case scenarios for identifying independent versus dependent variables

| Research scenarios | Independent variable (IV) | Dependent variable (DV) |
|---|---|---|
| Study that focuses on determining the impact of a particular teaching/learning method | • The teaching/learning method | • Impact of the method |
| Study that explores how much resources is invested to acquire some certain books for a library | • How many books were acquired | • Resources invested |
| An experiment that tests if the changes in the location of a certain work environment affect the outputs or productivity | • Change in locations | • Whether the productivity is affected or not |
| Sports experiment to determine if the number of hours spent training improves the athlete's performance | • Number of hours spent training | • Athletes performance |
| A researcher sets out to determine which type of educational model mostly supports the students in meeting their learning goals | • The types of educational model | • Level of support or meeting the students' learning goal |
| Medical experiment that focuses on determining how increase in the temperature of patients affects their response to treatment | • Patient's temperature | • Response to treatment |
| Agricultural research that seeks to establish how different factors (e.g., sunlight, water, etc.) affects the growth of certain plants | • The different factors, sunlight, water, etc | • How tall the plants grow |
| A social science research that investigates how the level of ones' education affects ones' maturity or level of thought | • Level of education | • Level of thought |

## 5.2 Summary

The relationship between the *independent* (IV) and *dependent* (DV) variables is the key foundation of most statistical data analysis or scientific tests. An ample understanding or identification of the independent versus dependent variables is paramount to having a good knowledge about the outcome or impact of the scientific research or how the experimentations are being done/conducted. In typical scientific research, the researchers can establish whether there is a significant correlation between the two variables (IV versus DV). In turn, the outcome of the procedures or methods (tests) enables the investigators to draw conclusions by either accepting or rejecting the pre-defined research hypothesis.

Quite often, it can be anything from really cumbersome or easy to identify the independent (IV) and dependent (DV) variables for any research study or data sample. According to Sarikas (2020), an easy way to identify the independent or dependent variable during an experiment is: independent variables (IV) are what the researchers change or changes on its own, whereas dependent variables (DV) are what changes as a result of the change in the independent variable (IV). Thus, independent variables (IV) are the *cause* while dependent variables (DV) are the *effect*.

Finally, it is important to remember that while some studies are likely to have one dependent variable (DV) and one independent variable (IV), it is also possible to have several of each type of the variables, i.e, more than one independent or dependent variables, in an experiment, as we will illustrate in Chapter 6 with some examples. Researchers might also want to learn or explore how variables in a single independent variable or factor affect several distinct dependent variables, or vice and versa (Cherry, 2019).

## References

Agravante, M. (2018). *What Is the Meaning of Variables in Research?* https://sciencing.com/meaning-variables-research-6164255.html.

Boyd, H. H. (2008). *Independent Variable In: Encyclopedia of Survey Research Methods*. https://doi.org/10.4135/9781412963947.

Cao, X. (2008). *Dependent Variable In: Encyclopedia of Survey Research Methods*. https://doi.org/10.4135/9781412963947.

Cherry, K. (2019). *Dependent Variable in Experiments*. https://www.verywellmind.com/what-is-a-dependent-variable-2795099.

Dul, J. (2016). *Necessary Condition Analysis (NCA): Logic and Methodology of "Necessary but Not Sufficient" Causality*, *19*(1), 10–52. https://doi.org/10.1177/1094428115584005.

McLeod, S. (2019). *What are Independent and Dependent Variables?* Simply Psychology. https://www.simplypsychology.org/variables.html.

Okoye, K., Tawil, A. R. H., Naeem, U., Bashroush, R., & Lamine, E. (2014). A semantic rule-based approach supported by process mining for personalised adaptive learning. *Procedia Computer Science*. https://doi.org/10.1016/j.procs.2014.08.031

Sarikas, C. (2020). *Independent and Dependent Variables: Which Is Which?* https://blog.prepscholar.com/independent-and-dependent-variables.

Shuttleworth, Martyn, & Wilson, L. T. (2008). *Dependent Variable*. https://explorable.com/dependent-variable

Shuttleworth, M., & Wilson, L. T. (2008). *Independent Variable*. https://explorable.com/independent-variable.

Stone-Romero, E. F. (2002). The relative validity and usefulness of various empirical research designs. In *Handbook of research methods in industrial and organizational psychology*. (pp. 77–98). Blackwell Publishing.

Tynan, M. C., Credé, M., & Harms, P. D. (2020). Are individual characteristics and behaviors necessary-but-not-sufficient conditions for academic success ?: A demonstration of Dul ' s ( 2016 ) necessary condition analysis. *Learning and Individual Differences, 77* (December 2019), 101815. https://doi.org/10.1016/j.lindif.2019.101815.

# Chapter 6
# Understanding the Different Types of Statistical Data Analysis and Methods

## 6.1 Introduction to Statistical Data Analysis

This chapter provides the readers (e.g. researchers, data analysts, statisticians) with basic guidelines toward a comprehensive understanding of the different types of statistical data analysis and methods particularly for scientific research. This includes a description of when best to apply each particular type of analysis following our explanations of the prerequisites in statistical data analysis discussed in the previous chapters of the book (see Chaps. 4 and 5). Quite often, the researchers tend to choose the type of analysis for their work based on their expert knowledge or experience about the readily available tools and methods rather than considering the fact that in real practice, the type of analysis for any research work depends on the type of data that is collected and/or the variables being considered.

To this effect, this chapter discusses in detail the different types of statistical data analysis methods to guide the work of the researchers and data analysts when carrying out their investigations. In addition, it provides some examples of use case scenarios for each of the methods being discussed.

## 6.2 Statistical Data Analysis and Methods in Scientific Research

Some of the most frequently applied methods used to carry out the statistical data analysis and hypotheses testing in literature are discussed here. Before we look at the different types of statistical methods, it is important for the researchers at all stages, particularly during the planning and design stage of their research, to bear in mind that the *type of data analysis or method* to be used depends on the type of data collected or the research design (see Chaps. 2 and 5).

K. Okoye and S. Hosseini, *R Programming*,
https://doi.org/10.1007/978-981-97-3385-9_6

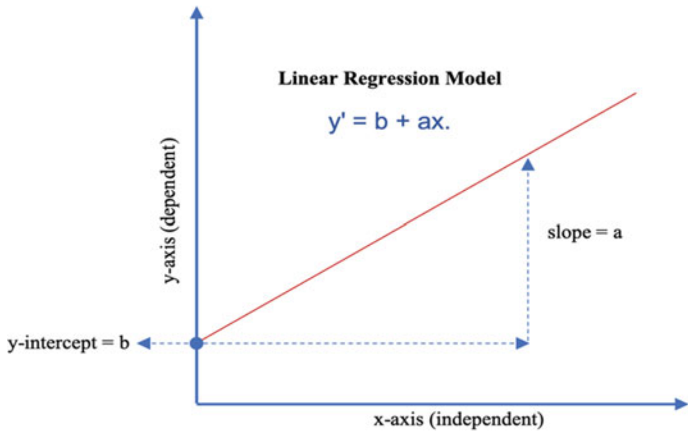


**Fig. 6.1** Linear regression graph

### 6.2.1 *Linear Regression*

Linear regression (best known type is often referred to as ordinary least square—OLS) (Zdaniuk, 2014) is a statistical method used to estimate the "relationships between variables" (Kronthaler & Zöllner, 2021). The test includes different techniques for modeling and analyzing the association between two or more variables. The linear regression is mostly applied when the focus is on the relationship between a *dependent variable* (DV) plus one or more *independent variables* (IV) (see Chap. 5).

The linear regression analysis assumes that the data points generally, but not exactly, fall along a straight line as shown in Fig. 6.1 (Montgomery et al., 2012).

**Example of Use Case Scenario for Linear Regression:**

A setting where the linear regression analysis can be applied is when a dependent variable (e.g., the student's grades) is expected to increase in proportion to their study time (independent variable).

### 6.2.2 *Logistic Regression*

The logistic regression is a type of statistical method used to predict the outcome of a *categorical variable* (dependent) as a function of the independent (predictor) variables (Connelly, 2020). The method is useful to model the probability of an event occurring as a function of other factors. The logistic test is mostly applied for machine learning and data prediction, otherwise referred to as a statistical method for measuring the likelihood of data at different intervals.

In a logistic regression test (see Fig. 6.2), separation occurs when a linear combination of the predictors can perfectly classify part or all of the observations in the

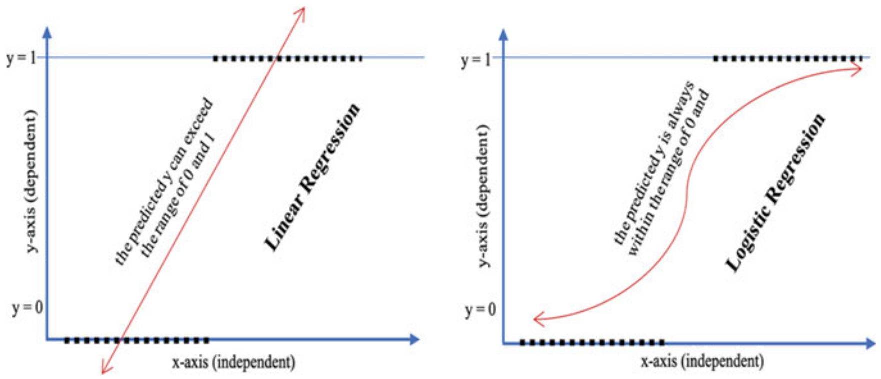


**Fig. 6.2** Graphical representation of the linear regression versus logistic regression models

sample (Ghosh et al., 2018). Consequentially, a finite maximum likelihood estimate of the regression coefficients tends to not exist.

**Linear Regression versus Logistic Regression:**

In the example shown in Fig. 6.2, it can be seen that while the linear regression analysis is used to estimate the *relationships* between variables; on the other hand, the logistic regression is very useful when *classifying* samples within a range or categories and tend to make use of different types of data to perform the classification.

The logistic regression method can also be used to determine or assess what variables are useful for classifying the data samples.

**Example of Use Case Scenario of Logistic Regression:**

Consider a dataset containing information about students who are considered to have a learning difficulty. There are some certain features such as cognitive impairment, visual impairment, mobility impairment, etc. that may be seen as the determining factors. Therefore, the data analysts or researchers' task could be to find the correlation between those listed features and their dependencies on each other.

Thus, for research purposes, the following questions can be answered using the logistic regression:

- Are the cognitive impaired students more prone to be classified as students with learning difficulty?
- What is the probability that a visually impaired student could have a relationship with students considered to have learning difficulties?
- Does mobility impairment have any impact in classifying a student as having a learning difficulty?

In essence, performing a logistic regression test using the above variables (features) will fit better to the available dataset in question. For instance, the analyst or researcher can make use of the regression (logistic) to build a predictive model for a new set of records that is capable of determining whether the students have a learning difficulty or not. However, in any case, the most important factor to note or consider in such type of analysis is the predictive accuracy of the resultant model.

### *6.2.3 Linear-Log Model*

The estimated unit change in the dependent variable (DV) for a percentage change in the independent variable (IV) can be represented or calculated through the coefficients in a linear-log model. Thus, if we use natural log values to represent the independent variables (x) and keep the dependent variable (y) in its original scale, the econometric specification is called a *linear-log model* (basically the mirror image of the *log-linear* model).

The linear-log models are typically used when the impact of the independent variable (x) on the dependent variable (y) decreases as a result of an increase in the independent variable (in contrast to the linear regression analysis). The resultant models (or linear-log models) can sometimes correct for the lack of homoscedasticity that is usually associated with the linear regression analysis, thus, it allows for heteroscedasticity in the residual distribution (Glick & Figliozzi, 2019).

For instance, when the researcher estimates a linear-log regression, a number of outcomes for the coefficient on the **x**-axis tend to produce the most likely relationships, as described in Fig. 6.3.

Where:

- Part (**a**)—Fig. 6.3 shows a linear-log function where the impact of the independent variable is positive and
- Part (**b**)—Fig. 6.3 shows a linear-log function where the impact of the independent variable is negative.

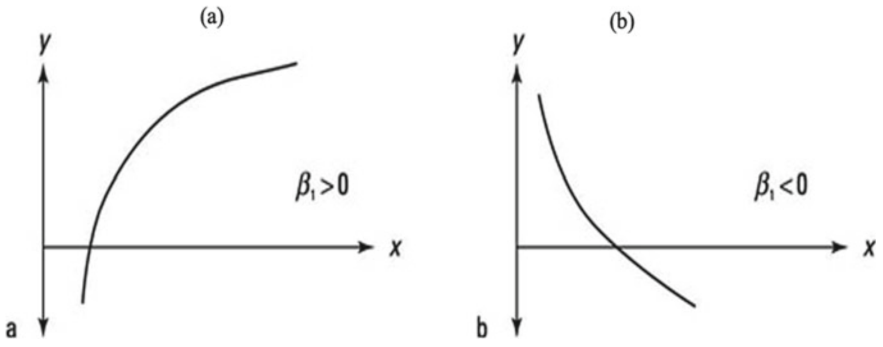


**Fig. 6.3** Linear-log model graph

### *6.2.4 T-test*

The *t-test* is a type of inferential statistic used to determine if there is a significant difference between the *means* of two groups of a variable which may be related in some certain features or characteristics (Novak, 2020). The t-test is one of the many common tests used for the purpose of hypothesis testing in statistics or research experiments. The method (t-test) can be regarded as one of the "multivariate" types of statistical analysis technique used to analyze datasets that contain more than one group of variable, and the methods are especially valuable when working with correlated variables (Chatfield, 2018; Novak, 2020).

The following formula is used to calculate the t-test:

$$\mathrm{T} = \frac{\text{Variance between Groups}}{\text{Variance within Groups}}$$

whereby

- A non-trivial (large) T-value equals to different groups.
- A trivial (small) T-value equals to similar groups.

**Example of Use Case Scenario for T-test:**

Consider a situation whereby a researcher is interested in whether men and women have different average heights. In a real-world scenario, it is not practically possible to measure the height of every man and woman across the globe. Instead, the researcher can decide to measure a selected sample of each, maybe, 500 men and 500 women in order to determine the average mean difference. The t-test will seek to determine whether that difference is probably a representative of a real difference between men and women in general, or, otherwise, whether the analysis is most likely a meaningless statistical hypothesis. Therefore, considering the scenario above one may ask: whether there were, in fact, no difference between the average heights in men and women? or focus on what are the chances that the randomly selected groups from those populations (men and women across the globe) will or will not be enough to accept the hypothesis.

**Limitations of the T-test**

- The results of inferential statistics, such as T-test, can only be applied to populations that resemble the sample in question or that is being tested.
- In T-test analysis, the sample and population are expected to be normal in distribution. Hence, most scores are often around the *mean* with fewer scores further out that may be resembling a bell curve.
- Each group in T-test is expected to have about the same number of data point or distribution. In other words, measuring large and small groups together may give inaccurate results.

- To perform a T-test, all data must be independent. Thus, the scores should not be influenced by each other.
- The datasets used to perform T-test are approximately interval level or higher. In turn, each unit of measurement is considered to be about equal to any other unit.

**How to Resolve the Shortcomings with the T-test**

- Non-parametric tests, such as the Mann–Whitney U and Kruskal–Wallis H test can perform the same type of analysis as T-test, with just the added benefit of being able to be applied with non-normal distributions and ordered-level data. Even though these tests (i.e., Mann–Whitney U and Kruskal–Wallis H test, etc.) could also be considered less powerful in some settings, depending on the type of the analysis being done.

### *6.2.5 Analysis of Variance—ANOVA (F-test)*

Researchers can make use of the ANOVA (short name for Analysis of Variance) test to determine the influence that an independent variable(s) has on the dependent variable in a regression study. With ANOVA F-test (which are regarded as one of the multivariate families of analysis), the datasets are split into two parts, namely, systematic factors and random factors (Chatfield, 2018; Christensen, 2020; Nibrad, 2019).

The analysis of variance (ANOVA) as the name implies is defined as a collection of models and their associated procedures, in which the variance is partitioned into certain components due to different explanatory variables. Similar to the T-test method, ANOVA also makes use of the *variance* between the groups and variance within the groups to calculate the ratio. Thus, the result of the ANOVA as shown in the following formula (i.e., F-statistic, also called the F-ratio) allows for the analysis of multiple groups of data to determine the variability between the samples and within samples. The researchers can make use of the ANOVA test results (F-test) to generate additional data or draw conclusions (facts) in alignment with the so-called regression models. For instance, with ANOVA tests, if there exists no real difference between the tested groups (also referred to as the null hypothesis) the result of the ANOVA's F-ratio statistic will be close to 1. Thus, the larger the ratio, the more likely that the groups are different.

The formula for **ANOVA test** is as follows:

$$\mathrm{F} = \frac{\mathrm{MST}}{\mathrm{MSE}}$$

where

| | |
|---|---|
| F | = ANOVA Coefficient, |
| MST | = Mean sum of squares due to treatment, and |
| MSE | = Mean sum of squares due to error. |

Consequently, if most of the variation (ratio) is between groups, then the researcher or data analysts can considerably claim that there is probably a significant effect. On the other hand, if most of the variation is within the groups, then there is probably not a significant effect.

Interestingly, the ANOVA analysis can be used to test for *fuzziness* in the datasets (Ahmed & Kilic, 2019). For instance, the one-way ANOVA (also referred to as a between-subject ANOVA or one-factor ANOVA) can help in determining statistical *differences between the mean* of continuous independent variables. Even though the method cannot tell which specific groups of data are significantly different from each other, rather, it just provides information that at least two of the groups are significantly different from each other (Ahmed & Kilic, 2019).

**Types of ANOVA and Use Case Scenarios:**

There are two main types of ANOVA analysis, namely, (i) one-way ANOVA, and (ii) two-way ANOVA (Christensen, 2020; Nibrad, 2019).

(i) The **One-way ANOVA** test for between groups can be used when the researcher wants to test two groups to see if there is a difference between them. In order to conduct a one-way ANOVA analysis, the following six assumptions must be satisfied (Ahmed & Kilic, 2019):

   (1) The dependent variables must be continuous.
   (2) Independent variable(s) must consist of two or more categorical, independent groups.
   (3) There should be no relationship between the observations in each group or between the groups themselves, i.e., independence of observations must hold.
   (4) There should be no significant outliers, which might have a negative effect on the one-way ANOVA, thus reducing the validity of the results.
   (5) Dependent variable should be approximately normally distributed for each category of the independent variable. Even though, one-way ANOVA only requires approximately normal data because it is quite "robust" to violations of normality, meaning that assumption can be a little violated and still provide valid results.
   (6) Homogeneity of variances must hold.

(ii) The **Two-way ANOVA** can be with or without replication:

   - The **Two-way ANOVA with replication** is used for two groups where the members of those groups are doing more than one thing. For example, two groups of patients from different hospitals trying two different therapies.
   - The **Two-way ANOVA without replication** is used when the analyst has one group and is double-testing that same group. For example, a research experiment testing one set of individuals before and after they take a particular medication to see if it works or not.

**Difference between ANOVA versus T-test:**

- **T-test** calculates the *Mean.*
- **ANOVA** calculates the *Ratio.*

Also, as shown in Fig. 6.4, a T-test analysis test two groups of variable, whereas ANOVA tests more than two groups to determine the differences or dependency of the variables. Thus, to conduct a test with three or more categories of variable, one must use an Analysis of Variance **(ANOVA).**

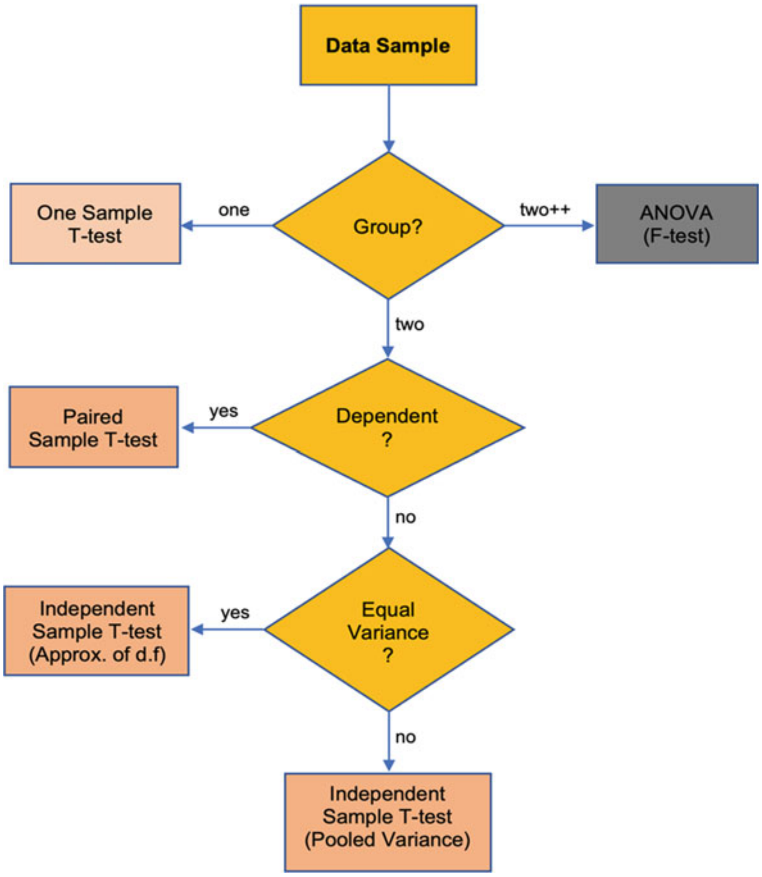


**Fig. 6.4** Flowchart showing the T-test versus ANOVA analysis

### 6.2.6 Mann–Whitney U Test

Mann–Whitney U test is a *non-parametric* equivalent to independent sample T-test. The test is used to compare whether there is a difference in the dependent variable for two independent groups of variable (McKnight & Najab, 2010), as shown in Fig. 6.5. For example, the probable effect of an exam-administration mode (the independent variable) over the test-takers' scores (dependent variable) can be analyzed using the Mann–Whitney U test (Oz & Ozturan, 2018) provided the exam-administration mode is of two categories or group (e.g., online and paper based). Quite often, researchers interpret the Mann–Whitney U test by comparing the *medians* between the two populations.

The formula to calculate the Mann–Whitney test is as follows:

$$\begin{aligned} U_1 &= n_1 n_2 + \frac{n_1(n_1+1)}{2} - R_1 \\ U_2 &= n_1 n_2 + \frac{n_2(n_2+1)}{2} - R_2. \end{aligned}$$

where

$R_1$ = sum of the ranks for group 1 and
$R_2$ = sum of the ranks for group 2.

It is important to mention that the Mann–Whitney U method functions by pooling the observations from the two samples into one combined sample. This is done by

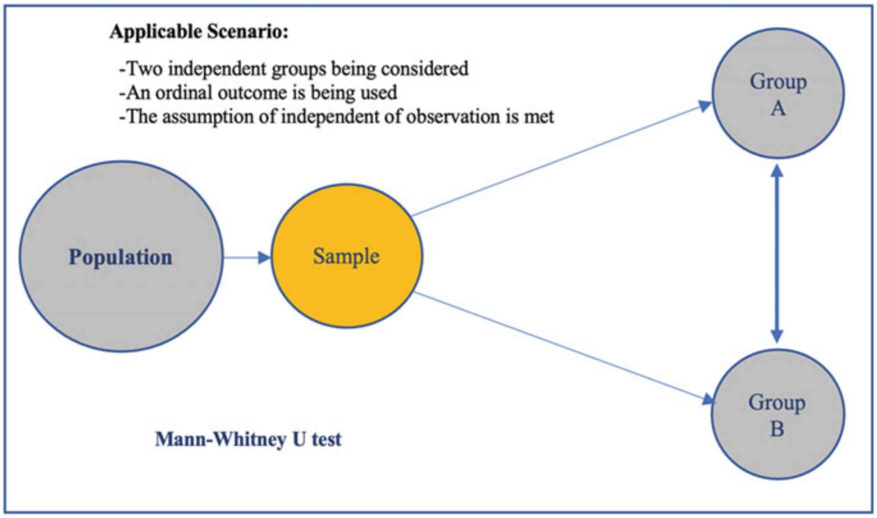

Applicable Scenario:
-Two independent groups being considered
-An ordinal outcome is being used
-The assumption of independent of observation is met
Population
Sample
Group A
Group B
Mann-Whitney U test

**Fig. 6.5** Description of Mann–Whitney U test

keeping track of which sample each observation comes from and then ranking them according to lowest to highest from 1 to $R_1 + R_2$, respectively.

For instance, the Mann–Whitney U test has proved itself as one of the many multivariate tests that can combine the *primary* and *mortality* endpoints of a dataset into a single composite endpoint and can be analyzed through the ranking of those combined outcomes (Matsouaka et al., 2016). The testing of those combined endpoints can be performed as a weighted test where the optimal weights are determined by maximizing the power of the statistical analysis, perhaps, under a particular alternative hypothesis.

**Example of Use Case Scenario of Mann–Whitney U Test:**

Consider a students' assessment system designed to determine the effectiveness of a new teaching program or strategy to improve the students' learning outcome. To this effect, a total of *n* participants is selected randomly to undergo either the new or a previously existing program. The students are asked to take note of the record of the number of times they feel overwhelmed as a result of the assigned program over a specified period of time. The Mann–Whitney U test in this scenario can be used to determine:

- If there is a difference in the number of times the students feel overwhelmed over the period of participating in the new program compared to those undergoing the previously existing program?
- If so, are the observations statistically significant?

## 6.2.7 *Chi-Squared ($\chi^2$)*

*Chi-squared* ($\chi^2$) statistics is a test that measures how *expectations* compare to the actual *observed* data (or model results) (McHugh, 2012). Datasets used in calculating a chi-squared statistic must be random, raw, mutually exclusive, drawn from independent groups or population, and from a large enough sample size. For example, the results of tossing a coin 100 times meet these criteria.

Chi-squared tests have proved effective and are often used in hypothesis testing for calculating new similarity distance measures, which is an important measure in the applications of image analysis, for instance, and statistical inference (Ren et al., 2019).

The formula for calculating the chi-square ($\chi^2$) statistics is as follows:

$$x_c^2 = \Sigma \frac{(O_i - E_i)^2}{E_i}$$

where

C = degrees of freedom,
O = observed values(s), and

**Table 6.1** Example of a chi-squared (χ2) data distribution

| | Present | Absent | **Total—for each value (i.e., O and E)** |
|---|---|---|---|
| Female | 18 (O)<br>15 (E) | 7 (O)<br>10 (E) | 25 |
| Male | 42 (O)<br>45 (E) | 33 (O)<br>30 (E) | 75 |
| Total—for each value (i.e, O and E) | 60 | 40 | 100 |

E = expected values(s.)

As gathered in Table 6.1, the numbers denoted with (O) represent the **Observed** value, **O**, whereas the numbers denoted with (E) represent the Expected value, **E**.

**Example of Use Case Scenario for Chi-squared (χ2):**

A chi-squared test can be applied to determine, for instance, the level of effect or impact that gender bias has on the students' evaluation of teaching or expectations about the academic professors or performance.

### *6.2.8 Kruskal–Wallis H Test*

The Kruskal–Wallis H test proposed by William Kruskal and W. Allen Wallis (Kruskal & Wallis, 1952) is a non-parametric method used to determine whether a group of data comes from the same population. The Kruskal–Wallis H analysis is identical (an alternative) to the ANOVA with the data replaced by categories or ordinal level data. Just in the same manner as the Mann–Whitney U test (but in this case for more than two groups of variables); the Kruskal–Wallis tests can be used to determine if there exist statistical differences between the independent observations based on the dependent variables (Veerasamy et al., 2018).

In other words, the Kruskal–Wallis H test can be referred to as an extension of Mann–Whitney U test typically applied for three or more groups, as illustrated in Fig. 6.6.

The formula for computing the **Kruskal–Wallis H test** is as follows:

$$H = \left( \frac{12}{n(n+1)} \sum_{j=1}^{k} \frac{R_j^2}{n_j} \right) - 3(n+1)$$

where

K = number of comparison groups,
n = total sample size,
$n_j$ = sample size in the jth group, and

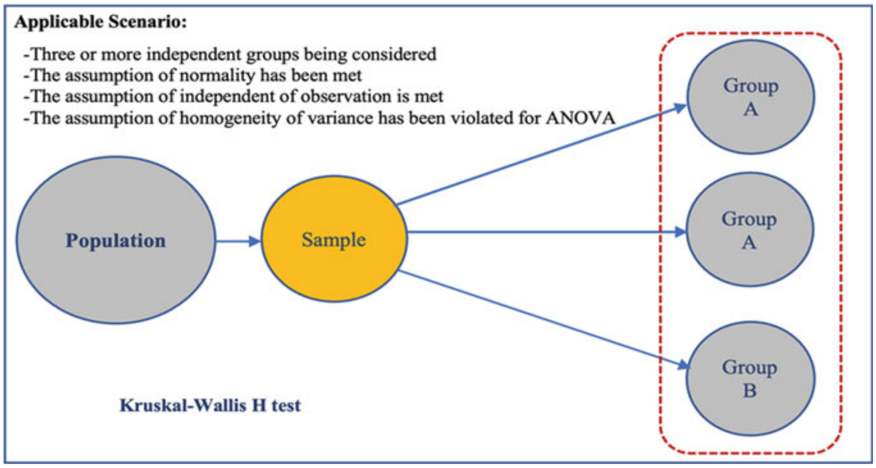

Applicable Scenario:
-Three or more independent groups being considered
-The assumption of normality has been met
-The assumption of independent of observation is met
-The assumption of homogeneity of variance has been violated for ANOVA
Population
Sample
Group A
Group A
Group B
Kruskal-Wallis H test

**Fig. 6.6** Kruskal–Wallis H test

$R_j$ = sum of the ranks in the jth group.

The following are the key features of the Kruskal–Wallis H test:

- All $n = n_1 + n_2 + \cdots + n_k$ measurements are jointly ranked (i.e., treated as one large sample).
- One can also use the sums of the ranks of the k samples to compare the distributions.

**Example of Use Case Scenario of Kruskal–Wallis H Test:**

Please refer to the use case scenario of Mann–Whitney U test (Section 6.2.6), but in this situation used for three or more groups.

## *6.2.9 Correlation*

Correlation is a bivariate analysis that measures the strength of association between two variables and the direction of the relationship (whether positive or negative).

### 6.2.9.1 Kendall Rank Correlation

The Kendall rank (also known as **Kendall's tau analysis**) is a non-parametric test mostly used to measure the strength of dependence between two variables (Brossart et al., 2018). In theory, Kendall's rank correlation coefficient can be applied as an efficient and robust way of identifying *monotone* relationships between two data sequences. Although when applied to digital data (i.e., discrete or discontinuous representation), the high number of ties can produce inconsistent results due to quantization (Couso et al., 2018).

The formula for **Kendall Rank Correlation** is as follows:

$$\text{Kendall's tau} = \frac{C - D}{C + D}$$

Spearman's Rank Correlation

Spearman rank (also known as **Spearman's rho analysis**) is a non-parametric test that is mostly used to measure the *degree* of association between two variables (Wang et al., 2019; Zar, 2014). For example, the researchers can use Spearman's rank correlation coefficient and multiple regression techniques to measure the relationship between some set of variables (Veerasamy et al., 2018).

The formula for **Spearman's Rank Correlation** is as follows:

$$\text{Spearman's.rho} = 1 - \frac{6\sum\left(d_i^2\right)}{n\left(n^2 - 1\right)}$$

**Kendall** versus **Spearman**:

As shown in Table 6.2 and the formula/calculations, an illustration of the difference between the Kendal tau and Spearman rho analysis is as follows (Hauke & Kossowski, 2011):

**When to use Kendall's tau Analysis?**

- The distribution of Kendall's tau analysis is most useful when the data analyst or researcher is interested in a test that has better statistical property.
- The interpretation of Kendall's tau test in terms of the probabilities of observing the agreeable (concordant) and non-agreeable (discordant) pairs is very direct.

**When to use Spearman's rho Analysis?**

- Spearman's rank correlation coefficient is the most widely used rank correlation coefficient analysis.

In summary, quite often the interpretation of Kendall's tau and Spearman's rho rank correlation coefficient are very similar. Thus, both methods tend to invariably lead to the same inferences.

**Table 6.2** Kendall versus Spearman data distribution

| | | Kendall tau | | Spearman rho | |
|---|---|---|---|---|---|
| Teacher | Student | C | D | D | $D^2$ |
| 1 | 2 | 10 | 1 | 1 | 1 |
| 2 | 1 | 10 | 0 | 1 | 1 |
| 3 | 4 | 8 | 1 | 1 | 1 |
| 4 | 3 | 8 | 0 | 1 | 1 |
| 5 | 6 | 6 | 1 | 1 | 1 |
| 6 | 5 | 6 | 0 | 1 | 1 |
| 7 | 8 | 4 | 1 | 1 | 1 |
| 8 | 7 | 4 | 0 | 1 | 1 |
| 9 | 10 | 2 | 1 | 1 | 1 |
| 10 | 9 | 2 | 0 | 1 | 1 |
| 11 | 12 | 0 | 1 | 1 | 1 |
| 12 | 11 | – | – | 1 | 1 |
| | | $\text{Kendall's tau} = \frac{60-6}{60+6}$ <br> $\text{Kendall's tau} = \frac{54}{66}$ <br> $\text{Kendall's tau} = .818$ | | $rho = 1 - \frac{6\sum(12_i^2)}{12(12^2-1)}$ <br> $rho = 1 - \frac{72}{1716}$ <br> $rho = .958$ | |

**Note:**
**Kendall's tau**: Most of the time, the outcome of the Kendall's tau correlation analysis is usually smaller values than Spearman's rho correlation. The calculations are (i) based on concordant and discordant pairs, (ii) insensitive to error, and (iii) the p-values are more accurate with smaller sample sizes.
**Spearman's rho**: The outcome of Spearman's rho test usually has larger values than Kendall's tau test. The calculations are (i) based on deviations, (ii) much more sensitive to error, and (iii) discrepancies in data.

### 6.2.10 Wilcoxon Test (Signed-Rank and Rank-Sum)

The Wilcoxon test, also referred to the *Wilcoxon-signed-rank* or *Wilcoxon-rank-sum* tests (Wilcoxon, 1945), is a non-parametric test and alternative version of the t-test (Rey & Neuhäuser, 2011). The test is mostly applied by the researchers to compare two dependent samples by testing whether the *median* values of the two groups differ significantly from each other. The resultant models assume that the data comes from two matched or dependent populations, following the same distribution through time or place (Hayes, 2023). The test can be applied to test the hypothesis that the median of a symmetrical distribution equals a given constant. And, as the name implies, and as with the many other non-parametric tests that we have already and previously described in this chapter (Chap. 6) and in Chap. 5, this distribution-free test is based on ranks (Rey & Neuhäuser, 2011). It is assumed that the independent variable in a

Wilcoxon test is dichotomous, and the dependent variable is a continuous variable whose measurement is at least ordinal.

The main types and summary of the Wilcoxon test include (Hayes, 2023):

- The Wilcoxon test compares two paired or independent groups of variables and comes in two versions, (i) the rank-sum test, and (ii) signed-rank test.
- The aim of the test is to determine if two or more sets of pairs in a data or variable are different from one another in a statistically significant manner.
- Both tests (whether rank-sum or signed-rank) assume that the pairs in the data sample come from the same dependent populations.
- Unlike t-test that calculates the *mean difference* of two groups of a variable, the Wilcoxon test is used to calculate the *median difference* between the two groups of variables.

The signed-ranked version of the Wilcoxon test is calculated based on differences in the samples' median scores but in addition to it taking into account the signs of the differences, thus, takes into consideration the magnitudes of the observed differences.

As the non-parametric equivalent of the *paired t-test*, the signed-rank can be used as an alternative to the t-test when the population data does not follow a normal distribution.

On the other hand, the Wilcoxon rank-sum test version is often used as the non-parametric equivalent of the *independent* or *two-sample t-test*.

The Wilcoxon rank-sum test is used to compare the median of two independent samples, while Wilcoxon signed-rank test is used to compare the median of two related (paired) samples.

The value of z (test statistics) in a Wilcoxon test is calculated with the following formula:

$$Z_T = \frac{T - \mu_T}{\sigma_T}$$

where

- T = sum of values from calculating the ranges of differences in the sample.

**Example of Use case Scenario of Wilcoxon Test:**

The following type of research questions can be answered using the Wilcoxon test:

- Are the test scores for students, e.g., 5th grade to 6th grade for the same group of students different from each other?
- Are the learning performance of a particular group of students better in the morning or in the evening?

## 6.3 Summary

The type of research methodology or design one chooses to carry out the research investigations determines the type of data that is required for the research purpose, and vice and versa. This includes the means or procedures that will be applied for collecting the samples (data collection) as well as the type of analysis (statistical data analysis) that would be performed. The authors have provided the data type and method matching in Table 6.3 as a guideline for the researchers in the selection of the most appropriate or suitable statistical analysis/method based on the type of data or sample (i.e., independent versus dependent variable).

Overall, a more comprehensive guide on how to choose the best and suitable statistical data analysis method based on the type of data (see Chap. 4) or available statistical tools for the research investigations can also be found in the following sources:

- NYU Elmer Homes Bobst Library (NYU Libraries, 2023) and
- UCLA—Institute for Digital Research and Education (UCLA, 2023).

**Table 6.3** The different types of statistical data analysis described in terms of the independent versus dependent Variables

| | Independent variable (IV) | | | | |
|---|---|---|---|---|---|
| | | Binary | Nominal | Ordinal | Interval |
| Dependent Variable (DV) | Binary | Chi-squared ($\chi 2$) | Chi-squared ($\chi 2$) | Linear-log models | Logistic regression |
| | Nominal | Chi-squared ($\chi 2$) | Chi-squared ($\chi 2$) | Linear-log models | Multiple regression |
| | Ordinal | Mann–Whitney's U | Kruskal–Wallis H | Correlation (Kendall, Spearman) | Correlation (Kendall, Spearman) |
| | Interval | T-test | ANOVA (F-test) | Correlation (Kendall, Spearman) | Linear regression |

## References

Ahmed, F., & Kilic, K. (2019). Fuzzy analytic hierarchy process: A performance analysis of various algorithms. *Fuzzy Sets and Systems*, *362*, 110–128. https://doi.org/10.1016/j.fss.2018.08.009

Brossart, D. F., Laird, V. C., & Armstrong, T. W. (2018). Interpreting Kendall's Tau and Tau-U for single-case experimental designs. *Cogent Psychology, 5*(1), 1518687. https://doi.org/10.1080/23311908.2018.1518687

Chatfield, C. (2018). *Introduction to multivariate analysis.* Tests in Statistical Science (Mathematics & Statistics) Taylor & Francis Group.

Christensen, R. (2020). *One-way ANOVA* (pp. 107–121). Springer, Cham. https://doi.org/10.1007/978-3-030-32097-3_4

Connelly, L. (2020). Logistic regression. *Medsurg Nursing, 29*(5), 353–354.

Couso, I., Strauss, O., & Saulnier, H. (2018). Kendall's rank correlation on quantized data: An interval-valued approach. *Fuzzy Sets and Systems*, *343*, 50–64. https://doi.org/10.1016/j.fss.2017.09.003

Ghosh, J., Li, Y., & Mitra, R. (2018). On the use of cauchy prior distributions for Bayesian logistic regression. *Bayesian Analysis, 13*(2), 359–383.

Glick, T. B., & Figliozzi, M. A. (2019). Analysis and application of log-linear and quantile regression models to predict bus dwell times. *Transportation Research Record, 2673*(10), 118–128. https://doi.org/10.1177/0361198119848701

Hauke, J., & Kossowski, T. (2011). Comparison of values of Pearson's and Spearman's correlation coefficient on the same sets of data. *Quaestiones Geographicae*, *30*(2), 87–93. https://repozytorium.amu.edu.pl/handle/10593/15580

Hayes, A. (2023). *Wilcoxon test: Definition in statistics, types, and calculation.* https://www.investopedia.com/terms/w/wilcoxon-test.asp

Kronthaler, F., & Zöllner, S. (2021). Linear regression with RStudio. In *Data analysis with RStudio* (pp. 87–106). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-662-62518-7_7

Kruskal, W. H., & Wallis, W. A. (1952). Use of ranks in one-criterion variance analysis. *Journal of the American Statistical Association*, *47*(260).

Matsouaka, R. A., Singhal, A. B., & Betensky, R. A. (2016). An optimal Wilcoxon–Mann–Whitney test of mortality and a continuous outcome. *Statistical Methods in Medical Research, 27*(8), 2384–2400. https://doi.org/10.1177/0962280216680524

McHugh, M. L. (2012). The Chi-square test of independence. *Biochemia Medica*, *23*(2), 143–149. https://doi.org/10.11613/BM.2013.018

McKnight, P. E., & Najab, J. (2010). Mann–Whitney U test. In *The corsini encyclopedia of psychology* (pp. 1–1). John Wiley & Sons, Inc. https://doi.org/10.1002/9780470479216.corpsy0524

Montgomery, D., Peck, E., & Vining, G. (2012). *Introduction to linear regression analysis*, vol. 821, Wiley Series in Probability and Statistics, Wiley Publishers.

Nibrad, G. M. (2019). Methodology and application of two-way ANOVA. *International Journal of Marketing and Technology, 9*(6), 1–8.

Novak, S. Y. (2020). On the T-test. In *Statistics theory*. http://arxiv.org/abs/2012.14530

NYU Libraries. (2023). *Quantitative analysis guide: Choose statistical test for 1 dependent variable.* https://guides.nyu.edu/quant/choose_test_1DV

Oz, H., & Ozturan, T. (2018). Computer-based and paper-based testing: Does the test administration mode influence the reliability and validity of achievement tests? *Journal of Language and Linguistic Studies, 14*(1), 67–85.

Ren, H., Xiao, S., & Zhou, H. (2019). A chi-square distance-based similarity measure of single-valued neutrosophic set and applications. *International Journal of Computers Communications Control*, *14*, 78–89. https://api.semanticscholar.org/CorpusID:86679389

Rey, D., & Neuhäuser, M. (2011). Wilcoxon-signed-rank test. In M. Lovric (Ed.), *International encyclopedia of statistical science* (pp. 1658–1659). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-04898-2_616

UCLA Institute for Digital Research and Education. (2023). *Data analysis examples*. https://stats.idre.ucla.edu/other/dae/

Veerasamy, A. K., D'Souza, D. J., Lindén, R., & Laakso, M. J. (2018). The impact of prior programming knowledge on lecture attendance and final exam. *Journal of Educational Computing Research*, *56*, 226–253. https://api.semanticscholar.org/CorpusID:67685012

Wang, B., Wang, R., & Wang, Y. (2019). Compatible matrices of Spearman's rank correlation. *Statistics and Probability Letters, 151*, 67–72. https://doi.org/10.1016/j.spl.2019.03.015

Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics Bulletin, 1*(6), 80–83. https://doi.org/10.2307/3001968

Zar, J. H. (2014). Spearman rank correlation: Overview. In: *Wiley statsref: Statistics reference online*. John Wiley & Sons, Ltd. https://doi.org/10.1002/9781118445112.stat05964

Zdaniuk, B. (2014). Ordinary least-squares (OLS) model. In A. C. Michalos (Ed.), *Encyclopedia of quality of life and well-being research* (pp. 4515–4517). Springer, Netherlands. https://doi.org/10.1007/978-94-007-0753-5_2008

# Part II
# Application and Implementation of Advanced Methods for Statistical Data Analysis in Research Using R

# Chapter 7
# Regression Analysis in R: Linear Regression and Logistic Regression

## 7.1 Introduction to Regression Analysis

Statistical data analysis methods or procedures are used by the researchers to test or scrutinize the collected data samples in order to determine if they should accept or reject the pre-defined research questions and hypothesis. Typically, the researchers or data analysts apply the (applicable) statistical methods and measurements to determine whether the assumptions or claims are scientifically (statistically) significant or not.

In this chapter, the authors introduce to the readers how to conduct the *Linear* and *Logistic Regression* analysis using R.

There are two main points to consider when conducting the regression analysis:

- First is either to check if a predictor variable (often referred to as the independent variable) is good enough (significant levels) in predicting the effect (outcome) or response of a targeted variable (referred to as the dependent variable).
- Or the second, which is to determine what variable(s), in particular, are the significant predictors of the outcome variable (dependent) in the case of multiple independent variables.

*Linear regression* is one of the most commonly used types of regression analysis, especially in predictive and diagnostic analytics. It is used by the researchers to determine the relationship that exists among one targeted (dependent) variable and another one or more predictor (independent) variable(s). The method assumes that the relationships between the independent and dependent variables are *linear*. Therefore, a constant unit of change in one of the variables implies a constant unit of change in the other (Pal et al., 2019; Schmidt & Finan, 2018). This is done by finding the best (fitting) straight line (see Chap. 6), otherwise referred to as the *regression line* through the points calculated by Least Squares (LS) modus considering the scrutinized variables (Darlington & Hayes, 2016; Pal et al., 2019).

K. Okoye and S. Hosseini, *R Programming*,
https://doi.org/10.1007/978-981-97-3385-9_7

The linear regression line is often represented as y = a + bx, where x is the predictor (independent) variable and y is the response (dependent) variable (see Chap. 6, Fig. 6.1). The slope of the line is denoted as b, whereas a represents the intercept (i.e., the value of y when x = 0). Thus, in theory, the basic formula or syntax for the linear regression test is defined as y = c + b*x, where y represents the estimated dependent variable score, c = constant, b = regression coefficient, and x representing the score on the independent variable. Consequently, the outcome of these regression estimates or formula is what the researchers use to explain the relationship between one targeted variable and another variable(s) (Kronthaler & Zöllner, 2021).

Typically, there are two main types of linear regression tests or method which the authors will be illustrating in this chapter using R. These are

(i) *Simple linear regression* which considers one dependent variable (interval or ratio) and one independent variable (interval or ratio or dichotomous) (Kaps & Lamberson, 2017).
(ii) *Multiple linear regression* which considers one dependent variable (interval or ratio) and two or more (2+) independent variables (interval or ratio or dichotomous) (Olive, 2017).

*Logistic regression*, on the other hand, (unlike the linear regression which uses continuous variables in its tests) is used when the dependent (targeted) variable is a categorical or dichotomous (binary) data, i.e., fits into one of two clear-cut categories. By definition, the "logistic regression" can be referred to as the statistical method or procedure that is used to analyze or explain the relationship between one dependent categorical (or binary) variable and one or more nominal, ordinal, interval, or ratio-level independent variables (Connelly, 2020). With logistic regression the linearity of the dependent (i.e., response variable) and independent (predictor) variable(s) is measured using weights or coefficient (beta values). The only major difference of *logistic regression* from *linear regression* is that the output of the method (logistic) is modeled or represented as binary values, i.e., 0 or 1, rather than just numerical values.

Therefore, the logistic regression equation can be defined as y = e^(b0 + b1*x) / (1 + e^(b0 + b1*x)) where y is the predicted output (dependent variable score), b0 is the bias or intercept term, while b1 is the coefficient for the input value x (i.e., independent variable score).

Logistic regression (an alternative to linear regression) is basically a method that provides means of using linear regression to solve problems that are not directly or commonly suitable for applying the linear regression procedures.

There are three main types of logistic regression analysis which the authors will illustrate in R in this chapter. These are

(i) *Binomial (binary) logistic regression* which considers dependent variables that only have two possible types of 0 or 1. For instance, male or female, win or lose, pass or fail, etc. (Prasetyo et al., 2020).
(ii) *Multinomial logistic regression* which considers dependent variables that have three or more (i.e., more than two levels) possible types but are not ordered.

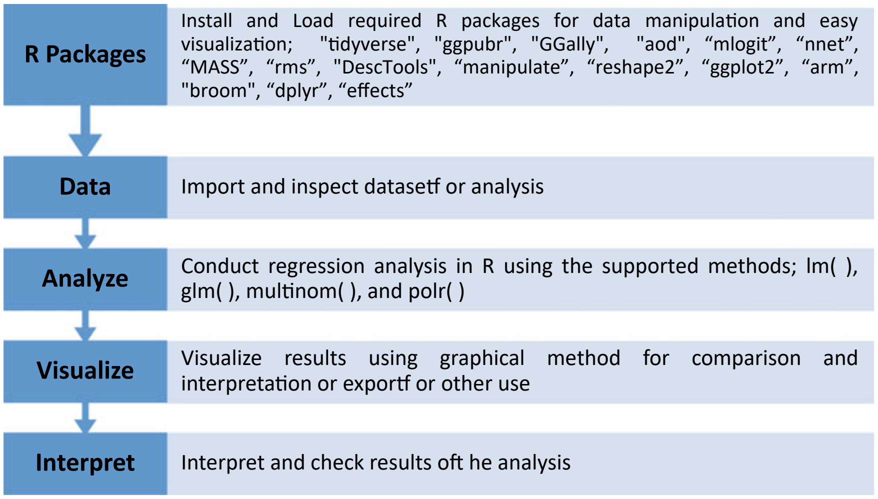

R Packages
Install and Load required R packages for data manipulation and easy visualization; "tidyverse", "ggpubr", "GGally", "aod", "mlogit", "nnet", "MASS", "rms", "DescTools", "manipulate", "reshape2", "ggplot2", "arm", "broom", "dplyr", "effects"
Data
Import and inspect datasetf or analysis
Analyze
Conduct regression analysis in R using the supported methods; lm( ), glm( ), multinom( ), and polr( )
Visualize
Visualize results using graphical method for comparison and interpretation or exportf or other use
Interpret
Interpret and check results oft he analysis

**Fig. 7.1** Steps to conducting the linear regression and logistic regression analysis in R

For instance, Place A vs Place B versus Place C (Austin & Merlo, 2017; Liang et al., 2020).

(iii) *Ordinal logistic regression* which considers more than two dependent variables but with ordered categories. For instance, Poor, Average, Good, Excellent, etc. (Fagerland & Hosmer, 2017; Liang et al., 2020).

Over the next sections of this chapter, authors will describe and look at how to conduct the different types of the Linear Regression analysis (see Sect. 7.2) and Logistic Regression analysis (Sect. 7.3) in R statistics (Rstudio, 2020).

In Fig. 7.1, the authors describe and illustrate the different steps, functions, and packages to performing the Linear and Logistic Regression tests in R as follows.

## 7.2 Linear Regression Analysis in R: Simple Regression and Multiple Regression

Linear regression is one of the most common scientific and widely used statistical method used to determine the relationship (linearity) that exists between a targeted variable (dependent or response variable) and another predictor variable (independent or explanatory variable).

The default hypothesis for testing whether certain or particular variables are related (correlated) is *IF* the p-value is less than or equal to 0.05 ($p \leq 0.05$), THEN correlation/relationship is assumed, *ELSE IF* the p-value is greater than 0.05 ($p > 0.05$) *THEN* dependency or association is not assumed.

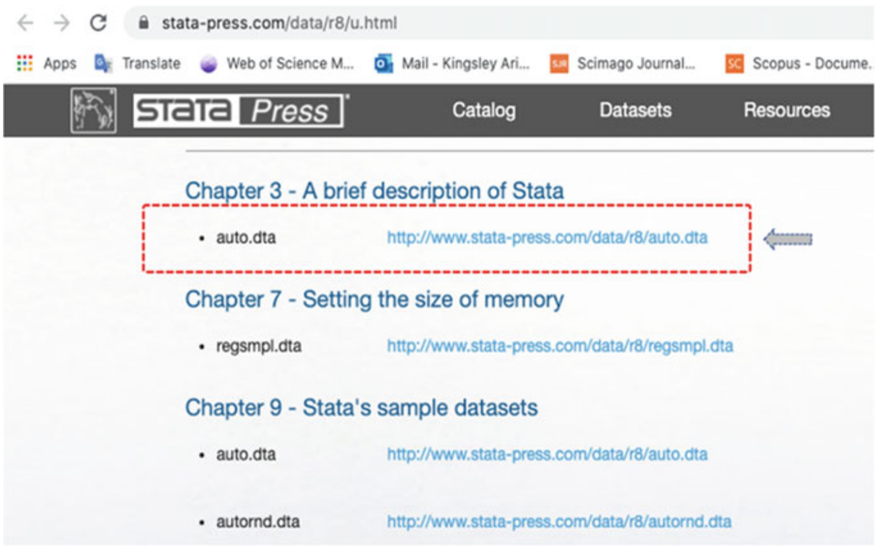
The authors will demonstrate how to conduct the two most common types of linear regression analysis (simple regression and multiple regression) tests by using the **lm( )** function or method in R. We will do this using the steps outlined in Fig. 7.1.

To start, **open RStudio** and **create a new or open an existing project**. Once you have RStudio and a project opened, **create a new RScript** and name it "**RegressionAnalysisTest**" or any name the user chooses (refer to Chap. 1 on how to do these steps if required or needed).

Now, let's download an example dataset that we will use to demonstrate the regression analysis (**the users are welcome to use any dataset or format they choose—see Chap. 2 for a step-by-step guide on how to work with different data types and format in R).

As shown in Fig. 7.2, download the example .dta file named "**auto.dat**" and save it on your local machine or computer (https://www.stata-press.com/data/r8/u.html). ***Note: the readers can also refer to the following repository (https://doi.org/https://doi.org/10.6084/m9.figshare.24728073) where the authors have uploaded all the example files used in this book if they wish to do so or not able to access the example file directly from the source page.

Once the user has successfully downloaded and saved the file on the computer, we can proceed to conduct the *linear regression analysis*.



Chapter 3 - A brief description of Stata

- auto.dta http://www.stata-press.com/data/r8/auto.dta

Chapter 7 - Setting the size of memory

- regsmpl.dta http://www.stata-press.com/data/r8/regsmpl.dta

Chapter 9 - Stata's sample datasets

- auto.dta http://www.stata-press.com/data/r8/auto.dta

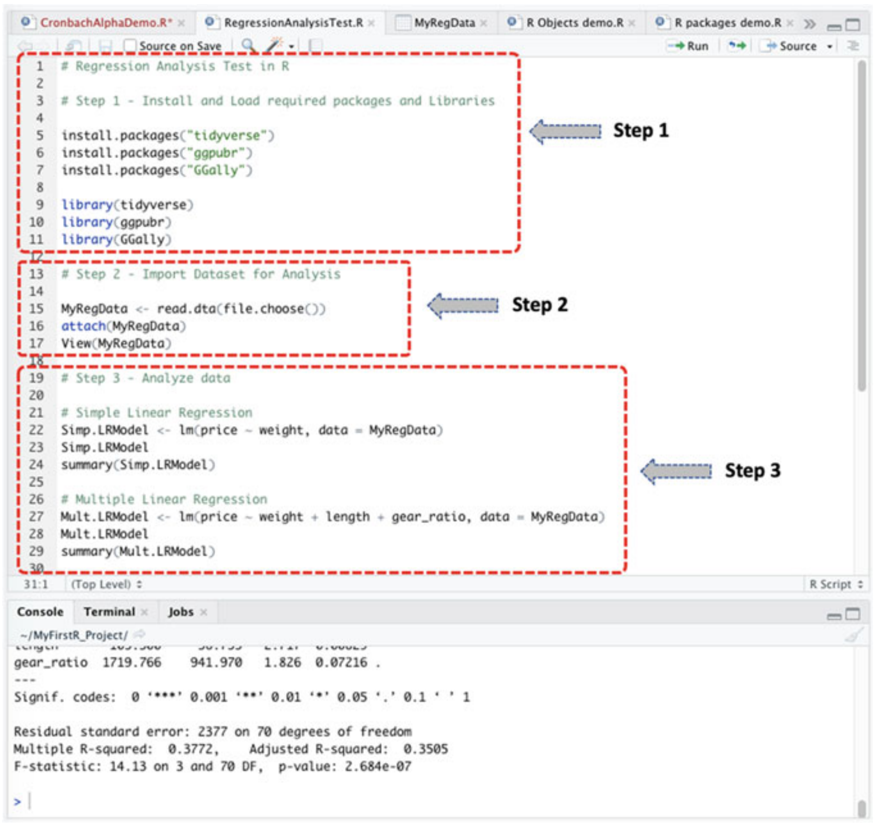
- autornd.dta http://www.stata-press.com/data/r8/autornd.dta

**Fig. 7.2** Example of .dta file format download. (*Source* https://www.stata-press.com/data/r8/u.html)

## # **Step 1**—Install and load the required R packages

**Install and Load** the following R packages (“**tidyverse**”, “**ggpubr**”, “**GGally**”). As shown and described in Fig. 7.3 (Step 1, Lines 3–11), the syntax to install and load the packages is as follows:

```
install.packages("tidyverse")
install.packages("ggpubr")
install.packages("GGally")

library(tidyverse)
library(ggpubr)
library(GGally)
```



```
1  # Regression Analysis Test in R
2
3  # Step 1 - Install and Load required packages and Libraries
4
5  install.packages("tidyverse")
6  install.packages("ggpubr")
7  install.packages("GGally")
8
9  library(tidyverse)
10 library(ggpubr)
11 library(GGally)
12
13 # Step 2 - Import Dataset for Analysis
14
15 MyRegData <- read.dta(file.choose())
16 attach(MyRegData)
17 View(MyRegData)
18
19 # Step 3 - Analyze data
20
21 # Simple Linear Regression
22 Simp.LRModel <- lm(price ~ weight, data = MyRegData)
23 Simp.LRModel
24 summary(Simp.LRModel)
25
26 # Multiple Linear Regression
27 Mult.LRModel <- lm(price ~ weight + length + gear_ratio, data = MyRegData)
28 Mult.LRModel
29 summary(Mult.LRModel)
30
```

```
gear_ratio  1719.766    941.970   1.826  0.07216 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2377 on 70 degrees of freedom
Multiple R-squared:  0.3772,	Adjusted R-squared:  0.3505
F-statistic: 14.13 on 3 and 70 DF,  p-value: 2.684e-07

>
```

**Fig. 7.3** Steps to conducting the simple and multiple linear regression in R

**# Step 2**—Import and Inspect Dataset for Regression analysis

As illustrated in Fig. 7.3 (see Step 2), import the dataset named "**auto.dta**" which we have downloaded earlier on the local computer, and store this in an R object named "**MyRegData**" (the users can use any name they choose if they want).

Once the user has successfully imported the dataset (see code below, Fig. 7.3, Lines 13–17), they will be able to view the details of the **auto.dta** dataset as highlighted in Fig. 7.4 with 74 observations and 12 variables in the imported data sample.

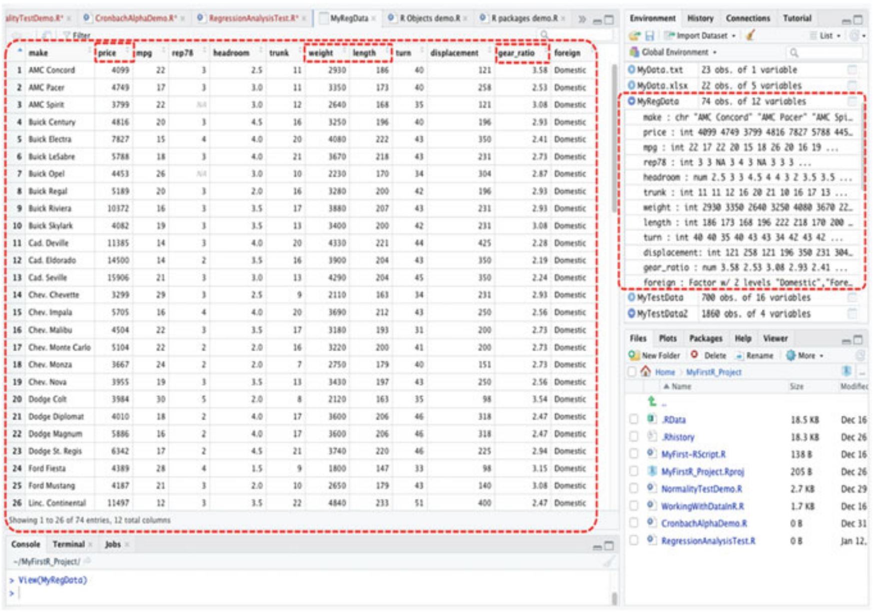
```
MyRegData <- read.dta(file.choose())
attach(MyTestData)
View(MyTestData)
```

**# Step 3**—Analyze the dataset

Now that the user has imported the dataset and stored it in an R object we named or defined as "**MyRegData**", we can proceed to analyze the data.

As shown and explained in Fig. 7.3 (Step 3, Lines 19–29), we will conduct a **Simple** and **Multiple Linear Regression** analysis using the **lm( )** function in R.

Note: As defined in the introduction section (Sect. 7.1)—(see also Chap. 5 for a description of the dependent versus independent variables).



| | make | price | mpg | rep78 | headroom | trunk | weight | length | turn | displacement | gear_ratio | foreign |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | AMC Concord | 4099 | 22 | 3 | 2.5 | 11 | 2930 | 186 | 40 | 121 | 3.58 | Domestic |
| 2 | AMC Pacer | 4749 | 17 | 3 | 3.0 | 11 | 3350 | 173 | 40 | 258 | 2.53 | Domestic |
| 3 | AMC Spirit | 3799 | 22 | NA | 3.0 | 12 | 2640 | 168 | 35 | 121 | 3.08 | Domestic |
| 4 | Buick Century | 4816 | 20 | 3 | 4.5 | 16 | 3250 | 196 | 40 | 196 | 2.93 | Domestic |
| 5 | Buick Electra | 7827 | 15 | 4 | 4.0 | 20 | 4080 | 222 | 43 | 350 | 2.41 | Domestic |
| 6 | Buick LeSabre | 5788 | 18 | 3 | 4.0 | 21 | 3670 | 218 | 43 | 231 | 2.73 | Domestic |
| 7 | Buick Opel | 4453 | 26 | NA | 3.0 | 10 | 2230 | 170 | 34 | 304 | 2.87 | Domestic |
| 8 | Buick Regal | 5189 | 20 | 3 | 2.0 | 16 | 3280 | 200 | 42 | 196 | 2.93 | Domestic |
| 9 | Buick Riviera | 10372 | 16 | 3 | 3.5 | 17 | 3880 | 207 | 43 | 231 | 2.93 | Domestic |
| 10 | Buick Skylark | 4082 | 19 | 3 | 3.5 | 13 | 3400 | 200 | 42 | 231 | 3.08 | Domestic |
| 11 | Cad. Deville | 11385 | 14 | 3 | 4.0 | 20 | 4330 | 221 | 44 | 425 | 2.28 | Domestic |
| 12 | Cad. Eldorado | 14500 | 14 | 2 | 3.5 | 16 | 3900 | 204 | 43 | 350 | 2.19 | Domestic |
| 13 | Cad. Seville | 15906 | 21 | 3 | 3.0 | 13 | 4290 | 204 | 45 | 350 | 2.24 | Domestic |
| 14 | Chev. Chevette | 3299 | 29 | 3 | 2.5 | 9 | 2110 | 163 | 34 | 231 | 2.93 | Domestic |
| 15 | Chev. Impala | 5705 | 16 | 4 | 4.0 | 20 | 3690 | 212 | 43 | 250 | 2.56 | Domestic |
| 16 | Chev. Malibu | 4504 | 22 | 3 | 3.5 | 17 | 3180 | 193 | 31 | 200 | 2.73 | Domestic |
| 17 | Chev. Monte Carlo | 5104 | 22 | 2 | 2.0 | 16 | 3220 | 200 | 41 | 200 | 2.73 | Domestic |
| 18 | Chev. Monza | 3667 | 24 | 2 | 2.0 | 7 | 2750 | 179 | 40 | 151 | 2.73 | Domestic |
| 19 | Chev. Nova | 3955 | 19 | 3 | 3.5 | 13 | 3430 | 197 | 43 | 250 | 2.56 | Domestic |
| 20 | Dodge Colt | 3984 | 30 | 5 | 2.0 | 8 | 2120 | 163 | 35 | 98 | 3.54 | Domestic |
| 21 | Dodge Diplomat | 4010 | 18 | 2 | 4.0 | 17 | 3600 | 206 | 46 | 318 | 2.47 | Domestic |
| 22 | Dodge Magnum | 5886 | 16 | 2 | 4.0 | 17 | 3600 | 206 | 46 | 318 | 2.47 | Domestic |
| 23 | Dodge St. Regis | 6342 | 17 | 2 | 4.5 | 21 | 3740 | 220 | 46 | 225 | 2.94 | Domestic |
| 24 | Ford Fiesta | 4389 | 28 | 4 | 1.5 | 9 | 1800 | 147 | 33 | 98 | 3.15 | Domestic |
| 25 | Ford Mustang | 4187 | 21 | 3 | 2.0 | 10 | 2650 | 179 | 43 | 140 | 3.08 | Domestic |
| 26 | Linc. Continental | 11497 | 12 | 3 | 3.5 | 22 | 4840 | 233 | 51 | 400 | 2.47 | Domestic |

Showing 1 to 26 of 74 entries, 12 total columns

```
~/MyFirstR_Project/
> View(MyRegData)
>
```

```
MyData.txt      23 obs. of 1 variable
MyData.xlsx     22 obs. of 5 variables
MyRegData       74 obs. of 12 variables
  make : chr "AMC Concord" "AMC Pacer" "AMC Spi…
  price : int 4099 4749 3799 4816 7827 5788 445…
  mpg : int 22 17 22 20 15 18 26 20 16 19 ...
  rep78 : int 3 3 NA 3 4 3 NA 3 3 3 ...
  headroom : num 2.5 3 3 4.5 4 4 3 2 3.5 3.5 ...
  trunk : int 11 11 12 16 20 21 10 16 17 13 ...
  weight : int 2930 3350 2640 3250 4080 3670 22…
  length : int 186 173 168 196 222 218 170 200 …
  turn : int 40 40 35 40 43 43 34 42 43 42 ...
  displacement: int 121 258 121 196 350 231 304…
  gear_ratio : num 3.58 2.53 3.08 2.93 2.41 ...
  foreign : Factor w/ 2 levels "Domestic","Fore…
MyTestData      700 obs. of 16 variables
MyTestData2     1860 obs. of 4 variables
```

| Name | Size | Modified |
|---|---|---|
| .. | | |
| .RData | 18.5 KB | Dec 16 |
| .Rhistory | 18.3 KB | Dec 26 |
| MyFirst-RScript.R | 138 B | Dec 16 |
| MyFirstR_Project.Rproj | 205 B | Dec 26 |
| NormalityTestDemo.R | 2.7 KB | Dec 29 |
| WorkingWithDataInR.R | 1.7 KB | Dec 16 |
| CronbachAlphaDemo.R | 0 B | Dec 31 |
| RegressionAnalysisTest.R | 0 B | Jan 12, |

**Fig. 7.4** The dataset example imported and stored as R object in RStudio

- Simple linear regression tests consider only one independent (predictor) variable, while
- Multiple linear regression uses two or more independent variables.

Therefore, to illustrate the two tests, we will use the following variables highlighted in the example dataset (**MyRegData):**

- For *simple linear regression*, we will check the relationship that "**weight**" (independent variable) has with the "**price**" (dependent variable) of the automobiles.
- For the *multiple linear regression*, we will check the relationship that "**price**" (dependent variable) has with the following independent variables "**weight**", "**length**", and "**gear_ratio**".

The syntax to conduct the two tests (see Step 3) in R is shown in the code below (see Fig. 7.3, Lines 19–29):

```
# Simple Linear Regression
Simp.LRModel <- lm(price ~ weight, data = MyRegData)
Simp.LRModel
summary(Simp.LRModel)

# Multiple Linear Regression
Mult.LRModel <- lm(price ~ weight + length + gear_ratio, data = MyRegData)
Mult.LRModel
summary(Mult.LRModel)
```

When the user has successfully run the commands (as described in Fig. 7.3, Step 3, Lines 19 to 29), they will be presented with the results of the linear regression analysis in the Console similar to the ones we have reported in Figs. 7.5a and b.

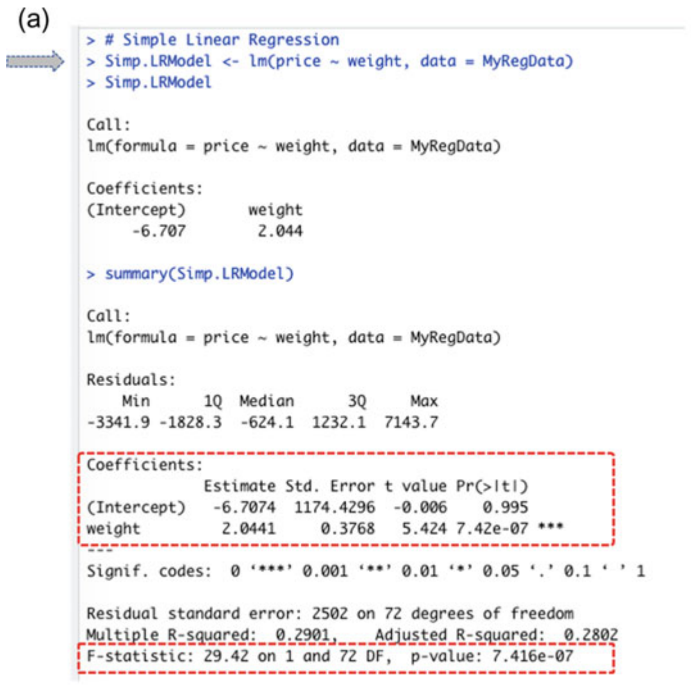
As shown in Figs. 7.5a and b, we conducted the simple and multiple linear regression analysis using the following variables (price, weight, length, gear_ratio) which are in the right format (continuous or interval ratio) for conducting the regression analysis (parametric test— see Chap. 4), and stored the results of the tests in an R object we called or defined as "Simp.LRModel" and "Mult.LRModel", respectively.

# **Step 4**—Visualize linear relationship

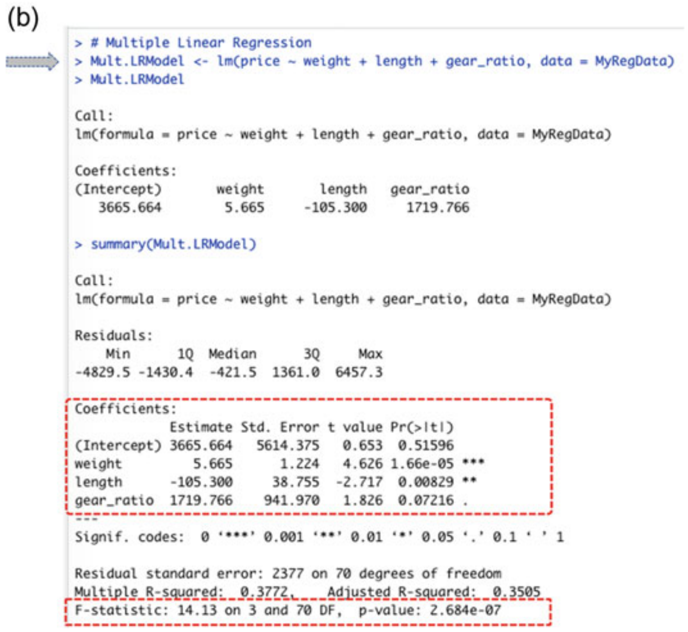
An additional and useful way to check the relationship that exists between a variable and another in R is by plotting them as graph which, we demonstrated in Chap. 2 in Sect. 2.7. In this way, the user can draw a correlation between the target or analyzed variables.

To do this, we will use the following functions: **plot( ), abline( ), ggplot( ),** and **ggpairs**( ) to visualize the relationship (linearity) between the variables, as shown in Fig. 7.6 and the resultant plots in Figs. 7.7 and 7.8.

The syntax used for the example plots (see Step 4 in Fig. 7.6, Lines 31–42) is as shown below, and the results reported and described in Figs. 7.7 and 7.8.

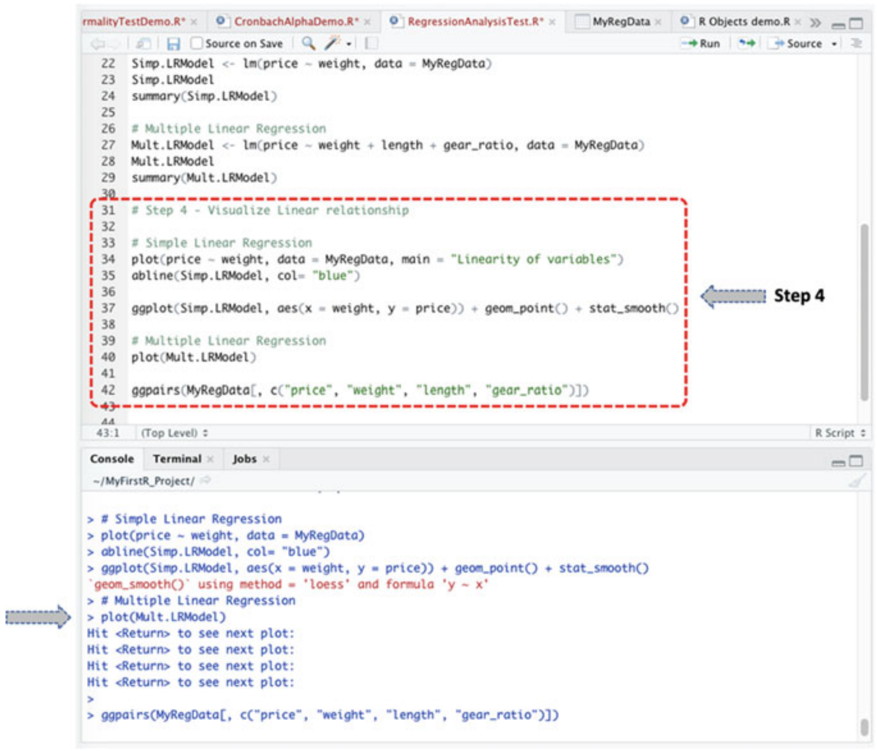


```
> # Simple Linear Regression
> Simp.LRModel <- lm(price ~ weight, data = MyRegData)
> Simp.LRModel

Call:
lm(formula = price ~ weight, data = MyRegData)

Coefficients:
(Intercept)       weight
     -6.707        2.044

> summary(Simp.LRModel)

Call:
lm(formula = price ~ weight, data = MyRegData)

Residuals:
    Min      1Q  Median      3Q     Max
-3341.9 -1828.3  -624.1  1232.1  7143.7

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -6.7074  1174.4296  -0.006    0.995
weight        2.0441     0.3768   5.424 7.42e-07 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2502 on 72 degrees of freedom
Multiple R-squared:  0.2901,	Adjusted R-squared:  0.2802
F-statistic: 29.42 on 1 and 72 DF,  p-value: 7.416e-07
```



```
> # Multiple Linear Regression
> Mult.LRModel <- lm(price ~ weight + length + gear_ratio, data = MyRegData)
> Mult.LRModel

Call:
lm(formula = price ~ weight + length + gear_ratio, data = MyRegData)

Coefficients:
(Intercept)       weight       length   gear_ratio
   3665.664        5.665     -105.300     1719.766

> summary(Mult.LRModel)

Call:
lm(formula = price ~ weight + length + gear_ratio, data = MyRegData)

Residuals:
    Min      1Q  Median      3Q     Max
-4829.5 -1430.4  -421.5  1361.0  6457.3

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 3665.664   5614.375   0.653  0.51596
weight         5.665      1.224   4.626 1.66e-05 ***
length      -105.300     38.755  -2.717  0.00829 **
gear_ratio  1719.766    941.970   1.826  0.07216 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2377 on 70 degrees of freedom
Multiple R-squared:  0.3772,	Adjusted R-squared:  0.3505
F-statistic: 14.13 on 3 and 70 DF,  p-value: 2.684e-07
```

**Fig. 7.5** **a** Result of simple linear regression analysis in R. **b** Results of multiple linear regression analysis in R



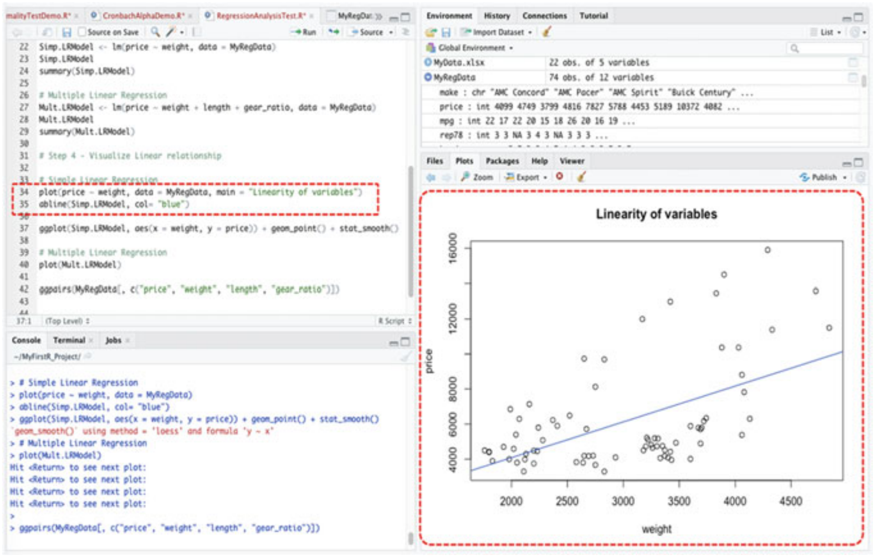
```
Simp.LRModel <- lm(price ~ weight, data = MyRegData)
Simp.LRModel
summary(Simp.LRModel)

# Multiple Linear Regression
Mult.LRModel <- lm(price ~ weight + length + gear_ratio, data = MyRegData)
Mult.LRModel
summary(Mult.LRModel)

# Step 4 - Visualize Linear relationship

# Simple Linear Regression
plot(price ~ weight, data = MyRegData, main = "Linearity of variables")
abline(Simp.LRModel, col= "blue")

ggplot(Simp.LRModel, aes(x = weight, y = price)) + geom_point() + stat_smooth()

# Multiple Linear Regression
plot(Mult.LRModel)

ggpairs(MyRegData[, c("price", "weight", "length", "gear_ratio")])
```

```
> # Simple Linear Regression
> plot(price ~ weight, data = MyRegData)
> abline(Simp.LRModel, col= "blue")
> ggplot(Simp.LRModel, aes(x = weight, y = price)) + geom_point() + stat_smooth()
`geom_smooth()` using method = 'loess' and formula 'y ~ x'
> # Multiple Linear Regression
> plot(Mult.LRModel)
Hit <Return> to see next plot:
Hit <Return> to see next plot:
Hit <Return> to see next plot:
Hit <Return> to see next plot:
>
> ggpairs(MyRegData[, c("price", "weight", "length", "gear_ratio")])
```

**Fig. 7.6** Visualizing linear relationship between two or more variables in R

```
# Simple Linear Regression
    plot(price ~ weight, data = MyRegData, main = "Linearity of variables")
    abline(Simp.LRModel, col= "blue")

    ggplot(Simp.LRModel, aes(x = weight, y = price)) + geom_point() +
    stat_smooth()


# Multiple Linear Regression
    plot(Mult.LRModel)

    ggpairs(MyRegData[, c("price", "weight", "length", "gear_ratio")])
```

**Note**: When applying the scatter plot function plot(Mult.LRModel) which we used for the multiple linear regression analysis (see as pointed in Fig. 7.6), the user will be prompted in the Console to press or hit the Return button (Enter key) on the keyboard to view the different plotted graphs.

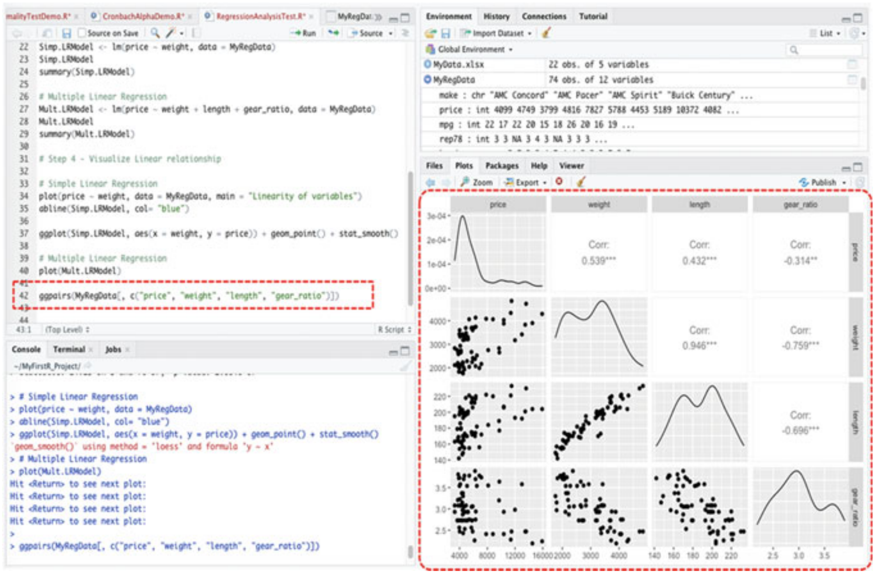


```
Simp.LRModel <- lm(price ~ weight, data = MyRegData)
Simp.LRModel
summary(Simp.LRModel)

# Multiple Linear Regression
Mult.LRModel <- lm(price ~ weight + length + gear_ratio, data = MyRegData)
Mult.LRModel
summary(Mult.LRModel)

# Step 4 - Visualize Linear relationship

# Simple Linear Regression
plot(price ~ weight, data = MyRegData, main = "Linearity of variables")
abline(Simp.LRModel, col= "blue")

ggplot(Simp.LRModel, aes(x = weight, y = price)) + geom_point() + stat_smooth()

# Multiple Linear Regression
plot(Mult.LRModel)

ggpairs(MyRegData[, c("price", "weight", "length", "gear_ratio")])
```

```
~/MyFirstR_Project/

> # Simple Linear Regression
> plot(price ~ weight, data = MyRegData)
> abline(Simp.LRModel, col= "blue")
> ggplot(Simp.LRModel, aes(x = weight, y = price)) + geom_point() + stat_smooth()
`geom_smooth()` using method = 'loess' and formula 'y ~ x'
> # Multiple Linear Regression
> plot(Mult.LRModel)
Hit <Return> to see next plot:
Hit <Return> to see next plot:
Hit <Return> to see next plot:
Hit <Return> to see next plot:
>
> ggpairs(MyRegData[, c("price", "weight", "length", "gear_ratio")])
```

```
MyData.xlsx     22 obs. of 5 variables
MyRegData       74 obs. of 12 variables
  make : chr "AMC Concord" "AMC Pacer" "AMC Spirit" "Buick Century" ...
  price : int 4099 4749 3799 4816 7827 5788 4453 5189 10372 4082 ...
  mpg : int 22 17 22 20 15 18 26 20 16 19 ...
  rep78 : int 3 3 NA 3 4 3 NA 3 3 3 ...
```

Fig. 7.7 Example of simple linear regression graph in R using the plot( ) and abline( ) functions



```
Simp.LRModel <- lm(price ~ weight, data = MyRegData)
Simp.LRModel
summary(Simp.LRModel)

# Multiple Linear Regression
Mult.LRModel <- lm(price ~ weight + length + gear_ratio, data = MyRegData)
Mult.LRModel
summary(Mult.LRModel)

# Step 4 - Visualize Linear relationship

# Simple Linear Regression
plot(price ~ weight, data = MyRegData, main = "Linearity of variables")
abline(Simp.LRModel, col= "blue")

ggplot(Simp.LRModel, aes(x = weight, y = price)) + geom_point() + stat_smooth()

# Multiple Linear Regression
plot(Mult.LRModel)

ggpairs(MyRegData[, c("price", "weight", "length", "gear_ratio")])
```

```
~/MyFirstR_Project/

> # Simple Linear Regression
> plot(price ~ weight, data = MyRegData)
> abline(Simp.LRModel, col= "blue")
> ggplot(Simp.LRModel, aes(x = weight, y = price)) + geom_point() + stat_smooth()
`geom_smooth()` using method = 'loess' and formula 'y ~ x'
> # Multiple Linear Regression
> plot(Mult.LRModel)
Hit <Return> to see next plot:
Hit <Return> to see next plot:
Hit <Return> to see next plot:
Hit <Return> to see next plot:
>
> ggpairs(MyRegData[, c("price", "weight", "length", "gear_ratio")])
```

```
MyData.xlsx     22 obs. of 5 variables
MyRegData       74 obs. of 12 variables
  make : chr "AMC Concord" "AMC Pacer" "AMC Spirit" "Buick Century" ...
  price : int 4099 4749 3799 4816 7827 5788 4453 5189 10372 4082 ...
  mpg : int 22 17 22 20 15 18 26 20 16 19 ...
  rep78 : int 3 3 NA 3 4 3 NA 3 3 3 ...
```

Fig. 7.8 Example of multiple linear regression graph in R using the ggpairs( ) function

**# Step 5**—Results Interpretation

The final step in the linear regression (simple and multiple) analysis is to understand and interpret the results of the tests for technical presentations or scientific use/ purposes.

By default, the hypothesis for testing whether certain variables are related (correlated) or not is: *IF* the p-value is less than or equal to 0.05 ($p \leq 0.05$), *THEN* linearity is assumed, *ELSE IF* the p-value is greater than 0.05 ($p > 0.05$) THEN association (linearity) is not assumed. Thus, linear regression test results with $p \leq 0.05$ (lower p-values) indicates that the predictor variable (independent) is related to the response variable (dependent). Whereas results with $p > 0.05$ (high p-value) means that changes or a change in the predictor variable(s) are not associated with a change in the response variable.

For example, as presented in the results below (coefficients) of the simple and multiple linear regressions which we reported in Figs. 7.5a and b, respectively; we can statistically conclude for the *simple linear regression* analysis that there exists a linear relationship between the "**price**" (dependent variable) and "**weight**" (independent variable) in the example dataset "**MyRegData**" with p-value of 7.416e-07 ($p \leq 0.05$), F = 29.42 (see Fig. 7.5a). Henceforth, we can confidently say or predict that an increase or decrease in the weight of the specified automobile will consequently lead to an increase or decrease in the price of the automobile, and vice and versa. In other words, the predictions are not construed as an event of chance, but are based on the statistical significance of the test.

```
Coefficients:

             Estimate  Std. Error  t value   Pr(>|t|)
(Intercept)   -6.7074  1174.4296   -0.006    0.995
 weight        2.0441     0.3768    5.424    7.42e-07 ***

 F-statistic: 29.42 on 1 and 72 DF,  p-value: 7.416e-07
```

Likewise, for the *Multiple Linear Regression* analysis (see: Fig. 7.5b), we can statistically conclude that in total ($p$ = 2.684e-07, F = 14.13) the predictor or independent variables (i.e., **weight, length,** and **gear-ratio**) have a significant relationship with "**price**" of the automobile. Although when we consider the differences among the individual (predictor or independent) variables, i.e., **weight** (p = 1.66e-05, t = 4.626), **length** (p = 0.00829, t = −2.717), and **gear-ratio** (p = 0.07216, t = 1.826), we can also find that the **gear_ratio** (where p = 0.07216) ($p > 0.05$) does not necessarily share a significant relationship with the "**price**" of the automobile in comparison to the other analyzed variables (**weight** and **length**) that does share a linear relationship with the "**price**" of the automobile.

```
Coefficients:
             Estimate   Std. Error  t value   Pr(>|t|)
(Intercept) 3665.664     5614.375   0.653     0.51596
 weight         5.665       1.224   4.626     1.66e-05 ***
 length      -105.300      38.755  -2.717     0.00829 **
 gear_ratio  1719.766     941.970   1.826     0.07216 .

F-statistic: 14.13 on 3 and 70 DF,  p-value: 2.684e-07
```

Accordingly, the above conclusions or interpretations of the results of the regression tests can also be visualized in the graphs represented in Figs. 7.7 and 7.8, respectively.

**Useful Tips**:

- The **Pr(>|t|)** or **p-value** (see Fig. 7.5a and b) is the probability that the user would get either a t-value as high (or higher) than the observed value(s) when the null hypothesis is true. In other words, when the coefficient is equal to zero or there is no relationship. Therefore, if the Pr(>|t|) is low, the coefficients are significant (i.e., significantly different from zero). Else if the Pr(>|t|) is high, the coefficients can consequently not come out to be significant.
- We can interpret the **t-value** as the larger the value (t-value), the less likely that the coefficient is not equal to zero by chance. Thus, the higher the t-value, the better the results.
- Standard **Error** is used to determine the distance between the point to the regression line in the model.
- Lastly, another valuable way of establishing how well a regression model fits the observed data is through the value of the **r-squared (R$^2$*).*** In most cases, an r-squared (R$^{2)}$ value of 0.5 and above are generally considered a better fit for the model.

Other useful functions or components the users can subsequently experiment as per the regression analysis and for further useful information and interpretation of the results are as follows:

- coefficients( ) computes the model coefficients. For example, in our example data coefficients(Simp.LRModel).
- confint(NameOfModel, level = 0.95) computes the confidence interval for the model parameter, e.g., confint(Mult.LRModel, level = 0.95).
- fitted( ) computes the predicted values, e.g., fitted(Simp.LRModel).
- residuals( ) computes the residuals, e.g., residuals(Mult.LRModel).
- anova( ) computes the analysis of variance table for the different variables, e.g., anova(Mult.LRModel).
- anova(Model1, Model2) also we can compare the results of different regression models, e.g., in our example case anova(Simp.LRModel, Mult.LRModel).

- vcov( ) computes the covariance matrix for the model parameters, e.g., vcov(Mult.LRModel).
- influence( ) computes the regression diagnostics, e.g., influence(Simp.LRModel).
- cor( ) computes the correlation coefficient between two specified variables, e.g., cor(Simp.LRModel).

## 7.3 Logistic Regression Analysis in R

Logistic regression (or logit models) is a great way to model the linear combination of *categorical* or *dichotomous* variables.

Just as with the many other statistical methods or analysis, the default hypothesis for testing whether certain categorical or binary variables are related (correlated) is: *IF* the p-value is less than or equal to 0.05 ($p \leq 0.05$), *THEN* association is assumed, *ELSE IF* the p-value is greater than 0.05 ($p > 0.05$) THEN association or relationship is not assumed.

Here, the authors will demonstrate to the readers how to conduct the different types of logistic regression analysis (i.e., binominal, multinominal, and ordinal) tests using the **glm( ), multinom( ), and polr( )** functions in R. We will show the readers the different computational procedures to conducting the three main types of logistic regression tests using the same steps we have outlined earlier in Fig. 7.1.

Now, let's download two example files that we will use to demonstrate the three different types of logistic regression analysis (the users are welcome to use any preexisting dataset or format so long as it meets the same criteria or type of variables).

As shown in Fig. 7.9, download the example files named "**TypeofCoverage.csv**" and "**VehChoicePrice.csv**" from the following address (http://www.ub.edu/rfa/R/regression_with_categorical_dependent_variables.html) and save the files on the computer.

***Note: The users can also directly access and download the example files from the following link or repository where the authors have uploaded all the example files used in the illustrations in this book: https://doi.org/https://doi.org/10.6084/m9.figshare.24728073.

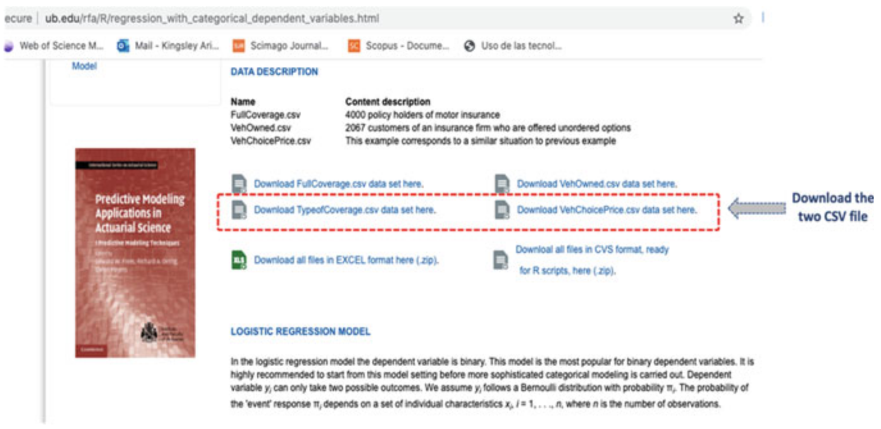
Once the user has downloaded the two CSV files and saved them on the local machine or computer, then we can proceed to conduct the logistic regression analysis.

To start, create a new RScript and name it "**LogisticRegressionTest**" or any name the user chooses (please note that the name chosen by the authors are only used to create and store the R objects for manipulation in our examples).

# **Step 1**—Install and load the required R packages

**Install and Load** the following *R packages* and *libraries*, as shown in Fig. 7.10 (Lines 3 to 31), which will be used to call the different R functions, data manipulations, and graphical visualizations for the logistic regression analysis.

**"aod", "mlogit", "nnet", "MASS", "rms", "DescTools", "manipulate", "reshape2", "ggplot2", "arm", "broom", "dplyr", "effects"**



ecure | ub.edu/rfa/R/regression_with_categorical_dependent_variables.html

DATA DESCRIPTION

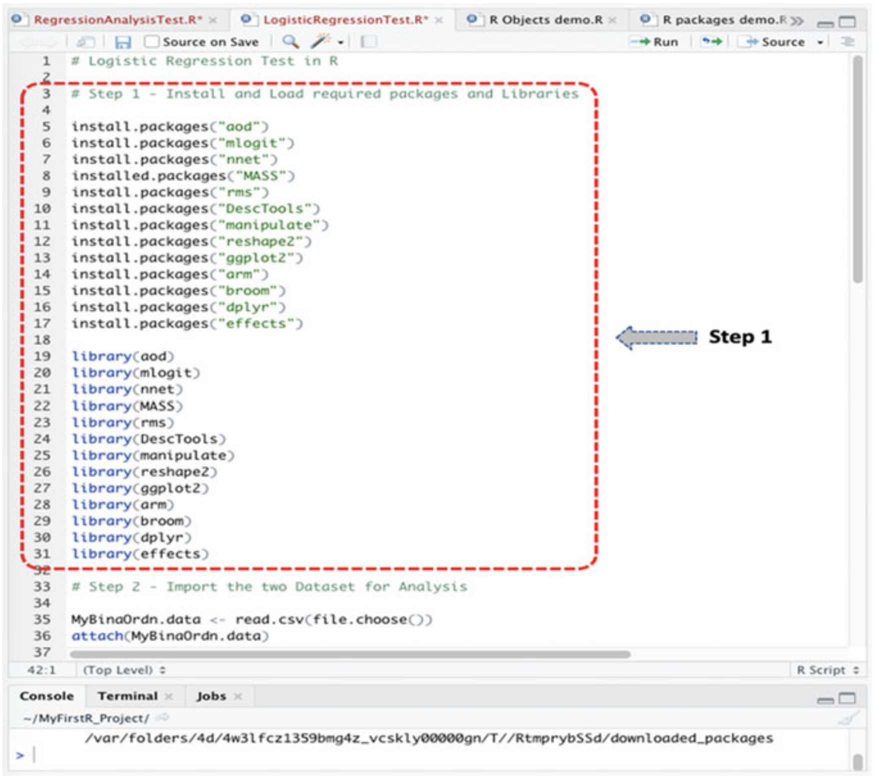
| Name | Content description |
|---|---|
| FullCoverage.csv | 4000 policy holders of motor insurance |
| VehOwned.csv | 2067 customers of an insurance firm who are offered unordered options |
| VehChoicePrice.csv | This example corresponds to a similar situation to previous example |

Download FullCoverage.csv data set here.
Download VehOwned.csv data set here.
Download TypeofCoverage.csv data set here.
Download VehChoicePrice.csv data set here.
Download all files in EXCEL format here (.zip).
Download all files in CVS format, ready for R scripts, here (.zip).

LOGISTIC REGRESSION MODEL

In the logistic regression model the dependent variable is binary. This model is the most popular for binary dependent variables. It is highly recommended to start from this model setting before more sophisticated categorical modeling is carried out. Dependent variable $y_i$ can only take two possible outcomes. We assume $y_i$ follows a Bernoulli distribution with probability $\pi_i$. The probability of the 'event' response $\pi_i$ depends on a set of individual characteristics $x_i$, $i = 1, \ldots, n$, where $n$ is the number of observations.

**Fig. 7.9** Example of binary, multinominal, and ordinal data download. (source: http://www.ub.edu/rfa/R/regression_with_categorical_dependent_variables.html)



```
# Logistic Regression Test in R

# Step 1 - Install and Load required packages and Libraries

install.packages("aod")
install.packages("mlogit")
install.packages("nnet")
installed.packages("MASS")
install.packages("rms")
install.packages("DescTools")
install.packages("manipulate")
install.packages("reshape2")
install.packages("ggplot2")
install.packages("arm")
install.packages("broom")
install.packages("dplyr")
install.packages("effects")

library(aod)
library(mlogit)
library(nnet)
library(MASS)
library(rms)
library(DescTools)
library(manipulate)
library(reshape2)
library(ggplot2)
library(arm)
library(broom)
library(dplyr)
library(effects)

# Step 2 - Import the two Dataset for Analysis

MyBinaOrdn.data <- read.csv(file.choose())
attach(MyBinaOrdn.data)
```

```
~/MyFirstR_Project/
      /var/folders/4d/4w31fcz1359bmg4z_vcskly00000gn/T//RtmprybSSd/downloaded_packages
>
```

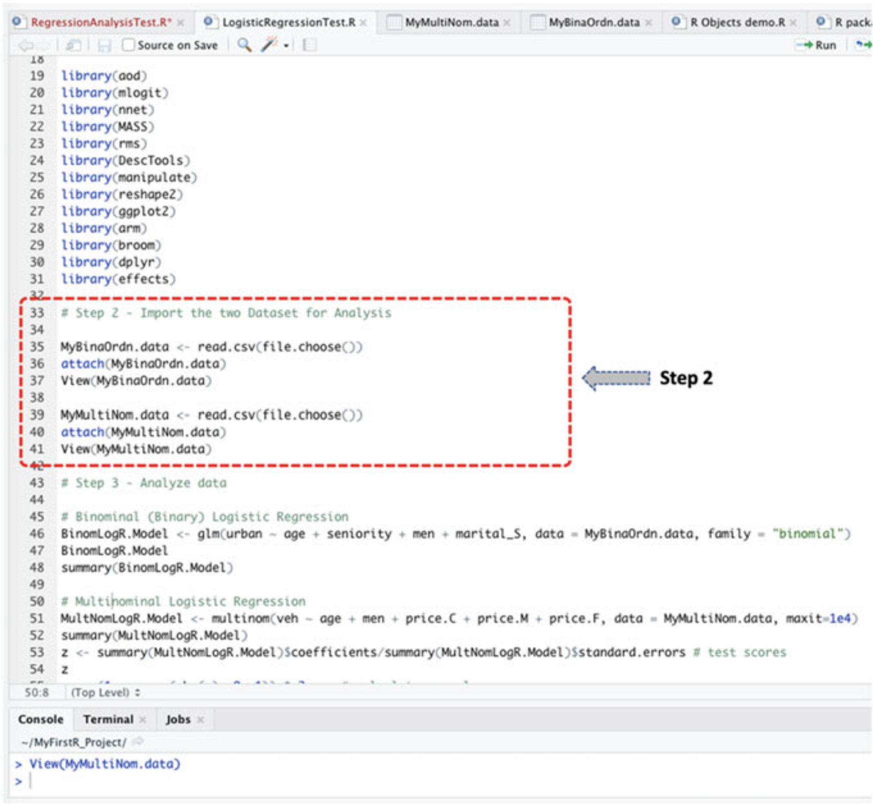
**Fig. 7.10** Installing R packages for logistic regression test and analysis

The syntax and code to install and load the required R packages are as follows:

```
install.packages("aod")
install.packages("mlogit")
install.packages("nnet")
installed.packages("MASS")
install.packages("rms")
install.packages("DescTools")
install.packages("manipulate")
install.packages("reshape2")
install.packages("ggplot2")
install.packages("arm")
install.packages("broom")
install.packages("dplyr")
install.packages("effects")

library(aod)
library(mlogit)
library(nnet)
library(MASS)
library(rms)
library(DescTools)
library(manipulate)
library(reshape2)
library(ggplot2)
library(arm)
library(broom)
library(dplyr)
library(effects)
```

# **Step 2**—Import and inspect example datasets for Logistic regression test

As highlighted and described in Fig. 7.11 (Step 2, Lines 33–41) and the code below, import the two datasets named "**TypeofCoverage.csv**" and "**VehChoicePrice.csv**" and store this in R objects named "**MyBinaOrdn.data**" and "**MyMultiNom.data**",



```
library(aod)
library(mlogit)
library(nnet)
library(MASS)
library(rms)
library(DescTools)
library(manipulate)
library(reshape2)
library(ggplot2)
library(arm)
library(broom)
library(dplyr)
library(effects)

# Step 2 - Import the two Dataset for Analysis

MyBinaOrdn.data <- read.csv(file.choose())
attach(MyBinaOrdn.data)
View(MyBinaOrdn.data)

MyMultiNom.data <- read.csv(file.choose())
attach(MyMultiNom.data)
View(MyMultiNom.data)

# Step 3 - Analyze data

# Binominal (Binary) Logistic Regression
BinomLogR.Model <- glm(urban ~ age + seniority + men + marital_S, data = MyBinaOrdn.data, family = "binomial")
BinomLogR.Model
summary(BinomLogR.Model)

# Multinominal Logistic Regression
MultNomLogR.Model <- multinom(veh ~ age + men + price.C + price.M + price.F, data = MyMultiNom.data, maxit=1e4)
summary(MultNomLogR.Model)
z <- summary(MultNomLogR.Model)$coefficients/summary(MultNomLogR.Model)$standard.errors # test scores
z
```

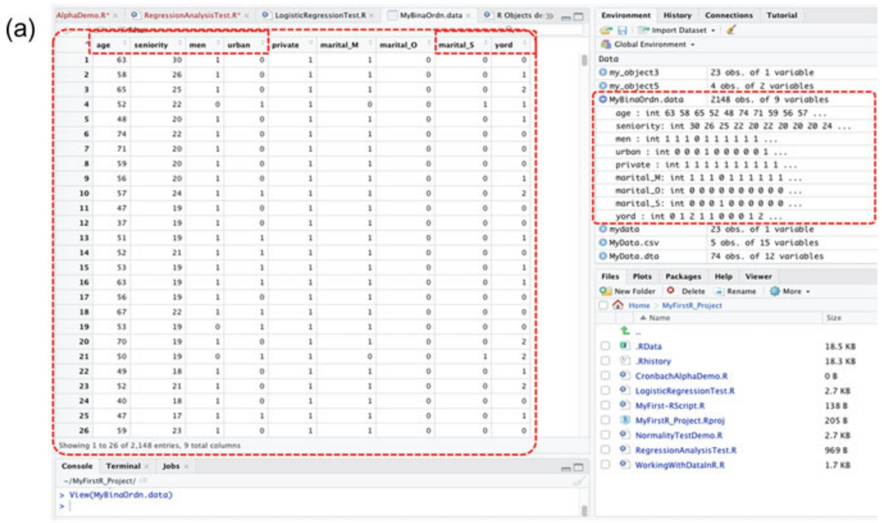
```
> View(MyMultiNom.data)
>
```

**Fig. 7.11** Importing data into R and storing them as R objects

respectively (remember you are welcome to use any name of your choice if you wish to do so).

```
MyBinaOrdn.data <- read.csv(file.choose())
attach(MyBinaOrdn.data)
View(MyBinaOrdn.data)

MyMultiNom.data <- read.csv(file.choose())
attach(MyMultiNom.data)
View(MyMultiNom.data)
```

Once successfully imported, the user will be able to view the details of the imported and stored files as highlighted and described in Fig. 7.12a and b.



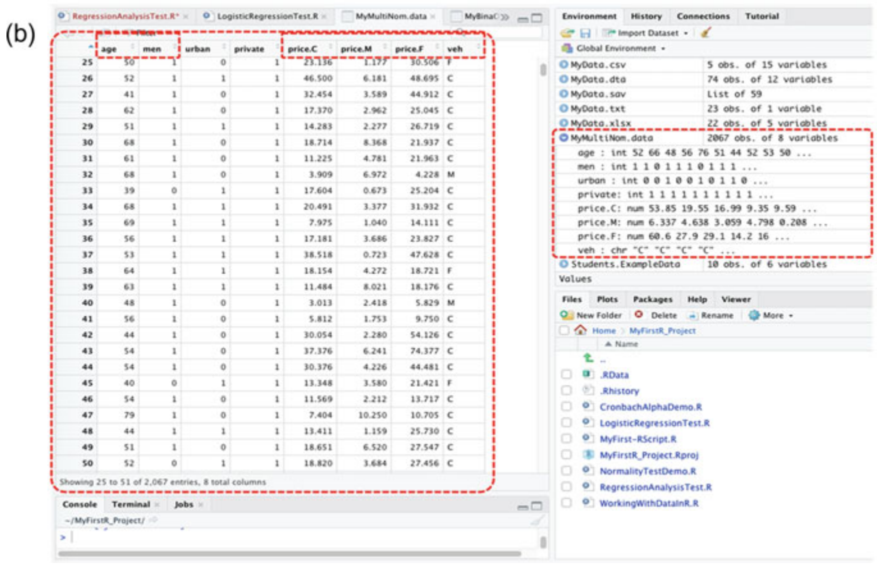
| | age | seniority | men | urban | private | marital_M | marital_O | marital_S | yord |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 63 | 30 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| 2 | 58 | 26 | 1 | 0 | 1 | 1 | 0 | 0 | 1 |
| 3 | 65 | 25 | 1 | 0 | 1 | 1 | 0 | 0 | 2 |
| 4 | 52 | 22 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 5 | 48 | 20 | 1 | 0 | 1 | 1 | 0 | 0 | 1 |
| 6 | 74 | 22 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| 7 | 71 | 20 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| 8 | 59 | 20 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| 9 | 56 | 20 | 1 | 0 | 1 | 1 | 0 | 0 | 1 |
| 10 | 57 | 24 | 1 | 1 | 1 | 1 | 0 | 0 | 2 |
| 11 | 47 | 19 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| 12 | 37 | 19 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| 13 | 51 | 19 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| 14 | 52 | 21 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 15 | 53 | 19 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| 16 | 63 | 19 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| 17 | 56 | 19 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| 18 | 67 | 22 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 19 | 53 | 19 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 20 | 70 | 19 | 1 | 0 | 1 | 1 | 0 | 0 | 2 |
| 21 | 50 | 19 | 0 | 1 | 1 | 0 | 0 | 1 | 2 |
| 22 | 49 | 18 | 1 | 0 | 1 | 1 | 0 | 0 | 1 |
| 23 | 52 | 21 | 1 | 0 | 1 | 1 | 0 | 0 | 2 |
| 24 | 40 | 18 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| 25 | 47 | 17 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| 26 | 59 | 23 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |

Showing 1 to 26 of 2,148 entries, 9 total columns

```
~/MyFirstR_Project/
> View(MyBinaOrdn.data)
>
```

```
Data
my_object3        23 obs. of 1 variable
my_object5        4 obs. of 2 variables
MyBinaOrdn.data   2148 obs. of 9 variables
  age : int 63 58 65 52 48 74 71 59 56 57 ...
  seniority: int 30 26 25 22 20 22 20 20 20 24 ...
  men : int 1 1 1 0 1 1 1 1 1 1 ...
  urban : int 0 0 0 1 0 0 0 0 0 1 ...
  private : int 1 1 1 1 1 1 1 1 1 1 ...
  marital_M: int 1 1 1 0 1 1 1 1 1 1 ...
  marital_O: int 0 0 0 0 0 0 0 0 0 0 ...
  marital_S: int 0 0 0 1 0 0 0 0 0 0 ...
  yord : int 0 1 2 1 1 0 0 0 1 2 ...
mydata            23 obs. of 1 variable
MyData.csv        5 obs. of 15 variables
MyData.dta        74 obs. of 12 variables
```

| Name | Size |
|---|---|
| .. | |
| .RData | 18.5 KB |
| .Rhistory | 18.3 KB |
| CronbachAlphaDemo.R | 0 B |
| LogisticRegressionTest.R | 2.7 KB |
| MyFirst-RScript.R | 138 B |
| MyFirstR_Project.Rproj | 205 B |
| NormalityTestDemo.R | 2.7 KB |
| RegressionAnalysisTest.R | 969 B |
| WorkingWithDataInR.R | 1.7 KB |



| | age | men | urban | private | price.C | price.M | price.F | veh |
|---|---|---|---|---|---|---|---|---|
| 25 | 50 | 1 | 0 | 1 | 23.136 | 1.177 | 30.506 | F |
| 26 | 52 | 1 | 1 | 1 | 46.500 | 6.181 | 48.695 | C |
| 27 | 41 | 1 | 0 | 1 | 32.454 | 3.589 | 44.912 | C |
| 28 | 62 | 1 | 0 | 1 | 17.370 | 2.962 | 25.045 | C |
| 29 | 51 | 1 | 1 | 1 | 14.283 | 2.277 | 26.719 | C |
| 30 | 68 | 1 | 0 | 1 | 18.714 | 8.368 | 21.937 | C |
| 31 | 61 | 1 | 0 | 1 | 11.225 | 4.781 | 21.963 | C |
| 32 | 68 | 1 | 0 | 1 | 3.909 | 6.972 | 4.228 | M |
| 33 | 39 | 0 | 1 | 1 | 17.604 | 0.673 | 25.204 | C |
| 34 | 68 | 1 | 1 | 1 | 20.491 | 3.377 | 31.932 | C |
| 35 | 69 | 1 | 1 | 1 | 7.975 | 1.040 | 14.111 | C |
| 36 | 56 | 1 | 1 | 1 | 17.181 | 3.686 | 23.827 | C |
| 37 | 53 | 1 | 1 | 1 | 38.518 | 0.723 | 47.628 | C |
| 38 | 64 | 1 | 1 | 1 | 18.154 | 4.272 | 18.721 | F |
| 39 | 63 | 1 | 1 | 1 | 11.484 | 8.021 | 18.176 | C |
| 40 | 48 | 1 | 0 | 1 | 3.013 | 2.418 | 5.829 | M |
| 41 | 56 | 1 | 0 | 1 | 5.812 | 1.753 | 9.750 | C |
| 42 | 44 | 1 | 0 | 1 | 30.054 | 2.280 | 54.126 | C |
| 43 | 54 | 1 | 0 | 1 | 37.376 | 6.241 | 74.377 | C |
| 44 | 54 | 1 | 0 | 1 | 30.376 | 4.226 | 44.481 | C |
| 45 | 40 | 0 | 1 | 1 | 13.348 | 3.580 | 21.421 | F |
| 46 | 54 | 1 | 0 | 1 | 11.569 | 2.212 | 13.717 | C |
| 47 | 79 | 1 | 0 | 1 | 7.404 | 10.250 | 10.705 | C |
| 48 | 44 | 1 | 1 | 1 | 13.411 | 1.159 | 25.730 | C |
| 49 | 51 | 1 | 0 | 1 | 18.651 | 6.520 | 27.547 | C |
| 50 | 52 | 0 | 1 | 1 | 18.820 | 3.684 | 27.456 | C |

Showing 25 to 51 of 2,067 entries, 8 total columns

```
~/MyFirstR_Project/
>
```

```
MyData.csv              5 obs. of 15 variables
MyData.dta              74 obs. of 12 variables
MyData.sav              List of 59
MyData.txt              23 obs. of 1 variable
MyData.xlsx             22 obs. of 5 variables
MyMultiNom.data         2067 obs. of 8 variables
  age : int 52 66 48 56 76 51 44 52 53 50 ...
  men : int 1 1 0 1 1 1 0 1 1 1 ...
  urban : int 0 0 1 0 0 1 0 1 1 0 ...
  private: int 1 1 1 1 1 1 1 1 1 1 ...
  price.C: num 53.85 19.55 16.99 9.35 9.59 ...
  price.M: num 6.337 4.638 3.059 4.798 0.208 ...
  price.F: num 60.6 27.9 29.1 14.2 16 ...
  veh : chr "C" "C" "C" "C" ...
Students.ExampleData    10 obs. of 6 variables
Values
```

| Name |
|---|
| .. |
| .RData |
| .Rhistory |
| CronbachAlphaDemo.R |
| LogisticRegressionTest.R |
| MyFirst-RScript.R |
| MyFirstR_Project.Rproj |
| NormalityTestDemo.R |
| RegressionAnalysisTest.R |
| WorkingWithDataInR.R |

**Fig. 7.12** **a** Example of dataset with categorical (binary) and ordinal variables imported in R. **b** Example of dataset with multinominal variable imported in R

# **Step 3**—Data analysis and implementation of logistic regression model

As explained in the introduction section (Sect. 7.1), the authors will illustrate how to conduct the three main types of logistic regression analysis. This includes

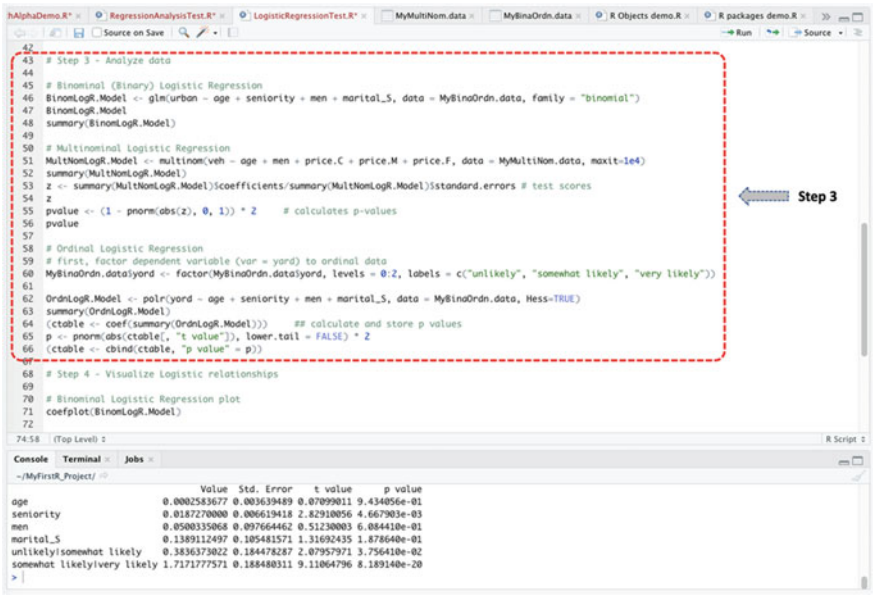
(i) **Binomial (binary) logistic regression** which uses dependent variables that only have two possible types or categories of 0 or 1.
(ii) **Multinomial logistic regression** uses dependent variables that have more than 2 (i.e., 3 or more) possible categories but are not ordered.
(iii) **Ordinal logistic regression** uses dependent variables that have 3 or more possible categories but with ordered categories.

To demonstrate how to conduct the following tests or experiment using the example datasets "**TypeofCoverage.csv**" and "**VehChoicePrice.csv**" which we have stored or defined as R objects, "**MyBinaOrdn.data**" and "**MyMultiNom.data**", in R, we will analyze the variables we have highlighted in the datasets (see highlighted columns in Figs. 7.12a and b).

To do this, we will conduct the following test or hypothesis:

- **Binominal (binary) logistic regression**: We will test the relationship that the following independent (predictor) variables—age (var = **age**), position (var = **seniority**), gender (var = **men**), and marital status (var = **marital_S**)—share with the type of coverage the customers of the insurance policies get (var = **urban**). Note that the var = urban is the binary dependent (response) variable: where we assume 1 = private and 0 = public). Data = **MyBinaOrdn.data** (stored as R object—see Fig. 7.12a).
- **Multinominal Logistic regression**: We will test the relationship that the following independent (predictor) variables—age (var = **age**), gender (var = **men**), and price of policies (vars = **price.C, price.M, price.F**)—have with the type of vehicle the customer purchases or acquires (var = **veh**). Note that var = veh is the multinominal dependent (response) variable with three unordered levels C, M, and F. Data = **MyMultiNom.data** (see Fig. 7.12b).
- **Ordinal Logistic regression**: We will test the relationship that the following independent (predictor) variables—age (var = **age**), position (var = **seniority**), gender (var = **men**), and marital status (var = **marital_S**)—share with the likelihood that the customers will buy the insurance policies (var = **yord**). Note that the var = yord is the ordinal dependent (response) variable with 3 ordered levels: where we assumed 0 = unlikely, 1 = somewhat likely, and 2 = very likely. Data = **MyBinaOrdn.data** (see Fig. 7.12a).

As shown in the codes provided by the authors below (see Fig. 7.13, Step 3, Lines 43 to 66), the syntax to performing the aforelisted tests (binominal, multinominal, and ordinal logistic regression) in R is as follows:



```
# Step 3 - Analyze data

# Binominal (Binary) Logistic Regression
BinomLogR.Model <- glm(urban ~ age + seniority + men + marital_S, data = MyBinaOrdn.data, family = "binomial")
BinomLogR.Model
summary(BinomLogR.Model)

# Multinominal Logistic Regression
MultNomLogR.Model <- multinom(veh ~ age + men + price.C + price.M + price.F, data = MyMultiNom.data, maxit=1e4)
summary(MultNomLogR.Model)
z <- summary(MultNomLogR.Model)$coefficients/summary(MultNomLogR.Model)$standard.errors # test scores
z
pvalue <- (1 - pnorm(abs(z), 0, 1)) * 2      # calculates p-values
pvalue

# Ordinal Logistic Regression
# first, factor dependent variable (var = yord) to ordinal data
MyBinaOrdn.data$yord <- factor(MyBinaOrdn.data$yord, levels = 0:2, labels = c("unlikely", "somewhat likely", "very likely"))

OrdnLogR.Model <- polr(yord ~ age + seniority + men + marital_S, data = MyBinaOrdn.data, Hess=TRUE)
summary(OrdnLogR.Model)
(ctable <- coef(summary(OrdnLogR.Model)))      ## calculate and store p values
p <- pnorm(abs(ctable[, "t value"]), lower.tail = FALSE) * 2
(ctable <- cbind(ctable, "p value" = p))

# Step 4 - Visualize Logistic relationships

# Binominal Logistic Regression plot
coefplot(BinomLogR.Model)
```

```
                                  Value  Std. Error    t value      p value
age                        0.0002583677 0.003639489 0.07099011 9.434056e-01
seniority                  0.0187270000 0.006619418 2.82910056 4.667903e-03
men                        0.0500335068 0.097664462 0.51230003 6.084410e-01
marital_S                  0.1389112497 0.105481571 1.31692435 1.878640e-01
unlikely|somewhat likely   0.3836373022 0.184478287 2.07957971 3.756410e-02
somewhat likely|very likely 1.7171777571 0.188480311 9.11064796 8.189140e-20
> 
```

**Fig. 7.13** Binominal, multinominal, and ordinal logistic regression tests in R

```
# Binominal (Binary) Logistic Regression
BinomLogR.Model <- glm(urban ~ age + seniority + men + marital_S, data =
MyBinaOrdn.data, family = "binomial")
BinomLogR.Model
summary(BinomLogR.Model)


# Multinominal Logistic Regression
MultNomLogR.Model <- multinom(veh ~ age + men + price.C + price.M +
price.F, data = MyMultiNom.data, maxit=1e4)
summary(MultNomLogR.Model)
z <-
summary(MultNomLogR.Model)$coefficients/summary(MultNomLogR.Model)$stan
dard.errors # test scores
z
pvalue <- (1 - pnorm(abs(z), 0, 1)) * 2      # calculates p-values
pvalue
```

```
# Ordinal Logistic Regression
# first, factor dependent variable (var = yard) to ordinal data
MyBinaOrdn.data$yord <- factor(MyBinaOrdn.data$yord, levels = 0:2, labels
= c("unlikely", "somewhat likely", "very likely"))


OrdnLogR.Model <- polr(yord ~ age + seniority + men + marital_S, data =
MyBinaOrdn.data, Hess=TRUE)
summary(OrdnLogR.Model)
(ctable <- coef(summary(OrdnLogR.Model)))      ## calculate and store p
values
p <- pnorm(abs(ctable[, "t value"]), lower.tail = FALSE) * 2
(ctable <- cbind(ctable, "p value" = p))
```

When the user has successfully run the codes or tests described in Step 3, Fig. 7.13, they will be presented with the results of the logistic regression models in the Console, which is similar to the ones the authors have reported in Figs. 7.14a, b, and c, respectively.

As gathered and explained in the Figs. 7.14a, b, and c, we conducted a binominal, mutinominal, and ordinal logistic regression analysis as defined in Step 3, and stored the results of the tests in R objects we called “BinomLogR.Model”, “MultNomLogR.Model” and “OrdnLogR.Model”, respectively.

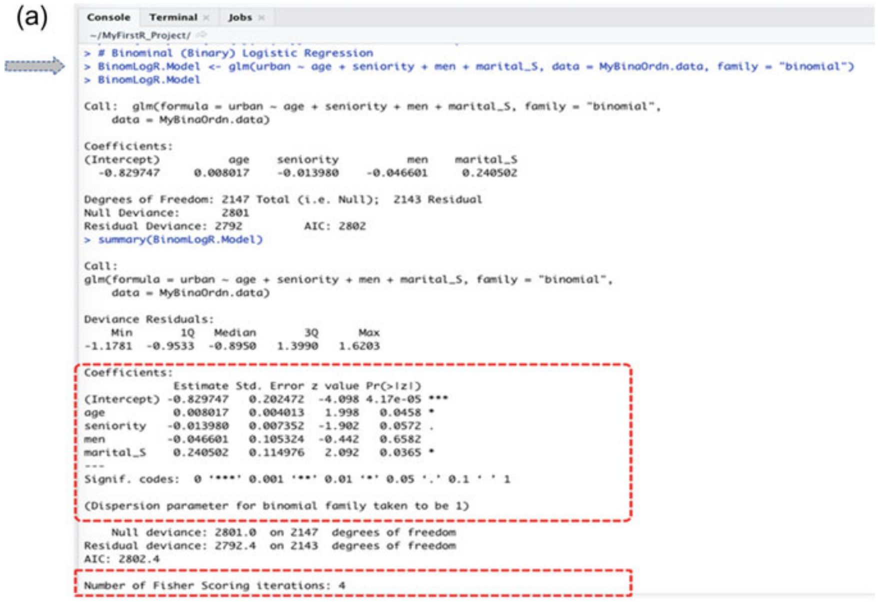
# **Step 4**—Visualize the results of the logistic regression analysis

As outlined in Fig. 7.1, another great way to check the relationship that exist between variables in R is by plotting them as graph. We will show how to plot the results of the logistic regression tests in R.

To do this, we will use the **coefplot( ), broom( ), plot( ), effects( ),** and **ggplot( )** functions to visualize the results.

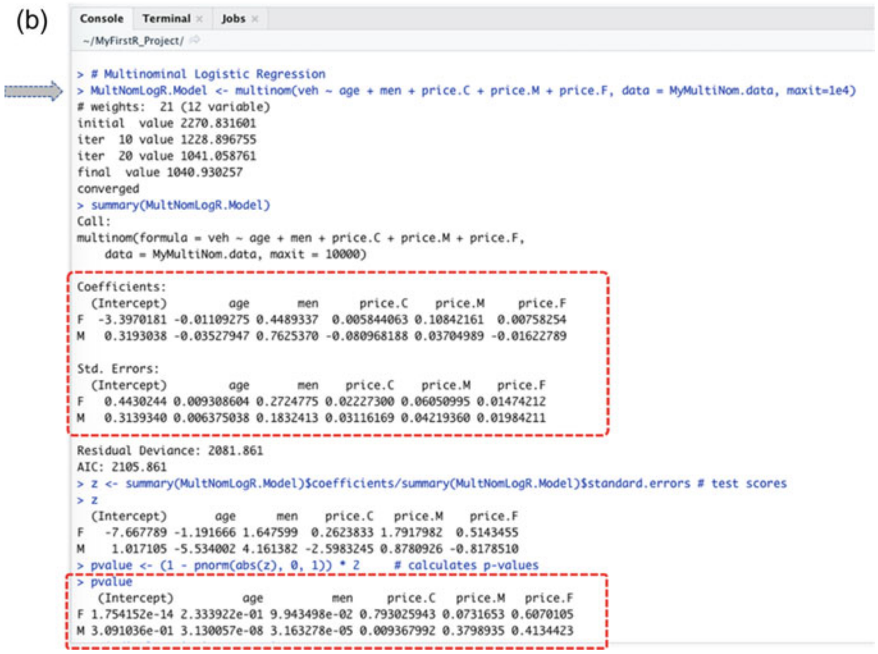
The syntax for plotting the respective graphs of the logistic regression tests or data is as shown in the codes below (see Fig. 7.15, Lines 68 to 86), and the resultant plots represented in Figs. 7.16a, b, and c, respectively.

(a)



```
Console   Terminal   Jobs
~/MyFirstR_Project/
> # Binominal (Binary) Logistic Regression
> BinomLogR.Model <- glm(urban ~ age + seniority + men + marital_S, data = MyBinaOrdn.data, family = "binomial")
> BinomLogR.Model

Call:  glm(formula = urban ~ age + seniority + men + marital_S, family = "binomial",
    data = MyBinaOrdn.data)

Coefficients:
(Intercept)          age    seniority          men    marital_S
  -0.829747     0.008017    -0.013980    -0.046601     0.240502

Degrees of Freedom: 2147 Total (i.e. Null);  2143 Residual
Null Deviance:      2801
Residual Deviance: 2792        AIC: 2802
> summary(BinomLogR.Model)

Call:
glm(formula = urban ~ age + seniority + men + marital_S, family = "binomial",
    data = MyBinaOrdn.data)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.1781  -0.9533  -0.8950   1.3990   1.6203

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.829747   0.202472  -4.098 4.17e-05 ***
age          0.008017   0.004013   1.998   0.0458 *
seniority   -0.013980   0.007352  -1.902   0.0572 .
men         -0.046601   0.105324  -0.442   0.6582
marital_S    0.240502   0.114976   2.092   0.0365 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2801.0  on 2147  degrees of freedom
Residual deviance: 2792.4  on 2143  degrees of freedom
AIC: 2802.4

Number of Fisher Scoring iterations: 4
```

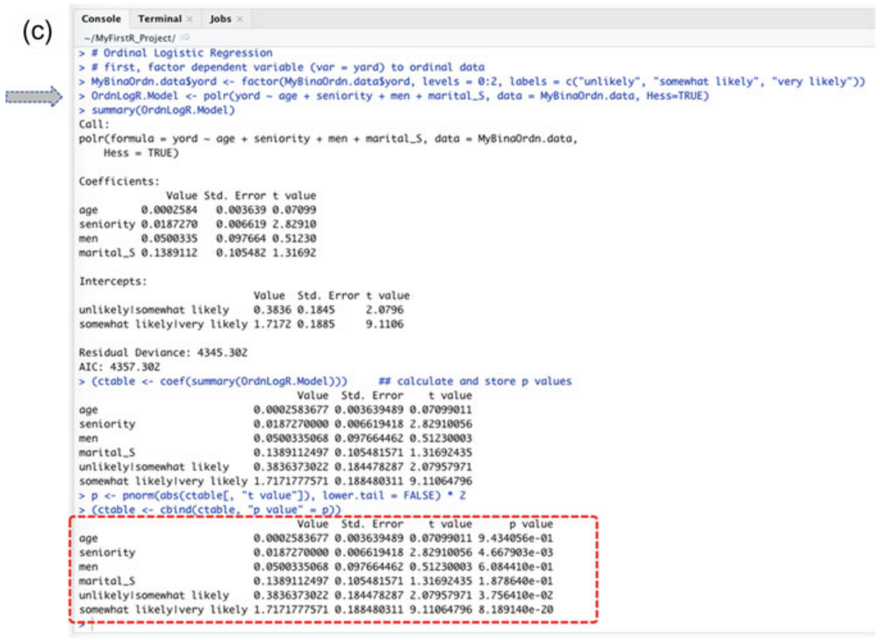
(b)



```
Console   Terminal   Jobs
~/MyFirstR_Project/

> # Multinominal Logistic Regression
> MultNomLogR.Model <- multinom(veh ~ age + men + price.C + price.M + price.F, data = MyMultiNom.data, maxit=1e4)
# weights:  21 (12 variable)
initial  value 2270.831601
iter  10 value 1228.896755
iter  20 value 1041.058761
final  value 1040.930257
converged
> summary(MultNomLogR.Model)
Call:
multinom(formula = veh ~ age + men + price.C + price.M + price.F,
    data = MyMultiNom.data, maxit = 10000)

Coefficients:
  (Intercept)         age       men      price.C    price.M     price.F
F  -3.3970181 -0.01109275 0.4489337  0.005844063 0.10842161  0.00758254
M   0.3193038 -0.03527947 0.7625370 -0.080968188 0.03704989 -0.01622789

Std. Errors:
  (Intercept)         age       men    price.C    price.M    price.F
F   0.4430244 0.009308604 0.2724775 0.02227300 0.06050995 0.01474212
M   0.3139340 0.006375038 0.1832413 0.03116169 0.04219360 0.01984211

Residual Deviance: 2081.861
AIC: 2105.861
> z <- summary(MultNomLogR.Model)$coefficients/summary(MultNomLogR.Model)$standard.errors # test scores
> z
  (Intercept)       age      men    price.C   price.M    price.F
F   -7.667789 -1.191666 1.647599  0.2623833 1.7917982  0.5143455
M    1.017105 -5.534002 4.161382 -2.5983245 0.8780926 -0.8178510
> pvalue <- (1 - pnorm(abs(z), 0, 1)) * 2      # calculates p-values
> pvalue
   (Intercept)          age          men     price.C   price.M   price.F
F 1.754152e-14 2.333922e-01 9.943498e-02 0.793025943 0.0731653 0.6070105
M 3.091036e-01 3.130057e-08 3.163278e-05 0.009367992 0.3798935 0.4134423
```

**Fig. 7.14** **a** Result of binominal (binary) logistic regression analysis in R. **b** Result of the multinominal logistic regression test in R. **c** Result of the ordinal logistic regression test in R

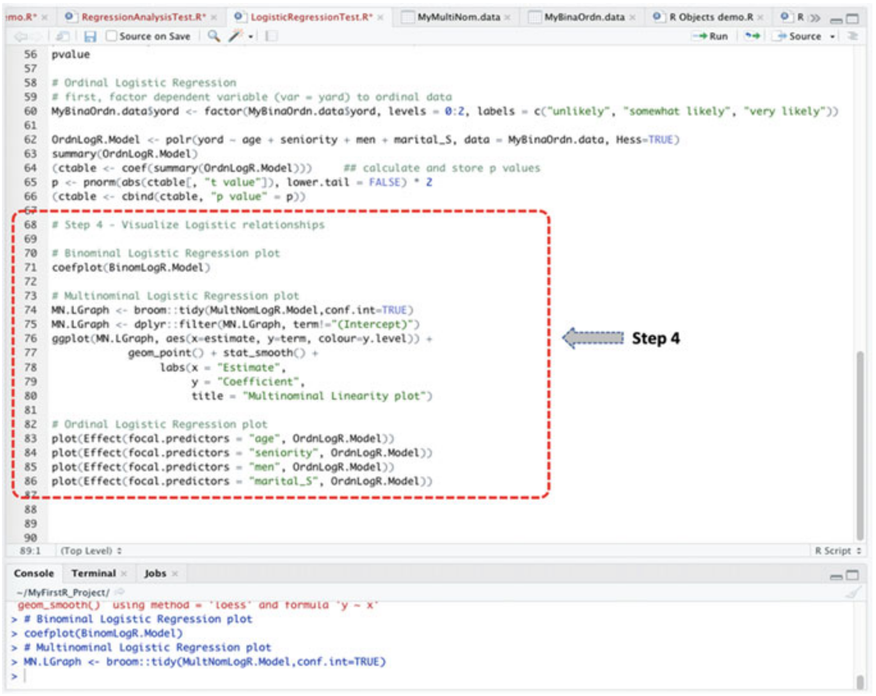
(c)



```
> # Ordinal Logistic Regression
> # first, factor dependent variable (var = yard) to ordinal data
> MyBinaOrdn.data$yord <- factor(MyBinaOrdn.data$yord, levels = 0:2, labels = c("unlikely", "somewhat likely", "very likely"))
> OrdnLogR.Model <- polr(yord ~ age + seniority + men + marital_S, data = MyBinaOrdn.data, Hess=TRUE)
> summary(OrdnLogR.Model)
Call:
polr(formula = yord ~ age + seniority + men + marital_S, data = MyBinaOrdn.data,
    Hess = TRUE)

Coefficients:
              Value Std. Error t value
age       0.0002584   0.003639 0.07099
seniority 0.0187270   0.006619 2.82910
men       0.0500335   0.097664 0.51230
marital_S 0.1389112   0.105482 1.31692

Intercepts:
                            Value  Std. Error t value
unlikely|somewhat likely    0.3836 0.1845     2.0796
somewhat likely|very likely 1.7172 0.1885     9.1106

Residual Deviance: 4345.302
AIC: 4357.302
> (ctable <- coef(summary(OrdnLogR.Model)))     ## calculate and store p values
                                Value  Std. Error    t value
age                         0.0002583677 0.003639489 0.07099011
seniority                   0.0187270000 0.006619418 2.82910056
men                         0.0500335068 0.097664462 0.51230003
marital_S                   0.1389112497 0.105481571 1.31692435
unlikely|somewhat likely    0.3836373022 0.184478287 2.07957971
somewhat likely|very likely 1.7171777571 0.188480311 9.11064796
> p <- pnorm(abs(ctable[, "t value"]), lower.tail = FALSE) * 2
> (ctable <- cbind(ctable, "p value" = p))
                                Value  Std. Error    t value      p value
age                         0.0002583677 0.003639489 0.07099011 9.434056e-01
seniority                   0.0187270000 0.006619418 2.82910056 4.667903e-03
men                         0.0500335068 0.097664462 0.51230003 6.084410e-01
marital_S                   0.1389112497 0.105481571 1.31692435 1.878640e-01
unlikely|somewhat likely    0.3836373022 0.184478287 2.07957971 3.756410e-02
somewhat likely|very likely 1.7171777571 0.188480311 9.11064796 8.189140e-20
>
```

**Fig. 7.14** (continued)

```
# Binominal Logistic Regression plot
coefplot(BinomLogR.Model)


# Multinominal Logistic Regression plot
MN.LGraph <- broom::tidy(MultNomLogR.Model,conf.int=TRUE)
MN.LGraph <- dplyr::filter(MN.LGraph, term!="(Intercept)")
ggplot(MN.LGraph, aes(x=estimate, y=term, colour=y.level)) +
            geom_point() + stat_smooth() +
                 labs(x = "Estimate",
                      y = "Coefficient",
                      title = "Multinominal Linearity plot")


# Ordinal Logistic Regression plot
plot(Effect(focal.predictors = "age", OrdnLogR.Model))
plot(Effect(focal.predictors = "seniority", OrdnLogR.Model))
plot(Effect(focal.predictors = "men", OrdnLogR.Model))
plot(Effect(focal.predictors = "marital_S", OrdnLogR.Model))
```



```
pvalue

# Ordinal Logistic Regression
# first, factor dependent variable (var = yard) to ordinal data
MyBinaOrdn.data$yord <- factor(MyBinaOrdn.data$yord, levels = 0:2, labels = c("unlikely", "somewhat likely", "very likely"))

OrdnLogR.Model <- polr(yord ~ age + seniority + men + marital_S, data = MyBinaOrdn.data, Hess=TRUE)
summary(OrdnLogR.Model)
(ctable <- coef(summary(OrdnLogR.Model)))      ## calculate and store p values
p <- pnorm(abs(ctable[, "t value"]), lower.tail = FALSE) * 2
(ctable <- cbind(ctable, "p value" = p))

# Step 4 - Visualize Logistic relationships

# Binominal Logistic Regression plot
coefplot(BinomLogR.Model)

# Multinominal Logistic Regression plot
MN.LGraph <- broom::tidy(MultNomLogR.Model,conf.int=TRUE)
MN.LGraph <- dplyr::filter(MN.LGraph, term!="(Intercept)")
ggplot(MN.LGraph, aes(x=estimate, y=term, colour=y.level)) +
           geom_point() + stat_smooth() +
                labs(x = "Estimate",
                     y = "Coefficient",
                     title = "Multinominal Linearity plot")

# Ordinal Logistic Regression plot
plot(Effect(focal.predictors = "age", OrdnLogR.Model))
plot(Effect(focal.predictors = "seniority", OrdnLogR.Model))
plot(Effect(focal.predictors = "men", OrdnLogR.Model))
plot(Effect(focal.predictors = "marital_S", OrdnLogR.Model))
```

```
~/MyFirstR_Project/
geom_smooth() using method = 'loess' and formula 'y ~ x'
> # Binominal Logistic Regression plot
> coefplot(BinomLogR.Model)
> # Multinominal Logistic Regression plot
> MN.LGraph <- broom::tidy(MultNomLogR.Model,conf.int=TRUE)
> 
```

**Fig. 7.15** Plotting the result of logistic regression analysis in R

## # **Step 5**—Interpretation of the logistic regression analysis results

The last step in our analysis is to interpret the different logistic regression models or methods in order to help the readers understand the results of the test.

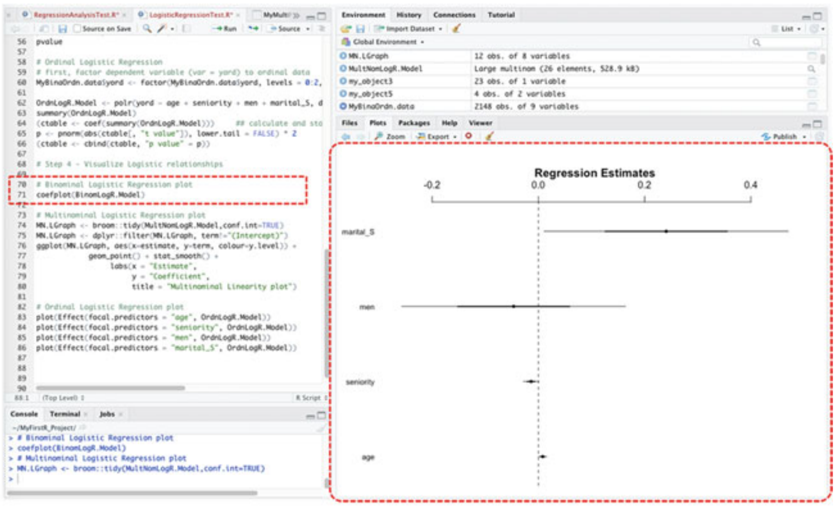
By default, the hypothesis for testing is: *IF* the p-value of the coefficients or estimate of linearity is less than or equal to 0.05 ($p \leq 0.05$), THEN the output is assumed to be statistically significant, *ELSE IF* the p-value is greater than 0.05 ($p > 0.05$) THEN the estimate of linearity is not significant.

We explain the results as follows:

**(I). Binominal (Binary) Logistic Regression Result**.

```
Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.829747   0.202472  -4.098 4.17e-05 ***
age          0.008017   0.004013   1.998   0.0458 *
seniority   -0.013980   0.007352  -1.902   0.0572 .
men         -0.046601   0.105324  -0.442   0.6582
marital_S    0.240502   0.114976   2.092   0.0365 *
```

(a)

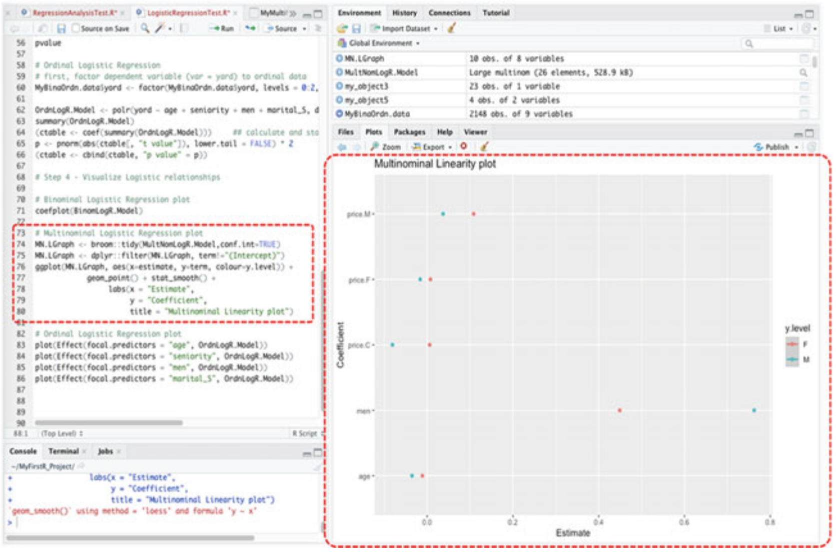


```
pvalue

# Ordinal Logistic Regression
# first, factor dependent variable (var = yord) to ordinal data
MyBinaOrdn.data$yord <- factor(MyBinaOrdn.data$yord, levels = 0:2,

OrdnLogR.Model <- polr(yord ~ age + seniority + men + marital_S, d
summary(OrdnLogR.Model)
(ctable <- coef(summary(OrdnLogR.Model)))      ## calculate and sto
p <- pnorm(abs(ctable[, "t value"]), lower.tail = FALSE) * 2
(ctable <- cbind(ctable, "p value" = p))

# Step 4 - Visualize Logistic relationships

# Binominal Logistic Regression plot
coefplot(BinomLogR.Model)

# Multinominal Logistic Regression plot
MN.LGraph <- broom::tidy(MultNomLogR.Model,conf.int=TRUE)
MN.LGraph <- dplyr::filter(MN.LGraph, term!="(Intercept)")
ggplot(MN.LGraph, aes(x=estimate, y=term, colour=y.level)) +
             geom_point() + stat_smooth() +
                 labs(x = "Estimate",
                      y = "Coefficient",
                      title = "Multinominal Linearity plot")

# Ordinal Logistic Regression plot
plot(Effect(focal.predictors = "age", OrdnLogR.Model))
plot(Effect(focal.predictors = "seniority", OrdnLogR.Model))
plot(Effect(focal.predictors = "men", OrdnLogR.Model))
plot(Effect(focal.predictors = "marital_S", OrdnLogR.Model))
```

```
~/MyFirstR_Project/
> # Binominal Logistic Regression plot
> coefplot(BinomLogR.Model)
> # Multinominal Logistic Regression plot
> MN.LGraph <- broom::tidy(MultNomLogR.Model,conf.int=TRUE)
>
```

| | |
|---|---|
| MN.LGraph | 12 obs. of 8 variables |
| MultNomLogR.Model | Large multinom (26 elements, 528.9 kB) |
| my_object3 | 23 obs. of 1 variable |
| my_object5 | 4 obs. of 2 variables |
| MyBinaOrdn.data | 2148 obs. of 9 variables |

(b)



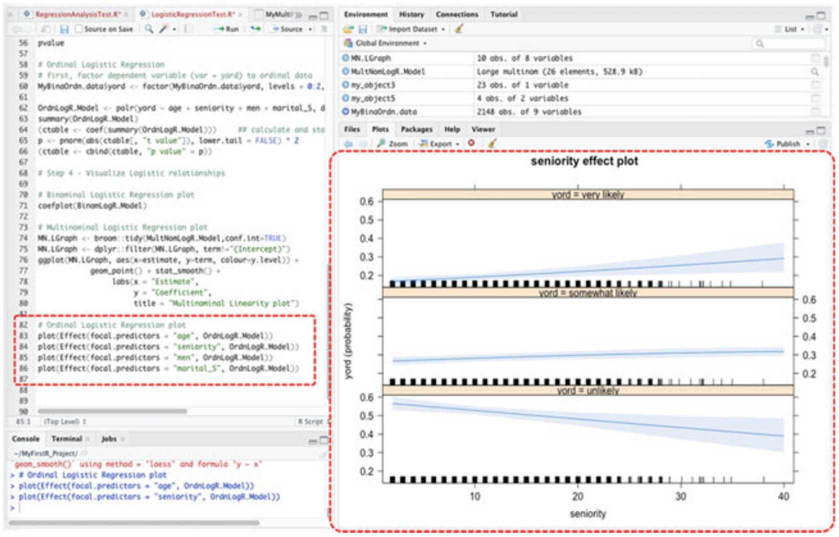
```
pvalue

# Ordinal Logistic Regression
# first, factor dependent variable (var = yord) to ordinal data
MyBinaOrdn.data$yord <- factor(MyBinaOrdn.data$yord, levels = 0:2,

OrdnLogR.Model <- polr(yord ~ age + seniority + men + marital_S, d
summary(OrdnLogR.Model)
(ctable <- coef(summary(OrdnLogR.Model)))      ## calculate and sto
p <- pnorm(abs(ctable[, "t value"]), lower.tail = FALSE) * 2
(ctable <- cbind(ctable, "p value" = p))

# Step 4 - Visualize Logistic relationships

# Binominal Logistic Regression plot
coefplot(BinomLogR.Model)

# Multinominal Logistic Regression plot
MN.LGraph <- broom::tidy(MultNomLogR.Model,conf.int=TRUE)
MN.LGraph <- dplyr::filter(MN.LGraph, term!="(Intercept)")
ggplot(MN.LGraph, aes(x=estimate, y=term, colour=y.level)) +
             geom_point() + stat_smooth() +
                 labs(x = "Estimate",
                      y = "Coefficient",
                      title = "Multinominal Linearity plot")

# Ordinal Logistic Regression plot
plot(Effect(focal.predictors = "age", OrdnLogR.Model))
plot(Effect(focal.predictors = "seniority", OrdnLogR.Model))
plot(Effect(focal.predictors = "men", OrdnLogR.Model))
plot(Effect(focal.predictors = "marital_S", OrdnLogR.Model))
```

```
~/MyFirstR_Project/
+                 labs(x = "Estimate",
+                      y = "Coefficient",
+                      title = "Multinominal Linearity plot")
`geom_smooth()` using method = 'loess' and formula 'y ~ x'
>
```

| | |
|---|---|
| MN.LGraph | 10 obs. of 8 variables |
| MultNomLogR.Model | Large multinom (26 elements, 528.9 kB) |
| my_object3 | 23 obs. of 1 variable |
| my_object5 | 4 obs. of 2 variables |
| MyBinaOrdn.data | 2148 obs. of 9 variables |

**Fig. 7.16** **a** Binominal (binary) logistic regression plot example in R using the coefplot( ) function. **b** Multinominal logistic regression plot example in R using the broom( ) and ggplot( ) functions. **c** Ordinal logistic regression plot example in R using the plot( ) and effects( ) function

As highlighted in Fig. 7.14a and the above result, the coefficient value of -0.829747 is the log(odds) estimate for the type of coverage the customers of the insurance policies are likely to get (i.e., var = **urban**, where we assume 1 = private and 0 = public) when taking into account the estimates of the predictor variables (age,

(c)

```
pvalue

# Ordinal Logistic Regression
# first, factor dependent variable (var = yord) to ordinal data
MyBinaOrdn.data$yord <- factor(MyBinaOrdn.data$yord, levels = 0:2,

OrdnLogR.Model <- polr(yord ~ age + seniority + men + marital_S, d
summary(OrdnLogR.Model)
(ctable <- coef(summary(OrdnLogR.Model)))      ## calculate and sto
p <- pnorm(abs(ctable[, "t value"]), lower.tail = FALSE) * 2
(ctable <- cbind(ctable, "p value" = p))

# Step 4 - Visualize Logistic relationships

# Binominal Logistic Regression plot
coefplot(BinomLogR.Model)

# Multinominal Logistic Regression plot
MN.LGraph <- broom::tidy(MultNomLogR.Model,conf.int=TRUE)
MN.LGraph <- dplyr::filter(MN.LGraph, term!="(Intercept)")
ggplot(MN.LGraph, aes(x=estimate, y=term, colour=y.level)) +
              geom_point() + stat_smooth() +
                  labs(x = "Estimate",
                       y = "Coefficient",
                       title = "Multinominal Linearity plot")

# Ordinal Logistic Regression plot
plot(Effect(focal.predictors = "age", OrdnLogR.Model))
plot(Effect(focal.predictors = "seniority", OrdnLogR.Model))
plot(Effect(focal.predictors = "men", OrdnLogR.Model))
plot(Effect(focal.predictors = "marital_S", OrdnLogR.Model))
```

```
MN.LGraph            10 obs. of 8 variables
MultNomLogR.Model    Large multinom (26 elements, 528.9 kB)
my_object3           23 obs. of 1 variable
my_object5           4 obs. of 2 variables
MyBinaOrdn.data      2148 obs. of 9 variables
```

```
~/MyFirstR_Project/
`geom_smooth()` using method = 'loess' and formula 'y ~ x'
> # Ordinal Logistic Regression plot
> plot(Effect(focal.predictors = "age", OrdnLogR.Model))
> plot(Effect(focal.predictors = "seniority", OrdnLogR.Model))
>
```

**Fig. 7.16** (continued)

seniority, men, marital_S). This result can be interpreted to be statistically significant, with p-value of 4.17e-05 ($p \leq 0.05$) and t-value (z) of -4.098.

Also, when considering the coefficient of the log(odds ratio) for the predictor (explanatory or independent) variables, we can find that they are mostly also statistically significant (var = **age**, $p = 0.0458$, z = 1.998), (var = **seniority**, $p = 0.0572$, z = -1.902), and (var = **marital_S**, $p = 0.0365$, z = 2.092), except for the gender variable (var = **men,** $p = 0.6582$, z = -0.442) that presented to be non-significant in the implemented model.

Therefore, we can draw from the example data (data = **MyBinaOrdn.data**) that the gender of the customers is not a major predictor or are not associated to the type of insurance policies the customers will get based on the data or model (model = BinomLogR.Model), whereas the age, position, and marital status are associated to the type of insurance policies the customers are likely to get.

**(Ii). Multinominal Logistic Regression Result**

```
Coefficients:
    (Intercept)    age         men        price.C      price.M     price.F
F  -3.3970181 -0.01109275 0.4489337  0.005844063   0.10842161   0.00758254
M   0.3193038 -0.03527947 0.7625370 -0.080968188   0.03704989  -0.01622789

Z (t-value)
    (Intercept)     age         men        price.C     price.M     price.F
F    -7.667789    -1.191666   1.647599   0.2623833  1.7917982   0.5143455
M     1.017105    -5.534002   4.161382  -2.5983245  0.8780926  -0.8178510

> pvalue
    (Intercept)     age          men          price.C     price.M    price.F
F 1.754152e-14 2.333922e-01 9.943498e-02 0.793025943 0.0731653  0.6070105
M 3.091036e-01 3.130057e-08 3.163278e-05 0.009367992 0.3798935  0.4134423
```

As gathered in Fig. 7.14b and the result presented above, we calculated the log(odds ratio) or relationship that the following predictor (independent) variables age (var = **age**, z = -1.191666, p = 2.333922e-01), gender (var = **men**, z = 1.647599, p = 9.943498e-02), and price of the different policies (i.e., vars = **price.C** (z = 0.2623833, p = 0.793025943), **price.M** (z = 1.7917982, p = 0.0731653), **price.F** (z = 0.5143455, p = 0.6070105) have with the type of vehicle the customer acquires (var = **veh).** Here var = veh is the dependent variable with three different unordered types or categories of C, M, and F, respectively. The coefficient value of -3.3970181 is the log(odds) estimate predicted by the model with t-value (z) of -7.667789 and p-value of 1.754152e-14 ($p \leq 0.05$) which presented to be statistically significant.

Consequentially, considering the results of the implemented multinominal method (model = MultNomLogR.Model), we can say that the explanatory/predictor factors such as age and gender (see significant values above) contributes or are associated to the type of vehicle the customers acquire. Whereas the prices of the insurance policies (with non-significant value) are not associated or are not a major predictor of the type of vehicle the customers acquire based on the analyzed data (data = **MyMultiNom.data**).

**(iii). Ordinal Logistic Regression Results**

```
Coefficients:
                               Value        Std. Error   t value     p value
age                          0.0002583677 0.003639489 0.07099011 9.434056e-01
seniority                    0.0187270000 0.006619418 2.82910056 4.667903e-03
men                          0.0500335068 0.097664462 0.51230003 6.084410e-01
marital_S                    0.1389112497 0.105481571 1.31692435 1.878640e-01
unlikely|somewhat likely      0.3836373022 0.184478287 2.07957971 3.756410e-02
somewhatlikely|verylikely    1.7171777571 0.188480311 9.11064796  8.189140e-20
```

As shown in Fig. 7.14c and the result presented above, we estimated (predicted) the likelihood or log(odds) that a customer will buy the insurance policies (var = **yord**, ordered dependent variable with 3 levels where 0 = unlikely, 1 = somewhat likely, and 2 = very likely) taking into account the following predictor (independent) variables—age (var = **age**), position (var = **seniority**), gender (var = **men**), and marital status (var = **marital_S**).

The coefficient values of the unlikely versus somewhat likely (z = 2.07957971, p = 3.756410e-02) and somewhat likely versus very likely (z = 9.11064796, p = 8.189140e-20) appear to be statistically significant ($p \leq 0.05$) with coefficient value estimations of 0.3836373022 and 1.7171777571, respectively.

Moreover, we can see that the log(odds ratio) of the predictor variables are all statistically significant: (var = **age**, z = 0.07099011, *p* = 9.434056e-01), (var = **seniority**, z = 2.82910056, *p* = 4.667903e-03), (var = **men**, z = 0.51230003, *p* = 6.084410e-01), and (var = **marital_S**, z = 1.31692435, *p* = 1.878640e-01). Therefore, we can statistically say based on the analyzed example data (data = **MyBinaOrdn.data)** that the age, position, gender, and marital status of the customers are a major predictor or are associated with the possibility (likelihood) that a customer will buy the insurance policies based on the defined model (model = OrdnLogR.Model).

*** the readers can refer to the further useful functions and information provided earlier at the end of Sect. 7.2 to experiment and explore more details and useful tips about the logistic regression analysis and results in R.

## 7.4 Summary

This chapter presents a hands-on example and experimentations on the use of R programming in conducting statistical analysis such as the Linear and Logistics regression test and analysis to the readers. It practically shows in detail how to conduct the different types of *linear* and *logistic regression* tests using R.

In Sect. 7.2, the authors demonstrated how to perform the two main types of linear regression analysis, namely,

(i) Simple linear regression which uses only one independent continuous variable.
(ii) Multiple linear regression which uses two or more independent variables.

In Sect. 7.3, we demonstrated how to conduct the three different types of logistic regression analysis, namely,

(i) Binomial (binary) logistic regression which uses dependent variables that only have two possible levels or types of 0 or 1.
(ii) Multinomial logistic regression which uses dependent variables that have 3 or more possible types but are not ordered.
(iii) Ordinal logistic regression which uses dependent variables that have 3 or more possible types but with ordered categories.

In addition, this chapter also covered and exemplified in detail how to plot or graphically represent the results of the different test and variables, including how to interpret and understand the results of both the linear regression and logistic regression analysis in R.

## References

Austin, P. C., & Merlo, J. (2017). Intermediate and advanced topics in multilevel logistic regression analysis. *Statistics in Medicine, 36*(20), 3257–3277. https://doi.org/10.1002/sim.7336

Connelly, L. (2020). Logistic regression. *Medsurg Nursing, 29*(5), 353–354.

Darlington, R. B., & Hayes, A. F. (2016). *Regression analysis and linear models: Concepts, applications, and implementation.* Guilford Press. https://www.routledge.com/Regression-Analysis-and-Linear-Models-Concepts-Applications-and-Implementation/Darlington-Hayes/p/book/9781462521135

Fagerland, M. W., & Hosmer, D. W. (2017). How to test for goodness of fit in ordinal logistic regression models. *Stata Journal, 17*(3), 668–686. https://doi.org/10.1177/1536867x1701700308

Kaps, M., & Lamberson, W. R. (2017). Simple linear regression. In *Biostatistics for animal science* (pp. 131–165). CABI. https://doi.org/10.1079/9781786390356.0131

Kronthaler, F., & Zöllner, S. (2021). Linear regression with RStudio. In *Data analysis with RStudio* (pp. 87–106). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-662-62518-7_7

Liang, J., Bi, G., & Zhan, C. (2020). Multinomial and ordinal Logistic regression analyses with multi-categorical variables using R. *Annals of Translational Medicine*, *8*(16), 982–982. https://doi.org/10.21037/atm-2020-57

Olive, D. J. (2017). Multiple linear regression. In *Linear regression* (pp. 17–83). Springer International Publishing. https://doi.org/10.1007/978-3-319-55252-1_2

Pal, M., Bharati, P., Pal, M., & Bharati, P. (2019). Introduction to correlation and linear regression analysis. In *Applications of regression techniques* (pp. 1–18). Springer Singapore. https://doi.org/10.1007/978-981-13-9314-3_1

Prasetyo, R. B., Kuswanto, H., Iriawan, N., & Ulama, B. S. S. (2020). Binomial regression models with a flexible generalized logit link function. *Symmetry, 12*(2), 221. https://doi.org/10.3390/sym12020221

Rstudio. (2020). *RStudio—RStudio.* https://rstudio.com/products/rstudio/

Schmidt, A. F., & Finan, C. (2018). Linear regression and the normality assumption. *Journal of Clinical Epidemiology*, *98*, 146–151. Elsevier USA. https://doi.org/10.1016/j.jclinepi.2017.12.006

# Chapter 8
# T-test Statistics in R: Independent Samples, Paired Sample, and One Sample T-tests

## 8.1 Introduction

*T-test* is one of the most widely used "parametric" procedures or statistical tests applied by researchers or data analysts to compare the *means* of two groups of independent variables or data (Kim, 2015; NCSS, 2020; Novak, 2020; Ramtin, 2023). It is classified as one of the inferential statistics or *bivariate tests* that can be used to determine whether there is a significant difference in the "mean" values between two groups of data samples.

In practice, there are different types of t-tests that can be utilized to analyze the difference(s) in a mean between the targeted samples depending on the type of the available dataset or the kind of analysis that is being required/performed. In any of the individual cases, performing the *t*-test analysis involves determining three main properties or features (data values) of the samples being analyzed. This includes:

(i) determining the *mean difference*, otherwise, referred to as the difference between the values of the underlying mean from or for each of the data sample(s).
(ii) determining the *standard deviation* of each of the group(s) being analyzed, and
(iii) determining the summary or *number of the data values* for each group.

Therefore, supposing the researcher or users wants to check whether two samples have the same mean values, where the null hypothesis is given as $H_0$: $\mu_0 = \mu_1$ (Xu et al., 2017); The formula for calculating the *t*-test is represented as follows:

$$T = \textit{sample mean difference/sample standard deviation of the sample mean difference}$$

Typically, the main assumptions or conditions for conducting the T-test statistics are summarized as follows (Okunev, 2022; Ramtin, 2023):

- The data should result in a bell-shaped distribution or curve (normally distributed) when plotted or graphically represented (see Chap. 3 in Part I). Although scientific evidence has shown that parametric procedures (see Chap. 4), such as the *t*-test discussed here, can still be performed for data sample sizes of above 30 (i.e., $n > 30$) regardless of whether the dataset that is being analyzed or measured is normally distributed or not (Adam, 2020; Roscoe, 1975).

- The existence of a simple random sampling technique, in such a way that the sampled data is randomly selected as a representative (selected) portion of the total population from which the data was collected.
- The scale of measurement for the data samples follows a continuous or ordinal scale and is independently sampled.
- Homogeneity of variance is assumed.

As a general rule of thumb, to arrive at a statistical or scientifically tested conclusion about the data samples' *mean* assumed to have a *t*-distribution (see Chaps. 3 and 6 for more details) the available data sample is expected to meet certain conditions. Such as (i) satisfying the assumption of normality, (ii) whereby the two samples under consideration must be independently selected/sampled from the same population, (iii) thus, independence of sample groups, and (iv) with equal variance (Kim, 2015).

In theory, there are three main types of *t*-tests that are commonly used in the current works of literature (Kim, 2015; Novak, 2020; Ramtin, 2023; Skaik, 2015; Xu et al., 2017). This includes.

- **Independent samples t-test** (*also referred to as Unpaired or Two-sample t-test*): which compares the means for two independently sampled groups, where the two groups under consideration are independent of each other.
- **Paired sample t-test** (*also known as Dependent sample t-test*): which compares the means of a sample collected from the same group or population but at different time or interval (e.g., pre and post test).
- **One sample t-test**: which compares the mean of a single group of variables or data alongside a known mean, i.e., test whether the given sample mean is equal to the hypothesized data value (otherwise known as the *test mean*).

In the remainder part of this chapter, the authors will be demonstrating to the readers how to conduct the three main types of T-tests (Independent, Paired, and One sample) in R. We will explain and illustrate the different statistical steps and main functions that are used to conduct the t-tests in R using the outlined steps in Fig. 8.1.

## 8.2 Independent Samples T-test in R

*Independent samples T-test* (also known as "Unpaired" or "Two-sample" t-test) is used when the dataset the researchers or data analysts wants to investigate are of two samples or groups and are statistically *independent*. In essence, the two-sample t-test as the name implies is used to compare the mean of two independent groups of variables or data samples. It is the most common type of *t*-test in the current literature as it allows the researchers to *compare the means* of two distinctive sets

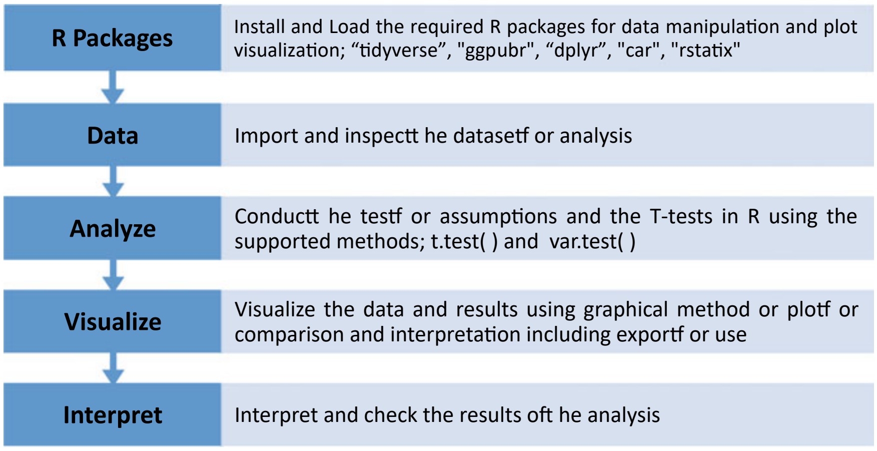

R Packages
Install and Load the required R packages for data manipulation and plot visualization; “tidyverse”, "ggpubr", “dplyr”, "car", "rstatix"
Data
Import and inspectt he datasetf or analysis
Analyze
Conductt he testf or assumptions and the T-tests in R using the supported methods; t.test( ) and var.test( )
Visualize
Visualize the data and results using graphical method or plotf or comparison and interpretation including exportf or use
Interpret
Interpret and check the results oft he analysis

**Fig. 8.1** Steps to conducting the T-test analysis in R

of data randomly drawn from two different populations, and to determine if the difference (if any or found) could have occurred randomly by chance or not.

By default, the hypothesis for testing whether there is a *difference in mean* of two specified (independent) data samples and whether this could supposedly occur by chance is; *IF* the p-value is less than or equal to 0.05 ($p \leq 0.05$), *THEN* we can assume that the *mean* of the two sets of data or groups of variables/population in the sample is statistically different and that this is not by chance ($H_1$), *ELSE IF* the p-value is greater than 0.05 ($p > 0.05$) *THEN* we presume that there is no difference in the mean of the two groups and any difference observed could only have occurred by chance ($H_0$).

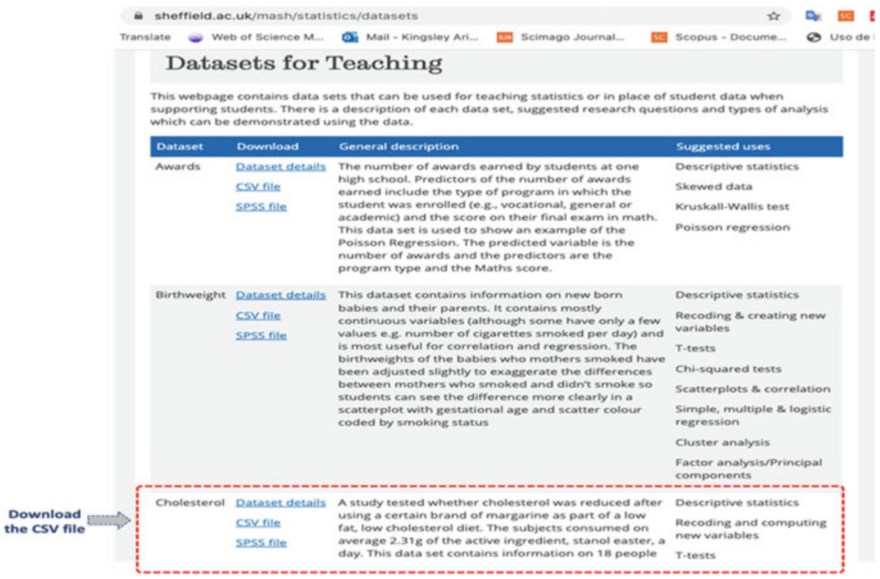
We will demonstrate how to conduct the Independent (unpaired) sample T-test in R using the **t.test( )** and **var.test( )** functions in R.

As defined in the previous section (Sect. 8.1), we will do this using the steps outlined in Fig. 8.1.

To begin, **Open RStudio** and **Create a new or open an existing project**. Once the user have RStudio and an R Project opened, **Create a new RScript** and name it “**IndSmpT-testDemo**” or any name of choice (see Chaps. 1 and 2 on how to do these steps if needed).

Now, let’s download an example data that will be used to demonstrate the Independent (unpaired) and subsequently the Paired and One sample *t*-tests in R in this chapter. ***Note: the users are welcome to use any existing data or format they may wish to use for the analysis. The example dataset the authors use here are only for illustration purposes (the users can see Chap. 2 for a step-by-step guide on how to work with different data types and formats in R).

As shown in Fig. 8.2, download the example csv data named “**Choresterol_R.csv**” from the following source: https://www.sheffield.ac.uk/mash/statistics/datasets and save it on the computer. ***Note: the readers can also refer to the following repository



Datasets for Teaching

This webpage contains data sets that can be used for teaching statistics or in place of student data when supporting students. There is a description of each data set, suggested research questions and types of analysis which can be demonstrated using the data.

| Dataset | Download | General description | Suggested uses |
|---|---|---|---|
| Awards | Dataset details<br>CSV file<br>SPSS file | The number of awards earned by students at one high school. Predictors of the number of awards earned include the type of program in which the student was enrolled (e.g., vocational, general or academic) and the score on their final exam in math. This data set is used to show an example of the Poisson Regression. The predicted variable is the number of awards and the predictors are the program type and the Maths score. | Descriptive statistics<br>Skewed data<br>Kruskall-Wallis test<br>Poisson regression |
| Birthweight | Dataset details<br>CSV file<br>SPSS file | This dataset contains information on new born babies and their parents. It contains mostly continuous variables (although some have only a few values e.g. number of cigarettes smoked per day) and is most useful for correlation and regression. The birthweights of the babies who mothers smoked have been adjusted slightly to exaggerate the differences between mothers who smoked and didn't smoke so students can see the difference more clearly in a scatterplot with gestational age and scatter colour coded by smoking status | Descriptive statistics<br>Recoding & creating new variables<br>T-tests<br>Chi-squared tests<br>Scatterplots & correlation<br>Simple, multiple & logistic regression<br>Cluster analysis<br>Factor analysis/Principal components |
| Cholesterol | Dataset details<br>CSV file<br>SPSS file | A study tested whether cholesterol was reduced after using a certain brand of margarine as part of a low fat, low cholesterol diet. The subjects consumed on average 2.31g of the active ingredient, stanol easter, a day. This data set contains information on 18 people | Descriptive statistics<br>Recoding and computing new variables<br>T-tests |

**Fig. 8.2** Example of csv data download for t-test. (*Source* https://www.sheffield.ac.uk/mash/statistics/datasets)

(https://doi.org/10.6084/m9.figshare.24728073) where the authors have uploaded all the example files and datasets used in this book if they cannot access the example files directly from the different source pages.

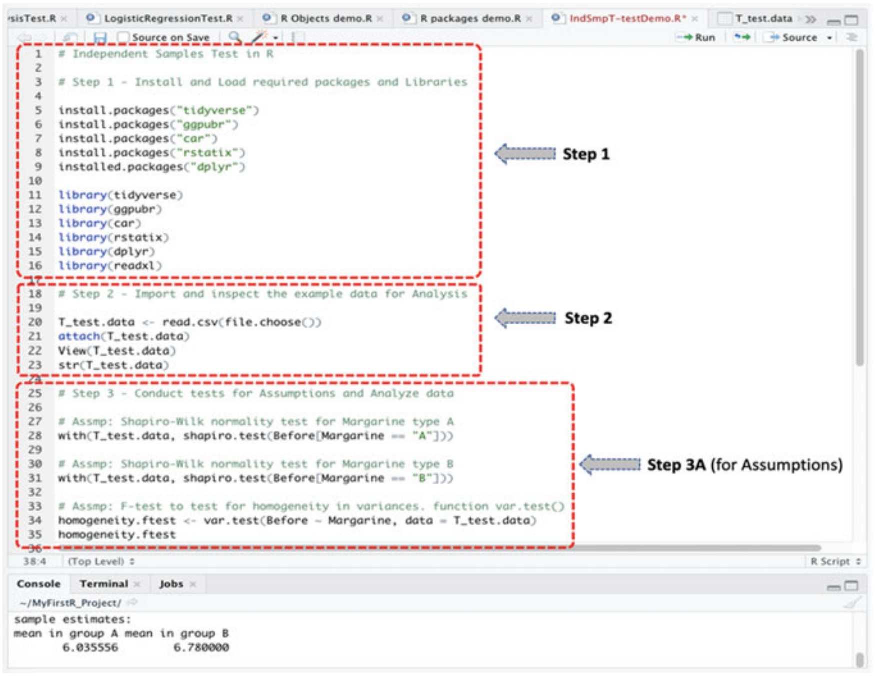
Once the user have downloaded the example file and saved this on the local machine or computer, we can proceed to conduct the *independent sample*s t-test.

## # Step 1—Install and Load the required R Packages

**Install** and **Load** the following *R packages* and *libraries* (see Fig. 8.3, Step1, Lines 3 to 16), which we will be using to call the different R functions, data manipulations, and graphical visualizations for the T-test analysis.

The code and syntax to install and load the required R packages are as follows:

```
install.packages("tidyverse")
install.packages("ggpubr")
install.packages("car")
install.packages("rstatix")
installed.packages("dplyr")

library(tidyverse)
library(ggpubr)
library(car)
library(rstatix)
library(dplyr)
library(readxl)
```



```
1   # Independent Samples Test in R
2
3   # Step 1 - Install and Load required packages and Libraries
4
5   install.packages("tidyverse")
6   install.packages("ggpubr")
7   install.packages("car")
8   install.packages("rstatix")
9   installed.packages("dplyr")
10
11  library(tidyverse)
12  library(ggpubr)
13  library(car)
14  library(rstatix)
15  library(dplyr)
16  library(readxl)
17
18  # Step 2 - Import and inspect the example data for Analysis
19
20  T_test.data <- read.csv(file.choose())
21  attach(T_test.data)
22  View(T_test.data)
23  str(T_test.data)
24
25  # Step 3 - Conduct tests for Assumptions and Analyze data
26
27  # Assmp: Shapiro-Wilk normality test for Margarine type A
28  with(T_test.data, shapiro.test(Before[Margarine == "A"]))
29
30  # Assmp: Shapiro-Wilk normality test for Margarine type B
31  with(T_test.data, shapiro.test(Before[Margarine == "B"]))
32
33  # Assmp: F-test to test for homogeneity in variances. function var.test()
34  homogeneity.ftest <- var.test(Before ~ Margarine, data = T_test.data)
35  homogeneity.ftest
36
```

```
sample estimates:
mean in group A mean in group B
       6.035556        6.780000
```

**Fig. 8.3** Steps to performing independent (unpaired) samples T-test in R

## # Step 2—Import and Inspect the example dataset for the T-test Analysis

As illustrated in Fig. 8.3 (Step 2, Lines 18 to 23), import the dataset named "**Choresterol_R.csv**" which the user have downloaded earlier and store this in an R object named "**T_test.data**" (the users can use any name of choice if they want).

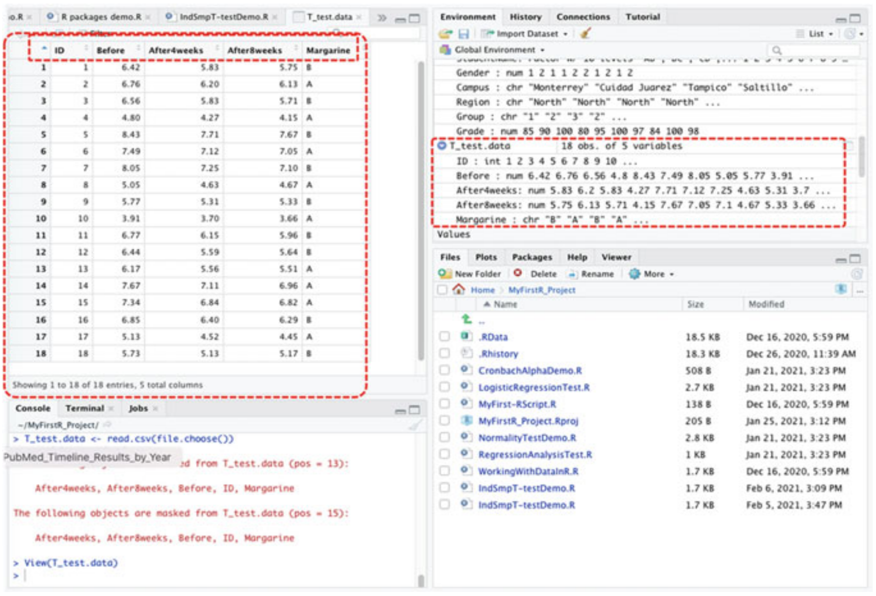
Once the user have successfully imported the dataset, you can view the details of the **Choresterol_R.csv** dataset as shown in Fig. 8.4 in R with 18 observations and 5 variables (column) in the data sample.

```
T_test.data <- read.csv(file.choose())
attach(T_test.data)
View(T_test.data)
str(T_test.data)
```

## # Step 3—Conduct the tests for Assumptions and Analyze data

Now that we have imported the dataset and stored it in an R object that we named "**T_test.data**", we can proceed to analyze the data.

As defined in Fig. 8.3 (see Step 3A, Lines 25 to 35), we will conduct the different tests of assumptions (data normality, homogeneity of variance, etc.) as mentioned earlier in Sect. 8.1, and then conduct the Independent Samples T-test if all the



|  | ID | Before | After4weeks | After8weeks | Margarine |
|---|---|---|---|---|---|
| 1 | 1 | 6.42 | 5.83 | 5.75 | B |
| 2 | 2 | 6.76 | 6.20 | 6.13 | A |
| 3 | 3 | 6.56 | 5.83 | 5.71 | B |
| 4 | 4 | 4.80 | 4.27 | 4.15 | A |
| 5 | 5 | 8.43 | 7.71 | 7.67 | B |
| 6 | 6 | 7.49 | 7.12 | 7.05 | A |
| 7 | 7 | 8.05 | 7.25 | 7.10 | B |
| 8 | 8 | 5.05 | 4.63 | 4.67 | A |
| 9 | 9 | 5.77 | 5.31 | 5.33 | B |
| 10 | 10 | 3.91 | 3.70 | 3.66 | A |
| 11 | 11 | 6.77 | 6.15 | 5.96 | B |
| 12 | 12 | 6.44 | 5.59 | 5.64 | B |
| 13 | 13 | 6.17 | 5.56 | 5.51 | A |
| 14 | 14 | 7.67 | 7.11 | 6.96 | A |
| 15 | 15 | 7.34 | 6.84 | 6.82 | A |
| 16 | 16 | 6.85 | 6.40 | 6.29 | B |
| 17 | 17 | 5.13 | 4.52 | 4.45 | A |
| 18 | 18 | 5.73 | 5.13 | 5.17 | B |

Showing 1 to 18 of 18 entries, 5 total columns

```
Gender : num 1 2 1 1 2 2 1 2 1 2
Campus : chr "Monterrey" "Cuidad Juarez" "Tampico" "Saltillo" ...
Region : chr "North" "North" "North" "North" ...
Group : chr "1" "2" "3" "2" ...
Grade : num 85 90 100 80 95 100 97 84 100 98
T_test.data          18 obs. of 5 variables
  ID : int 1 2 3 4 5 6 7 8 9 10 ...
  Before : num 6.42 6.76 6.56 4.8 8.43 7.49 8.05 5.05 5.77 3.91 ...
  After4weeks: num 5.83 6.2 5.83 4.27 7.71 7.12 7.25 4.63 5.31 3.7 ...
  After8weeks: num 5.75 6.13 5.71 4.15 7.67 7.05 7.1 4.67 5.33 3.66 ...
  Margarine : chr "B" "A" "B" "A" ...
Values
```

```
~/MyFirstR_Project/
> T_test.data <- read.csv(file.choose())
                             ed from T_test.data (pos = 13):

    After4weeks, After8weeks, Before, ID, Margarine

The following objects are masked from T_test.data (pos = 15):

    After4weeks, After8weeks, Before, ID, Margarine

> View(T_test.data)
>
```

| Name | Size | Modified |
|---|---|---|
| .RData | 18.5 KB | Dec 16, 2020, 5:59 PM |
| .Rhistory | 18.3 KB | Dec 26, 2020, 11:39 AM |
| CronbachAlphaDemo.R | 508 B | Jan 21, 2021, 3:23 PM |
| LogisticRegressionTest.R | 2.7 KB | Jan 21, 2021, 3:23 PM |
| MyFirst-RScript.R | 138 B | Dec 16, 2020, 5:59 PM |
| MyFirstR_Project.Rproj | 205 B | Jan 25, 2021, 3:12 PM |
| NormalityTestDemo.R | 2.8 KB | Jan 21, 2021, 3:23 PM |
| RegressionAnalysisTest.R | 1 KB | Jan 21, 2021, 3:23 PM |
| WorkingWithDataInR.R | 1.7 KB | Dec 16, 2020, 5:59 PM |
| IndSmpT-testDemo.R | 1.7 KB | Feb 6, 2021, 3:09 PM |
| IndSmpT-testDemo.R | 1.7 KB | Feb 5, 2021, 3:47 PM |

**Fig. 8.4** Example dataset.csv imported and stored as an R object in R

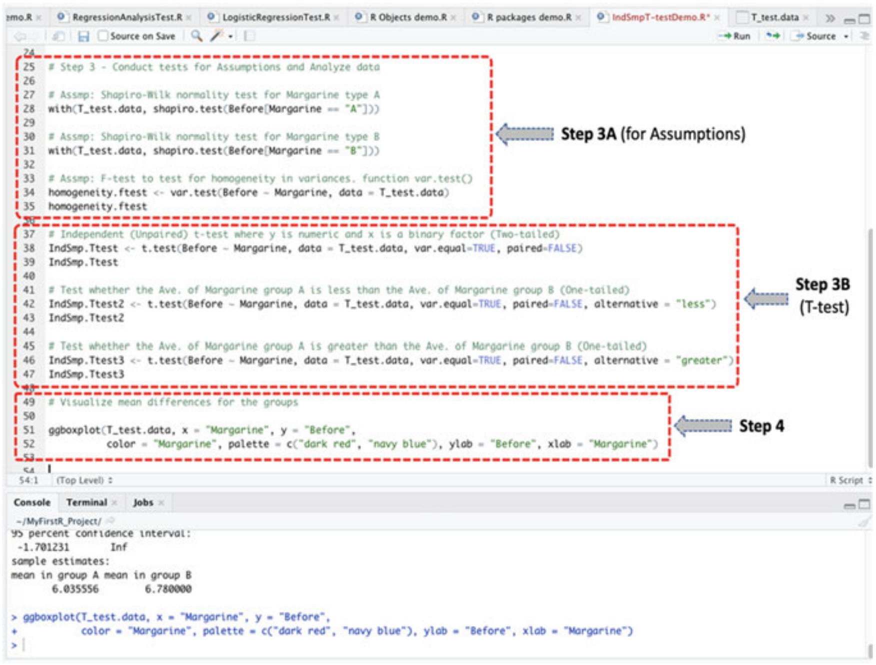
necessary conditions are met, by using the **var.test( )** and **t.test( )** functions in R, respectively (see Fig. 8.5, Step 3B, Lines 37 to 47).

As defined earlier in the Introduction section (Sect. 8.1);

- **Independent Sample t-test** compares the *means* for two independently sampled groups from two different populations whereby the two variables under consideration are independent of each other.
- The targeted grouping "independent" variable (***x***) is often a categorical or binary type, while the ***y*** variable must be numeric.

To illustrate this test using the example dataset we stored as "**T_test.data"** in R (see: highlighted columns in Fig. 8.4):

1. We will test whether the mean of the **group A** of the "**Margarine**" type or variable is *equal to* the mean of the **group B** of the "**Margarine**" considering the "**Before**" intervention variable contained in the dataset? (**two-tailed test**)
2. Also, we will check whether the *mean* of the **group A** (Margarine) is *less than* the mean of **group B** (Margarine)? (**one-tailed test**)
3. Then we will check whether the *mean* of **group A** (Margarine) is *greater than* the mean of **group B** (Margarine)? (**one-tailed test**).



```
24
25 # Step 3 - Conduct tests for Assumptions and Analyze data
26
27 # Assmp: Shapiro-Wilk normality test for Margarine type A
28 with(T_test.data, shapiro.test(Before[Margarine == "A"]))
29
30 # Assmp: Shapiro-Wilk normality test for Margarine type B
31 with(T_test.data, shapiro.test(Before[Margarine == "B"]))
32
33 # Assmp: F-test to test for homogeneity in variances. function var.test()
34 homogeneity.ftest <- var.test(Before ~ Margarine, data = T_test.data)
35 homogeneity.ftest
36
37 # Independent (Unpaired) t-test where y is numeric and x is a binary factor (Two-tailed)
38 IndSmp.Ttest <- t.test(Before ~ Margarine, data = T_test.data, var.equal=TRUE, paired=FALSE)
39 IndSmp.Ttest
40
41 # Test whether the Ave. of Margarine group A is less than the Ave. of Margarine group B (One-tailed)
42 IndSmp.Ttest2 <- t.test(Before ~ Margarine, data = T_test.data, var.equal=TRUE, paired=FALSE, alternative = "less")
43 IndSmp.Ttest2
44
45 # Test whether the Ave. of Margarine group A is greater than the Ave. of Margarine group B (One-tailed)
46 IndSmp.Ttest3 <- t.test(Before ~ Margarine, data = T_test.data, var.equal=TRUE, paired=FALSE, alternative = "greater")
47 IndSmp.Ttest3
48
49 # Visualize mean differences for the groups
50
51 ggboxplot(T_test.data, x = "Margarine", y = "Before",
52           color = "Margarine", palette = c("dark red", "navy blue"), ylab = "Before", xlab = "Margarine")
53
54
```

```
95 percent confidence interval:
 -1.701231       Inf
sample estimates:
mean in group A mean in group B
       6.035556        6.780000

> ggboxplot(T_test.data, x = "Margarine", y = "Before",
+           color = "Margarine", palette = c("dark red", "navy blue"), ylab = "Before", xlab = "Margarine")
> 
```

**Fig. 8.5** Independent sample (unpaired) T-test in R

The syntax to conduct the defined tests and experiment in R (Fig. 8.5, Step 3A and 3B, Lines 25 to 47), are as shown in the codes below:

**Test of Assumption**:

```
# Assmp: Shapiro-Wilk normality test for Margarine type A
    with(T_test.data, shapiro.test(Before[Margarine == "A"]))

# Assmp: Shapiro-Wilk normality test for Margarine type B
    with(T_test.data, shapiro.test(Before[Margarine == "B"]))

# Assmp: F-test to test for homogeneity in variances. function var.test()
    homogeneity.ftest <- var.test(Before ~ Margarine, data = T_test.data)
  homogeneity.ftest
```

**T-Test Analysis**:

```
# Independent (Unpaired) t-test where y is numeric and x is a binary
  factor (Two-tailed)
IndSmp.Ttest <- t.test(Before ~ Margarine, data = T_test.data,
  var.equal=TRUE, paired=FALSE)
  IndSmp.Ttest

# Test whether the Ave. of Margarine group A is less than the Ave. of
  Margarine group B (One-tailed)
IndSmp.Ttest2 <- t.test(Before ~ Margarine, data = T_test.data,
  var.equal=TRUE, paired=FALSE, alternative = "less")
  IndSmp.Ttest2

# Test whether the Ave. of Margarine group A is greater than the Ave. of
  Margarine group B (One-tailed)
IndSmp.Ttest3 <- t.test(Before ~ Margarine, data = T_test.data,
  var.equal=TRUE, paired=FALSE, alternative = "greater")
  IndSmp.Ttest3
```

**Useful Tips**:

- The users must use the `paired = FALSE` option to specify the Independent Samples (unpaired) t-test.
- The following function: `paired = TRUE` is used to specify the Paired (dependent) sample t-test (which the authors will cover in the next Sect. 8.3).
- Users need to use the `var.equal = TRUE` option to specify equal variances or a pooled variance estimate for the *t*-test analysis.
- Users must use the `alternative = "less"` and `alternative = "greater"` options to specify a "one-tailed" *t*-test.

To continue in our illustrations, once the user have successfully run the codes as defined in **Steps 3A** and **3B** (see Fig. 8.5, Lines 25 to 47), they will be presented with the results of the "tests for assumptions" and the "Independent sample t-test" in the Console as shown in Fig. 8.6a and b.

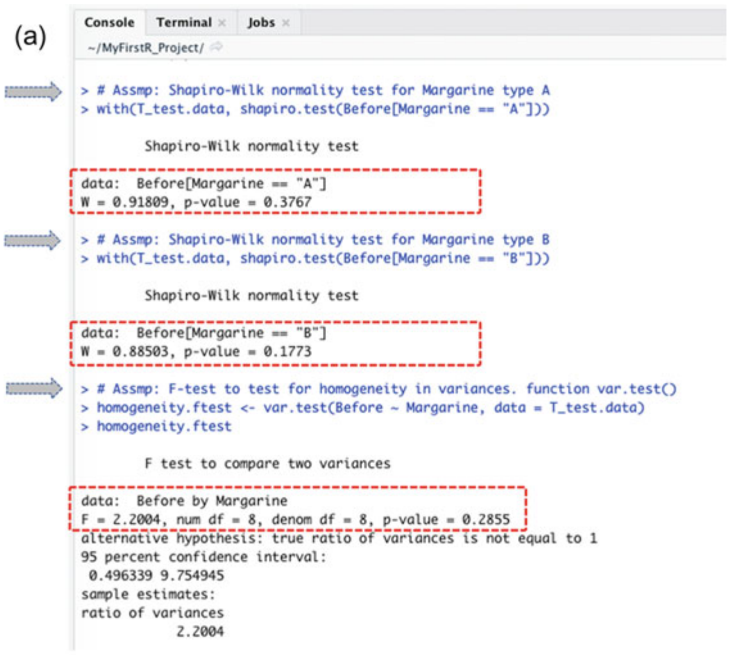
In Fig. 8.6a, which represents the outcome of the Step 3A (Fig. 8.5, test of assumptions); we conducted the different assumptions for the *t*-test in order to determine if the analyzed dataset is fitting and/or valid for the test (Independent Samples t-test).

As highlighted in the figure (Fig. 8.6a), the **normality test** using Shapiro–Wilk's method (where we assume a test statistics, W, greater than 0.5 and p-value greater than 0.05, i.e., $p > 0.05$, is normal) shows that the distribution of the two groups of the independent variable (**A: whereby W = 0.91809, p-value=0.3767;** and **B: whereby W = 0.88503, p-value=0.1773** of "**Margarine**") are normality distributed when measured against the "**Before**" variable which is the second targeted variable in our analysis.

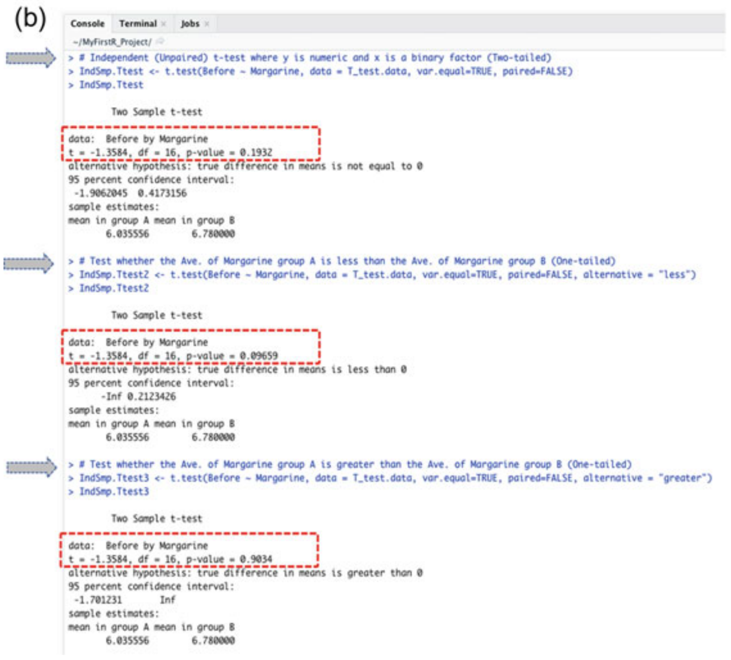
Thus:

(`Before[Margarine == "A"], p-value=0.3767`) and

(`Before[Margarine == "B"], p-value=0.1773`)



```
> # Assmp: Shapiro-Wilk normality test for Margarine type A
> with(T_test.data, shapiro.test(Before[Margarine == "A"]))

        Shapiro-Wilk normality test

data:  Before[Margarine == "A"]
W = 0.91809, p-value = 0.3767

> # Assmp: Shapiro-Wilk normality test for Margarine type B
> with(T_test.data, shapiro.test(Before[Margarine == "B"]))

        Shapiro-Wilk normality test

data:  Before[Margarine == "B"]
W = 0.88503, p-value = 0.1773

> # Assmp: F-test to test for homogeneity in variances. function var.test()
> homogeneity.ftest <- var.test(Before ~ Margarine, data = T_test.data)
> homogeneity.ftest

        F test to compare two variances

data:  Before by Margarine
F = 2.2004, num df = 8, denom df = 8, p-value = 0.2855
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
 0.496339 9.754945
sample estimates:
ratio of variances
            2.2004
```



```
> # Independent (Unpaired) t-test where y is numeric and x is a binary factor (Two-tailed)
> IndSmp.Ttest <- t.test(Before ~ Margarine, data = T_test.data, var.equal=TRUE, paired=FALSE)
> IndSmp.Ttest

        Two Sample t-test

data:  Before by Margarine
t = -1.3584, df = 16, p-value = 0.1932
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -1.9062045  0.4173156
sample estimates:
mean in group A mean in group B
       6.035556        6.780000

> # Test whether the Ave. of Margarine group A is less than the Ave. of Margarine group B (One-tailed)
> IndSmp.Ttest2 <- t.test(Before ~ Margarine, data = T_test.data, var.equal=TRUE, paired=FALSE, alternative = "less")
> IndSmp.Ttest2

        Two Sample t-test

data:  Before by Margarine
t = -1.3584, df = 16, p-value = 0.09659
alternative hypothesis: true difference in means is less than 0
95 percent confidence interval:
      -Inf 0.2123426
sample estimates:
mean in group A mean in group B
       6.035556        6.780000

> # Test whether the Ave. of Margarine group A is greater than the Ave. of Margarine group B (One-tailed)
> IndSmp.Ttest3 <- t.test(Before ~ Margarine, data = T_test.data, var.equal=TRUE, paired=FALSE, alternative = "greater")
> IndSmp.Ttest3

        Two Sample t-test

data:  Before by Margarine
t = -1.3584, df = 16, p-value = 0.9034
alternative hypothesis: true difference in means is greater than 0
95 percent confidence interval:
 -1.701231       Inf
sample estimates:
mean in group A mean in group B
       6.035556        6.780000
```

**Fig. 8.6** **a** Results of normality and homogeneity of variance test displayed in the console in R. **b** Results of the independent samples (Two-sample) T-test in R

Also, in the assumption test, we tested the **homogeneity of variance** for the two targeted variables (**Before ~ Margarine**) using the **var.test( )** method; whereby we assume that a value of $p > 0.05$ indicates "equality in variance". Consequently, as highlighted in Fig. 8.6a, we note that there is no difference in the variance for the two sets of analyzed variables with `p = 0.2855`. and `F = 2.2004`, respectively. Thus, we accept that the assumption of equality in variance is met.

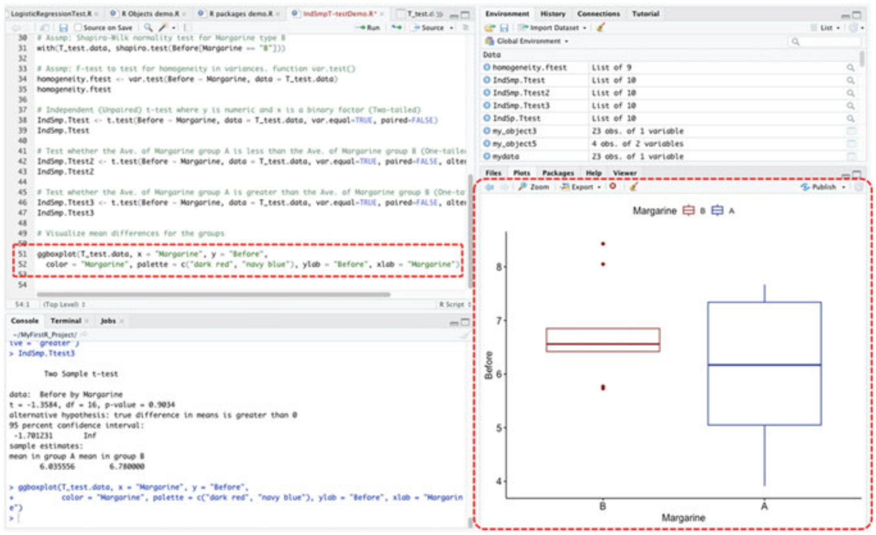
Therefore, with all necessary conditions met, we proceeded to conduct the "Independent Samples T-test" as defined in Step 3B (Fig. 8.5) and the results are as reported in Fig. 8.6b.

As shown in Fig. 8.6b, we performed the Independent Samples (Two-sample) t-test by considering the two variables (**Before ~ Margarine**). The results of the test are stored in R objects we called "`IndSmp.Ttest`" for the **two-tailed** analysis, and then "`IndSmp.Ttest2`" and "`IndSmp.Ttest3`" for the **one-tailed** analysis, respectively.

## # Step 4—Plot and visualize the mean differences for the independent groups

Another way to check whether there is a difference in the mean of the two independent variables is by plotting them as graphs. By so doing, the users will be able to visualize the difference in the mean (if any) between the two variables.

To do this, as defined in Step 4 in Fig. 8.5 (Lines 49 to 52) and the resultant boxplot represented in Fig. 8.7; the authors applied the **ggplot( )** function in R to visualize the *mean* between the two groups of the independent variable "**Margarine**" (**group A** and **group B**) by taking into account the second target variable **"Before"** in the example data "**T_test.data**".



```
30  # Assmp: Shapiro-Wilk normality test for Margarine type B
31  with(T_test.data, shapiro.test(Before[Margarine == "B"]))
32
33  # Assmp: F-test to test for homogeneity in variances. function var.test()
34  homogeneity.ftest <- var.test(Before ~ Margarine, data = T_test.data)
35  homogeneity.ftest
36
37  # Independent (Unpaired) t-test where y is numeric and x is a binary factor (Two-tailed)
38  IndSmp.Ttest <- t.test(Before ~ Margarine, data = T_test.data, var.equal=TRUE, paired=FALSE)
39  IndSmp.Ttest
40
41  # Test whether the Ave. of Margarine group A is less than the Ave. of Margarine group B (One-taile
42  IndSmp.Ttest2 <- t.test(Before ~ Margarine, data = T_test.data, var.equal=TRUE, paired=FALSE, alte
43  IndSmp.Ttest2
44
45  # Test whether the Ave. of Margarine group A is greater than the Ave. of Margarine group B (One-ta
46  IndSmp.Ttest3 <- t.test(Before ~ Margarine, data = T_test.data, var.equal=TRUE, paired=FALSE, alte
47  IndSmp.Ttest3
48
49  # Visualize mean differences for the groups
50
51  ggboxplot(T_test.data, x = "Margarine", y = "Before",
52    color = "Margarine", palette = c("dark red", "navy blue"), ylab = "Before", xlab = "Margarine")
53
54
```

```
ive = "greater")
> IndSmp.Ttest3

        Two Sample t-test

data:  Before by Margarine
t = -1.3584, df = 16, p-value = 0.9034
alternative hypothesis: true difference in means is greater than 0
95 percent confidence interval:
 -1.701231       Inf
sample estimates:
mean in group A mean in group B
       6.035556        6.780000

> ggboxplot(T_test.data, x = "Margarine", y = "Before",
+           color = "Margarine", palette = c("dark red", "navy blue"), ylab = "Before", xlab = "Margarin
e")
>
```

```
Data
homogeneity.ftest   List of 9
IndSmp.Ttest        List of 10
IndSmp.Ttest2       List of 10
IndSmp.Ttest3       List of 10
IndSp.Ttest         List of 10
my_object3          23 obs. of 1 variable
my_object5          4 obs. of 2 variables
mydata              23 obs. of 1 variable
```

Fig. 8.7 Plotting and visualizing the mean differences for two groups of independent variables in R using the ggplot( ) function

The syntax used in plotting the mean is shown below, and the results are represented in Fig. 8.7.

```
# Visualize mean difference for the groups of independent variables
 ggboxplot(T_test.data, x = "Margarine", y = "Before",
         color = "Margarine", palette = c("dark red", "navy blue"),
         ylab = "Before", xlab = "Margarine")
```

## # Step 5—T-tests Results Interpretation

The last step for the Independent Samples *t*-test analysis or test is to understand and interpret the results of the method.

By default, the hypothesis for conducting the t-test (Independent t-test) is; *IF* the p-value is less than or equal to 0.05 ($p \leq 0.05$), THEN we assume that the *mean* of the two sets of data or groups of variables are statistically different and that this is not by chance ($H_1$), *ELSE IF* the p-value is greater than 0.05 ($p > 0.05$) THEN we can presume that there is no difference in the mean of the two groups of variable/ population or that any difference observed may supposedly be by chance ($H_0$).

```
> IndSmp.Ttest <- t.test(Before ~ Margarine, data = T_test.data,
var.equal=TRUE, paired=FALSE)
> IndSmp.Ttest
      Two Sample t-test
data:  Before by Margarine
t = -1.3584, df = 16, p-value = 0.1932
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -1.9062045  0.4173156
sample estimates:
mean in group A mean in group B
       6.035556        6.780000
```

As shown in the result reported above and gathered from the outcome of the Independent Sample (Two-sample) *t*-test for the example dataset (**T_test.data**) in Fig. 8.6b; the meaning of the results of the test by using the **t.test( )** function which we applied for the (independent) **two-tailed** tests (**Before ~ Margarine**) and saved in an R object we called "`IndSmp.Ttest`" can be explained as a list containing the following:

- **Statistics**: `t = -1.3584` which denotes the value of the *t*-test analysis.
- **Parameter**: `df = 16` signifies the degrees of freedom for the t-test statistics.
- **p-value**: `p-value = 0.1932` is the p-value or significance levels of the test.
- **Confidence interval**: `Conf.Int(95%, -1.9062045 0.4173156)` represents the confidence interval for the mean assumed to be appropriate to the specified alternative hypothesis.
- **Sample estimates**: `group` **`A`** `mean = 6.035556, group` **`B`** `mean = 6.780000` which is the means of the two groups of population being compared considering the two variables (Before ~ Margarine).

Statistically, the p-value of the Independent sample t-test (`IndSmp.Ttest`) is `p=0.1932` (where p is expected at $P \leq 0.05$). As we can see, the value is greater than the stated or acceptable significance levels ($p \leq 0.05$). Therefore, we can conclude that there is no significant difference between the means of the two groups (A and B) of **Margarine** taking into account the **Before** variable or intervention in the dataset. In other words, the *mean* of the **group A** of the "**Margarine**" variable is *equal to* the mean of **group B** of the "**Margarine**" when analyzed against the "**Before**" intervention (**two-tailed test**).

Furthermore, as shown in the next results presented below and reported in Fig. 8.6b for the "one-tailed" t-tests:

– We checked whether the mean of the **group A** is *less than* the mean of the **group B** of the **Margarine** (`IndSmp.Ttest2`), and
– Whether the mean of **group A** is *greater than* the mean of **group B** of the **Margarine** variable (`IndSmp.Ttest3`), respectively.

```
> IndSmp.Ttest2 <- t.test(Before ~ Margarine, data = T_test.data,
var.equal=TRUE, paired=FALSE, alternative = "less")
> IndSmp.Ttest2
      Two Sample t-test
data:  Before by Margarine
t = -1.3584, df = 16, p-value = 0.09659
alternative hypothesis: true difference in means is less than 0
95 percent confidence interval:
      -Inf 0.2123426
sample estimates:
mean in group A mean in group B
       6.035556        6.780000
```

```
> IndSmp.Ttest3 <- t.test(Before ~ Margarine, data = T_test.data,
var.equal=TRUE, paired=FALSE, alternative = "greater")
> IndSmp.Ttest3
      Two Sample t-test
data:  Before by Margarine
t = -1.3584, df = 16, p-value = 0.9034
alternative hypothesis: true difference in means is greater than 0
95 percent confidence interval:
 -1.701231       Inf
sample estimates:
mean in group A mean in group B
       6.035556        6.780000
```

As gathered in the above results of the one-tailed tests (`IndSmp.Ttest2, where p=0.09659` and `IndSmp.Ttest3, where p=0.9034`); while we can see that there are no significant differences in the means of the two groups (i.e., the value of the p-value is above the threshold of $p \leq 0.05$, and the mean for group A=`6.035556` and group B=`6.780000`).

On the other hand, the one-tailed test (`IndSmp.Ttest2, p=0.09659`) reveals that the mean of the group **A (6.035556)** is slightly less than mean of group **B (6.780000)**, and likewise the mean of group **A** is not greater than the mean of group B for the other one-tailed test (`IndSmp.Ttest3, p=0.9034`), vice and versa.

Thus, in summary, we can statistically conclude that the *means* of the **group A** is slightly less than and not greater than the *means* of the **group B** with mean sample estimate of A = `6.035556` and B = `6.780000`, respectively.

## 8.3 Paired (Dependent) Sample T-test in R

*Paired* or *Dependent Samples* T-test is another type of statistical test used by the researchers to determine the mean difference between two sets of data or variables. But, in this case or scenario, the test is used to compare the means of a sample collected from the same group but at different time or interval. In other words, it is used when the researchers or data analyst are interested in knowing the *difference in mean* between two measures (e.g., pre and post tests) in a sample.

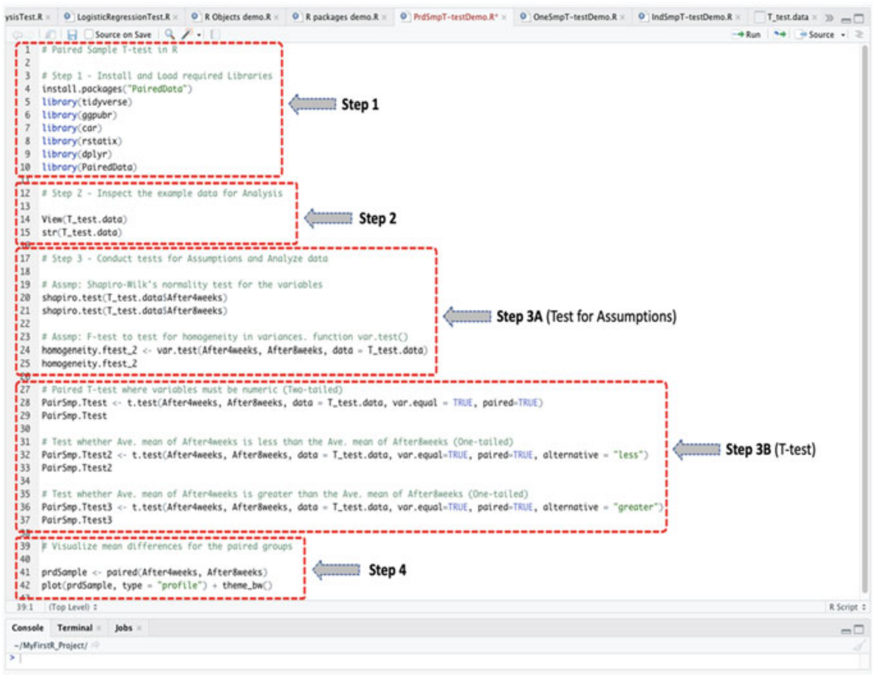
To demonstrate the paired sample t-test, we will continue to use the same example dataset "**Choresterol_R.csv**" that we have downloaded and imported earlier in Sect. 8.2 that we stored as an R object named "**T_test.data**" (see: Fig. 8.4). ***The list of example datasets used in this book can also be directly accessed via the following link: https://doi.org/10.6084/m9.figshare.24728073 if the user needs to access the file again.

To begin with conducting the Paired Sample t-test, create a **new R Script** and name it "**PrdSmpT-testDemo**" (the user may use any name of their choice).

**# Step 1—Load the required R Packages and Libraries**

Since we have previously installed the necessary R packages for our previous t-test analysis (see Sect. 8.2), we do not need to install the packages again, rather we just need to call or load the libraries for the necessary R packages as shown in the code below and in Fig. 8.8 (see Step1, Lines 3 to 10). ***Note: the only new R package that the authors have installed this time is the `"PairedData"` which have not been previously installed in the previous program. If the user have directly visited this particular section of the chapter or have previously exited and resumed R, then they may need to Install or re-install all the necessary R packages and libraries listed below (see: Chap. 2, Sect. 2.6, for further guidance if you require to do so) depending on the user's case.

We will be using the **paired( )** function when plotting the graphical display of the two paired variables or data.



```
1   # Paired Sample T-test in R
2
3   # Step 1 - Install and Load required Libraries
4   install.packages("PairedData")
5   library(tidyverse)
6   library(ggpubr)
7   library(car)
8   library(rstatix)
9   library(dplyr)
10  library(PairedData)
11
12  # Step 2 - Inspect the example data for Analysis
13
14  View(T_test.data)
15  str(T_test.data)
16
17  # Step 3 - Conduct tests for Assumptions and Analyze data
18
19  # Assmp: Shapiro-Wilk's normality test for the variables
20  shapiro.test(T_test.data$After4weeks)
21  shapiro.test(T_test.data$After8weeks)
22
23  # Assmp: F-test to test for homogeneity in variances. function var.test()
24  homogeneity.ftest_2 <- var.test(After4weeks, After8weeks, data = T_test.data)
25  homogeneity.ftest_2
26
27  # Paired T-test where variables must be numeric (Two-tailed)
28  PairSmp.Ttest <- t.test(After4weeks, After8weeks, data = T_test.data, var.equal = TRUE, paired=TRUE)
29  PairSmp.Ttest
30
31  # Test whether Ave. mean of After4weeks is less than the Ave. mean of After8weeks (One-tailed)
32  PairSmp.Ttest2 <- t.test(After4weeks, After8weeks, data = T_test.data, var.equal=TRUE, paired=TRUE, alternative = "less")
33  PairSmp.Ttest2
34
35  # Test whether Ave. mean of After4weeks is greater than the Ave. mean of After8weeks (One-tailed)
36  PairSmp.Ttest3 <- t.test(After4weeks, After8weeks, data = T_test.data, var.equal=TRUE, paired=TRUE, alternative = "greater")
37  PairSmp.Ttest3
38
39  # Visualize mean differences for the paired groups
40
41  prdSample <- paired(After4weeks, After8weeks)
42  plot(prdSample, type = "profile") + theme_bw()
```

**Fig. 8.8** Conducting paired samples (dependent) T-test in R

```
install.packages("PairedData")
library(tidyverse)
library(ggpubr)
library(car)
library(rstatix)
library(dplyr)
library(PairedData)
```

## # Step 2—Inspect example dataset for Paired T-test Analysis

As illustrated in Fig. 8.8 (Step 2), again since we will be using the already imported dataset (**T_test.data**) for our new analysis (see: Fig. 8.4), we do not need to import the data again rather we can view and/or check the data to make sure that the necessary variables we need or require for our analysis are in the data. This can be done by using the **View( )** or **str( )** commands as shown in the code below (Fig. 8.8, Lines 12 to 15).

```
View(T_test.data)
str(T_test.data)
```

*****Note**: the users can use the following code presented below to import and attach the example dataset if this is their first time or if they have exited R or directly visited this particular section of the chapter. The users that have continued from our previous analysis in Sect. 8.2 and can ignore using this command again and move straight to step 3.

```
T_test.data <- read.csv(file.choose())
attach(T_test.data)
View(T_test.data)
str(T_test.data)
```

**# Step 3—Conduct tests for Assumptions and Analyze the data**

Now, as shown in Fig. 8.8 (Step 3A, Lines 17 to 25), we conducted the different necessary tests of assumptions (i.e., data normality and homogeneity of variance) in R for the selected items or variables (i.e., "**After4weeks**" and "**After8weeks**"—see Fig. 8.4) in our analysis for the paired sample t-test using the **var.test( )** function, as defined earlier in Sect. 8.1. Then, we performed the Paired Sample T-test (Fig. 8.8, Step 3B, Lines 27 to 37) using the **t.test( )** function.

As defined earlier in the introduction section (Sect. 8.1);

- **Paired Sample T-test** statistics compares the means for two sets of data from a single population but analyzed at different time intervals (e.g., pre and post test).
- The targeted variables must be numeric.

To illustrate this test or experiment (paired sample t-test) using the example dataset "**T_test.data"** (see: highlighted columns in Fig. 8.4):

1. We will test whether the mean of the "**After4weeks**" variable is *equal to* the mean of the "**After8weeks**" variable after the intervention? **(two-tailed test)**
2. Also, we will check whether the mean of the "**After4weeks**" is *less than* the mean of the "**After8weeks**"? **(one-tailed test)**
3. Then we will check whether the mean of the "**After4weeks**" is *greater than* the mean of the "**After8weeks**"? **(one-tailed test)**.

The syntax to conduct the stated tests listed above in R (see Fig. 8.8, Step 3A and 3B, Lines 17 to 37), are as shown in the codes below (see: Fig. 8.8):

```
# Assmp: Shapiro-Wilk's normality test for the variables
shapiro.test(T_test.data$After4weeks)
shapiro.test(T_test.data$After8weeks)

# Assmp: F-test to test for homogeneity in variances. function var.test()
homogeneity.ftest_2  <-  var.test(After4weeks,  After8weeks,  data  =
T_test.data)
homogeneity.ftest_2

# Paired T-test where the given variables must be numeric (Two-tailed)
PairSmp.Ttest <- t.test(After4weeks, After8weeks, data = T_test.data,
var.equal = TRUE, paired=TRUE)
PairSmp.Ttest

# Test whether Ave. mean of After4weeks is less than the Ave. mean of
After8weeks (One-tailed)
PairSmp.Ttest2 <- t.test(After4weeks, After8weeks, data = T_test.data,
var.equal=TRUE, paired=TRUE, alternative = "less")
PairSmp.Ttest2

# Test whether Ave. mean of After4weeks is greater than the Ave. mean of
After8weeks (One-tailed)
PairSmp.Ttest3 <- t.test(After4weeks, After8weeks, data = T_test.data,
var.equal=TRUE, paired=TRUE, alternative = "greater")
PairSmp.Ttest3
```

**Useful Tip**:

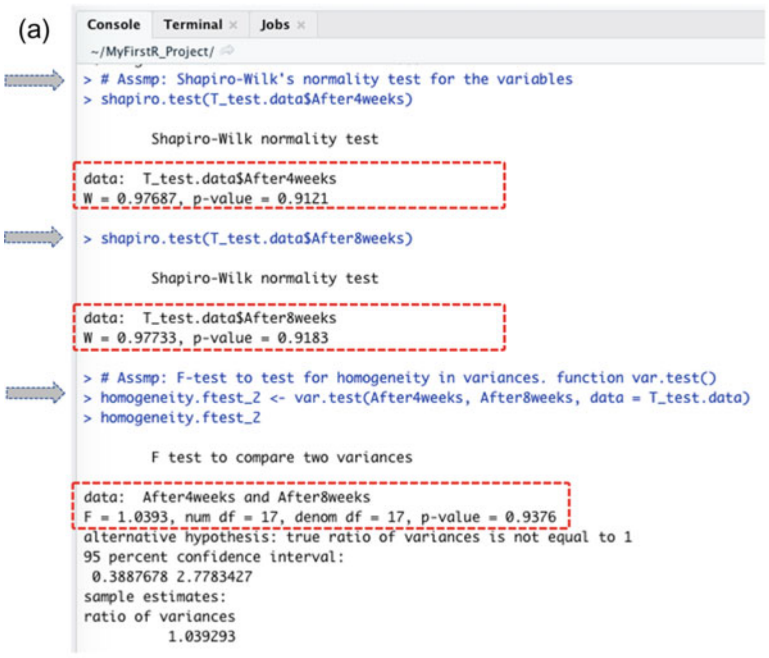
- As shown in the codes above, the users always need to specify the `paired = TRUE` option when conducting the Paired sample *t*-test.

Once the user have successfully run the codes as defined in the **Step 3A** and **3B** (Lines 17 to 37) in Fig. 8.8, they will be presented with the results of the "tests for assumptions" and the "Paired Sample T-test" in the Console as shown in Figs. 8.9a and b, respectively.

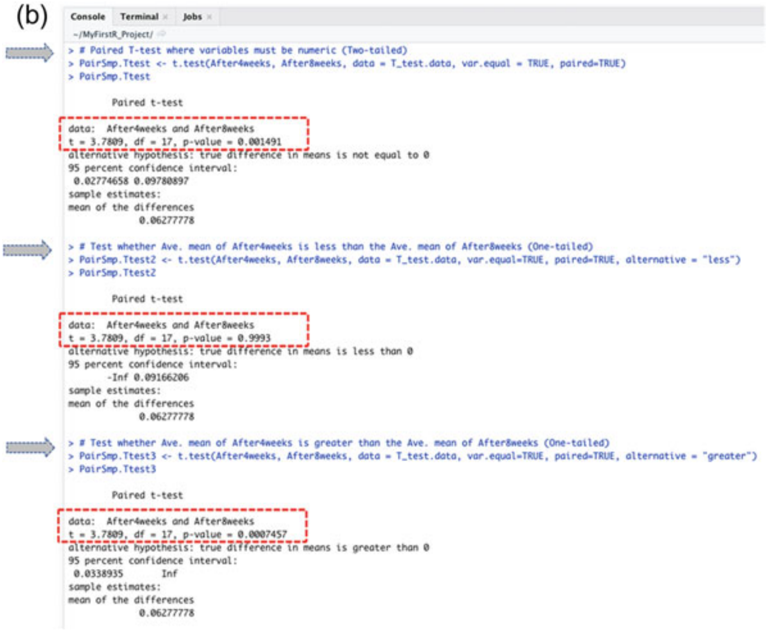
In Figs. 8.9a, which represents the outcome of the Step 3A (see: Fig. 8.8) for the paired *t*-test analysis, we conducted the different necessary assumptions tests for the t-test in order to determine if the available dataset and selected variables are valid to perform the test.

As highlighted in the figure (Fig. 8.9a), for the test of assumptions, the **normality test** using Shapiro–Wilk's method in which we hypothetically assume that a score or statistics value, W, of above 0.5 (p-value expected at p > 0.05), and value of the **homogeneity of variance** of *p-value > 0.05,* is normal and acceptable, shows that the distribution of the two sets of data (i.e., `After4weeks: W=0.97687 where p-value=0.9121`, and `After8weeks:` W=0.97733 `where p-value=0.9183`) are normality distributed and suitable for the paired t-test analysis.

The **homogeneity of variance** for the two targeted variables using the **var.test( )** method, whereby we assume that a value of *p > 0.05* indicates equality in variance. As highlighted in Fig. 8.9a, we note that there is no difference in the variance for the two variables (`After4weeks, After8weeks`) with `p-value=0.9376` and `F=1.0393`. Hence, we accepted the assumption that equality in variance is met.

(a)


```
> # Assmp: Shapiro-Wilk's normality test for the variables
> shapiro.test(T_test.data$After4weeks)

        Shapiro-Wilk normality test

data:  T_test.data$After4weeks
W = 0.97687, p-value = 0.9121

> shapiro.test(T_test.data$After8weeks)

        Shapiro-Wilk normality test

data:  T_test.data$After8weeks
W = 0.97733, p-value = 0.9183

> # Assmp: F-test to test for homogeneity in variances. function var.test()
> homogeneity.ftest_2 <- var.test(After4weeks, After8weeks, data = T_test.data)
> homogeneity.ftest_2

        F test to compare two variances

data:  After4weeks and After8weeks
F = 1.0393, num df = 17, denom df = 17, p-value = 0.9376
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
 0.3887678 2.7783427
sample estimates:
ratio of variances
          1.039293
```

(b)


```
> # Paired T-test where variables must be numeric (Two-tailed)
> PairSmp.Ttest <- t.test(After4weeks, After8weeks, data = T_test.data, var.equal = TRUE, paired=TRUE)
> PairSmp.Ttest

        Paired t-test

data:  After4weeks and After8weeks
t = 3.7809, df = 17, p-value = 0.001491
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 0.02774658 0.09780897
sample estimates:
mean of the differences
             0.06277778

> # Test whether Ave. mean of After4weeks is less than the Ave. mean of After8weeks (One-tailed)
> PairSmp.Ttest2 <- t.test(After4weeks, After8weeks, data = T_test.data, var.equal=TRUE, paired=TRUE, alternative = "less")
> PairSmp.Ttest2

        Paired t-test

data:  After4weeks and After8weeks
t = 3.7809, df = 17, p-value = 0.9993
alternative hypothesis: true difference in means is less than 0
95 percent confidence interval:
       -Inf 0.09166206
sample estimates:
mean of the differences
             0.06277778

> # Test whether Ave. mean of After4weeks is greater than the Ave. mean of After8weeks (One-tailed)
> PairSmp.Ttest3 <- t.test(After4weeks, After8weeks, data = T_test.data, var.equal=TRUE, paired=TRUE, alternative = "greater")
> PairSmp.Ttest3

        Paired t-test

data:  After4weeks and After8weeks
t = 3.7809, df = 17, p-value = 0.0007457
alternative hypothesis: true difference in means is greater than 0
95 percent confidence interval:
 0.0338935       Inf
sample estimates:
mean of the differences
             0.06277778
```

**Fig. 8.9** **a** Results of normality and homogeneity of variance test displayed in the console in R. **b** Results of paired sample (dependent) T-test displayed in console in R

Accordingly, Fig. 8.9b is the result of the Paired Sample T-test analysis as defined in Step 3B in Fig. 8.8.

As reported in Fig. 8.9b, we performed the Paired Sample *t*-test by considering the mean differences for the targeted variables (`After4weeks, After8weeks`). The results of the test were stored in R objects we called "`PairSmp.Ttest`" for the **two-tailed** analysis, and "`PairSmp.Ttest2`" and "`PairSmp.Ttest3`" for the **one-tailed** analysis, respectively.

**# Step 4—Plot and visualize the mean differences for the two paired samples or variables**

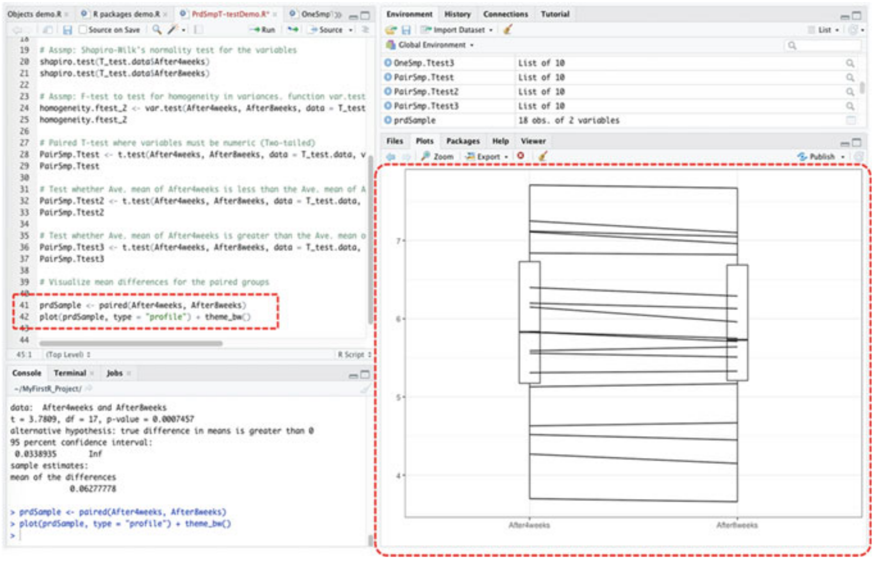
As defined in Fig. 8.8 (Step 4, Lines 39 to 42) and the resultant plot represented in Fig. 8.10, we graphically plotted and visualized the mean between the two paired groups of variables (`After4weeks, After8weeks`) in the example dataset "**T_test.data**".

The code used to create or plot the mean of the two variables (`After4weeks, After8weeks`) is shown below, and the resultant graph is represented in Fig. 8.10.

```
# Visualize mean differences for the paired groups of variables
prdSample <- paired(After4weeks, After8weeks)
plot(prdSample, type = "profile") + theme_bw()
```

**# Step 5—Results Interpretation for the paired sample t-test**

The final step for the "paired sample *t*-test" analysis is to understand and interpret the results of the test.



```
19 # Assmp: Shapiro-Wilk's normality test for the variables
20 shapiro.test(T_test.data$After4weeks)
21 shapiro.test(T_test.data$After8weeks)
22
23 # Assmp: F-test to test for homogeneity in variances. function var.test
24 homogeneity.ftest_2 <- var.test(After4weeks, After8weeks, data = T_test
25 homogeneity.ftest_2
26
27 # Paired T-test where variables must be numeric (Two-tailed)
28 PairSmp.Ttest <- t.test(After4weeks, After8weeks, data = T_test.data, v
29 PairSmp.Ttest
30
31 # Test whether Ave. mean of After4weeks is less than the Ave. mean of A
32 PairSmp.Ttest2 <- t.test(After4weeks, After8weeks, data = T_test.data,
33 PairSmp.Ttest2
34
35 # Test whether Ave. mean of After4weeks is greater than the Ave. mean o
36 PairSmp.Ttest3 <- t.test(After4weeks, After8weeks, data = T_test.data,
37 PairSmp.Ttest3
38
39 # Visualize mean differences for the paired groups
40
41 prdSample <- paired(After4weeks, After8weeks)
42 plot(prdSample, type = "profile") + theme_bw()
43
44
```

```
data:  After4weeks and After8weeks
t = 3.7809, df = 17, p-value = 0.0007457
alternative hypothesis: true difference in means is greater than 0
95 percent confidence interval:
 0.0338935       Inf
sample estimates:
mean of the differences
             0.06277778

> prdSample <- paired(After4weeks, After8weeks)
> plot(prdSample, type = "profile") + theme_bw()
>
```

```
OneSmp.Ttest3     List of 10
PairSmp.Ttest     List of 10
PairSmp.Ttest2    List of 10
PairSmp.Ttest3    List of 10
prdSample         18 obs. of 2 variables
```

**Fig. 8.10** Plotting the mean difference for two paired groups of variables in R

By default, the hypothesis for conducting the test (Paired t-test) is; *IF* the p-value is less than or equal to 0.05 ($p \leq 0.05$), *THEN* we assume that the means of the two sets of variable or data are statistically different and that this is not by chance ($H_1$), *ELSE IF* the p-value is greater than 0.05 ($p > 0.05$) *THEN* we presume that there is no difference in the mean of the two sets of data ($H_0$).

```
> PairSmp.Ttest
      Paired t-test
data:  After4weeks and After8weeks
t = 3.7809, df = 17, p-value = 0.001491
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 0.02774658 0.09780897
sample estimates:
mean of the differences
             0.06277778
```

As reported in the results above (see different components of the t-test explained in detail in previous Sect. 8.2); the **t.test( )** function which we applied for the **Two-tailed** Paired Sample (`PairSmp.Ttest`) *t*-test shows that there is a difference in the mean between the two sets of analyzed data (`After4weeks, After8weeks`). The p-value for the **Two-tailed** test was statistically found as `p=0.001491,` which is significantly less than the scientifically accepted levels ($p \leq 0.05$). Therefore, we can statistically conclude that there is a significant difference between the means of the two sets of data (`After4weeks, After8weeks`) considering the two intervention periods.

Furthermore, as gathered in the results reported below for the **one-tailed** *t*-tests (see: Figs. 8.8 and 8.9b);

- We also checked whether the mean of the **After4weeks** variable is *less than* the mean of the **After8weeks** (`PairSmp.Ttest2`).
- Then checked whether the mean of the **After4weeks** is *greater than* the mean of the **After8weeks** (`PairSmp.Ttest3`).

```
> PairSmp.Ttest2
      Paired t-test
data:  After4weeks and After8weeks
t = 3.7809, df = 17, p-value = 0.9993
alternative hypothesis: true difference in means is less than 0
95 percent confidence interval:
       -Inf 0.09166206
sample estimates:
mean of the differences
             0.06277778
```

```
> PairSmp.Ttest3
        Paired t-test
data:  After4weeks and After8weeks
t = 3.7809, df = 17, p-value = 0.0007457
alternative hypothesis: true difference in means is greater than 0
95 percent confidence interval:
 0.0338935       Inf
sample estimates:
mean of the differences
             0.06277778
```

In the results of the **one-tailed** t-tests, we can see that when we analyzed whether the mean of the "**After4weeks**" variable is *less than* the mean of the "**After8weeks**" that there was no significant difference (`PairSmp.Ttest2, p=0.9993`). But when we analyzed whether the mean of the "**After4weeks**" is *greater than* the mean of the "**After8weeks"** that there was a significant difference (`PairSmp.Ttest3, p=0.0007457`). Therefore, it can be said from the tests that the mean of the "**After4weeks"** is *greater* and *not less* than the mean of the "**After8weeks"** which was statistically difference by a margin of `0.0627777` (see: mean of the differences) presented in both tests (one-tailed).

Henceforth, in summary, we can statistically say that there was a significant change or variation in the *mean* of the targeted variables (After4weeks and After8weeks) over the period of the intervention or experiment based on the example dataset we stored as "**T_test.data**". Perhaps, this result suggests a drop in the cholesterol levels of the participants across those periods, for instance.

## 8.4 One Sample T-test in R

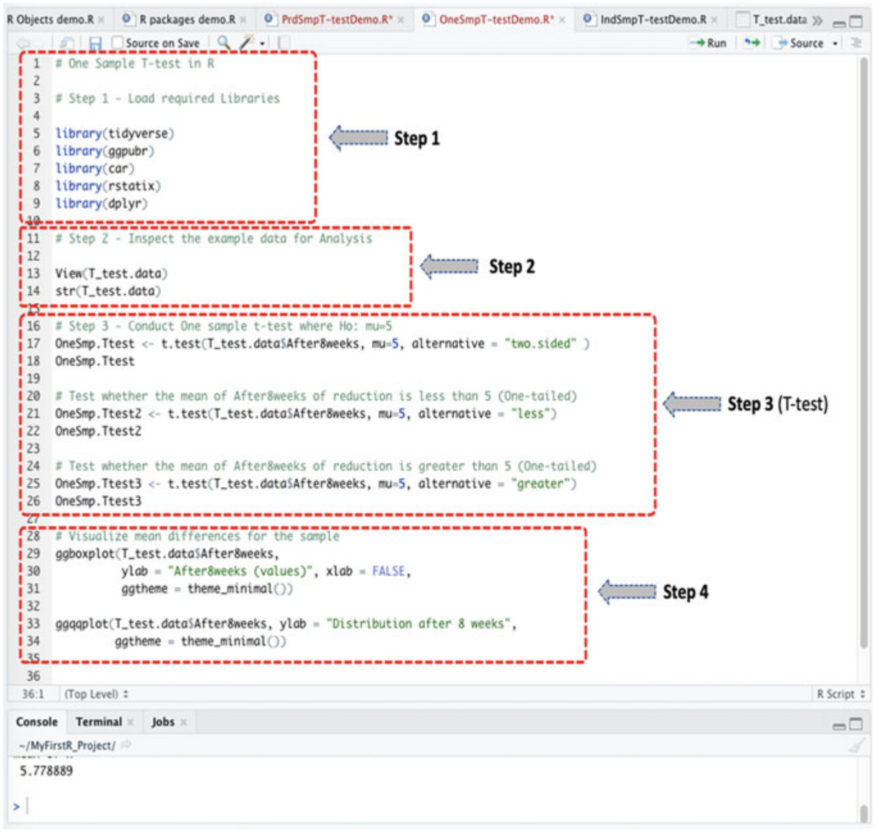
*One Sample T-test* is the third type of t-test that is used in statistical analysis or by the researchers to compare the mean of a single group of variable or data alongside a *known mean value*. The test as the name implies (one sample t-test) is usually performed to determine whether the specified sample mean is equal to the "hypothesized" data value (otherwise known as the *test mean*).

To demonstrate the One sample *t*-test, we will continue to use the same example dataset "**Choresterol_R.csv**" that we have stored as an R object we name "**T_test.data**" in R (see: Fig. 8.4).

To demonstrate the One sample *t*-test in R, create a **new R Script** and name it "**OneSmpT-testDemo**" (**remember the users can use any name they want provided reference to them are correctly made in the codes or program).

**# Step 1—Load the Required Libraries**

Again, since we have previously installed the necessary R packages in R for the previous t-tests we have looked into so far (see Sects. 8.2 and 8.3), we only need



```
1  # One Sample T-test in R
2
3  # Step 1 - Load required Libraries
4
5  library(tidyverse)
6  library(ggpubr)
7  library(car)
8  library(rstatix)
9  library(dplyr)
10
11 # Step 2 - Inspect the example data for Analysis
12
13 View(T_test.data)
14 str(T_test.data)
15
16 # Step 3 - Conduct One sample t-test where Ho: mu=5
17 OneSmp.Ttest <- t.test(T_test.data$After8weeks, mu=5, alternative = "two.sided" )
18 OneSmp.Ttest
19
20 # Test whether the mean of After8weeks of reduction is less than 5 (One-tailed)
21 OneSmp.Ttest2 <- t.test(T_test.data$After8weeks, mu=5, alternative = "less")
22 OneSmp.Ttest2
23
24 # Test whether the mean of After8weeks of reduction is greater than 5 (One-tailed)
25 OneSmp.Ttest3 <- t.test(T_test.data$After8weeks, mu=5, alternative = "greater")
26 OneSmp.Ttest3
27
28 # Visualize mean differences for the sample
29 ggboxplot(T_test.data$After8weeks,
30           ylab = "After8weeks (values)", xlab = FALSE,
31           ggtheme = theme_minimal())
32
33 ggqqplot(T_test.data$After8weeks, ylab = "Distribution after 8 weeks",
34          ggtheme = theme_minimal())
35
36
```

```
5.778889

>
```

**Fig. 8.11** Different steps to performing the one sample T-tests in R

to load the necessary libraries as shown in the code below and in Fig. 8.11 (Step1, Lines 3 to 9).

```
library(tidyverse)
library(ggpubr)
library(car)
library(rstatix)
library(dplyr)
```

## # Step 2—Inspect the example dataset for Analysis

As illustrated in Fig. 8.11 (Step 2), the users do not need to import the example dataset again. Just inspect or view the data (if the user wish) to make sure to choose the correct variable that you would like to analyze or that are contained there in the data. Use the following R functions **“View( )** or **str( )”** to view the details of the example dataset we named and stored as “**T_test.data”** (Fig. 8.11, Lines 11 to 14).

```
View(T_test.data)
str(T_test.data)
```

***__Note__: the users can use the following code presented below to import and attach the example dataset if this is their first time or if they have exited R or directly visited this particular section of the chapter. The users that have continued from our previous examples and analysis in Sect. 8.2 and 8.3 can ignore using this command again and move straight to step 3.

```
T_test.data <- read.csv(file.choose())
attach(T_test.data)
View(T_test.data)
str(T_test.data)
```

## # Step 3—Conduct the One Sample *t*-test

As defined earlier in the introduction section (Sect. 8.1);

- **One Sample *t*-test** is used to determine the mean of a particular (one) group of data or sample drawn from the same population compared against a standard *mean* value. For example, in our example dataset, we will analyze the mean value of the "**After8weeks**" variable across the data.
- Note that the target variable must be numeric.

As a general rule of thumb, the users are required to conduct the tests of assumptions before applying the t-tests. However, since we have performed the tests of assumptions for the "**After8weeks**" variable in the previous section (see: Sect. 8.3, Fig. 8.8, Step 3A, Line 21), we will skip this phase in this section and proceed directly with conducting the One sample T-test (Fig. 8.11, Step 3, Lines 16 to 26).

For the One Sample *t*-test:

- We will check whether the mean effect of the cholesterol reduction for "**After8weeks**" variable is *equal to* the standard mean value of 5, for instance, from the look of our example data. In other words, whether the mean cholesterol reduction of "After8weeks" in the population is different from 5 (two-tailed)
- Also, we will check whether the mean reduction in the cholesterol for "**After8weeks**" is *less than* 5 (one-tailed test), and then
- Test whether the mean reduction in cholesterol for the "**After8weeks**" is *greater than* 5 (one-tailed test).

The syntax to conduct the above-listed One sample t-tests (Fig. 8.11, Step 3) is presented in the codes below:

```
# Step 3 - Conduct One sample t-test where Ho: mu=5
OneSmp.Ttest  <-  t.test(T_test.data$After8weeks,  mu=5,  alternative  =
"two.sided" )
OneSmp.Ttest

# Test whether the mean of After8weeks is less than 5 (One-tailed)
OneSmp.Ttest2  <-  t.test(T_test.data$After8weeks,  mu=5,  alternative  =
"less")
OneSmp.Ttest2

# Test whether the mean of After8weeks is greater than 5 (One-tailed)
OneSmp.Ttest3  <-  t.test(T_test.data$After8weeks,  mu=5,  alternative  =
"greater")
OneSmp.Ttest3
```

**Note**: `mu` represents the hypothetical mean assumed or anticipated by the researcher. For instance, in our case we chose 5 (`mu = 5`) based on the various likely fields contained in the dataset (see: Fig. 8.4). It is noteworthy to mention that the value of the `mu` can be changed usually based on the frequent data values or samples.

Once the user have successfully run the codes (see Fig. 8.11, Step 3), you will be presented with the results of the One sample T-tests analysis in the Console as shown in Fig. 8.12.

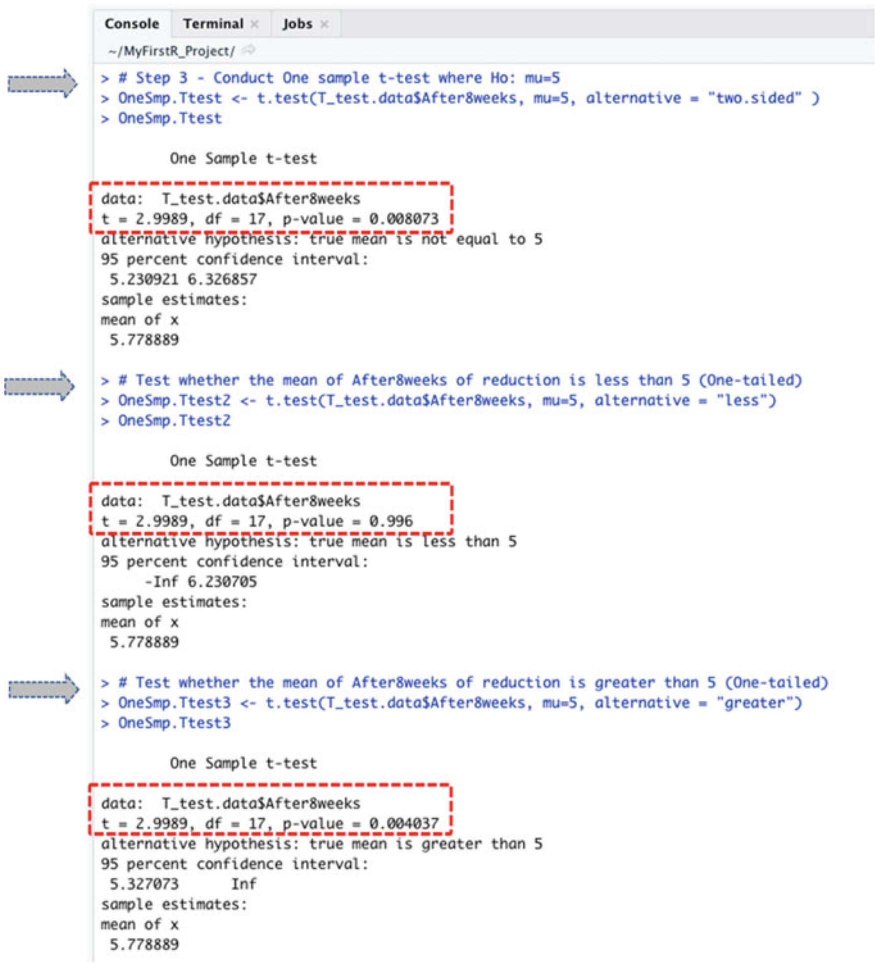
As reported in Fig. 8.12, we conducted the One Sample *t*-test by comparing the mean of the "**After8weeks"** variable after cholesterol reduction in the example dataset we named "**T_test.data" in R** against a hypothesized standard mean value of 5 (mu = 5). The result of the tests was stored in R objects we called "`OneSmp.Ttest`" for the **two-tailed** analysis, and "`OneSmp.Ttest2`" and "`OneSmp.Ttest3`" for the **one-tailed** analysis, respectively.

## # Step 4—Plot and visualize the mean difference of the analyzed data

As defined in Fig. 8.11 (Step 4, Lines 28 to 34) and the resultant boxplot (Fig. 8.13a) and Quantile–quantile plot (Fig. 8.13b); we utilized the various types of **ggplot( )** functions in R to visualize the mean and/or relationship between the "**After8weeks"** variable and the standard mean values based on the example dataset "**T_test.data**".

The code used for the mean boxplot using the ggboxplot( ) function, and quantile–quantile plot using the ggqqplot( ) function is as shown in the code below, and the results represented in Fig. 8.13a and b.

```
# Visualize estimated mean difference for the sample data
  ggboxplot(T_test.data$After8weeks,
          ylab = "After8weeks (values)", xlab = FALSE,
          ggtheme = theme_minimal())

ggqqplot(T_test.data$After8weeks,  ylab  =  "Distribution  after  8
weeks",
         ggtheme = theme_minimal())
```



Console Terminal Jobs

~/MyFirstR_Project/

```
> # Step 3 - Conduct One sample t-test where Ho: mu=5
> OneSmp.Ttest <- t.test(T_test.data$After8weeks, mu=5, alternative = "two.sided" )
> OneSmp.Ttest

        One Sample t-test

data:  T_test.data$After8weeks
t = 2.9989, df = 17, p-value = 0.008073
alternative hypothesis: true mean is not equal to 5
95 percent confidence interval:
 5.230921 6.326857
sample estimates:
mean of x
 5.778889

> # Test whether the mean of After8weeks of reduction is less than 5 (One-tailed)
> OneSmp.Ttest2 <- t.test(T_test.data$After8weeks, mu=5, alternative = "less")
> OneSmp.Ttest2

        One Sample t-test

data:  T_test.data$After8weeks
t = 2.9989, df = 17, p-value = 0.996
alternative hypothesis: true mean is less than 5
95 percent confidence interval:
     -Inf 6.230705
sample estimates:
mean of x
 5.778889

> # Test whether the mean of After8weeks of reduction is greater than 5 (One-tailed)
> OneSmp.Ttest3 <- t.test(T_test.data$After8weeks, mu=5, alternative = "greater")
> OneSmp.Ttest3

        One Sample t-test

data:  T_test.data$After8weeks
t = 2.9989, df = 17, p-value = 0.004037
alternative hypothesis: true mean is greater than 5
95 percent confidence interval:
 5.327073      Inf
sample estimates:
mean of x
 5.778889
```

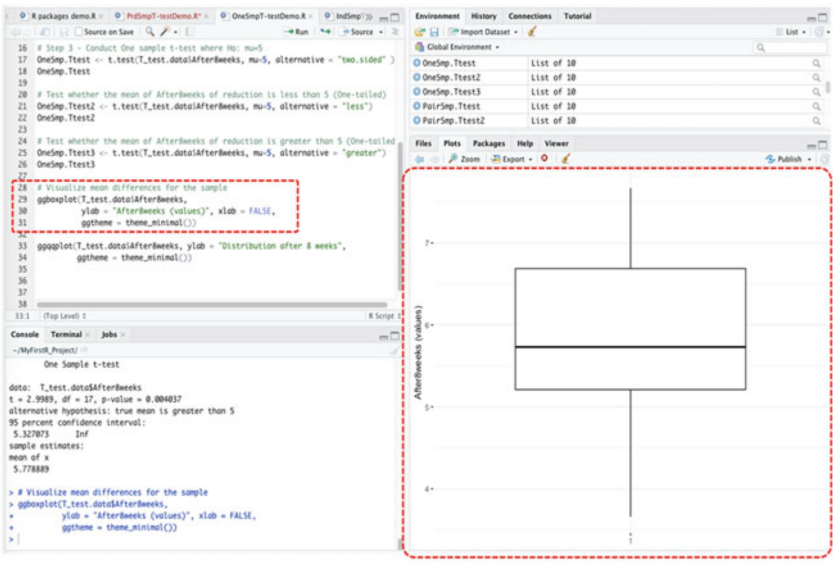
**Fig. 8.12** Result of one sample T-test displayed in the console in R

## # Step 5—Interpretation of Result of One sample t-test

The final step in the One sample *t*-test analysis is to understand and interpret the results of the test.

By default, the hypothesis for conducting the test (One sample t-test) is; *IF* the p-value is less than or equal to 0.05 ($p \leq 0.05$), *THEN* we assume that the *mean* of the single variable is statistically different from the standard mean score or value ($H_1$), *ELSE IF* the p-value is greater than 0.05 ($p > 0.05$) THEN we can presume that there is no difference in the mean of the variable and the standard mean value ($H_0$).

(a)

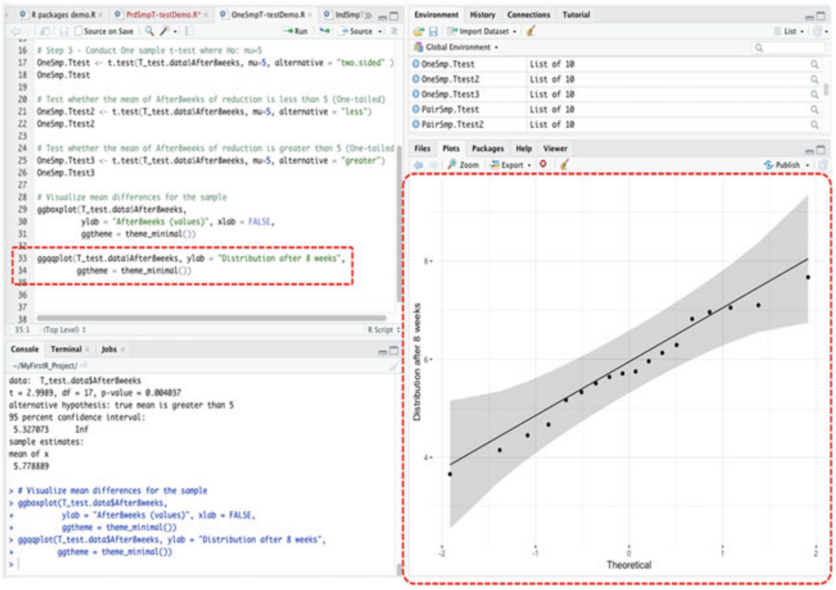


```
# Step 3 - Conduct One sample t-test where Ho: mu=5
OneSmp.Ttest <- t.test(T_test.data$After8weeks, mu=5, alternative = "two.sided" )
OneSmp.Ttest

# Test whether the mean of After8weeks of reduction is less than 5 (One-tailed)
OneSmp.Ttest2 <- t.test(T_test.data$After8weeks, mu=5, alternative = "less")
OneSmp.Ttest2

# Test whether the mean of After8weeks of reduction is greater than 5 (One-tailed
OneSmp.Ttest3 <- t.test(T_test.data$After8weeks, mu=5, alternative = "greater")
OneSmp.Ttest3

# Visualize mean differences for the sample
ggboxplot(T_test.data$After8weeks,
          ylab = "After8weeks (values)", xlab = FALSE,
          ggtheme = theme_minimal())

ggqqplot(T_test.data$After8weeks, ylab = "Distribution after 8 weeks",
         ggtheme = theme_minimal())
```

```
        One Sample t-test

data:  T_test.data$After8weeks
t = 2.9989, df = 17, p-value = 0.004037
alternative hypothesis: true mean is greater than 5
95 percent confidence interval:
 5.327073      Inf
sample estimates:
mean of x
 5.778889

> # Visualize mean differences for the sample
> ggboxplot(T_test.data$After8weeks,
+           ylab = "After8weeks (values)", xlab = FALSE,
+           ggtheme = theme_minimal())
> 
```

```
OneSmp.Ttest     List of 10
OneSmp.Ttest2    List of 10
OneSmp.Ttest3    List of 10
PairSmp.Ttest    List of 10
PairSmp.Ttest2   List of 10
```

(b)



```
# Step 3 - Conduct One sample t-test where Ho: mu=5
OneSmp.Ttest <- t.test(T_test.data$After8weeks, mu=5, alternative = "two.sided" )
OneSmp.Ttest

# Test whether the mean of After8weeks of reduction is less than 5 (One-tailed)
OneSmp.Ttest2 <- t.test(T_test.data$After8weeks, mu=5, alternative = "less")
OneSmp.Ttest2

# Test whether the mean of After8weeks of reduction is greater than 5 (One-tailed
OneSmp.Ttest3 <- t.test(T_test.data$After8weeks, mu=5, alternative = "greater")
OneSmp.Ttest3

# Visualize mean differences for the sample
ggboxplot(T_test.data$After8weeks,
          ylab = "After8weeks (values)", xlab = FALSE,
          ggtheme = theme_minimal())

ggqqplot(T_test.data$After8weeks, ylab = "Distribution after 8 weeks",
         ggtheme = theme_minimal())
```

```
data:  T_test.data$After8weeks
t = 2.9989, df = 17, p-value = 0.004037
alternative hypothesis: true mean is greater than 5
95 percent confidence interval:
 5.327073      Inf
sample estimates:
mean of x
 5.778889

> # Visualize mean differences for the sample
> ggboxplot(T_test.data$After8weeks,
+           ylab = "After8weeks (values)", xlab = FALSE,
+           ggtheme = theme_minimal())
> ggqqplot(T_test.data$After8weeks, ylab = "Distribution after 8 weeks",
+          ggtheme = theme_minimal())
> 
```

```
OneSmp.Ttest     List of 10
OneSmp.Ttest2    List of 10
OneSmp.Ttest3    List of 10
PairSmp.Ttest    List of 10
PairSmp.Ttest2   List of 10
```

**Fig. 8.13** **a** Plotting estimated mean for single variable in R using the ggboxplot( ) function. **b** Quantile–Quantile plot for relationship between a single variable and the mean values in R using the ggqqplot( ) function

```
> OneSmp.Ttest <- t.test(T_test.data$After8weeks, mu=5) # Ho: mu=5
> OneSmp.Ttest
        One Sample t-test
data:  T_test.data$After8weeks
t = 2.9989, df = 17, p-value = 0.008073
alternative hypothesis: true mean is not equal to 5
95 percent confidence interval:
 5.230921 6.326857
sample estimates:
mean of x
 5.778889
```

Accordingly, as gathered in the result of the One Sample T-test presented above; the p-value for the test (`OneSmp.Ttest`) is `p=0.008073` which is far less than the default significant levels of $p \leq 0.05$. Therefore, we can statistically conclude that there is a significant difference between the mean of the `After8weeks` variable and the pre-defined standard mean value, where `mu = 5` (two-tailed). Thus, we reject the null hypothesis that the mean of the `After8weeks` is equal to 5 whereby the means **sample estimate** is equal to `5.778889.`

***meaning of the different elements or components of t-tests results have been explained in detail in the previous Sect. 8.2***.

As gathered in the tests result presented below for the **“**one-tailed” one sample *t*-tests (see: Figs. 8.8 and 8.9b);

- We also checked whether the mean of the **After8weeks** variable is *less than* the mean of the hypothesized value of 5, where `mu=5 (OneSmp.Ttest2)`, and then
- Checked whether the mean of the **After8weeks** variable is *greater than* the mean of the hypothesized value of 5, where `mu=5 (OneSmp.Ttest3)`.

```
> OneSmp.Ttest2 <- t.test(T_test.data$After8weeks, mu=5, alternative="less")
> OneSmp.Ttest2
   One Sample t-test
data:  T_test.data$After8weeks
t = 2.9989, df = 17, p-value = 0.996
alternative hypothesis: true mean is less than 5
95 percent confidence interval:
     -Inf 6.230705
sample estimates:
mean of x
 5.778889
```

```
> OneSmp.Ttest3 <- t.test(T_test.data$After8weeks, mu=5, alternative =
"greater")
> OneSmp.Ttest3
    One Sample t-test
data:  T_test.data$After8weeks
t = 2.9989, df = 17, p-value = 0.004037
alternative hypothesis: true mean is greater than 5
95 percent confidence interval:
 5.327073      Inf
sample estimates:
mean of x
 5.778889
```

From the results of the **one-tailed** tests above, we can see that when we analyzed whether the mean of the "After8weeks" variable is *less than* the hypothesized "standard mean of 5" (mu=5) that there was no significant value (`OneSmp.Ttest2, p=0.996`). But when we analyzed whether the mean of "After8weeks" is *greater than* the hypothesized "standard mean of 5" that there was a significant score or value (`OneSmp.Ttest3, p=0.004037`). This means that most of the values representing the analyzed single variable (After8weeks) for the different participants have a value greater than 5.

## 8.5 Summary

In this chapter, the authors illustrated how to conduct the three main types of T-tests (Independent, Paired, and One sample) analysis in R. In Sect. 8.2, we explained and illustrated to the readers how to perform the Independent samples (unpaired or Two-sample) *t*-test. Section 8.3 covers how to conduct the Paired sample (Dependent) *t*-test. While in Sect. 8.4, we demonstrated how to perform the One sample *t*-test using R.

Also, this chapter covered how to graphically plot the *mean* of the different analyzed variables in the data and/or results of the t-tests. We discussed in detail how to interpret and understand the results of the three main types of T-tests in R.

To summarize the contents, the main topics covered in this chapter of the book are as follows:

- *t-test* is one of the inferential (parametric) statistics that are used for hypothesis testing, and for determining the significant differences (where applicable) between the *means* of two independent groups of data or single variable in a given data sample.

When choosing whether to conduct an *Independent*, *Paired* or *One sample* t-test? The researcher or data analyst should:

- Perform the "*independent sample t-test*" if the groups come from *two different populations* (i.e., statistically independent). For example, two different categories of people, thing, or place (gender: male and female, state: on and off, region: north and south, etc.).

- Perform "*paired sample t-test*" if the targeted group or variable comes from a single population but is analyzed at different time intervals (e.g., longitudinal study or a *pre* and *post* test intervention for the same group of people or place).
- Perform "*one sample t-test*" for one single group from the same population being analyzed against a standard *mean* value. For example, comparing the standard mean difference or effect of a particular teaching model on a specified group of students in a particular class with the average mean tested on sample estimate of 5 scores.

Additionally, the users will need to perform the "*two-tailed test*" statistics or calculation, if they only want to determine whether the mean of two data samples are different from one another. Whereas, on the other hand, they will also need to perform a "*one-tailed test*" if their goal includes also to determine whether the mean of a specific sample or variable is *less than* or *greater than* the mean of another variable or mean value, as the case may be.

## References

Adam, A. M. (2020). Sample size determination in survey research. *Journal of Scientific Research and Reports*, 90–97.https://doi.org/10.9734/jsrr/2020/v26i530263.

Kim, T. K. (2015). T test as a parametric statistic. *Korean Journal of Anesthesiology*, *68*(6), 540–546. https://doi.org/10.4097/kjae.2015.68.6.540.

NCSS. (2020). One-Sample T-Test. In *NCSS Statistical Software*. NCSS.com.

Novak, S. Y. (2020). On the T-test. In *Statistics theory*. http://arxiv.org/abs/2012.14530.

Okunev, R. (2022). Independent T-Test. In *Analytics for retail* (pp. 107–114). Apress. https://doi.org/10.1007/978-1-4842-7830-7_9.

Ramtin, S. (2023). How to choose appropriate bivariate test. In A. E. M. Eltorai, J. A. Bakal, S. F. DeFroda, & B. D. Owens (Eds.), *Translational sports medicine* (pp. 145–149). Academic Press. https://doi.org/10.1016/B978-0-323-91259-4.00115-6.

Roscoe, J. T. (1975). *Fundamental research statistics for the behavioral sciences* (2nd ed.). Holt, Rinehart, and Winston.

Skaik, Y. (2015). The bread and butter of statistical analysis "t-test": Uses and misuses. In *Pakistan Journal of Medical Sciences* (Vol. 31, Issue 6, pp. 1558–1559). Professional Medical Publications. https://doi.org/10.12669/pjms.316.8984.

Xu, M., Fralick, D., Zheng, J. Z., Wang, B., Tu, X. M., & Feng, C. (2017). The differences and similarities between two-sample t-test and paired t-test. *Shanghai Archives of Psychiatry*, *29*(3), 184–188. https://doi.org/10.11919/j.issn.1002-0829.217070.

# Chapter 9
# Analysis of Variance (ANOVA) in R: One-Way and Two-Way ANOVA

## 9.1 Introduction

*Analysis of variance* (ANOVA), also known as $F$-test, is one of the inferential statistical tests of hypothesis mostly applied by researchers or the data analysts to compare/determine the *differences in mean* between data samples that are represented in more than two *independent* comparison groups (usually categorical or ordinal) and a continuous dependent variable (Connelly, 2021; Sullivan, 2020). The ANOVA statistics can be regarded as an extension of the Independent Samples $t$-test (see: Chap. 8), that is mainly used when there are specifically two or more groups of independent variables(s) being compared against a continuous dependent variable. Therefore, ANOVA tests are only applicable when the data sample that is being analyzed is made up of *more than two groups* (i.e., a minimum of three) of an independent variable(s). The main aim of using the tests (ANOVA) is to statistically examine the differences (variability) that may exist within the groups of the (independent) variables being compared, as well as, among the groups that are being compared. Thus, statistically ANOVA tests determine whether the *means* of the three or more groups of an independent variable(s) are different taking into account the influence (usually referred to as Between-subject effect) that they may have on the dependent variable. With ANOVA, researchers or data analysts can ascertain the statistical significance of both the main effects (the variation) and their interaction (i.e., between-subjects effects) based on the significant values, usually determined through the p-values ($p \leq 0.05$).

The formula for calculating ANOVA is explained as follows: it uses the *F-test* to determine whether the group *means* are equal by including the correct *variances* in the *ratio* (Connelly, 2021). In other words, the F-statistic is the ratio where:

$$F = \textit{variation between sample means} / \textit{variation within the samples}$$

Thus,

K. Okoye and S. Hosseini, *R Programming*,
https://doi.org/10.1007/978-981-97-3385-9_9

$$F = MSE/MST$$

where:

F = ANOVA coefficient, MST = Mean sum of squares due to treatment**,** and MSE = Mean sum of squares due to error.

There are two main types of ANOVA tests commonly used in the works of literature (Christensen, 2020; Guillén-Gámez et al., 2021; Nibrad, 2019). These are:

- **One-way ANOVA**: used to compare the differences in mean between one (categorical or ordinal) independent variable and one (continuous) dependent variable, whereby the independent variable must have at least three levels, i.e., a minimum of three different groups or categories.
- **Two-way ANOVA**: used to compare the differences in mean between two independent variables (with three or more multilevel) and one (continuous) dependent variable. For example, it is used for examining the effects that two factors (independent variables) may have on the population of the study (continuous dependent variable) simultaneously or all at the same time.

Other types of ANOVA statistics or multivariate analysis are also used in the existing literature or statistical analysis, such as the multivariate analysis of variance (MANOVA) (Dugard et al., 2022; Okoye et al., 2022), analysis of co-variance (ANCOVA) (Kaltenecker & Okoye, 2023; Li & Chen, 2019), multivariate analysis of co-variance (MANCOVA) (Li & Chen, 2019; Okoye et al., 2023), etc.

Just like many of the different types of *parametric* tests or statistical procedures; the main "assumptions" or "conditions" that are necessary for performing the ANOVA tests both for research experiments or data analytics are summarized as follows (Connelly, 2021; Sullivan, 2020)—see Chap. 6, Sect. 6.2.5:

- *Independence of cases*: there should be no relationship among the observations in each group or among the groups of the variables themselves, i.e., independence of observations must hold.
- *Normality of data*: there should be no significant outliers, that might have a negative effect on the ANOVA test. The dependent variable should have an approximately normal distribution for each category of the target independent variable.
- *Homogeneity of variances*: the variance among the groups must hold or should be approximately equal.
- The *independent variable(s)* must consist of more than two independent groups or categories, i.e., a minimum of three groups or levels.
- The *independent variable(s)* must be categorical or ordinal.
- The *dependent variable* must be continuous.

In the next sections of this chapter (Sects. 9.2 and 9.3); the authors will explain and demonstrate to the readers how to conduct the One-way and Two-way ANOVA tests in R. We will illustrate the different steps to performing the two tests using the following steps in R outlined in Fig. 9.1.

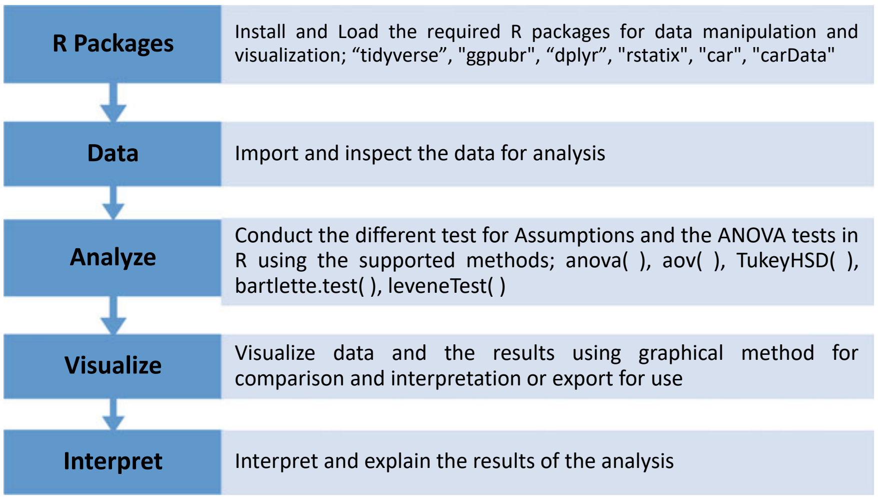

R Packages
Install and Load the required R packages for data manipulation and visualization; "tidyverse", "ggpubr", "dplyr", "rstatix", "car", "carData"
Data
Import and inspect the data for analysis
Analyze
Conduct the different test for Assumptions and the ANOVA tests in R using the supported methods; anova( ), aov( ), TukeyHSD( ), bartlette.test( ), leveneTest( )
Visualize
Visualize data and the results using graphical method for comparison and interpretation or export for use
Interpret
Interpret and explain the results of the analysis

**Fig. 9.1** Steps to conducting the ANOVA test and analysis in R

## 9.2 One-Way ANOVA Test in R

*One-way ANOVA* is used when the dataset the researcher or analysts wants to investigate are made up of *more than two groups* of an independent variable and are statistically *independent*, and a continuous dependent variable. Thus, the One-way ANOVA test as the name implies, is statistically used to compare the *differences in mean* between one (categorical or ordinal) independent variable and one (continuous) dependent variable, whereby the independent variable must have at least three levels, i.e., a minimum of three different groups or categories.

By default, the hypothesis for testing whether there is a *difference* or *variation in the mean* of the more than two (> 2) specified groups of independent data or variable against the one dependent (usually continuous) variable is; *IF* the p-value of the test is less than or equal to 0.05 ($p \leq 0.05$), *THEN* we assume that the mean of the groups of population (minimum of three) in the data sample are statistically different (i.e., varies) and that this is not by chance ($H_1$), *ELSE IF* the p-value is greater than 0.05 ($p > 0.05$) *THEN* we can conclude that there is no difference in the mean of the groups and any difference observed could only occur by chance ($H_0$).

The authors will practically demonstrate to the readers how to conduct the One-way ANOVA test in R using the **anova( ), aov( ),** and **bartlette.test( )** functions. We will do this by using the outlined steps in Fig. 9.1.

To begin with the illustration, **Open RStudio** and **create a new or open an existing project**. Once the user have RStudio and an R Project opened, **Create a new RScript** and name it "**OneWayANOVADemo**" or any name the user chooses (see: Chaps. 1 and 2).

Now, download an example file that we will use to demonstrate the two types of ANOVA analysis (One-Way and Two-way). ***Note the users can use any dataset or format of their choice provided they are able to follow the different steps described in the code by the authors in the illustration).

As shown in Fig. 8.2, download the example data named "**Diet_R.csv**" from the following source (https://www.sheffield.ac.uk/mash/statistics/datasets) and save it on your local machine or computer. ***Note: the readers can also refer to the following repository (https://doi.org/10.6084/m9.figshare.24728073) where the authors have uploaded all the example files used in this book to download the file.

Once the user have downloaded the file and saved it on the computer, we can proceed to conduct the first ANOVA analysis (One-way ANOVA) in R.

## # Step 1—Install and Load the required R Packages and Libraries

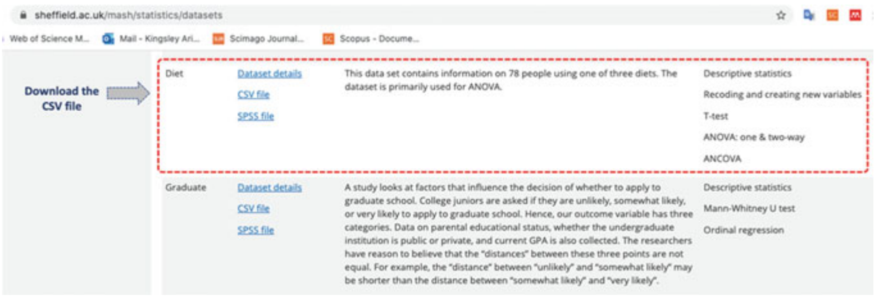
**Install** and **load** the following *R packages* and *libraries* (see Fig. 9.3, Step1, Lines 3 to 11) that will be used to call the different R functions, data manipulations, and graphical visualizations for the One-way ANOVA test.

The syntax and code to install and load the R packages and Libraries are as follows:

```
install.packages("tidyverse")
install.packages("ggpubr")
installed.packages("dplyr")

library(tidyverse)
library(ggpubr)
library(dplyr)
```

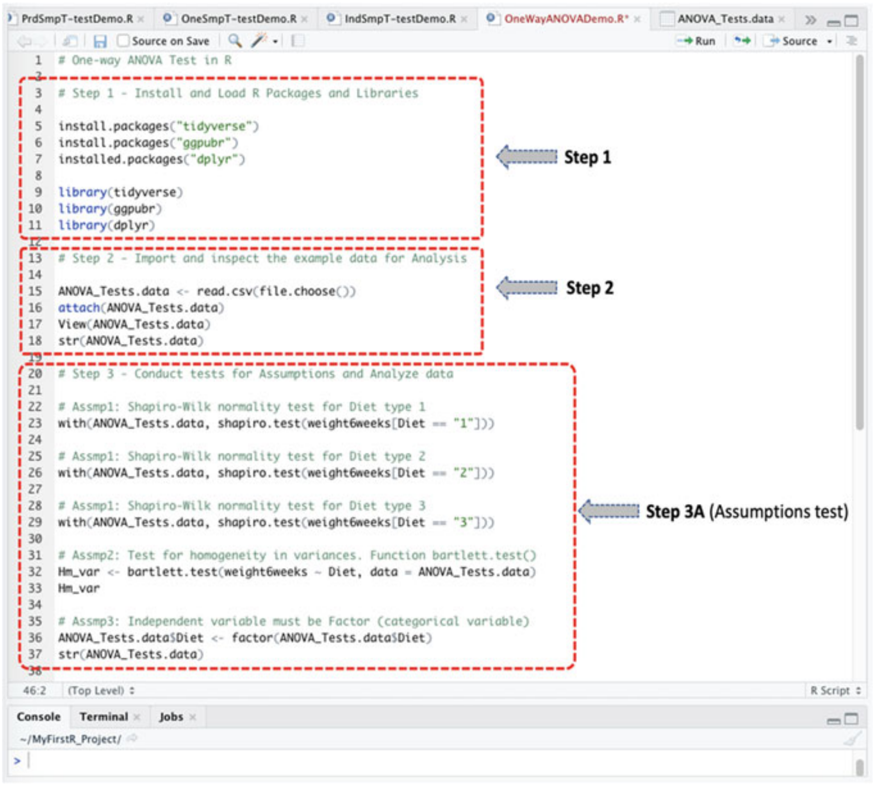
## # Step 2—Import and Inspect example dataset for Analysis

As defined in Fig. 7.3 (Step 2, Lines 13 to 18); import the dataset named "**Diet_R.csv**" that we have downloaded earlier, and store this in an R object named "**ANOVA_Tests.data**" (the users can use any name of their choice if they wish to do so).



| | | | |
|---|---|---|---|
| Diet | Dataset details<br>CSV file<br>SPSS file | This data set contains information on 78 people using one of three diets. The dataset is primarily used for ANOVA. | Descriptive statistics<br>Recoding and creating new variables<br>T-test<br>ANOVA: one & two-way<br>ANCOVA |
| Graduate | Dataset details<br>CSV file<br>SPSS file | A study looks at factors that influence the decision of whether to apply to graduate school. College juniors are asked if they are unlikely, somewhat likely, or very likely to apply to graduate school. Hence, our outcome variable has three categories. Data on parental educational status, whether the undergraduate institution is public or private, and current GPA is also collected. The researchers have reason to believe that the "distances" between these three points are not equal. For example, the "distance" between "unlikely" and "somewhat likely" may be shorter than the distance between "somewhat likely" and "very likely". | Descriptive statistics<br>Mann-Whitney U test<br>Ordinal regression |

**Fig. 9.2** Example of file download for ANOVA test. (*Source* https://www.sheffield.ac.uk/mash/statistics/datasets)



```
1  # One-way ANOVA Test in R
2
3  # Step 1 - Install and Load R Packages and Libraries
4
5  install.packages("tidyverse")
6  install.packages("ggpubr")
7  installed.packages("dplyr")
8
9  library(tidyverse)
10 library(ggpubr)
11 library(dplyr)
12
13 # Step 2 - Import and inspect the example data for Analysis
14
15 ANOVA_Tests.data <- read.csv(file.choose())
16 attach(ANOVA_Tests.data)
17 View(ANOVA_Tests.data)
18 str(ANOVA_Tests.data)
19
20 # Step 3 - Conduct tests for Assumptions and Analyze data
21
22 # Assmp1: Shapiro-Wilk normality test for Diet type 1
23 with(ANOVA_Tests.data, shapiro.test(weight6weeks[Diet == "1"]))
24
25 # Assmp1: Shapiro-Wilk normality test for Diet type 2
26 with(ANOVA_Tests.data, shapiro.test(weight6weeks[Diet == "2"]))
27
28 # Assmp1: Shapiro-Wilk normality test for Diet type 3
29 with(ANOVA_Tests.data, shapiro.test(weight6weeks[Diet == "3"]))
30
31 # Assmp2: Test for homogeneity in variances. Function bartlett.test()
32 Hm_var <- bartlett.test(weight6weeks ~ Diet, data = ANOVA_Tests.data)
33 Hm_var
34
35 # Assmp3: Independent variable must be Factor (categorical variable)
36 ANOVA_Tests.data$Diet <- factor(ANOVA_Tests.data$Diet)
37 str(ANOVA_Tests.data)
38
```

**Fig. 9.3** Steps to conducting one-way ANOVA test in R

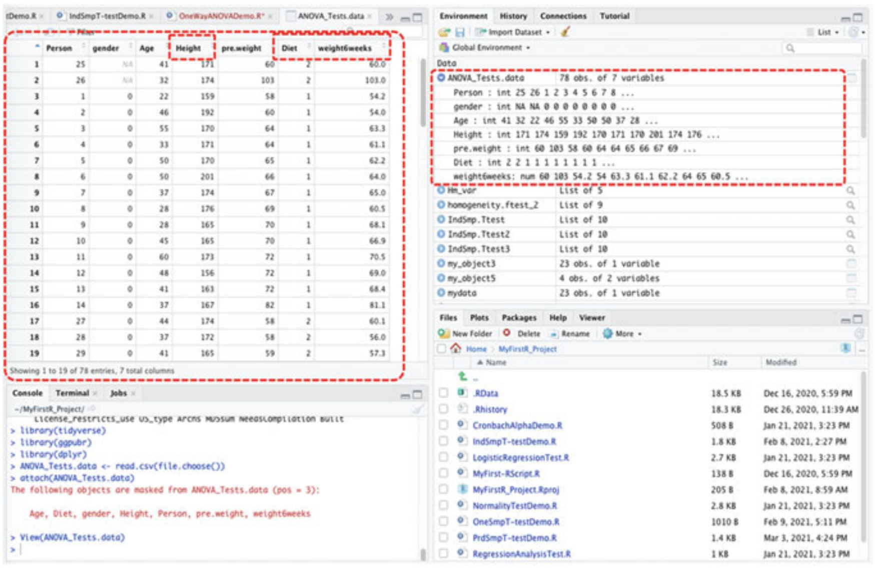
Once the user have successfully imported the dataset in R, you will be able to view the details of the **Diet_R.csv** file as shown in Fig. 9.4 with 78 observations and 7 variables in the sample data.

```
ANOVA_Tests.data <- read.csv(file.choose())
attach(ANOVA_Tests.data)
View(ANOVA_Tests.data)
str(ANOVA_Tests.data)
```

## # Step 3—Conduct the tests for Assumptions and Analyze the data

To analyze the imported dataset that we stored as **ANOVA_Tests.data** (see Fig. 9.4). First, the authors will be conducting the different tests of assumptions, e.g., normality test and homogeneity of variance (see: Sect. 9.1), before performing the actual One-way ANOVA analysis if the dataset in question meets or satisfies the necessary assumptions or test condition for the One-way ANOVA.

The syntax and code for conducting the different tests of assumptions are presented below and highlighted in Fig. 9.3 (Step 3A, Lines 20 to 37):



| | Person | gender | Age | Height | pre.weight | Diet | weight6weeks |
|---|---|---|---|---|---|---|---|
| 1 | 25 | NA | 41 | 171 | 60 | 2 | 60.0 |
| 2 | 26 | NA | 32 | 174 | 103 | 2 | 103.0 |
| 3 | 1 | 0 | 22 | 159 | 58 | 1 | 54.2 |
| 4 | 2 | 0 | 46 | 192 | 60 | 1 | 54.0 |
| 5 | 3 | 0 | 55 | 170 | 64 | 1 | 63.3 |
| 6 | 4 | 0 | 33 | 171 | 64 | 1 | 61.1 |
| 7 | 5 | 0 | 50 | 170 | 65 | 1 | 62.2 |
| 8 | 6 | 0 | 50 | 201 | 66 | 1 | 64.0 |
| 9 | 7 | 0 | 37 | 174 | 67 | 1 | 65.0 |
| 10 | 8 | 0 | 28 | 176 | 69 | 1 | 60.5 |
| 11 | 9 | 0 | 28 | 165 | 70 | 1 | 68.1 |
| 12 | 10 | 0 | 45 | 165 | 70 | 1 | 66.9 |
| 13 | 11 | 0 | 60 | 173 | 72 | 1 | 70.5 |
| 14 | 12 | 0 | 48 | 156 | 72 | 1 | 69.0 |
| 15 | 13 | 0 | 41 | 163 | 72 | 1 | 68.4 |
| 16 | 14 | 0 | 37 | 167 | 82 | 1 | 81.1 |
| 17 | 27 | 0 | 44 | 174 | 58 | 2 | 60.1 |
| 18 | 28 | 0 | 37 | 172 | 58 | 2 | 56.0 |
| 19 | 29 | 0 | 41 | 165 | 59 | 2 | 57.3 |

Showing 1 to 19 of 78 entries, 7 total columns

```
Data
ANOVA_Tests.data      78 obs. of 7 variables
  Person : int 25 26 1 2 3 4 5 6 7 8 ...
  gender : int NA NA 0 0 0 0 0 0 0 0 ...
  Age : int 41 32 22 46 55 33 50 50 37 28 ...
  Height : int 171 174 159 192 170 171 170 201 174 176 ...
  pre.weight : int 60 103 58 60 64 64 65 66 67 69 ...
  Diet : int 2 2 1 1 1 1 1 1 1 1 ...
  weight6weeks: num 60 103 54.2 54 63.3 61.1 62.2 64 65 60.5 ...
Hm_var                List of 5
homogeneity.ftest_2   List of 9
IndSmp.Ttest          List of 10
IndSmp.Ttest2         List of 10
IndSmp.Ttest3         List of 10
my_object3            23 obs. of 1 variable
my_object5            4 obs. of 2 variables
mydata                23 obs. of 1 variable
```

Home > MyFirstR_Project

| Name | Size | Modified |
|---|---|---|
| .. | | |
| .RData | 18.5 KB | Dec 16, 2020, 5:59 PM |
| .Rhistory | 18.3 KB | Dec 26, 2020, 11:39 AM |
| CronbachAlphaDemo.R | 508 B | Jan 21, 2021, 3:23 PM |
| IndSmpT-testDemo.R | 1.8 KB | Feb 8, 2021, 2:27 PM |
| LogisticRegressionTest.R | 2.7 KB | Jan 21, 2021, 3:23 PM |
| MyFirst-RScript.R | 138 B | Dec 16, 2020, 5:59 PM |
| MyFirstR_Project.Rproj | 205 B | Feb 8, 2021, 8:59 AM |
| NormalityTestDemo.R | 2.8 KB | Jan 21, 2021, 3:23 PM |
| OneSmpT-testDemo.R | 1010 B | Feb 9, 2021, 5:11 PM |
| PrdSmpT-testDemo.R | 1.4 KB | Mar 3, 2021, 4:24 PM |
| RegressionAnalysisTest.R | 1 KB | Jan 21, 2021, 3:23 PM |

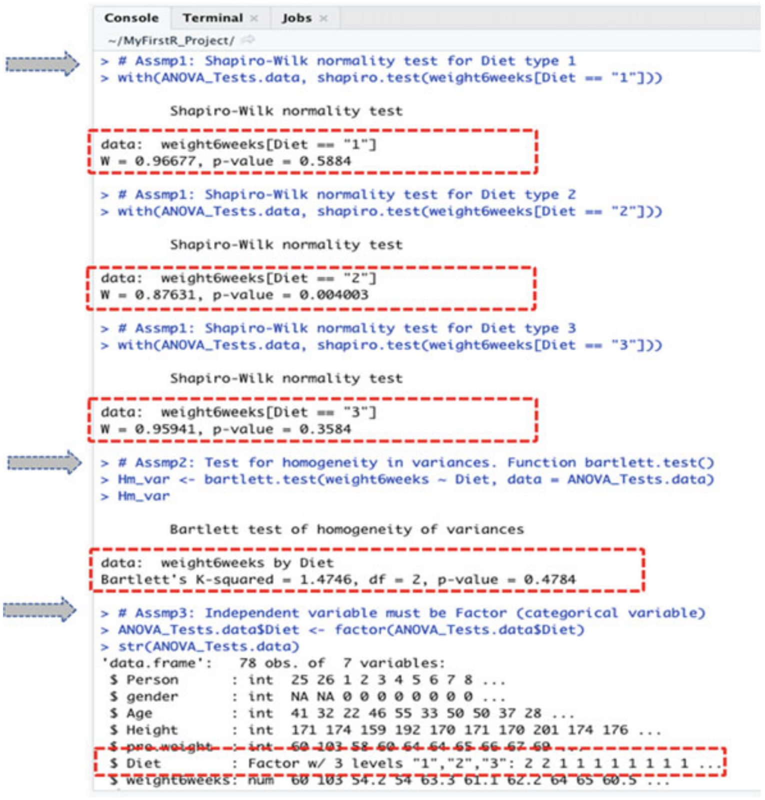
```
~/MyFirstR_Project/
    License_restricts_use OS_type Archs MD5sum NeedsCompilation Built
> library(tidyverse)
> library(ggpubr)
> library(dplyr)
> ANOVA_Tests.data <- read.csv(file.choose())
> attach(ANOVA_Tests.data)
The following objects are masked from ANOVA_Tests.data (pos = 3):

    Age, Diet, gender, Height, Person, pre.weight, weight6weeks

> View(ANOVA_Tests.data)
> |
```

**Fig. 9.4** Example of dataset imported and stored in RStudio environment as an R object

```
# Assmp1: Shapiro-Wilk normality test for Diet type 1
with(ANOVA_Tests.data, shapiro.test(weight6weeks[Diet == "1"]))

# Assmp1: Shapiro-Wilk normality test for Diet type 2
with(ANOVA_Tests.data, shapiro.test(weight6weeks[Diet == "2"]))

# Assmp1: Shapiro-Wilk normality test for Diet type 3
with(ANOVA_Tests.data, shapiro.test(weight6weeks[Diet == "3"]))

# Assmp2: Test for homogeneity in variances. Function bartlett.test()
Hm_var <- bartlett.test(weight6weeks ~ Diet, data = ANOVA_Tests.data)
Hm_var

# Assmp3: Independent variable must be Factor (categorical variable)
ANOVA_Tests.data$Diet <- factor(ANOVA_Tests.data$Diet)
str(ANOVA_Tests.data)
```

Once the user have successfully run the lines of codes defined in **Step 3A** (Fig. 9.3, Lines 20 to 37), they will be presented with the results of the "tests for assumptions" in the Console as shown in Fig. 9.5.

As highlighted in the results in Fig. 9.5; the *normality test* using Shapiro–Wilk's method shows that the distribution for majority of the different groups of "**Diet**" variable were normally distributed when considered against the target variable "**weight6weeks**", assuming *p-value of greater than 0.05, i.e., p > 0.05* and test statistics, W, of value greater than 0.5 as the threshold whereby: (`weight6weeks[Diet == "1"], W=0.96677, p-value=0.5884`), (`weight6weeks[Diet == "2"], W=0.87631, p-value=0.004003`), (`weight6weeks[Diet == "3"], W=0.95941, p-value=0.3584`). Therefore, from the results, we can assume or proceed to conduct the One-way ANOVA (parametric) analysis



```
Console   Terminal ×   Jobs ×
~/MyFirstR_Project/
> # Assmp1: Shapiro-Wilk normality test for Diet type 1
> with(ANOVA_Tests.data, shapiro.test(weight6weeks[Diet == "1"]))

        Shapiro-Wilk normality test

data:  weight6weeks[Diet == "1"]
W = 0.96677, p-value = 0.5884

> # Assmp1: Shapiro-Wilk normality test for Diet type 2
> with(ANOVA_Tests.data, shapiro.test(weight6weeks[Diet == "2"]))

        Shapiro-Wilk normality test

data:  weight6weeks[Diet == "2"]
W = 0.87631, p-value = 0.004003

> # Assmp1: Shapiro-Wilk normality test for Diet type 3
> with(ANOVA_Tests.data, shapiro.test(weight6weeks[Diet == "3"]))

        Shapiro-Wilk normality test

data:  weight6weeks[Diet == "3"]
W = 0.95941, p-value = 0.3584

> # Assmp2: Test for homogeneity in variances. Function bartlett.test()
> Hm_var <- bartlett.test(weight6weeks ~ Diet, data = ANOVA_Tests.data)
> Hm_var

        Bartlett test of homogeneity of variances

data:  weight6weeks by Diet
Bartlett's K-squared = 1.4746, df = 2, p-value = 0.4784

> # Assmp3: Independent variable must be Factor (categorical variable)
> ANOVA_Tests.data$Diet <- factor(ANOVA_Tests.data$Diet)
> str(ANOVA_Tests.data)
'data.frame':   78 obs. of  7 variables:
 $ Person      : int  25 26 1 2 3 4 5 6 7 8 ...
 $ gender      : int  NA NA 0 0 0 0 0 0 0 0 ...
 $ Age         : int  41 32 22 46 55 33 50 50 37 28 ...
 $ Height      : int  171 174 159 192 170 171 170 201 174 176 ...
 $ pre.weight  : int  60 103 58 60 64 64 65 66 67 69 ...
 $ Diet        : Factor w/ 3 levels "1","2","3": 2 2 1 1 1 1 1 1 1 1 ...
 $ weight6weeks: num  60 103 54.2 54 63.3 61.1 62.2 64 65 60.5 ...
```

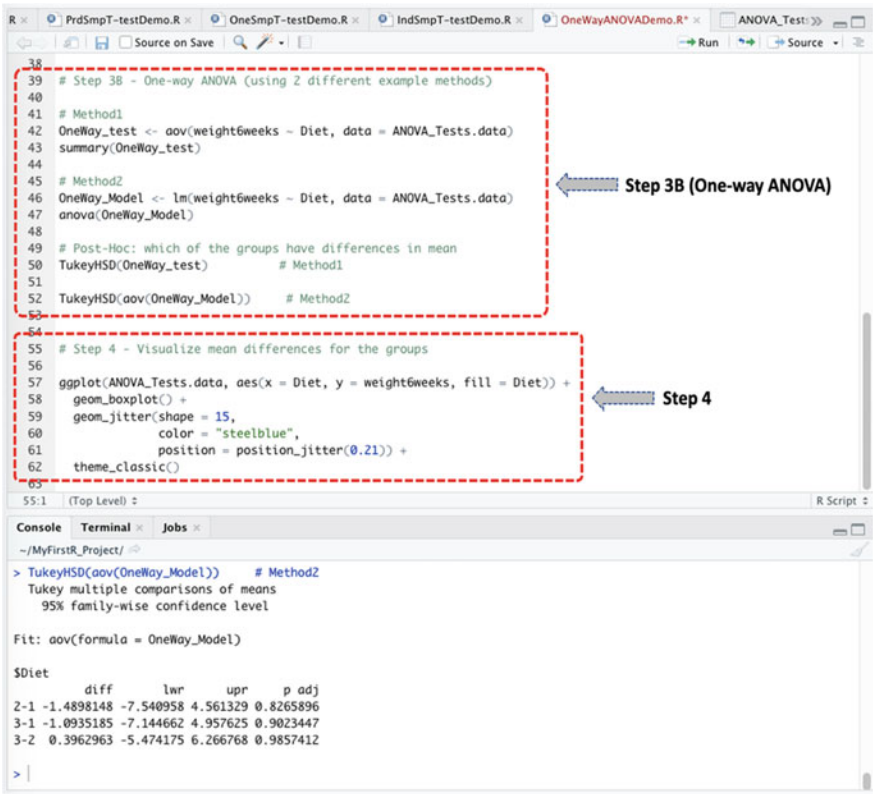
**Fig. 9.5** Result of tests for assumption prior to conducting the one-way ANOVA analysis in R

since the normality test and all the other necessary conditions are met. Moreover, it is important to mention that datasets which contains more than $n > 40$ samples or observations (see: Chap. 3) is considered also a scientifically acceptable sample size for conducting any type of the *parametric* tests in scientific research or statistical analysis purposes (Roscoe, 1975).

Furthermore, in the second test of assumption, we tested the *homogeneity of variance* for the two targeted/analyzed variables (weight6weeks ~ Diet) using the **bartlett.test( )** function in R; whereby we assume that a value of $p > 0.05$ indicates "equality in variance". As shown and highlighted in the results presented in Fig. 9.5, we can see that there are no difference in the *homogeneity of variance* for the two analyzed variables with `p-value = 0.4784`. Therefore, we accept that the assumption of equality in variance is met.

Lastly, in the third assumption (Fig. 9.5), we converted the independent variable "**Diet**" with 3 levels (1, 2, and 3) to a factor format to represent categorical values—see Chap. 2 for more details on Factorization in R.

With all the necessary conditions met, we can proceed to conduct the "**One-way ANOVA**" test using the **anova( )**, **aov( ),** and **TukeyHSD( )** methods or function in



```
38
39  # Step 3B - One-way ANOVA (using 2 different example methods)
40
41  # Method1
42  OneWay_test <- aov(weight6weeks ~ Diet, data = ANOVA_Tests.data)
43  summary(OneWay_test)
44
45  # Method2
46  OneWay_Model <- lm(weight6weeks ~ Diet, data = ANOVA_Tests.data)
47  anova(OneWay_Model)
48
49  # Post-Hoc: which of the groups have differences in mean
50  TukeyHSD(OneWay_test)          # Method1
51
52  TukeyHSD(aov(OneWay_Model))      # Method2
53
54
55  # Step 4 - Visualize mean differences for the groups
56
57  ggplot(ANOVA_Tests.data, aes(x = Diet, y = weight6weeks, fill = Diet)) +
58    geom_boxplot() +
59    geom_jitter(shape = 15,
60                color = "steelblue",
61                position = position_jitter(0.21)) +
62    theme_classic()
63
```

```
> TukeyHSD(aov(OneWay_Model))      # Method2
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = OneWay_Model)

$Diet
          diff       lwr      upr     p adj
2-1 -1.4898148 -7.540958 4.561329 0.8265896
3-1 -1.0935185 -7.144662 4.957625 0.9023447
3-2  0.3962963 -5.474175 6.266768 0.9857412

> 
```

**Fig. 9.6** One-way ANOVA test in R using two different types of method or approach

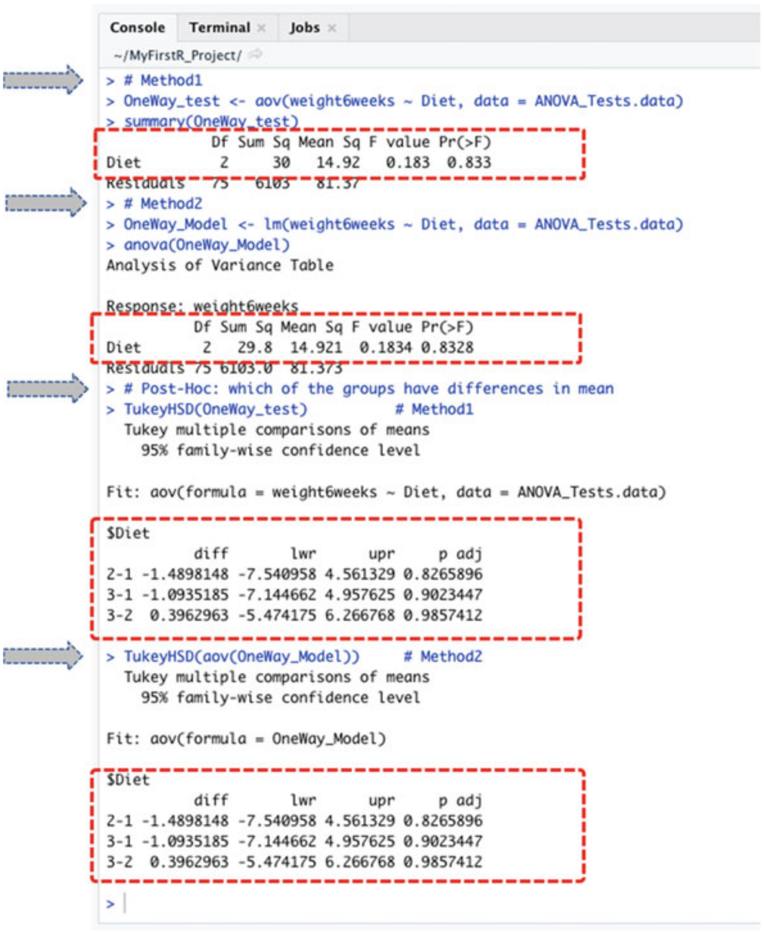
R, as described in **Step 3B** (Fig. 9.6, Lines 39 to 52) and consequently in the outcome of the ANOVA test or results represented in Fig. 9.7.

**Note:** as defined in the introduction section (Sect. 9.1);

- **One-way ANOVA** test compares the differences in mean between *one* independent variable (with three or more multilevel or groups) and *one* dependent (continuous) variable.
- The targeted "independent" variable (***x***) is often a categorical or ordinal type, while the "dependent" variable (***y***) must be numeric.

To demonstrate the One-way ANOVA using the example dataset we called "**ANOVA_Tests.data**" in R (see: highlighted columns and data in Fig. 9.4).

- We will test whether the mean of the 3 groups of **Diet** (the independent variable) varies, and if so, which diet was best for losing weight taking into account the "**weight6weeks**" (dependent) variable.

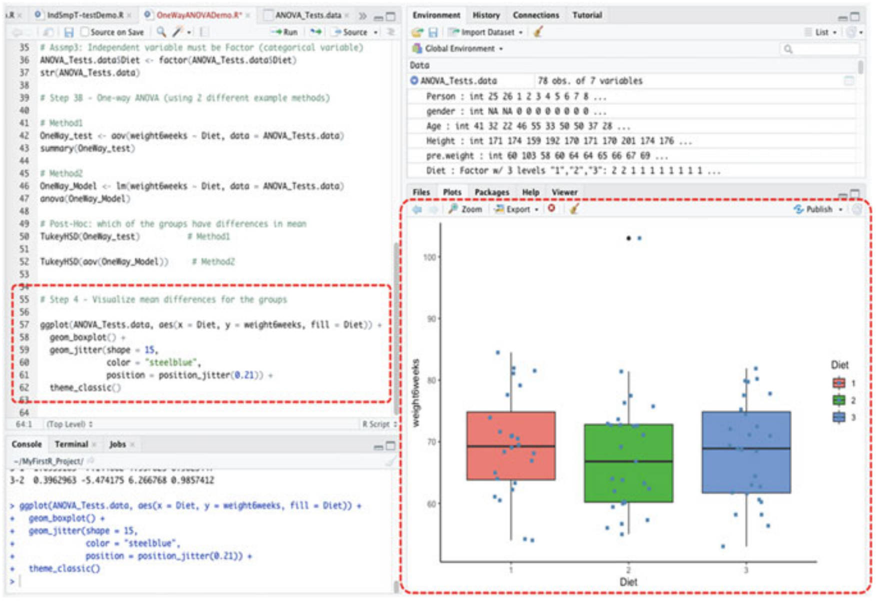


```
> # Method1
> OneWay_test <- aov(weight6weeks ~ Diet, data = ANOVA_Tests.data)
> summary(OneWay_test)
            Df Sum Sq Mean Sq F value Pr(>F)
Diet         2     30   14.92   0.183  0.833
Residuals   75   6103   81.37
> # Method2
> OneWay_Model <- lm(weight6weeks ~ Diet, data = ANOVA_Tests.data)
> anova(OneWay_Model)
Analysis of Variance Table

Response: weight6weeks
          Df Sum Sq Mean Sq F value Pr(>F)
Diet       2   29.8  14.921  0.1834 0.8328
Residuals 75 6103.0  81.373
> # Post-Hoc: which of the groups have differences in mean
> TukeyHSD(OneWay_test)           # Method1
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = weight6weeks ~ Diet, data = ANOVA_Tests.data)

$Diet
          diff       lwr      upr     p adj
2-1 -1.4898148 -7.540958 4.561329 0.8265896
3-1 -1.0935185 -7.144662 4.957625 0.9023447
3-2  0.3962963 -5.474175 6.266768 0.9857412

> TukeyHSD(aov(OneWay_Model))     # Method2
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = OneWay_Model)

$Diet
          diff       lwr      upr     p adj
2-1 -1.4898148 -7.540958 4.561329 0.8265896
3-1 -1.0935185 -7.144662 4.957625 0.9023447
3-2  0.3962963 -5.474175 6.266768 0.9857412

> |
```

**Fig. 9.7** Results of one-way ANOVA test in R

The syntax to performing this test in R is as shown in the codes below and in Fig. 9.6 (Step 3B, Lines 39 to 52).

```
# Method1
OneWay_test <- aov(weight6weeks ~ Diet, data = ANOVA_Tests.data)
summary(OneWay_test)

# Method2
OneWay_Model <- lm(weight6weeks ~ Diet, data = ANOVA_Tests.data)
anova(OneWay_Model)


# Post-Hoc: which of the groups have differences in mean
TukeyHSD(OneWay_test)          # Method1

TukeyHSD(aov(OneWay_Model))     # Method2
```



```
35 # Assmp3: Independent variable must be Factor (categorical variable)
36 ANOVA_Tests.data$Diet <- factor(ANOVA_Tests.data$Diet)
37 str(ANOVA_Tests.data)
38
39 # Step 3B - One-way ANOVA (using 2 different example methods)
40
41 # Method1
42 OneWay_test <- aov(weight6weeks ~ Diet, data = ANOVA_Tests.data)
43 summary(OneWay_test)
44
45 # Method2
46 OneWay_Model <- lm(weight6weeks ~ Diet, data = ANOVA_Tests.data)
47 anova(OneWay_Model)
48
49 # Post-Hoc: which of the groups have differences in mean
50 TukeyHSD(OneWay_test)          # Method1
51
52 TukeyHSD(aov(OneWay_Model))     # Method2
53
54
55 # Step 4 - Visualize mean differences for the groups
56
57 ggplot(ANOVA_Tests.data, aes(x = Diet, y = weight6weeks, fill = Diet)) +
58   geom_boxplot() +
59   geom_jitter(shape = 15,
60               color = "steelblue",
61               position = position_jitter(0.21)) +
62   theme_classic()
63
64
```

```
ANOVA_Tests.data        78 obs. of 7 variables
  Person : int 25 26 1 2 3 4 5 6 7 8 ...
  gender : int NA NA 0 0 0 0 0 0 0 0 ...
  Age : int 41 32 22 46 55 33 50 50 37 28 ...
  Height : int 171 174 159 192 170 171 170 201 174 176 ...
  pre.weight : int 60 103 58 60 64 64 65 66 67 69 ...
  Diet : Factor w/ 3 levels "1","2","3": 2 2 1 1 1 1 1 1 1 1 ...
```

```
~/MyFirstR_Project/
3-2  0.3962963 -5.474175 6.266768 0.9857412

> ggplot(ANOVA_Tests.data, aes(x = Diet, y = weight6weeks, fill = Diet)) +
+   geom_boxplot() +
+   geom_jitter(shape = 15,
+               color = "steelblue",
+               position = position_jitter(0.21)) +
+   theme_classic()
> 
```

Fig. 9.8 Plot of the mean difference for the 3 groups of the independent variable versus the dependent variable in R using the ggplot( ) function

As presented in Figs. 9.6 and 9.7, we conducted the **One-way ANOVA** test by considering the two variables (**weight6weeks ~ Diet**). We illustrated the ANOVA analysis using two different methods or functions in R; the aov( ) and anova( ) methods. Both methods (named **Method1** and **Method2, respectively**) produced the same results (see Fig. 9.7) and are explained in detail in the **Step 5** in the later part of this section.

## # Step 4—Plot and visualize the mean differences for the results or data

As illustrated in Fig. 9.8 (Step 4, Lines 55 to 62), the authors used the **ggplot( )** function in R to visualize the mean differences between the 3 groups of "**Diet**" (**1, 2** and **3**) representing the independent variable by taking into account the **"weight6weeks"** (dependent variable) as contained in the analyzed data "**ANOVA_Tests.data**".

The syntax to plot the mean or results of the analyzed variables is as shown in the code below, and the resultant graph represented in Fig. 9.8.

```
ggplot(ANOVA_Tests.data, aes(x = Diet, y = weight6weeks, fill = Diet)) +
geom_boxplot() +
geom_jitter(shape = 15,
            color = "steelblue",
            position = position_jitter(0.21)) +
theme_classic()
```

## # Step 5—Results Interpretation (One-way ANOVA)

The last step for One-way ANOVA analysis is to interpret and understand the results of the test.

By default, the hypothesis for conducting the test (One-way ANOVA) is; *IF* the p-value of the test result is less than or equal to 0.05 ($p \leq 0.05$), *THEN* we assume that the mean of the group (minimum of three levels or categories) of population in the data sample are statistically different (varies) and that this is not by chance ($H_1$), *ELSE IF* the p-value is greater than 0.05 ($p > 0.05$) *THEN* we can conclude that there is no difference in the mean of the groups and any difference observed could only occur by chance ($H_0$).

```
 > OneWay_Model <- lm(weight6weeks ~ Diet, data = ANOVA_Tests.data)
> anova(OneWay_Model)
Analysis of Variance Table

Response: weight6weeks
          Df  Sum Sq Mean Sq F value Pr(>F)
Diet       2    29.8  14.921  0.1834 0.8328
Residuals 75 6103.0  81.373
```

As shown in the result of the test presented above (see: Fig. 9.7); the different component or meaning of the **One-way ANOVA** test and outcome can be explained as a list containing the following:

- **Statistics**: `F = 0.1834` which signifies the ratio or value of the analysis of variance test.
- **p-value**: `p-value = 0.8328` is the p-value or significance levels of the test.

Statistically, as we can see from the results, the p-value (p=0.8328) is greater than the defined or acceptable significance levels ($p \leq 0.05$). Therefore, we can statistically conclude that there is no difference between the means of effect of the different groups of "**Diet**" after the 6 weeks of intervention considering the "**weight6weeks**" variable.

Also, to confirm the results of the One-way ANOVA test, a good practice by the researchers or statisticians is to check where the significant differences lies (if there was any).

To show the readers how to carry out this post-hoc test in R supposing we found any significant difference which the authors will be explaining more in detail in other chapters of this book; we conducted a post-hoc test using the **TukeyHSD( )** method by comparing the individual groups of diet against each other (see Fig. 9.7).

```
> TukeyHSD(aov(OneWay_Model))
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = OneWay_Model)

$Diet
          diff       lwr      upr     p adj
2-1 -1.4898148 -7.540958 4.561329 0.8265896
3-1 -1.0935185 -7.144662 4.957625 0.9023447
3-2    0.3962963 -5.474175 6.266768 0.9857412
```

As seen in the results above (see Fig. 9.7), we can see that there were no differences found between the subjects or comparisons for the 3 groups (group `2-1, p=0.8265896;` group `3-1, p=0.9023447;` group `3-2, p=0.9857412`), respectively. Thus, also confirming the results of the One-way ANOVA analysis we have explained earlier in this section.

## 9.3 Two-Way ANOVA Test in R

*Two-way ANOVA* is used when the dataset the researchers or analyst wants to analyze consists of "two independent variables" (with more than two groups) that are statistically *independent*. Unlike the One-way ANOVA that considers only one independent variable, the Two-way ANOVA is applied to compare the effects or differences in mean between two independent variables (categorical or ordinal) against one dependent (continuous) variable, whereby the independent variables must have at least three levels, i.e., a minimum of three different groups or categories.

***It is also noteworthy to mention that ANOVA tests can be performed for independent variables with *two groups* (although it is best recommended to use the Independent Samples *t*-test in this type of scenario)***.

By default, the hypothesis for testing whether there is a *difference* or *variation in the mean* of two specified groups of independent data samples (with three or more levels) against one dependent (usually continuous) variable is; *IF* the p-value of the test is less than or equal to 0.05 ($p \leq 0.05$), *THEN* we can assume that the impact or mean effect of the groups (usually minimum of three groups) of population in the data sample are statistically different (varies) and that this is not by chance ($H_1$), *ELSE IF* the p-value is greater than 0.05 ($p > 0.05$) *THEN* we can say that there is no effect or difference in the mean of the groups of variables and any difference observed could only occur by chance ($H_0$).

Let's continue to use the **Diet_R.csv** data we imported earlier and stored as an object we called "**ANOVA_Tests.data**" in R (see: Fig. 9.4) to illustrate how to perform the **Two-way ANOVA** using the **anova( ), aov( )** and **leveneTest( )** functions in R. We will do this using the same steps we have previously outlined in Fig. 9.1. ***Users can refer to the following repository to download the example file if they need to: https://doi.org/10.6084/m9.figshare.24728073.

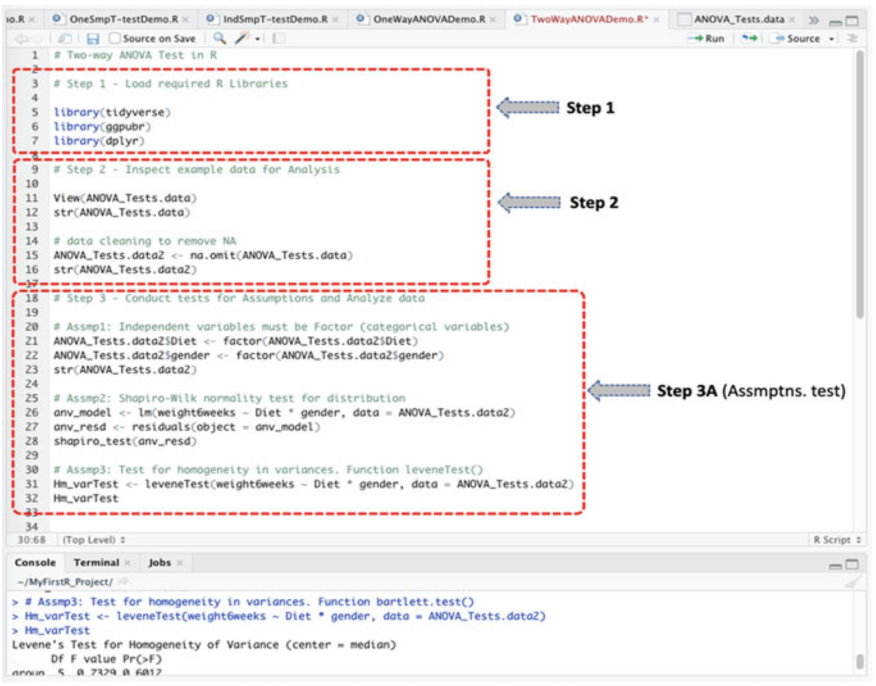
To begin, **Create a new R Script** in the current R project (this can be done by using the **file menu** option, see also Chaps. 1 and 2) and name it as "**TwoWayANOVADemo**".

## # Step 1—Install and Load the required R Packages and Libraries

**Load** the following *R libraries* (Fig. 9.9, Step1, Lines 3 to 7), that we will be using to call the different R functions, data manipulations, and graphical visualizations for the Two-way ANOVA analysis.

**Note**: we did not need to repeat or re-install the required R packages again as this has already been previously installed in RStudio in the previous example in Sect. 9.2. However, if the user have directly visited this particular section for the first time or have previously exited or reinstalled R, then they may require to install or re-install the necessary R packages listed below again (see Chap. 2, Sect. 2.6 on how to install the R packages in RStudio).

The syntax and code to run/load the required R Libraries are as follows:



```
1  # Two-way ANOVA Test in R
2
3  # Step 1 - Load required R Libraries
4
5  library(tidyverse)
6  library(ggpubr)
7  library(dplyr)
8
9  # Step 2 - Inspect example data for Analysis
10
11 View(ANOVA_Tests.data)
12 str(ANOVA_Tests.data)
13
14 # data cleaning to remove NA
15 ANOVA_Tests.data2 <- na.omit(ANOVA_Tests.data)
16 str(ANOVA_Tests.data2)
17
18 # Step 3 - Conduct tests for Assumptions and Analyze data
19
20 # Assmp1: Independent variables must be Factor (categorical variables)
21 ANOVA_Tests.data2$Diet <- factor(ANOVA_Tests.data2$Diet)
22 ANOVA_Tests.data2$gender <- factor(ANOVA_Tests.data2$gender)
23 str(ANOVA_Tests.data2)
24
25 # Assmp2: Shapiro-Wilk normality test for distribution
26 anv_model <- lm(weight6weeks ~ Diet * gender, data = ANOVA_Tests.data2)
27 anv_resd <- residuals(object = anv_model)
28 shapiro_test(anv_resd)
29
30 # Assmp3: Test for homogeneity in variances. Function leveneTest()
31 Hm_varTest <- leveneTest(weight6weeks ~ Diet * gender, data = ANOVA_Tests.data2)
32 Hm_varTest
33
34
```

```
> # Assmp3: Test for homogeneity in variances. Function bartlett.test()
> Hm_varTest <- leveneTest(weight6weeks ~ Diet * gender, data = ANOVA_Tests.data2)
> Hm_varTest
Levene's Test for Homogeneity of Variance (center = median)
      Df F value Pr(>F)
group  5  0.7329 0.6012
```

**Fig. 9.9** Steps to performing the two-way ANOVA test in R

```
library(tidyverse)
library(ggpubr)
library(dplyr)
```

***Note: for the leveneTest of assumption for homogeneity in variances (see Fig. 9.9, Step 3A, Assump: 3) by using the leveneTest( ) function, the user may also need or require to install the following additional highlighted R packages and libraries if they should encounter an error depending on the updated version of software installed.

```
install.packages("rstatix")
install.packages("car")

library(rstatix)
library(car)
library(carData)


library(tidyverse)
library(ggpubr)
library(dplyr)
```

## # Step 2—Inspect the example dataset for Analysis

Since we have already imported the "**Diet_R.csv**" and stored the example data file in an R object named "**ANOVA_Tests.data**" (Fig. 9.3) in the previous example in Sect. 9.2, the users do not need to import the data again. Rather, as shown in Fig. 9.9 (Step 2, Lines 9 to 12) you can view the dataset to inspect the different variables and confirm the items or variables we will be using to conduct the Two-way ANOVA test.

The code to do this is as shown below (see Fig. 9.9, Step 2, Lines 9 to 12).

```
View(ANOVA_Tests.data)
str(ANOVA_Tests.data)
```

***Note: In the event that the reader has exited or closed RStudio and returned back to this current section at a later time, or directly visited this section of the book or example, then the user would need to use the following code to attach the example file or re-read the data again, as the case may be:

```
ANOVA_Tests.data2 <- read.csv(file.choose())
attach(ANOVA_Tests.data2)
View(ANOVA_Tests.data2)
str(ANOVA_Tests.data2)
```

Also, one important data cleaning task that the authors would like to bring the readers' attention to and to illustrate, which is a good practice in scientific research and statistics, is to remove the incomplete rows or data with *NA* otherwise referred to as empty cells (see Fig. 9.4). The incomplete datasets (NA) can be removed by using the **na.omit( )** function in R. Moreover, the reason for cleaning this dataset is because we will be including the "**gender**" variable (see Fig. 9.4) in our analysis in this particular example or section.

The syntax to remove the NAs or empty cells is as shown in the code below (Fig. 9.9, Step 2, Lines 14 to 16).

```
# data cleaning to remove NA
ANOVA_Tests.data2 <- na.omit(ANOVA_Tests.data)
str(ANOVA_Tests.data2)
```

*****Note:** as you can see, when the user have successfully run the codes, a new set of data "without the NAs" will be created, and we stored this new dataset in an R object we called "**`ANOVA_Tests.data2`**".

Now we can proceed to conduct the next steps in the Two-way ANOVA analysis using the new cleaned data (**`ANOVA_Tests.data2`**).

**# Step 3—Conduct tests for Assumptions and Analyze the data**

As a necessary procedure, as shown in Fig. 9.9 (Step 3A, Lines 18 to 32), we will conduct the different tests of assumptions (i.e., check the variable types and format, normality test, and homogeneity of variances) before performing the Two-way ANOVA test.

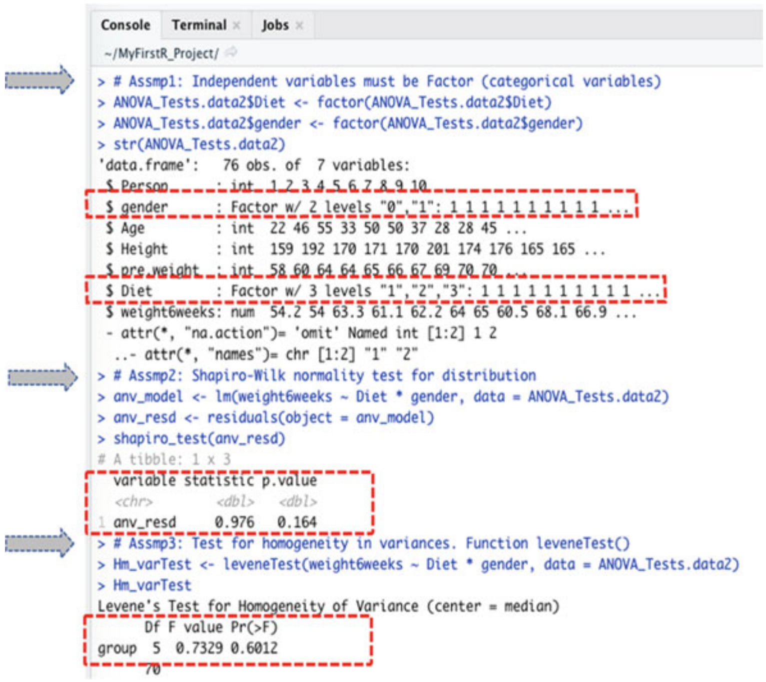
The code to conduct the different tests of assumptions is presented below (see Fig. 9.9, Step 3A):

```
# Assmp1: Independent variables must be Factor (categorical variables)
ANOVA_Tests.data2$Diet <- factor(ANOVA_Tests.data2$Diet)
ANOVA_Tests.data2$gender <- factor(ANOVA_Tests.data2$gender)
str(ANOVA_Tests.data2)

# Assmp2: Shapiro-Wilk normality test for distribution
anv_model <- lm(weight6weeks ~ Diet * gender, data = ANOVA_Tests.data2)
anv_resd <- residuals(object = anv_model)
shapiro_test(anv_resd)

# Assmp3: Test for homogeneity in variances. Function leveneTest()
Hm_varTest <- leveneTest(weight6weeks ~ Diet * gender, data =
ANOVA_Tests.data2)
Hm_varTest
```

Once the user have successfully run the codes as defined in the **Step 3A** above (Fig. 9.9, Lines 18 to 32), you will be presented with the results of the "tests for assumptions" in the Console in R as shown in Fig. 9.10.

As gathered in Fig. 9.10, in Assmp1: the authors have converted (factored) and ensured that the two Independent variables "**Diet**" and "**gender**" that we will be analyzing or using to illustrate the Two-way ANOVA analysis are stored or recognized as a **Factor** (categorical variable) in R.

Also, we conducted a *normality test* in Assmp2 by using Shapiro–Wilk's method to check the distribution of the data or targeted variables that we will be using to



```
> # Assmp1: Independent variables must be Factor (categorical variables)
> ANOVA_Tests.data2$Diet <- factor(ANOVA_Tests.data2$Diet)
> ANOVA_Tests.data2$gender <- factor(ANOVA_Tests.data2$gender)
> str(ANOVA_Tests.data2)
'data.frame':   76 obs. of  7 variables:
 $ Person      : int  1 2 3 4 5 6 7 8 9 10 ...
 $ gender      : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
 $ Age         : int  22 46 55 33 50 50 37 28 28 45 ...
 $ Height      : int  159 192 170 171 170 201 174 176 165 165 ...
 $ pre.weight  : int  58 60 64 64 65 66 67 69 70 70 ...
 $ Diet        : Factor w/ 3 levels "1","2","3": 1 1 1 1 1 1 1 1 1 1 ...
 $ weight6weeks: num  54.2 54 63.3 61.1 62.2 64 65 60.5 68.1 66.9 ...
 - attr(*, "na.action")= 'omit' Named int [1:2] 1 2
  ..- attr(*, "names")= chr [1:2] "1" "2"
> # Assmp2: Shapiro-Wilk normality test for distribution
> anv_model <- lm(weight6weeks ~ Diet * gender, data = ANOVA_Tests.data2)
> anv_resd <- residuals(object = anv_model)
> shapiro_test(anv_resd)
# A tibble: 1 x 3
  variable statistic p.value
  <chr>        <dbl>   <dbl>
1 anv_resd     0.976   0.164
> # Assmp3: Test for homogeneity in variances. Function leveneTest()
> Hm_varTest <- leveneTest(weight6weeks ~ Diet * gender, data = ANOVA_Tests.data2)
> Hm_varTest
Levene's Test for Homogeneity of Variance (center = median)
      Df F value Pr(>F)
group  5  0.7329 0.6012
      70
```

**Fig. 9.10** Results of the different tests of assumptions prior to conducting the two-way ANOVA test in R

build the model. By assuming *p-value of > 0.05* and test statistics value greater than 0.5 as the acceptable threshold. We can see that the distribution of the variables is normal with test result of 0.976, and `p-value=0.164`.

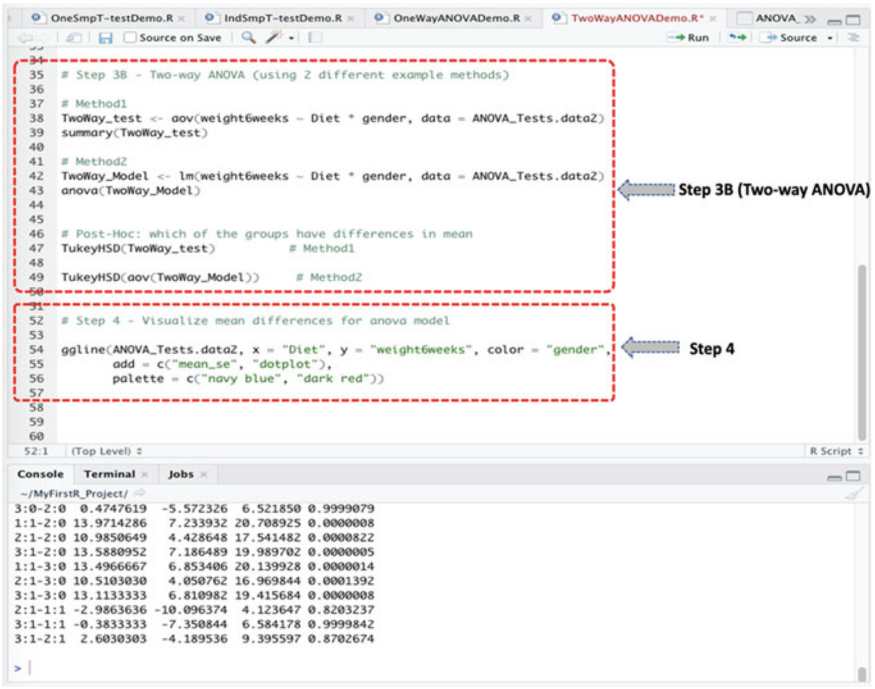
Lastly, in Assmp3: the authors tested the *homogeneity of variance* for the selected variables using the **leveneTest( )** function, whereby we assume that a value of *p > 0.05* indicates "equality in variance". Consequentially, as highlighted in the third assumption (Assmp3) in Fig. 9.10, we can see that there is no difference in variance for the analyzed variables with `p-value=0.6012`.

Therefore, we can accept that all the necessary conditions to perform the Two-way ANOVA test are met.

With all assumptions met, we can now proceed to conduct the "**Two-way ANOVA**" analysis using the **anova( )**, **aov( ),** and **TukeyHSD( )** methods as defined in Fig. 9.11 (Step 3B, Lines 35 to 49), and the results of the Two-way ANOVA test reported in Figs. 9.12a and b.

As defined earlier in the introduction section (Sect. 9.1);

- **Two-way ANOVA** is applied to compare the differences in mean between two independent variables and one dependent variable, whereby the independent variable(s) must have at least three or more levels or groups.
- The targeted "independent" variable (***x***) is often a categorical or ordinal type, while the "dependent" variable (***y***) must be numeric.



```
35  # Step 3B - Two-way ANOVA (using 2 different example methods)
36
37  # Method1
38  TwoWay_test <- aov(weight6weeks ~ Diet * gender, data = ANOVA_Tests.data2)
39  summary(TwoWay_test)
40
41  # Method2
42  TwoWay_Model <- lm(weight6weeks ~ Diet * gender, data = ANOVA_Tests.data2)
43  anova(TwoWay_Model)
44
45
46  # Post-Hoc: which of the groups have differences in mean
47  TukeyHSD(TwoWay_test)          # Method1
48
49  TukeyHSD(aov(TwoWay_Model))     # Method2
50
51
52  # Step 4 - Visualize mean differences for anova model
53
54  ggline(ANOVA_Tests.data2, x = "Diet", y = "weight6weeks", color = "gender",
55         add = c("mean_se", "dotplot"),
56         palette = c("navy blue", "dark red"))
57
```

```
3:0-2:0  0.4747619  -5.572326  6.521850 0.9999079
1:1-2:0 13.9714286   7.233932 20.708925 0.0000008
2:1-2:0 10.9850649   4.428648 17.541482 0.0000822
3:1-2:0 13.5880952   7.186489 19.989702 0.0000005
1:1-3:0 13.4966667   6.853406 20.139928 0.0000014
2:1-3:0 10.5103030   4.050762 16.969844 0.0001392
3:1-3:0 13.1133333   6.810982 19.415684 0.0000008
2:1-1:1 -2.9863636 -10.096374  4.123647 0.8203237
3:1-1:1 -0.3833333  -7.350844  6.584178 0.9999842
3:1-2:1  2.6030303  -4.189536  9.395597 0.8702674

>
```

**Fig. 9.11** Conducting Two-way ANOVA analysis in R using two different methods

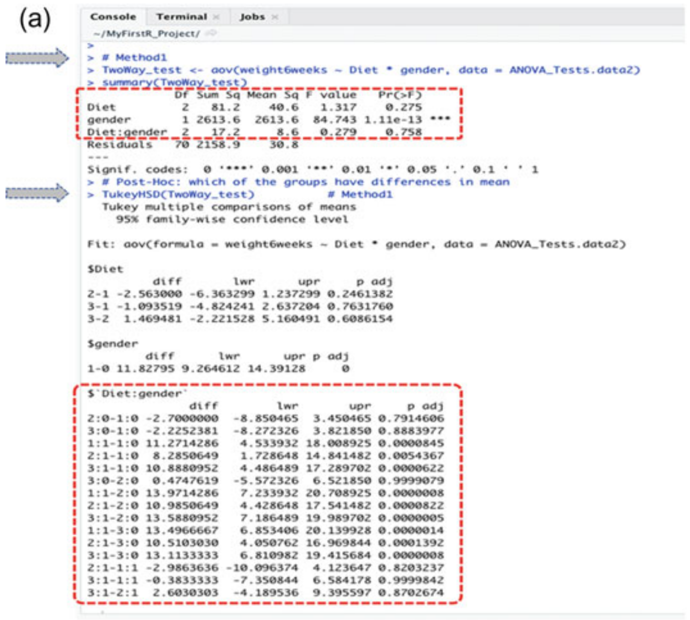
To illustrate the Two-way ANOVA using the cleaned example dataset (**Diet_R.csv**) which we have stored as "**ANOVA_Tests.data2**" in R.

- We will test whether the weight lost after 6 weeks ("**weight6weeks**") by the participants was influenced by the "**diet**" and "**gender**" variables. In other words, we will check the *effect* that the "diet" and "gender" variables (the independent variables) have on weight lost after 6 weeks ("**weight6weeks**") (dependent variable), and if so, where the differences may lie across the data.

The syntax for conducting this above test in R is as shown in the codes below (Fig. 9.11, Step 3B, Lines 35 to 49).

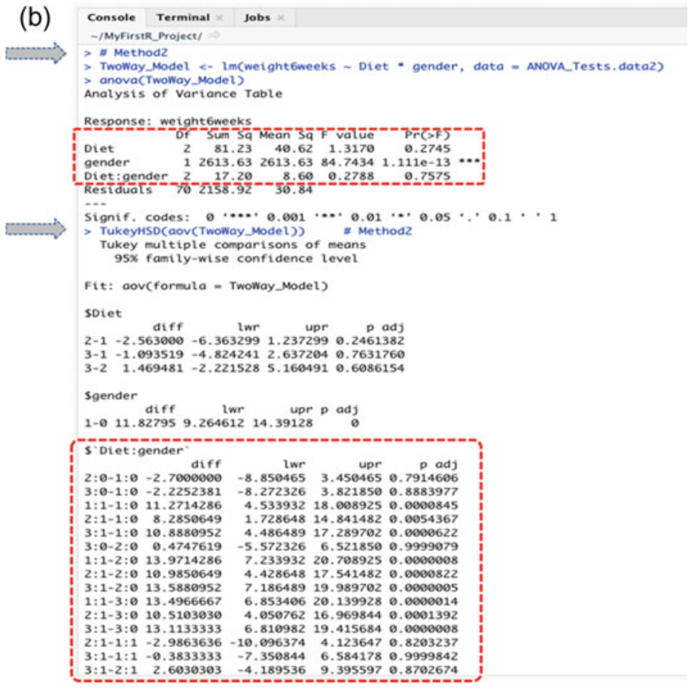
```
# Method1
TwoWay_test <- aov(weight6weeks ~ Diet * gender, data = ANOVA_Tests.data2)
summary(TwoWay_test)

# Method2
TwoWay_Model <- lm(weight6weeks ~ Diet * gender, data = ANOVA_Tests.data2)
anova(TwoWay_Model)

# Post-Hoc: which of the groups have differences in mean
TukeyHSD(TwoWay_test)          # Method1

TukeyHSD(aov(TwoWay_Model))     # Method2
```

(a)



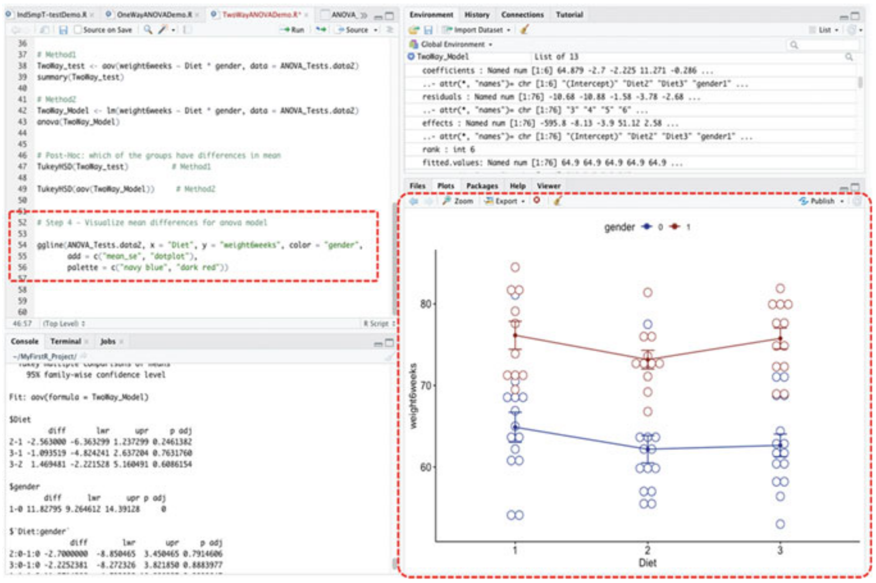
```
>
> # Method1
> TwoWay_test <- aov(weight6weeks ~ Diet * gender, data = ANOVA_Tests.data2)
> summary(TwoWay_test)
            Df Sum Sq Mean Sq F value   Pr(>F)    
Diet         2   81.2    40.6   1.317    0.275    
gender       1 2613.6  2613.6  84.743 1.11e-13 ***
Diet:gender  2   17.2     8.6   0.279    0.758    
Residuals   70 2158.9    30.8                     
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
> # Post-Hoc: which of the groups have differences in mean
> TukeyHSD(TwoWay_test)          # Method1
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = weight6weeks ~ Diet * gender, data = ANOVA_Tests.data2)

$Diet
         diff       lwr      upr     p adj
2-1 -2.563000 -6.363299 1.237299 0.2461382
3-1 -1.093519 -4.824241 2.637204 0.7631760
3-2  1.469481 -2.221528 5.160491 0.6086154

$gender
        diff      lwr      upr p adj
1-0 11.82795 9.264612 14.39128     0

$`Diet:gender`
                diff        lwr       upr     p adj
2:0-1:0 -2.7000000  -8.850465  3.450465 0.7914606
3:0-1:0 -2.2252381  -8.272326  3.821850 0.8883977
1:1-1:0 11.2714286   4.533932 18.008925 0.0000845
2:1-1:0  8.2850649   1.728648 14.841482 0.0054367
3:1-1:0 10.8880952   4.486489 17.289702 0.0000622
3:0-2:0  0.4747619  -5.572326  6.521850 0.9999079
1:1-2:0 13.9714286   7.233932 20.708925 0.0000008
2:1-2:0 10.9850649   4.428648 17.541482 0.0000822
3:1-2:0 13.5880952   7.186489 19.989702 0.0000005
1:1-3:0 13.4966667   6.853406 20.139928 0.0000014
2:1-3:0 10.5103030   4.050762 16.969844 0.0001392
3:1-3:0 13.1133333   6.810982 19.415684 0.0000008
2:1-1:1 -2.9863636 -10.096374  4.123647 0.8203237
3:1-1:1 -0.3833333  -7.350844  6.584178 0.9999842
3:1-2:1  2.6030303  -4.189536  9.395597 0.8702674
```

(b)



```
> # Method2
> TwoWay_Model <- lm(weight6weeks ~ Diet * gender, data = ANOVA_Tests.data2)
> anova(TwoWay_Model)
Analysis of Variance Table

Response: weight6weeks
            Df  Sum Sq Mean Sq F value    Pr(>F)    
Diet         2   81.23   40.62  1.3170    0.2745    
gender       1 2613.63 2613.63 84.7434 1.111e-13 ***
Diet:gender  2   17.20    8.60  0.2788    0.7575    
Residuals   70 2158.92   30.84                      
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
> TukeyHSD(aov(TwoWay_Model))      # Method2
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = TwoWay_Model)

$Diet
         diff       lwr      upr     p adj
2-1 -2.563000 -6.363299 1.237299 0.2461382
3-1 -1.093519 -4.824241 2.637204 0.7631760
3-2  1.469481 -2.221528 5.160491 0.6086154

$gender
        diff      lwr      upr p adj
1-0 11.82795 9.264612 14.39128     0

$`Diet:gender`
                diff        lwr       upr     p adj
2:0-1:0 -2.7000000  -8.850465  3.450465 0.7914606
3:0-1:0 -2.2252381  -8.272326  3.821850 0.8883977
1:1-1:0 11.2714286   4.533932 18.008925 0.0000845
2:1-1:0  8.2850649   1.728648 14.841482 0.0054367
3:1-1:0 10.8880952   4.486489 17.289702 0.0000622
3:0-2:0  0.4747619  -5.572326  6.521850 0.9999079
1:1-2:0 13.9714286   7.233932 20.708925 0.0000008
2:1-2:0 10.9850649   4.428648 17.541482 0.0000822
3:1-2:0 13.5880952   7.186489 19.989702 0.0000005
1:1-3:0 13.4966667   6.853406 20.139928 0.0000014
2:1-3:0 10.5103030   4.050762 16.969844 0.0001392
3:1-3:0 13.1133333   6.810982 19.415684 0.0000008
2:1-1:1 -2.9863636 -10.096374  4.123647 0.8203237
3:1-1:1 -0.3833333  -7.350844  6.584178 0.9999842
3:1-2:1  2.6030303  -4.189536  9.395597 0.8702674
```

**Fig. 9.12** **a** Result of two-way ANOVA (Method1) test in R with Post-Hoc test. **b** Result of two-way ANOVA (Method2) test in R with Post-Hoc test



```
36
37  # Method1
38  TwoWay_test <- aov(weight6weeks ~ Diet * gender, data = ANOVA_Tests.data2)
39  summary(TwoWay_test)
40
41  # Method2
42  TwoWay_Model <- lm(weight6weeks ~ Diet * gender, data = ANOVA_Tests.data2)
43  anova(TwoWay_Model)
44
45
46  # Post-Hoc: which of the groups have differences in mean
47  TukeyHSD(TwoWay_test)          # Method1
48
49  TukeyHSD(aov(TwoWay_Model))     # Method2
50
51
52  # Step 4 - Visualize mean differences for anova model
53
54  ggline(ANOVA_Tests.data2, x = "Diet", y = "weight6weeks", color = "gender",
55         add = c("mean_se", "dotplot"),
56         palette = c("navy blue", "dark red"))
57
58
59
60
```

```
  95% family-wise confidence level

Fit: aov(formula = TwoWay_Model)

$Diet
         diff       lwr      upr     p adj
2-1 -2.563000 -6.363299 1.237299 0.2461382
3-1 -1.093519 -4.824241 2.637204 0.7631760
3-2  1.469481 -2.221528 5.160491 0.6086154

$gender
        diff      lwr      upr p adj
1-0 11.82795 9.264612 14.39128     0

$`Diet:gender`
              diff       lwr      upr     p adj
2:0-1:0 -2.7000000 -8.850465 3.450465 0.7914606
3:0-1:0 -2.2252381 -8.272326 3.821850 0.8883977
```

```
TwoWay_Model            List of 13
   coefficients : Named num [1:6] 64.879 -2.7 -2.225 11.271 -0.286 ...
   ..- attr(*, "names")= chr [1:6] "(Intercept)" "Diet2" "Diet3" "gender1" ...
   residuals : Named num [1:76] -10.68 -10.88 -1.58 -3.78 -2.68 ...
   ..- attr(*, "names")= chr [1:76] "3" "4" "5" "6" ...
   effects : Named num [1:76] -595.8 -8.13 -3.9 51.12 2.58 ...
   ..- attr(*, "names")= chr [1:76] "(Intercept)" "Diet2" "Diet3" "gender1" ...
   rank : int 6
   fitted.values: Named num [1:76] 64.9 64.9 64.9 64.9 64.9 ...
```

**Fig. 9.13** Plot of the mean differences between the different groups of variables in the ANOVA model using the ggline( ) function in R

As shown in Figs. 9.11, 9.12a, and b, we conducted the **Two-way ANOVA** analysis by considering the following variables (**weight6weeks ~ Diet * gender**). We illustrated this using two different ways or methods in R. As we can see, both methods (defined as **Method1** and **Method2**) tend to produce the same results as shown in Figs. 9.12a and b, respectively. The results are explained in detail in the subsequent **Step 5** in this section.

# **Step 4**—Plot and visualize the mean differences for the ANOVA model

As shown in Fig. 9.13 (Step 4, Lines 52 to 56), we used the **ggline( )** function in R to visualize the mean differences that exist between the different groups of variables in the ANOVA model.

The code for the ANOVA model is shown below, and the result presented in the graph in Fig. 9.13.

```
ggline(ANOVA_Tests.data2, x = "Diet", y = "weight6weeks", color = "gender",
      add = c("mean_se", "dotplot"),
      palette = c("navy blue", "dark red"))
```

## # Step 5—Two-way ANOVA Results Interpretation

The final step for the Two-way ANOVA analysis is to interpret and understand the results of the test.

By default, the hypothesis for conducting the test (Two-way ANOVA) is; *IF* the p-value of the test is less than or equal to 0.05 ($p \leq 0.05$), *THEN* we can assume that the mean of the groups (minimum of three levels or groups) of variables or population (of which are two independent variables) in the data are statistically different (varies) and that this is not by chance ($H_1$), *ELSE IF* the p-value is greater than 0.05 ($p > 0.05$) *THEN* we can conclude that there is no difference in the mean of the analyzed group of variables and any difference observed could only occur by chance ($H_0$).

```
 > TwoWay_Model <- lm(weight6weeks ~ Diet * gender, data =
ANOVA_Tests.data2)
> anova(TwoWay_Model)
Analysis of Variance Table

Response: weight6weeks
            Df  Sum Sq Mean Sq F value    Pr(>F)
Diet         2   81.23   40.62  1.3170    0.2745
gender       1 2613.63 2613.63 84.7434 1.111e-13 ***
Diet:gender  2   17.20    8.60  0.2788    0.7575
Residuals   70 2158.92   30.84
```

As shown in the outcome of the Two-way ANOVA tests, with the same similar results observed for the **method 1** and **method 2** (see: Fig. 9.12a and b); statistically, we can see that the weight lost by the participants after 6 weeks "**weight6weeks**" was not influenced by the Diet (`p=0.2745`). Also, the "**weight6weeks**" was not influenced by the combination of the "**Diet**" and "**gender**" factors (`Diet:gender`) with p-value greater than the significance levels of $p \leq 0.05$ (i.e., `p-value=0.7575`). However, we can see also that even though the combination of the variables (`Diet:gender`) does not have any significant effect on weight lost after 6 weeks, there were differences in mean (variation) for the genders (1 = male, 0 = female) variables when taking into account the weight lost after 6 weeks "**weight6weeks**" with `p-value = 1.111e-13` ($p \leq 0.05$).

Therefore, it will be necessary and important to further conduct a post-hoc test, as shown below, to determine where the significant differences lies (see: Figs. 9.12a and b).

```
$`Diet:gender`
                 diff        lwr       upr     p adj
2:0-1:0 -2.7000000  -8.850465  3.450465 0.7914606
3:0-1:0 -2.2252381  -8.272326  3.821850 0.8883977
1:1-1:0 11.2714286   4.533932 18.008925 0.0000845
2:1-1:0  8.2850649   1.728648 14.841482 0.0054367
3:1-1:0 10.8880952   4.486489 17.289702 0.0000622
3:0-2:0  0.4747619  -5.572326  6.521850 0.9999079
1:1-2:0 13.9714286   7.233932 20.708925 0.0000008
2:1-2:0 10.9850649   4.428648 17.541482 0.0000822
3:1-2:0 13.5880952   7.186489 19.989702 0.0000005
1:1-3:0 13.4966667   6.853406 20.139928 0.0000014
2:1-3:0 10.5103030   4.050762 16.969844 0.0001392
3:1-3:0 13.1133333   6.810982 19.415684 0.0000008
2:1-1:1 -2.9863636 -10.096374  4.123647 0.8203237
3:1-1:1 -0.3833333  -7.350844  6.584178 0.9999842
3:1-2:1  2.6030303  -4.189536  9.395597 0.8702674
```

As reported in the pairwise multiple comparisons test by using the **TukeyHSD( )** method or function in R, we can see that most of the significant differences ($p \leq 0.05$) observed for the between-subjects effects were found mainly for the female gender group (0).

Consequently, we can statistically conclude that the mean of weight lost after the 6 weeks ("**weight6weeks**") by the participants varies by **gender** with `p-value=1.111e-13` ($p \leq 0.05$) but not influenced by Diet (`p=0.2745`).

*****Useful Tips:**

- The researchers or analysts can also analyze *more than two independent variables*. This is known as **N-Way ANOVA,** whereby *N* represents the number of independent variables the researcher or data analysts are testing against the one dependent (response) variable. For instance, in our example data (Fig. 9.4), the users can simultaneously analyze the influence or effects that the Diet, Gender, Age group, etc. have on the "weight6weeks" variable.

## 9.4 Summary

In this chapter, the authors practically demonstrate in detail how to perform the most commonly used type of ANOVA tests (One-way and Two-way) in R.

In Sect. 9.2, it illustrates how to perform the One-way ANOVA test, while in Sect. 9.3 it looked at how to conduct the Two-way ANOVA analysis or test.

The authors also covered how to graphically plot the mean differences or results of the ANOVA tests in R in this chapter, and then subsequently discussed how to interpret and understand the results of the tests in R.

In summary, the main topics and contents covered in this chapter includes:

- ANOVA (analysis of variance) is a statistical *test of variance* as the name implies or hypothesis used to compare the *differences in means* of data samples that are represented in more than two independent comparison groups or multilevel for the independent variable(s) (usually categorical or ordinal) and a continuous dependent variable.

When choosing whether to conduct a One-way or Two-way ANOVA test? The researcher or data analyst should:

- Perform the "*One-way ANOVA*" if the groups come from *one independent* variable (with a minimum of three groups) usually measured as categorical or ordinal values, and *one dependent* variable (continuous).
- Perform the "*Two-way ANOVA*" if the targeted groups come from *two independent* variables (with a minimum of three groups) usually measured as categorical or ordinal values, and *one dependent* variable (continuous).
- In either case (One-way or Two-way), the targeted "independent" variable ($\boldsymbol{x}$) is often a categorical or ordinal type, while the "dependent" variable ($\boldsymbol{y}$) must be numeric.

Other types of the ANOVA statistics or "multivariate analysis" as they are called are also used in the existing literature or statistical analysis, such as the multivariate analysis of variance (MANOVA) (Dugard et al., 2022; Okoye et al., 2022), analysis of co-variance (ANCOVA) (Kaltenecker & Okoye, 2023; Li & Chen, 2019), multivariate analysis of co-variance (MANCOVA) (Li & Chen, 2019; Okoye et al., 2023), etc.

## References

Christensen, R. (2020). One-way ANOVA. In: Plane answers to complex questions. Springer Texts in Statistics book series (STS), pp 107–121. Springer, Cham. https://doi.org/10.1007/978-3-030-32097-3_4.

Connelly, L. M. (2021). Introduction to analysis of variance (ANOVA). *Medsurg Nursing*, *30*(3), 218. https://www.proquest.com/docview/2542477790

Dugard, P., Todman, J., & Staines, H. (2022). Multivariate analysis of variance (MANOVA). In Approaching multivariate analysis, 2nd Edn. Routledge. https://www.taylorfrancis.com/chapters/edit/https://doi.org/10.4324/9781003343097-3/multivariate-analysis-variance-manova-pat-dugard-john-todman-harry-staines.

Guillén-Gámez, F. D., Mayorga-Fernández, M. J., & Ramos, M. (2021). Examining the use self-perceived by university teachers about ict resources: Measurement and comparative analysis in a one-way ANOVA design. *Contemporary Educational Technology*, *13*(1), 1–13. https://doi.org/10.30935/cedtech/8707.

Kaltenecker, E., & Okoye, K. (2023). How do location, accreditation, and faculty size affect business schools' ranking? *Journal of Education for Business*, 1–7. https://doi.org/10.1080/08832323.2023.2268800.

Li, Z., & Chen, M. Y. (2019). Application of ANCOVA and MANCOVA in language assessment research. In V. Aryadoust, & M. Raquel (Eds.), *Quantitative data analysis for language assessment volume I* (Vol. 1, p. 21). Routledge. https://doi.org/10.4324/9781315187815.

Nibrad, G. M. (2019). Methodology and application of two-way ANOVA. *International Journal of Marketing and Technology, 9*(6), 1–8.

Okoye, K., Nganji, J. T., Escamilla, J., Fung, J. M., & Hosseini, S. (2022). Impact of global government investment on education and research development: A comparative analysis and demystifying the science, technology, innovation, and education conundrum. *Global Transitions, 4*, 11–27. https://doi.org/10.1016/J.GLT.2022.10.001.

Okoye, K., Daruich, S. D. N., De La O, J. F. E., Castano, R., Escamilla, J., & Hosseini, S. (2023). A text mining and statistical approach for assessment of pedagogical impact of students' evaluation of teaching and learning outcome in education. *IEEE Access*, *11*, 9577–9596. https://doi.org/10.1109/ACCESS.2023.3239779.

Roscoe, J. T. (1975). *Fundamental research statistics for the behavioral sciences* (2nd ed.). Holt, Rinehart, and Winston.

Sullivan, L. (2020). *Hypothesis testing-analysis of variance (ANOVA)*. https://sphweb.bumc.bu.edu/otlt/mph-modules/bs/bs704_hypothesistesting-anova/bs704_hypothesistesting-anova_print.html.

# Chapter 10
# Chi-Squared ($X^2$) Statistical Test in R

## 10.1 Introduction

*Chi-Squared* ($\chi 2$) is a statistical test applied by researchers or data analysts to measure how *expectations* compares to actual *observed* data or results of a model (Biswal, 2023; Kishore & Jaswal, 2023). The test (Chi-squared) is mainly used to explain or determine whether there exists a relationship between "categorical" variables which must be raw data that are mutually exclusive and randomly drawn from independent populations and from a large enough sample size (Sayassatov & Cho, 2020; Turhan, 2020).

By default, the hypothesis for performing the Chi-squared ($\chi 2$) test is; *IF* the p-value of the test statistics is less than or equal to 0.05 ($p \leq 0.05$), *THEN* we can assume that there exists a relationship between the targeted (categorical) variables and that this is not by chance ($H_1$), *ELSE IF* the p-value is greater than 0.05 ($p > 0.05$) THEN we can say that there is no relationship (measure of independency) between the analyzed (categorical) variables or data.

The formula for calculating the Chi-square ($\chi 2$) statistics is as follows (Biswal, 2023; Kishore & Jaswal, 2023):

$$X_c^2 = \sum \frac{(O_i - E_i)^2}{E_i}$$

where:

$c$ = Degrees of freedom

$O$ = Observed value(s)

$E$ = Expected values(s)

Theoretically, there are two main types of analysis or test that can be performed using the Chi-squared ($X^2$) statistics or method (Preacher, 2001; Turhan, 2020). These are:

https://doi.org/10.1007/978-981-97-3385-9_10

- **Independence test:** which is a test of "relationship" that allows the researcher or data analysts to compare two (categorical) variables to determine whether they are related or not. In this scenario, the researcher can apply the Chi-squared test to tell how likely or if by random (chance) the resultant relationship, usually determined through the p-value ($p \leq 0.05$), can explain any difference found between the observed or actual (frequency) data and the expected data (McHugh, 2012; Taneichi et al., 2020).
- **Goodness of fit test:** is applied to determine whether a proportion of a data sample matches the larger population. In this scenario, the test (Chi-squared) allows the researcher to check how well the analyzed (drawn) sample matches the assumed (expected) characteristics or features of the larger population that the data is projected to represent (Cochran, 1952; Rolke & Gongora, 2020). Thus, *if* the analyzed data does not match or fit the assumed (expected) characteristics of the intended population, usually determined through the p-value ($p \leq 0.05$), *then* the researcher may not consequentially want to utilize the drawn data sample to make any ample or definite conclusion about the studied/larger population in question.

As a general rule of thumb, the following assumptions must be met in order to perform the Chi-squared ($X^2$) statistics or analysis (Turhan, 2020):

- The "observed" and "expected" observations must be randomly collected or drawn.
- All the groups of items in the data must be independent.
- None of the groups must contain very few items (e.g., not less than 10).
- The data sample size must be large (at least $n > 50$).

It is also important to mention, just like the many other versions of the *non-parametric* statistical methods (see Chap. 4), that the Chi-squared ($X^2$) test does not require the studied or analyzed dataset or sample to meet the "equality of variance" assumption among the groups of variables or yet "homoscedasticity" in the data (McHugh, 2012).

In the next section of this chapter (Sect. 10.2), the authors will demonstrate how to conduct the Chi-squared ($X^2$) test in R. Figure 1 is an outline of the different steps that we will apply in order to perform the test (Chi-squared) in RStudio.

## 10.2 Chi-Squared ($X^2$) Test in R

As defined in the previous section (Sect. 10.1)—using the Chi-Squared ($X^2$) test in R is a method that can be used to determine whether two *categorical* variables have a statistically significant *correlation* (association) between them. With the Chi-squared statistics, the two targeted variables must be *categorized* (e.g., sex, marital status, ethnicity, religious orientation, likelihood of events, etc.), and must be selected from the same population.

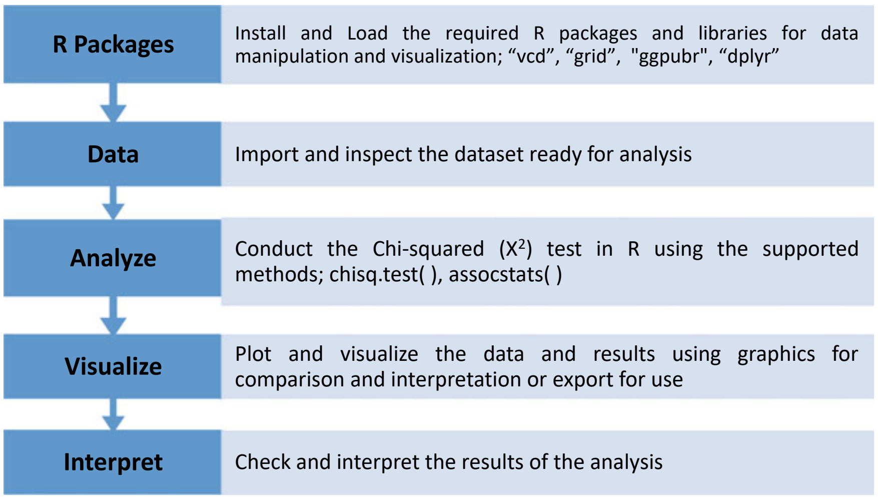

R Packages
Install and Load the required R packages and libraries for data manipulation and visualization; "vcd", "grid", "ggpubr", "dplyr"
Data
Import and inspect the dataset ready for analysis
Analyze
Conduct the Chi-squared (X²) test in R using the supported methods; chisq.test( ), assocstats( )
Visualize
Plot and visualize the data and results using graphics for comparison and interpretation or export for use
Interpret
Check and interpret the results of the analysis

**Fig. 10.1** Steps to conducting Chi-squared ($X^2$) test in R

In this section, the authors will be practically demonstrating to the readers how to conduct the two types of tests (i.e., Independence, and Goodness of fit) in R. This will be done by using the Chi-squared ($X^2$) function called **chisq.test( )** in R. We will do this by following the steps outlined in Fig. 10.1.

To begin, **Open RStudio** and **Create a new or Open an existing project**. Once the user have the RStudio and an R Project opened, **Create a new RScript** and name it "**ChiSquare_Demo**" or any name the user chooses (see Chap. 1 on how to do these steps if the user need to refresh on the topic).

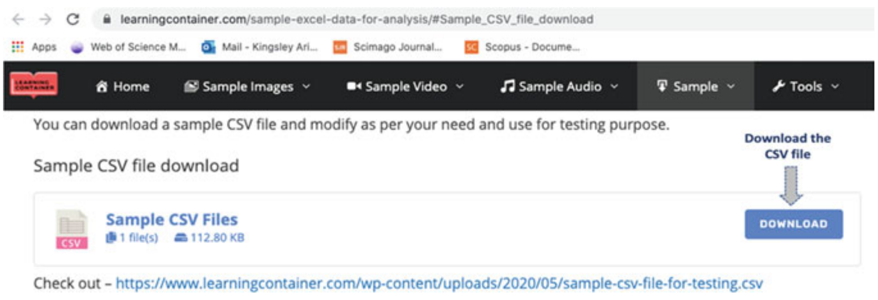
Once the user have created an R Script, now let's download an example dataset that we will use to demonstrate the Chi-squared ($X^2$) test (the users are welcome to use any dataset or format of their choice if they wish to do so).

As shown in Fig. 10.2, download the example data (**Sample CSV Files**) named as "sample-csv-file-for-testing" from the following link (https://www.learningcontainer.com/sample-excel-data-for-analysis/#Sample_CSV_file_download) and save the file on the local machine or computer.

Once the user have successfully downloaded and saved the example file (which is named as "sample-csv-file-for-testing" upon download) on the system or computer, we can proceed to conduct the Chi-squared ($X^2$) analysis in R.

# **Step 1**—Install and Load the required R Packages and Libraries

**Install** and **Load** the following *R packages* and *libraries* (Fig. 10.3, Step1, Lines 3 to 13) that we will be using to call the different R functions, data manipulations, and graphical visualizations for the Chi-squared ($X^2$) test.

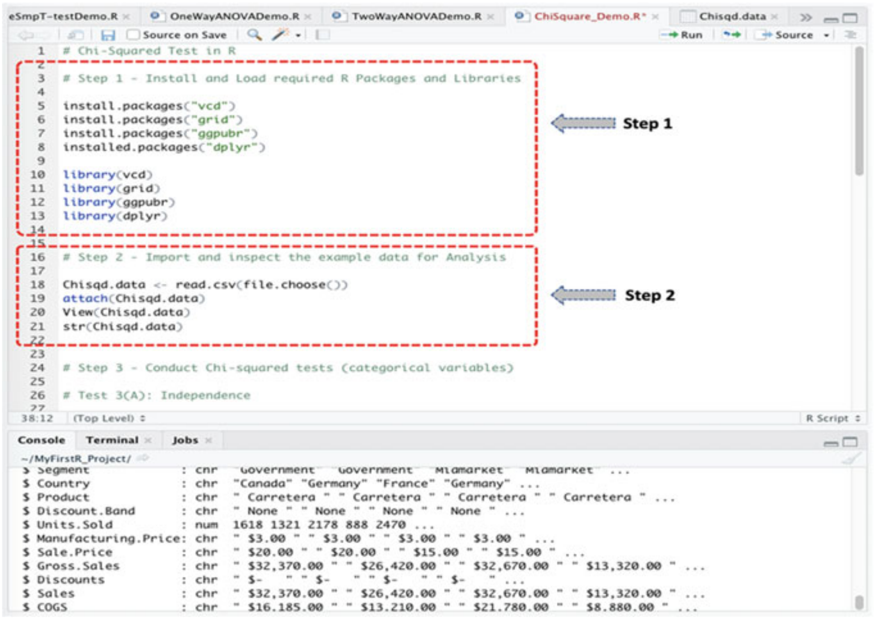


You can download a sample CSV file and modify as per your need and use for testing purpose.

Sample CSV file download

Check out – https://www.learningcontainer.com/wp-content/uploads/2020/05/sample-csv-file-for-testing.csv

**Fig. 10.2** Example of CSV file download. *Source* https://www.learningcontainer.com/sample-excel-data-for-analysis/#Sample_CSV_file_download. ***Note* the users can also directly access the example file through the following link (https://doi.org/https://doi.org/10.6084/m9.figshare.24728073) where the authors have uploaded all the example files used in this book



```
# Chi-Squared Test in R

# Step 1 - Install and Load required R Packages and Libraries

install.packages("vcd")
install.packages("grid")
install.packages("ggpubr")
installed.packages("dplyr")

library(vcd)
library(grid)
library(ggpubr)
library(dplyr)


# Step 2 - Import and inspect the example data for Analysis

Chisqd.data <- read.csv(file.choose())
attach(Chisqd.data)
View(Chisqd.data)
str(Chisqd.data)


# Step 3 - Conduct Chi-squared tests (categorical variables)

# Test 3(A): Independence
```

```
$ Segment            : chr  "Government" "Government" "Midmarket" "Midmarket" ...
$ Country            : chr  "Canada" "Germany" "France" "Germany" ...
$ Product            : chr  " Carretera " " Carretera " " Carretera " " Carretera " ...
$ Discount.Band      : chr  " None " " None " " None " " None " ...
$ Units.Sold         : num  1618 1321 2178 888 2470 ...
$ Manufacturing.Price: chr  " $3.00 " " $3.00 " " $3.00 " " $3.00 " ...
$ Sale.Price         : chr  " $20.00 " " $20.00 " " $15.00 " " $15.00 " ...
$ Gross.Sales        : chr  " $32,370.00 " " $26,420.00 " " $32,670.00 " " $13,320.00 " ...
$ Discounts          : chr  " $-    " " $-    " " $-    " " $-    " ...
$ Sales              : chr  " $32,370.00 " " $26,420.00 " " $32,670.00 " " $13,320.00 " ...
$ COGS               : chr  " $16,185.00 " " $13,210.00 " " $21,780.00 " " $8,880.00 " ...
```

**Fig. 10.3** Conducting Chi-squared ($X^2$) statistics in R

The syntax and code to install and load the required R packages and libraries are as follows:

```
install.packages("vcd")
install.packages("grid")
install.packages("ggpubr")
installed.packages("dplyr")

library(vcd)
library(grid)
library(ggpubr)
library(dplyr)
```

**# Step 2—Import and Inspect the example dataset for Analysis.**

As defined in Fig. 10.3 (Step 2, Lines 16 to 21), import the example dataset named "sample-csv-file-for-testing" (**Sample CSV Files**) that we downloaded earlier, and store this in an R object named "**Chisqd.data**" (the users can use any name of choice if they wish to do so).

The syntax for importing and attaching the example data file into R is as shown in the code below:

```
Chisqd.data <- read.csv(file.choose())
attach(Chisqd.data)
View(Chisqd.data)
str(Chisqd.data)
```

Once the data is successfully imported and stored in RStudio as an R object we called "Chisqd.data", the users will be able to view the details of the file originally named "sample-csv-file-for-testing" (Sample CSV Files) as shown in Fig. 10.4 with 700 observations and 16 variables in the data sample.

**# Step 3—Conduct Chi-squared ($X^2$) test (used for categorical variables only)**

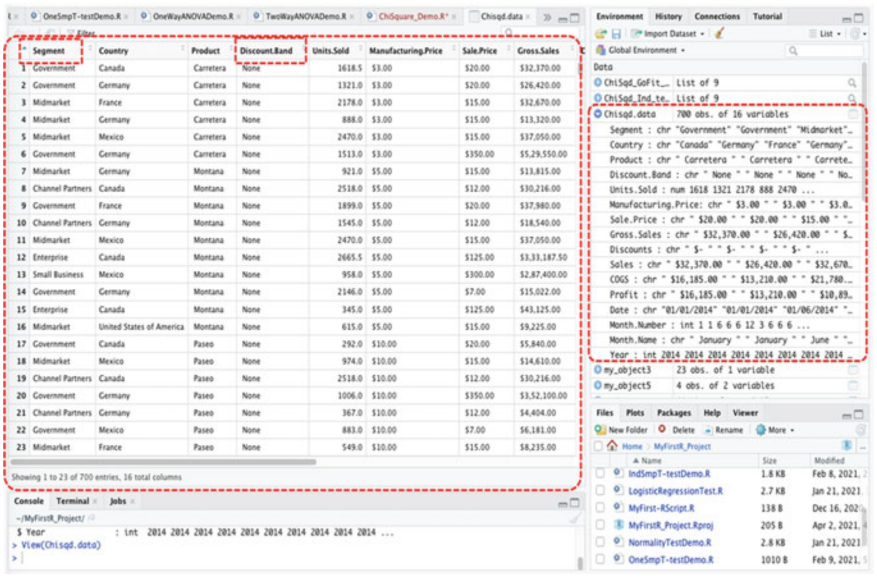
Now we can proceed to analyze the imported dataset that we stored as **Chisqd.data** in the R environment (see: Fig. 10.4).

As defined in the introduction section (Sect. 10.1);

- **Chi-squared ($X^2$)** statistics compares the *relationship* (correlation) between two sets of variables with two or more levels or independent groups.
- The target variable(s) must be a "categorical" data type.

To demonstrate how to perform the two main types of the Chi-squared (X2) test or analysis (i.e., Independence test, and Goodness of fit test), we will be using the R function called **chisq.test( )** to conduct the following test in R:

1. For the **Independence test**—we will test whether the two categorical variables "**Segment**" and "**Discount.Band**" in the imported dataset (see: Fig. 10.4) are related (correlated).
2. For the **Goodness of Fit test**—we will check how the *observed* groups in the "**Discount.Band**" variable matches or is capable of fitting the *expected* population.



|  | Segment | Country | Product | Discount.Band | Units.Sold | Manufacturing.Price | Sale.Price | Gross.Sales |
|---|---|---|---|---|---|---|---|---|
| 1 | Government | Canada | Carretera | None | 1618.5 | $3.00 | $20.00 | $32,370.00 |
| 2 | Government | Germany | Carretera | None | 1321.0 | $3.00 | $20.00 | $26,420.00 |
| 3 | Midmarket | France | Carretera | None | 2178.0 | $3.00 | $15.00 | $32,670.00 |
| 4 | Midmarket | Germany | Carretera | None | 888.0 | $3.00 | $15.00 | $13,320.00 |
| 5 | Midmarket | Mexico | Carretera | None | 2470.0 | $3.00 | $15.00 | $37,050.00 |
| 6 | Government | Germany | Carretera | None | 1513.0 | $3.00 | $350.00 | $5,29,550.00 |
| 7 | Midmarket | Germany | Montana | None | 921.0 | $5.00 | $15.00 | $13,815.00 |
| 8 | Channel Partners | Canada | Montana | None | 2518.0 | $5.00 | $12.00 | $30,216.00 |
| 9 | Government | France | Montana | None | 1899.0 | $5.00 | $20.00 | $37,980.00 |
| 10 | Channel Partners | Germany | Montana | None | 1545.0 | $5.00 | $12.00 | $18,540.00 |
| 11 | Midmarket | Mexico | Montana | None | 2470.0 | $5.00 | $15.00 | $37,050.00 |
| 12 | Enterprise | Canada | Montana | None | 2665.5 | $5.00 | $125.00 | $3,33,187.50 |
| 13 | Small Business | Mexico | Montana | None | 958.0 | $5.00 | $300.00 | $2,87,400.00 |
| 14 | Government | Germany | Montana | None | 2146.0 | $5.00 | $7.00 | $15,022.00 |
| 15 | Enterprise | Canada | Montana | None | 345.0 | $5.00 | $125.00 | $43,125.00 |
| 16 | Midmarket | United States of America | Montana | None | 615.0 | $5.00 | $15.00 | $9,225.00 |
| 17 | Government | Canada | Paseo | None | 292.0 | $10.00 | $20.00 | $5,840.00 |
| 18 | Midmarket | Mexico | Paseo | None | 974.0 | $10.00 | $15.00 | $14,610.00 |
| 19 | Channel Partners | Canada | Paseo | None | 2518.0 | $10.00 | $12.00 | $30,216.00 |
| 20 | Government | Germany | Paseo | None | 1006.0 | $10.00 | $350.00 | $3,52,100.00 |
| 21 | Channel Partners | Germany | Paseo | None | 367.0 | $10.00 | $12.00 | $4,404.00 |
| 22 | Government | Mexico | Paseo | None | 883.0 | $10.00 | $7.00 | $6,181.00 |
| 23 | Midmarket | France | Paseo | None | 549.0 | $10.00 | $15.00 | $8,235.00 |

Showing 1 to 23 of 700 entries, 16 total columns

```
~/MyFirstR_Project/
$ Year               : int  2014 2014 2014 2014 2014 2014 2014 2014 2014 2014 ...
> View(Chisqd.data)
>
```

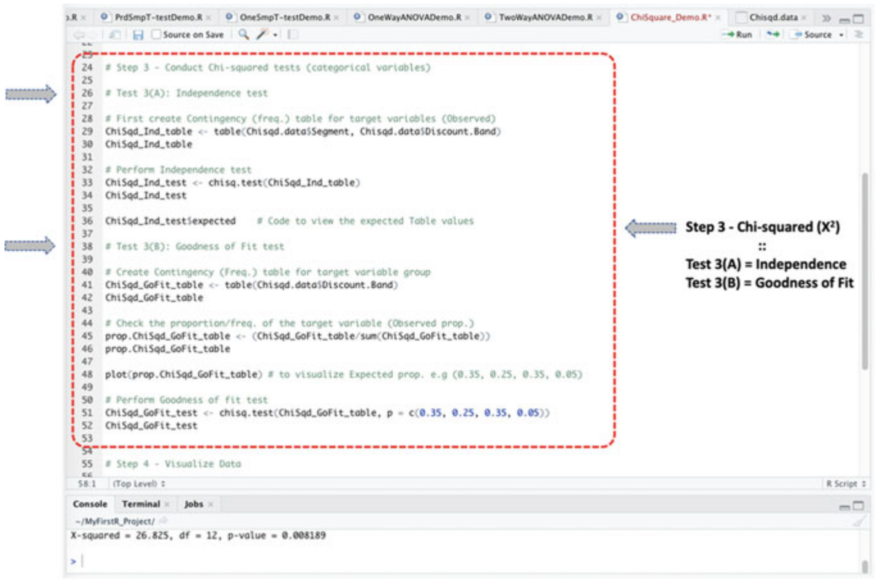
```
Data
ChiSqd_GoFit_…   List of 9
ChiSqd_Ind_te…   List of 9
Chisqd.data      700 obs. of 16 variables
  Segment : chr "Government" "Government" "Midmarket"…
  Country : chr "Canada" "Germany" "France" "Germany"…
  Product : chr " Carretera " " Carretera " " Carrete…
  Discount.Band : chr " None " " None " " None " " No…
  Units.Sold : num 1618 1321 2178 888 2470 ...
  Manufacturing.Price: chr " $3.00 " " $3.00 " " $3.0…
  Sale.Price : chr " $20.00 " " $20.00 " " $15.00 " "…
  Gross.Sales : chr " $32,370.00 " " $26,420.00 " " $…
  Discounts : chr " $- " " $- " " $- " " $- " ...
  Sales : chr " $32,370.00 " " $26,420.00 " " $32,670…
  COGS : chr " $16,185.00 " " $13,210.00 " " $21,780.…
  Profit : chr " $16,185.00 " " $13,210.00 " " $10,89…
  Date : chr "01/01/2014" "01/01/2014" "01/06/2014" "…
  Month.Number : int 1 1 6 6 6 12 3 6 6 6 ...
  Month.Name : chr " January " " January " " June " "…
  Year : int 2014 2014 2014 2014 2014 2014 2014 2014 …
my_object3       23 obs. of 1 variable
my_object5       4 obs. of 2 variables
```

| Name | Size | Modified |
|---|---|---|
| IndSmpT-testDemo.R | 1.8 KB | Feb 8, 2021, |
| LogisticRegressionTest.R | 2.7 KB | Jan 21, 2021 |
| MyFirst-RScript.R | 138 B | Dec 16, 202 |
| MyFirstR_Project.Rproj | 205 B | Apr 2, 2021, |
| NormalityTestDemo.R | 2.8 KB | Jan 21, 2021 |
| OneSmpT-testDemo.R | 1010 B | Feb 9, 2021, |

Fig. 10.4 Example dataset imported and stored as an R object in RStudio

The syntax and code for performing the above Chi-squared ($X^2$) tests in R is as shown in the codes below and represented in Fig. 10.5 (Step 3, Lines 24 to 52).

```
# Test 3(A): Independence test

# First create Contingency (freq.) table for target variables
(Observed)
ChiSqd_Ind_table <- table(Chisqd.data$Segment,
Chisqd.data$Discount.Band)
ChiSqd_Ind_table

# Perform Independence test
ChiSqd_Ind_test <- chisq.test(ChiSqd_Ind_table)
ChiSqd_Ind_test

ChiSqd_Ind_test$expected     # Code to view the expected Table values
# Test 3(B): Goodness of Fit test

# Create Contingency (Freq.) table for target variable group
```



```
24 # Step 3 - Conduct Chi-squared tests (categorical variables)
25
26 # Test 3(A): Independence test
27
28 # First create Contingency (freq.) table for target variables (Observed)
29 ChiSqd_Ind_table <- table(Chisqd.data$Segment, Chisqd.data$Discount.Band)
30 ChiSqd_Ind_table
31
32 # Perform Independence test
33 ChiSqd_Ind_test <- chisq.test(ChiSqd_Ind_table)
34 ChiSqd_Ind_test
35
36 ChiSqd_Ind_test$expected    # Code to view the expected Table values
37
38 # Test 3(B): Goodness of Fit test
39
40 # Create Contingency (Freq.) table for target variable group
41 ChiSqd_GoFit_table <- table(Chisqd.data$Discount.Band)
42 ChiSqd_GoFit_table
43
44 # Check the proportion/freq. of the target variable (Observed prop.)
45 prop.ChiSqd_GoFit_table <- (ChiSqd_GoFit_table/sum(ChiSqd_GoFit_table))
46 prop.ChiSqd_GoFit_table
47
48 plot(prop.ChiSqd_GoFit_table) # to visualize Expected prop. e.g (0.35, 0.25, 0.35, 0.05)
49
50 # Perform Goodness of fit test
51 ChiSqd_GoFit_test <- chisq.test(ChiSqd_GoFit_table, p = c(0.35, 0.25, 0.35, 0.05))
52 ChiSqd_GoFit_test
53
54
55 # Step 4 - Visualize Data
```

```
~/MyFirstR_Project/
X-squared = 26.825, df = 12, p-value = 0.008189

>
```

**Fig. 10.5** Chi-squared (X2) Test in R using the chisq.test( ) function

```
ChiSqd_GoFit_table <- table(Chisqd.data$Discount.Band)
ChiSqd_GoFit_table

# Check the proportion/freq. of the target variable (Observed prop.)
prop.ChiSqd_GoFit_table <-
(ChiSqd_GoFit_table/sum(ChiSqd_GoFit_table))
prop.ChiSqd_GoFit_table

plot(prop.ChiSqd_GoFit_table) # to visualize Expected prop. e.g
(0.35, 0.25, 0.35, 0.05)

# Perform Goodness of fit test
ChiSqd_GoFit_test <- chisq.test(ChiSqd_GoFit_table, p = c(0.35, 0.25,
0.35, 0.05))
ChiSqd_GoFit_test
```

**Note:** by adding or running the **$expected** command along with the results of the **chisq.test( )** function/method (see: Line 36, Fig. 10.5) returns a Contingency table that contains the expected counts which will or are considered to be "TRUE" under the null hypothesis ($H_0$).

Once the user have successfully run the codes or command (Step 3, Fig. 10.5), the user will be presented with the results of the method in the Console as shown in Figs. 10.6a and b, respectively.

As shown in results of the Chi-squared tests in Figs. 10.6a and b; we conducted the "**Independence**" (Test 3(A)) and "**Goodness of Fit**" (Test 3(B)) tests in R using the Chi-squared ($X^2$) method or function: **chisq.test( )**. This was illustrated by using the two *categorical* variables "**Segment**" and "**Discount.Band**" contained in the example dataset "sample-csv-file-for-testing" (**Sample CSV Files**) that we stored as R object named or defined as "**Chisqd.data**" in R. The results of the methods were stored as R objects which we called "`ChiSqd_Ind_test`" and "`ChiSqd_GoFit_test`", respectively (Figs. 10.6a and b).

**# Step 4**—Plot and visualize the categorical variables and data relationships.

In Fig. 10.7a (Step 4, Lines 57 to 62), the authors used the **ggplot( )** function in R to visualize the relationship (correlation) between the two categorical variables "**Segment**" and **"Discount.Band**" in the example dataset named "**Chisqd.data**".

The syntax and code used to plot the correlation are shown below, and the resultant chart represented in Fig. 10.7a.

```
# visualize association of the two categorical variables
ggplot(Chisqd.data) +
  aes(x = Discount.Band, fill = Segment) +
  geom_bar() +
  scale_fill_hue() +
  theme_minimal()
```

Additionally, in Fig. 10.7b (Step 4, Lines 65 to 71), we made use of the **assocstats( )** and **mosaic( )** functions in R to plot the results of the Chi-squared Independence test. Technically, the **mosaic( )** method has the advantage of combining the Contingency table and the result of the Chi-square ($X^2$) test of independence.

The code used to plot the contingency table and the result of the Chi-squared test of independence is shown below, and the resulting chart represented in Fig. 10.7b.

```
# Combine plot and statistical test
 assocstats(ChiSqd_Ind_table)
 mosaic(~ Segment + Discount.Band,
        direction = c("h", "v"),
        data = Chisqd.data, shade = TRUE, legend = TRUE,
        highlighting_direction = "right",
        main = "Chart Rep. Data plot and Chi-squared Result")
```

**# Step 5**—Results Interpretation for the Chi-squared test.

The final step for the Chi-squared ($X^2$) test and analysis is to interpret and understand the results of the test/method.

By default, the hypothesis for conducting the two main tests (Independence test, and Goodness of Fit test) by considering the selected categorical variables "**Segment**" and "**Discount.Band**" (see: Fig. 10.4) is as follows:

**Test of Independence:**

- **($H_1$)** *IF* the p-value of the test is less than or equal to 0.05 ($p \leq 0.05$), *THEN* we can assume that the two variables are associated, thus, there is a relationship (correlation) between the two categorical variables. In other words, determining the value of one variable helps to predict the value of the other.
- **($H_0$)** *ELSE* IF the p-value is greater than 0.05 ($p > 0.05$) THEN we can say that the two variables are not related, thus, there is no relationship (correlation) between the two categorical variables. Therefore, determining the value of one variable does not help to predict the value of the other, and vice and versa.

```
 > ChiSqd_Ind_test

        Pearson's Chi-squared test

data:  ChiSqd_Ind_table

X-squared = 26.825, df = 12, p-value = 0.008189
```

Consequentially, as shown in the results of the test represented above (see: Fig. 10.6a); the meaning of the **Independence Chi-squared ($X^2$)** test output can be explained as a list containing the following:

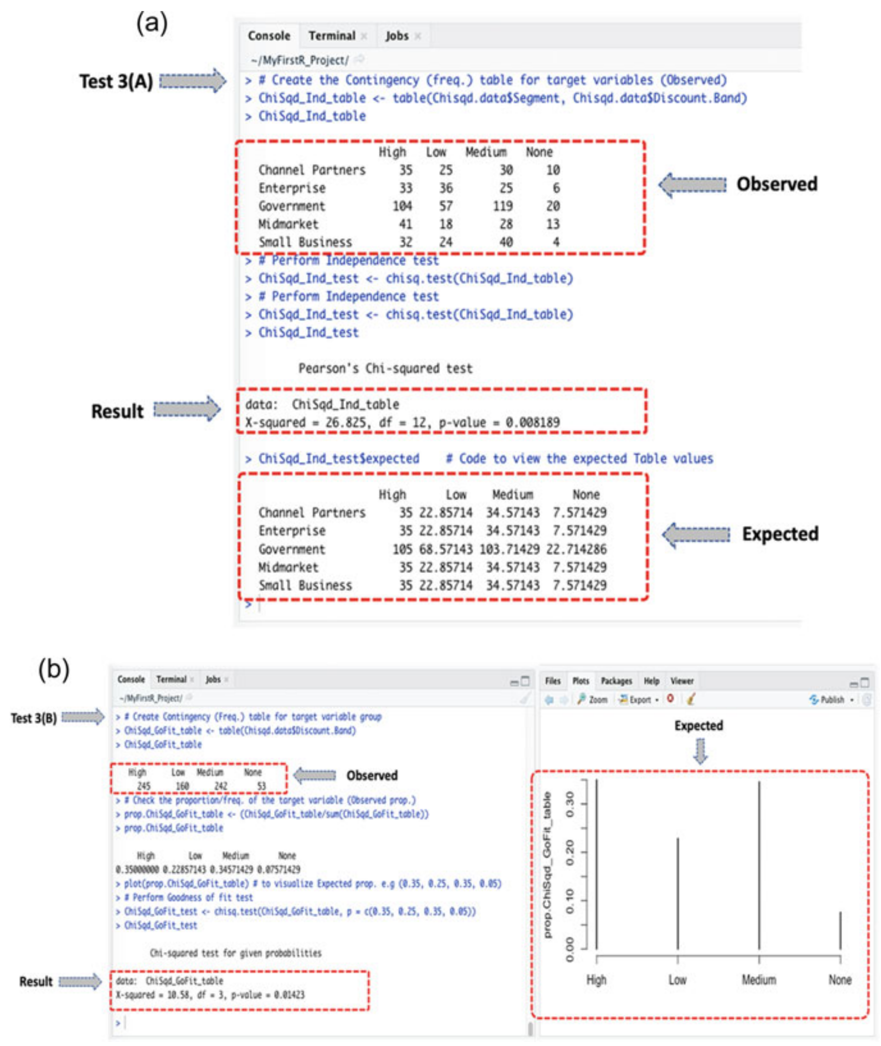
- **Statistics**: `X`$^2$ `(X-squared) = 26.825` represents the value of the correlation test.
- **p-value**: `p-value = 0.008189` is the significance level of the test.

Statistically, we can see from the reported result (`p-value=0.008189`) that the p-value is less than the slated significance level ($p \leq 0.05$) deemed scientifically acceptable. Therefore, we reject the $H_0$ and accept the $H_1$ by statistically concluding that there is a significant relationship (correlation) between the "**Segment**" and "**Discount.Band**" variables in the analyzed data.

Likewise, for the **Goodness of Fit test** we conducted to determine whether the *observed* group of the "**Discount.Band**" variable matches or is capable of fitting the *expected* population:

**Goodness of Fit test:**

- **($H_1$)** *IF* the p-value of the test is less than or equal to 0.05 ($p \leq 0.05$), *THEN* we assume that the observed group of the "**Discount.Band**" (categorical) variable does not match (i.e., varies or are not commonly distributed) and are not capable of fitting the expected population. In other words, the different groups in the "Discount.Band" variable are not the same and are not an expected representative (fits) of the studied population, and may consequently not be used to make ample conclusions about the studied population.
- **($H_0$)** *ELSE IF* the p-value is greater than 0.05 ($p > 0.05$) THEN we can say that the observed group of the "**Discount.Band**" (categorical) variable match (i.e.,



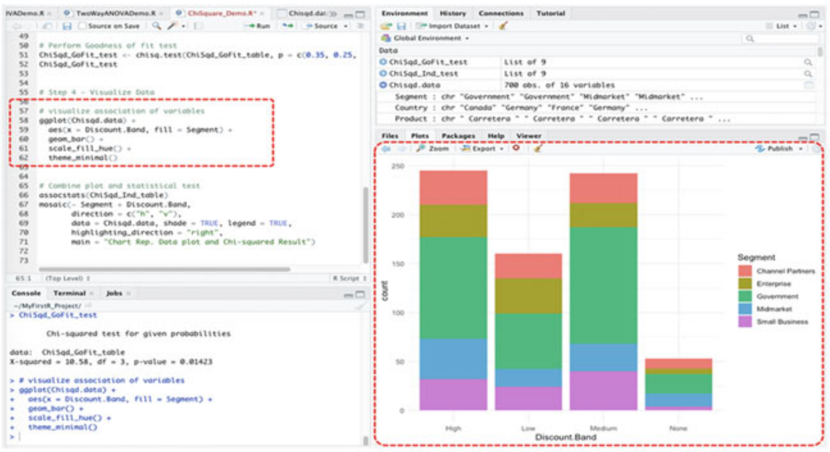
```
~/MyFirstR_Project/
> # Create the Contingency (freq.) table for target variables (Observed)
> ChiSqd_Ind_table <- table(Chisqd.data$Segment, Chisqd.data$Discount.Band)
> ChiSqd_Ind_table

                  High  Low  Medium  None
  Channel Partners   35   25      30    10
  Enterprise         33   36      25     6
  Government        104   57     119    20
  Midmarket          41   18      28    13
  Small Business     32   24      40     4
> # Perform Independence test
> ChiSqd_Ind_test <- chisq.test(ChiSqd_Ind_table)
> # Perform Independence test
> ChiSqd_Ind_test <- chisq.test(ChiSqd_Ind_table)
> ChiSqd_Ind_test

        Pearson's Chi-squared test

data:  ChiSqd_Ind_table
X-squared = 26.825, df = 12, p-value = 0.008189

> ChiSqd_Ind_test$expected    # Code to view the expected Table values

                  High      Low    Medium      None
  Channel Partners   35 22.85714  34.57143  7.571429
  Enterprise         35 22.85714  34.57143  7.571429
  Government        105 68.57143 103.71429 22.714286
  Midmarket          35 22.85714  34.57143  7.571429
  Small Business     35 22.85714  34.57143  7.571429
>
```

```
~/MyFirstR_Project/
> # Create Contingency (Freq.) table for target variable group
> ChiSqd_GoFit_table <- table(Chisqd.data$Discount.Band)
> ChiSqd_GoFit_table

  High    Low  Medium    None
   245    160     242      53
> # Check the proportion/freq. of the target variable (Observed prop.)
> prop.ChiSqd_GoFit_table <- (ChiSqd_GoFit_table/sum(ChiSqd_GoFit_table))
> prop.ChiSqd_GoFit_table

      High        Low     Medium       None
0.35000000 0.22857143 0.34571429 0.07571429
> plot(prop.ChiSqd_GoFit_table) # to visualize Expected prop. e.g (0.35, 0.25, 0.35, 0.05)
> # Perform Goodness of fit test
> ChiSqd_GoFit_test <- chisq.test(ChiSqd_GoFit_table, p = c(0.35, 0.25, 0.35, 0.05))
> ChiSqd_GoFit_test

        Chi-squared test for given probabilities

data:  ChiSqd_GoFit_table
X-squared = 10.58, df = 3, p-value = 0.01423

>
```

**Fig. 10.6** **a** Results of the (Independence) Chi-squared ($X^2$) test in R. **b** Results of the (Goodness of Fit) Chi-squared ($X^2$) test in R

fit or are commonly distributed) and are capable of representing the expected population. Thus, the different groups in the "Discount.Band" variable are an expected representative (fits) of the studied population and can be utilized to make conclusions about the studied population, vice and versa.

(a)

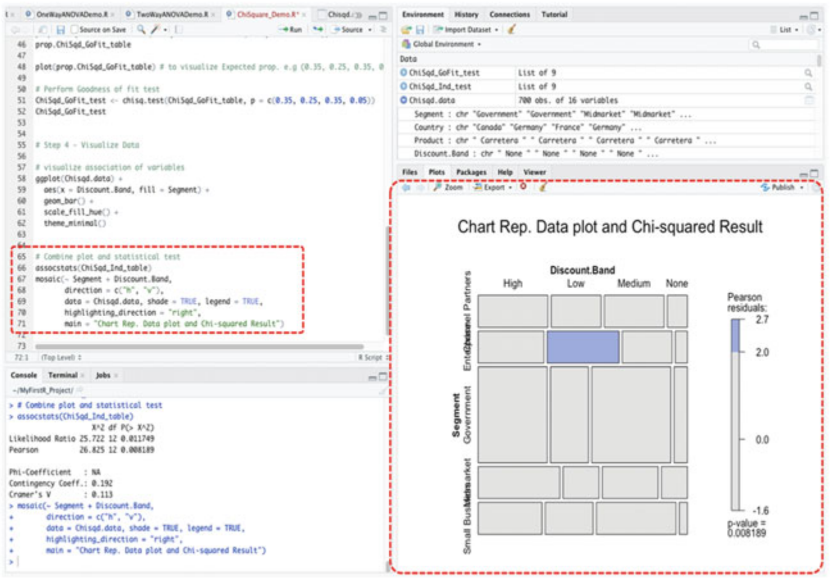


```
# Perform Goodness of fit test
ChiSqd_GoFit_test <- chisq.test(ChiSqd_GoFit_table, p = c(0.35, 0.25,
ChiSqd_GoFit_test


# Step 4 - Visualize Data

# visualize association of variables
ggplot(Chisqd.data) +
  aes(x = Discount.Band, fill = Segment) +
  geom_bar() +
  scale_fill_hue() +
  theme_minimal()


# Combine plot and statistical test
assocstats(ChiSqd_Ind_table)
mosaic(~ Segment + Discount.Band,
       direction = c("h", "v"),
       data = Chisqd.data, shade = TRUE, legend = TRUE,
       highlighting_direction = "right",
       main = "Chart Rep. Data plot and Chi-squared Result")
```

```
> ChiSqd_GoFit_test

	Chi-squared test for given probabilities

data:  ChiSqd_GoFit_table
X-squared = 10.58, df = 3, p-value = 0.01423

> # visualize association of variables
> ggplot(Chisqd.data) +
+   aes(x = Discount.Band, fill = Segment) +
+   geom_bar() +
+   scale_fill_hue() +
+   theme_minimal()
> |
```

```
Data
ChiSqd_GoFit_test     List of 9
ChiSqd_Ind_test       List of 9
Chisqd.data           700 obs. of 16 variables
  Segment : chr "Government" "Government" "Midmarket" "Midmarket" ...
  Country : chr "Canada" "Germany" "France" "Germany" ...
  Product : chr " Carretera " " Carretera " " Carretera " " Carretera " ...
```

(b)



```
prop.ChiSqd_GoFit_table

plot(prop.ChiSqd_GoFit_table) # to visualize Expected prop. e.g (0.35, 0.25, 0.35, 0

# Perform Goodness of fit test
ChiSqd_GoFit_test <- chisq.test(ChiSqd_GoFit_table, p = c(0.35, 0.25, 0.35, 0.05))
ChiSqd_GoFit_test


# Step 4 - Visualize Data

# visualize association of variables
ggplot(Chisqd.data) +
  aes(x = Discount.Band, fill = Segment) +
  geom_bar() +
  scale_fill_hue() +
  theme_minimal()


# Combine plot and statistical test
assocstats(ChiSqd_Ind_table)
mosaic(~ Segment + Discount.Band,
       direction = c("h", "v"),
       data = Chisqd.data, shade = TRUE, legend = TRUE,
       highlighting_direction = "right",
       main = "Chart Rep. Data plot and Chi-squared Result")
```

```
> # Combine plot and statistical test
> assocstats(ChiSqd_Ind_table)
                    X^2 df  P(> X^2)
Likelihood Ratio 25.722 12 0.011749
Pearson          26.825 12 0.008189

Phi-Coefficient   : NA
Contingency Coeff.: 0.192
Cramer's V        : 0.113
> mosaic(~ Segment + Discount.Band,
+        direction = c("h", "v"),
+        data = Chisqd.data, shade = TRUE, legend = TRUE,
+        highlighting_direction = "right",
+        main = "Chart Rep. Data plot and Chi-squared Result")
> |
```

```
Data
ChiSqd_GoFit_test     List of 9
ChiSqd_Ind_test       List of 9
Chisqd.data           700 obs. of 16 variables
  Segment : chr "Government" "Government" "Midmarket" "Midmarket" ...
  Country : chr "Canada" "Germany" "France" "Germany" ...
  Product : chr " Carretera " " Carretera " " Carretera " " Carretera " ...
  Discount.Band : chr " None " " None " " None " " None " ...
```

**Fig. 10.7** **a** Plot representing correlation (relationship) between two categorical variables using the ggplot( ) function in R. **b** Plot representing the Contingency table and the Chi-squared test of Independence using the assocstats( ) and mosaic( ) functions in R

```
> ChiSqd_GoFit_test

	Chi-squared test for given probabilities

data:  ChiSqd_GoFit_table
X-squared = 10.58, df = 3, p-value = 0.01423
```

Accordingly, as shown in the Goodness of Fit test result presented above (see: Fig. 10.6b); we can see that the p-value (`p-value = 0.01423`) is less than the stated significance level ($p \leq 0.05$). Therefore, we reject the $H_0$ and accept $H_1$ by concluding that the different groups of the "Discount.Band" variable are not proportionate or commonly distributed (does not fit or vary), or are not an expected representative of the studied population.

## 10.3 Conclusion

In this chapter, the authors demonstrated how to conduct the two main types of Chi-squared ($X^2$) tests in R. This includes the "Independence" and "Goodness of Fit" tests covered in Sect. 10.2.

Also, the chapter covers how to graphically plot the results of the Chi-squared tests and the Contingency table, and then discussed in detail how to interpret and understand the results of the test (Chi-squared) in R.

In summary, the main contents covered in this chapter are as follows:

- The *Chi-Squared* ($X^2$) statistics measures how *expectations* compares to actual *observed* data or results of a model. It is mainly used to determine whether there exists a relationship (correlation) between two categorical variables.
- The test of "Independence" checks the association between the two categorical variables, while
- The "Goodness of fit" test checks if there is a significant difference between the *observed* frequency values and the *expected* frequency values for a specific variable.
- In either case (Independence test or Goodness of Fit test), the researcher or data analyst must create a Contingency table, otherwise referred to as "frequency table" before applying the Chi-squared ($X^2$) method.

## References

Biswal, A. (2023). *What is a Chi-Square Test? Formula, Examples & Application.* https://www.simplilearn.com/tutorials/statistics-tutorial/chi-square-test.

Cochran, W. G. (1952). The $\chi^2$ Test of Goodness of Fit. *The Annals of Mathematical Statistics*, *23*(3), 315–345. http://www.jstor.org/stable/2236678.

Kishore, K., & Jaswal, V. (2023). Statistics corner: Chi-squared test. *Journal of Postgraduate Medicine, Education and Research, 57*(1), 40–44. https://doi.org/10.5005/jp-journals-10028-1618

McHugh, M. L. (2012). The Chi-square test of independence. *Biochemia Medica*, *23*(2), 143–149. https://doi.org/10.11613/BM.2013.018.

Preacher, K. J. (2001). *Calculation for the chi-square test: An interactive calculation tool for chi-square tests of goodness of fit and independence.* Quantpsy.Org. http://www.quantpsy.org/chisq/chisq.htm.

Rolke, W., & Gongora, C. G. (2020). A chi-square goodness-of-fit test for continuous distributions against a known alternative. *Computational Statistics, 1–16*,. https://doi.org/10.1007/s00180-020-00997-x

Sayassatov, D., & Cho, .namjae. (2020). The analysis of association between learning styles and a model of IoT-based education : Chi-square test for association. *J. Inf. Technol. Appl. Manag*, *27*(3), 19–36. https://doi.org/10.21219/jitam.2020.27.3.019.

Taneichi, N., Sekiya, Y., & Toyama, J. (2020). Improvement of the test of independence among groups of factors in a multi-way contingency table. *Japanese Journal of Statistics and Data Science, 1–33*,. https://doi.org/10.1007/s42081-020-00094-9

Turhan, N. S. (2020). Karl pearson's Chi-square tests. *Educational Research and Reviews, 15*(9), 575–580. https://doi.org/10.5897/ERR2019.3817

# Chapter 11
# Mann–Whitney U Test and Kruskal–Wallis H Test Statistics in R

## 11.1 Introduction

The likely effect of the *independent* variable(s) over a *dependent* variable can be analyzed or determined using the "Mann–Whitney U" and "Kruskal–Wallis H" tests. The two tests are used to determine if there exist statistically differences (significant levels usually measured through the p-values, where $p \leq 0.05$) between the independent observations in a given dataset based on the dependent variable or observation. In theory, the tests (Mann–Whitney U and Kruskal–Wallis H) are referred to as *non-parametric* procedures or methods used by the researchers or data analysts to statistically determine whether a group of data comes from the same population by considering the effect of the independent variable on the dependent variable (Frey, 2018; le Cessie et al., 2020; MacFarland et al., 2016; McKight & Najab, 2010; McKnight & Najab, 2010; Nachar, 2008; Okoye et al., 2022; Ortega, 2023; Ostertagová et al., 2014; Vargha & Delaney, 1998).

Furthermore, just like many of the other types of the "non-parametric" procedures, the two tests (Mann–Whitney U and Kruskal–Wallis H) are usually applied when the data sample in question are not normally distributed (i.e., violates the assumption of *t*-distribution) or the data sample size is too small to conduct the parametric methods or procedures (see Chaps. 3 and 4). Thus, while the measurement to establish whether the independent groups of variables being analyzed comes from the same population via the "mean" for the parametric procedures, the non-parametric equivalents or tests (such as the *Mann–Whitney U* and *Kruskal–Wallis H*), on the other hand, are measured by considering the "median" (see Chap. 4).

By definition, the *Mann–Whitney U* test, also known as the *U* test, is used to determine the differences in *median* between *two groups* of an independent variable with no specific distribution on a single ranked scale, and must be ordinal variable data type (McKnight & Najab, 2010; Ramtin, 2023). The test (Mann–Whitney) is often considered as the *non-parametric* version or equivalent of the Independent Samples *t*-test (a type of parametric test). Moreover, while the *t*-test (parametric) and Mann–Whitney U (non-parametric) tests may show to serve the same statistical purposes, due to the fact that they are both used to determine if there exists a statistically significant differences between the *two groups of an independent variable*. On the contrary, the Mann–Whitney U test is used with "ordinal" or "ranked" datasets that may have violated the assumptions of normality or small sample size, whereas, the *t*-test is used with "continuous" or "interval" datasets that happen to meet the assumptions of normality or large sample size (MacFarland et al., 2016). Therefore, the Mann–Whitney U test is most suitable when the data that is being analyzed by the researcher or data analyst is in ranked form, deviates from the acceptable *t*-distribution, or the probability that a randomly drawn member(s) of the first group (e.g., group A) of the population will exceed the second group (group B) of the population in a single independent variable or data (see: Chap. 6, Sect. 6.2.6).

Mathematically, the result of applying the Mann–Whitney U test is a U-Statistic or formula represented as follows (Mann & Whitney, 1947; Nachar, 2008):

$$U_1 = n_1 n_2 + \frac{n_1(n_1 + 1)}{2} - R_1$$

$$U_2 = n_1 n_2 + \frac{n_2(n_2 + 1)}{2} - R_2$$

where:

$R_1$= sum of the ranks for group 1
$R_2$ = sum of the ranks for group 2
$n_1$ = number of observations or participants for group 1
$n_2$ = number of observations or participants for group 2

As seen in the formula above, it is noteworthy to mention that the Mann–Whitney U statistics involves pooling the observations from the two groups of samples (e.g., group A and group B) into one combined sample, done by keeping track of which sample each observation comes from, and then *ranking* them according to lowest to highest, i.e., from 1 to $R_1 + R_2$, respectively.

On the other hand, the *Kruskal–Wallis* test, also referred to as *H* test, is described as an extension of the two-grouped Mann–Whitney U test (McKight & Najab, 2010). Thus, the method (Kruskal–Wallis H) (see Chap. 6, Sect. 6.2.8) is used when the researcher is comparing the median of more than two groups (i.e., three or more categories) of independent samples (Ortega, 2023; Vargha & Delaney, 1998). Just like the Mann–Whitney U test, the Kruskal–Wallis H method uses *ranked* (ordinal) datasets, a powerful alternative (non-parametric version) to the One-way analysis of variance (ANOVA), and proves to be a suitable statistical method when the data sample in question deviates from the acceptable *t*-distribution or is not normally distributed (Ostertagová et al., 2014).

Mathematically, the result of applying the Kruskal–Wallis test is an H-Statistic or formula represented as follows (Kruskal & Wallis, 1952; McKight & Najab, 2010):

$$H = \left( \frac{12}{n(n+1)} \sum\nolimits_{j=1}^{k} \frac{R_j^2}{n_j} \right) - 3(n+1)$$

where:

$$\begin{aligned} k &= number\ of\ groups\ being\ compared\ or\ analyzed \\ n &= total\ sample\ size \\ n_j &= sample\ size\ in\ the\ jth\ group \\ R_j &= sum\ of\ the\ ranks\ in\ the\ jth\ jth\ group \end{aligned}$$

As defined in the above formula, it is noteworthy to mention that with the Kruskal–Wallis H statistics, all of the $n$ values or measurements (e.g., $n = n_1 + n_2 + \ldots + n_k$) are jointly ranked (i.e., are treated as one single sample), and one can use the sums of the ranks of the $k$ samples to compare the distributions.

Accordingly, in Table 11.1 the authors provide a summary of the differences and similarities between the Mann–Whitney U test and Kruskal–Wallis H test, including the different conditions that are necessary or required to performing the tests, which are practically demonstrated in R in the next sections of this chapter (Sects. 11.2 and 11.3).

**Table 11.1** Differences and similarities between the Mann–Whitney U test and Kruskal–Wallis H tests and Assumptions

| Mann–Whitney U | Kruskal–Wallis H |
|---|---|
| Independent variable must be of two levels or groups, e.g., group A and group B | Independent variable must be more than two levels or groups (i.e., three or more), e.g., group A, group B, group C, …group $n^{\text{th}}$ |
| Used for Ranked or Ordinal datasets | Used for Ranked or Ordinal datasets |
| Dependent variable should be measured on an ordinal or continuous scale | Dependent variable should be measured on an ordinal or continuous scale |
| Data sample or observations are not normally distributed, i.e., skewed | Data sample or observations are not normally distributed, i.e., skewed |
| Data must be independent and randomly drawn from the population, i.e., no relationship should exist between the two groups or within each group | Data must be independent and randomly drawn from the population, i.e., no relationship should exist between the groups (minimum of three or more groups) or within each group |
| Measures or compares the “median”, unlike the parametric counterparts that compare the “mean” | Measures or compares the “median”, unlike the parametric counterparts that compare the “mean” |
| Non-parametric equivalent or version of the Independent sample *t*-test | Non-parametric equivalent or version of the One-way ANOVA test |

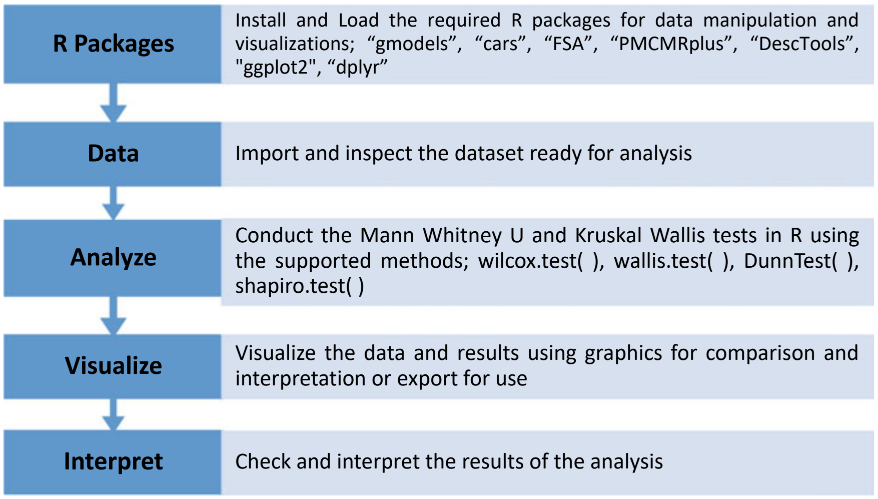

R Packages
Install and Load the required R packages for data manipulation and visualizations; "gmodels", "cars", "FSA", "PMCMRplus", "DescTools", "ggplot2", "dplyr"
Data
Import and inspect the dataset ready for analysis
Analyze
Conduct the Mann Whitney U and Kruskal Wallis tests in R using the supported methods; wilcox.test( ), wallis.test( ), DunnTest( ), shapiro.test( )
Visualize
Visualize the data and results using graphics for comparison and interpretation or export for use
Interpret
Check and interpret the results of the analysis

**Fig. 11.1** Steps to conducting Mann–Whitney U and Kruskal–Wallis H tests in R

In the next sections of this chapter (Sects. 11.2 and 11.3), the authors will be demonstrating to the readers how to conduct the "Mann–Whitney U" and "Kruskal–Wallis H" tests in R, respectively. We will illustrate the different steps to performing the two tests in R by using the following steps as outlined in Fig. 11.1.

## 11.2 Mann–Whitney U Test in R

*Mann–Whitney U* test is used when the data the researcher or analysts wants to analyze are made up of *two groups* and are statistically *independent*. Statistically, the test is used to compare the differences in *median* between an ordinal independent variable, and an ordinal or continuous dependent variable; whereby the independent variable must have two (ranked) levels. As defined earlier in Sect. 11.1, the test (Mann–Whitney U) is distribution free and has the powerful advantage of being used to analyze small sample sizes.

By default, the hypothesis for testing whether there is a *difference in the median* of the two specified groups of independent data (ordinal) against the dependent (usually ordinal or continuous) variable is; *IF* the p-value of the test (Mann–Whitney U) is less than or equal to 0.05 ($p \leq 0.05$), *THEN* we can assume that at least one sample of the two groups being analyzed comes from a population with a different distribution than the other, thus, the *median* of the groups of population (two groups) in the data sample are statistically different (varies), or yet in other words, that the first group are significantly larger than those of the second, or vice and versa, and that this is not by chance ($H_1$). *ELSE IF* the p-value is greater than 0.05 ($p > 0.05$) THEN we can assume that there is no difference in the *median* of the two groups, thus, the two independent groups are homogeneous and have the same distribution (stochastically equal), and any observed difference could only occur by chance ($H_0$).

Here, we will be demonstrating how to perform the Mann–Whitney U test by using the **wilcox.test( )** function in R. We will do this using the steps outlined in Fig. 11.1.

To begin, **Open RStudio** and **Create a new or Open an existing project**. Once the user have the RStudio and an R Project opened, **Create a new R Script** and name it "**MannWhitneyDemo**" or any name the user chooses (see Chap. 1 if the readers need to refresh on this step or topic).

We will download an example dataset that we will be using to demonstrate both the Mann–Whitney U test in this section (Sect. 11.2) and the Kruskal–Wallis H test in the next section (Sect. 11.3). ***Note: the users can use any dataset or format if they wish to do so***.

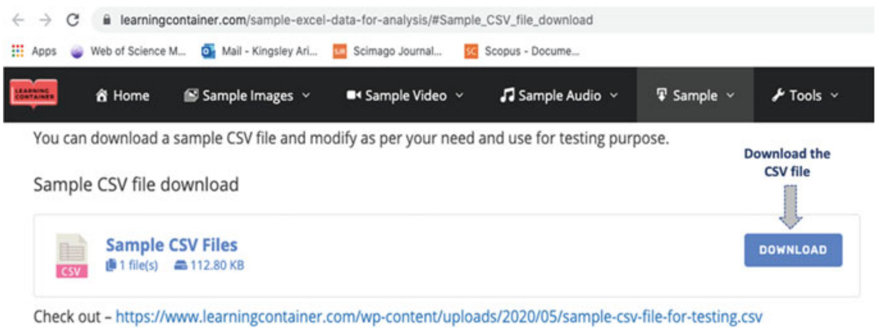
As shown in Fig. 11.2, download the example data named "**Sample CSV Files**" from the following link (source: https://www.learningcontainer.com/sample-excel-data-for-analysis/#Sample_CSV_file_download) if the user have not done so in the previous chapter (Chap. 10), and save the downloaded file on the computer. ***Also, the example dataset can accessed and downloaded via the following link (https://doi.org/https://doi.org/10.6084/m9.figshare.24728073) where the authors have uploaded all the example files used in this book.

Once the user have successfully downloaded and/or accessed the example file on the computer, we can proceed to conduct the Mann–Whitney U test in R.

# **Step 1**—Install and Load the required R Packages and Libraries

**Install** and **Load** the following *R packages* and *libraries* (Fig. 11.3, Step1, Lines 3 to 15) that will be used to call the different R functions, data manipulations, and graphical visualizations for the Mann–Whitney U test.

The syntax and code to install and load the R packages and Libraries are as follows:



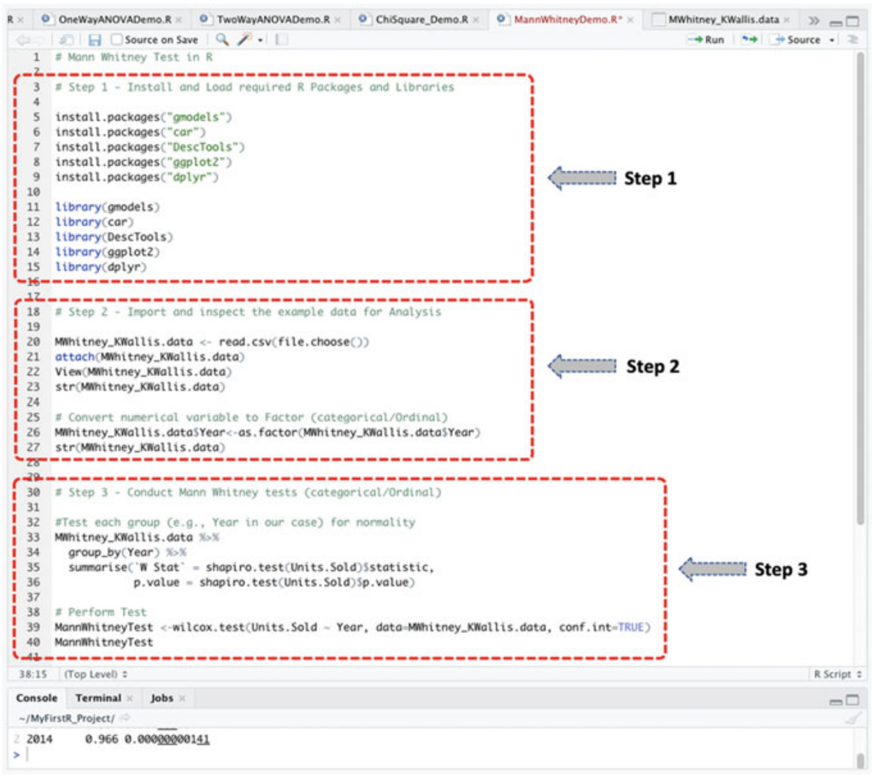
You can download a sample CSV file and modify as per your need and use for testing purpose.

Sample CSV file download

Sample CSV Files
1 file(s) 112.80 KB

Check out – https://www.learningcontainer.com/wp-content/uploads/2020/05/sample-csv-file-for-testing.csv

**Fig. 11.2** Example of CSV data sample and file download. *Source* https://www.learningcontainer.com/sample-excel-data-for-analysis/#Sample_CSV_file_download



```
1  # Mann Whitney Test in R
2
3  # Step 1 - Install and Load required R Packages and Libraries
4
5  install.packages("gmodels")
6  install.packages("car")
7  install.packages("DescTools")
8  install.packages("ggplot2")
9  install.packages("dplyr")
10
11 library(gmodels)
12 library(car)
13 library(DescTools)
14 library(ggplot2)
15 library(dplyr)
16
17
18 # Step 2 - Import and inspect the example data for Analysis
19
20 MWhitney_KWallis.data <- read.csv(file.choose())
21 attach(MWhitney_KWallis.data)
22 View(MWhitney_KWallis.data)
23 str(MWhitney_KWallis.data)
24
25 # Convert numerical variable to Factor (categorical/Ordinal)
26 MWhitney_KWallis.data$Year<-as.factor(MWhitney_KWallis.data$Year)
27 str(MWhitney_KWallis.data)
28
29
30 # Step 3 - Conduct Mann Whitney tests (categorical/Ordinal)
31
32 #Test each group (e.g., Year in our case) for normality
33 MWhitney_KWallis.data %>%
34   group_by(Year) %>%
35   summarise(`W Stat` = shapiro.test(Units.Sold)$statistic,
36             p.value = shapiro.test(Units.Sold)$p.value)
37
38 # Perform Test
39 MannWhitneyTest <-wilcox.test(Units.Sold ~ Year, data=MWhitney_KWallis.data, conf.int=TRUE)
40 MannWhitneyTest
41
```

```
2 2014      0.966 0.00000000141
>
```

**Fig. 11.3** Conducting Mann–Whitney U test in R

```
install.packages("gmodels")
install.packages("car")
install.packages("DescTools")
install.packages("ggplot2")
install.packages("dplyr")

library(gmodels)
library(car)
library(DescTools)
library(ggplot2)
library(dplyr)
```

## # Step 2—Import and Inspect the example dataset for Analysis.

As shown in the Step 2 in Fig. 11.3 (Lines 18 to 27), import the example dataset named “sample-csv-file-for-testing” (**Sample CSV Files**) that we have downloaded earlier on the computer, and store this as an R object named “**MWhitney_KWallis.data**” in RStudio (***the users can use any name of choice if they wish to do so).

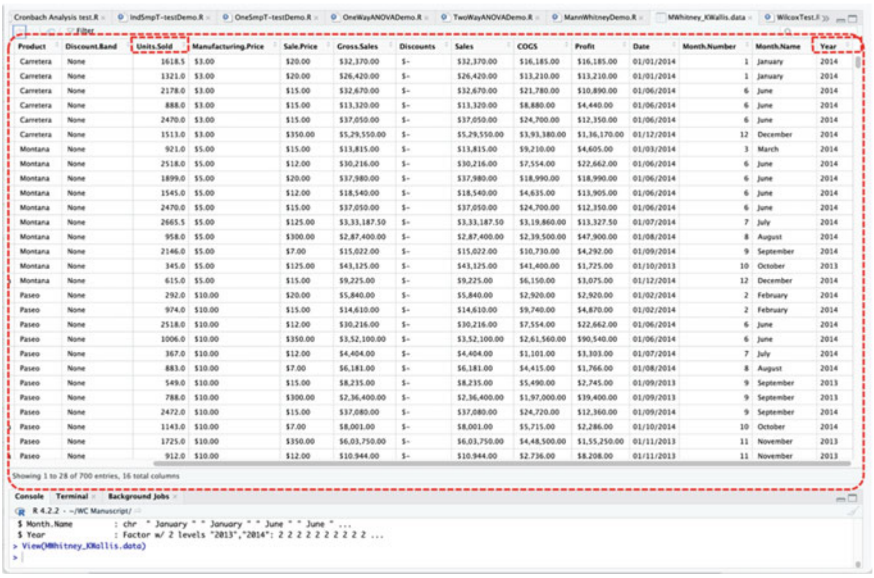
The code for importing and storing of the example file in R is as shown below:

```
MWhitney_KWallis.data <- read.csv(file.choose())
attach(MWhitney_KWallis.data)
View(MWhitney_KWallis.data)
str(MWhitney_KWallis.data)
```

Once the user has successfully imported and stored the dataset in RStudio environment, you will be able to view the details of the example file "sample-csv-file-for-testing" (Sample CSV Files) named "**MWhitney_KWallis.data**" in the file environment as shown in Fig. 11.4 with 700 observations and 16 variables contained in the stored data sample (see data fragment in Fig. 11.4).

**Note**: the authors have also perfomed an important step by converting the variable named "Year" with two levels or groups (i.e., 2013, 2014) (see: Figs. 11.3 and 11.4) to a Categorial (ordinal) variable as we will be using this to illustrate the Mann–Whitney U test (see code below—Step 2, Fig. 11.3).

```
#Convert numerical variable to Factor (categorical/Ordinal)
MWhitney_KWallis.data$Year<-as.factor(MWhitney_KWallis.data$Year)
str(MWhitney_KWallis.data)
```



| Product | Discount.Band | Units.Sold | Manufacturing.Price | Sale.Price | Gross.Sales | Discounts | Sales | COGS | Profit | Date | Month.Number | Month.Name | Year |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Carretera | None | 1618.5 | $3.00 | $20.00 | $32,370.00 | $- | $32,370.00 | $16,185.00 | $16,185.00 | 01/01/2014 | 1 | January | 2014 |
| Carretera | None | 1321.0 | $3.00 | $20.00 | $26,420.00 | $- | $26,420.00 | $13,210.00 | $13,210.00 | 01/01/2014 | 1 | January | 2014 |
| Carretera | None | 2178.0 | $3.00 | $15.00 | $32,670.00 | $- | $32,670.00 | $21,780.00 | $10,890.00 | 01/06/2014 | 6 | June | 2014 |
| Carretera | None | 888.0 | $3.00 | $15.00 | $13,320.00 | $- | $13,320.00 | $8,880.00 | $4,440.00 | 01/06/2014 | 6 | June | 2014 |
| Carretera | None | 2470.0 | $3.00 | $15.00 | $37,050.00 | $- | $37,050.00 | $24,700.00 | $12,350.00 | 01/06/2014 | 6 | June | 2014 |
| Carretera | None | 1513.0 | $3.00 | $350.00 | $5,29,550.00 | $- | $5,29,550.00 | $3,93,380.00 | $1,36,170.00 | 01/12/2014 | 12 | December | 2014 |
| Montana | None | 921.0 | $5.00 | $15.00 | $13,815.00 | $- | $13,815.00 | $9,210.00 | $4,605.00 | 01/03/2014 | 3 | March | 2014 |
| Montana | None | 2518.0 | $5.00 | $12.00 | $30,216.00 | $- | $30,216.00 | $7,554.00 | $22,662.00 | 01/06/2014 | 6 | June | 2014 |
| Montana | None | 1899.0 | $5.00 | $20.00 | $37,980.00 | $- | $37,980.00 | $18,990.00 | $18,990.00 | 01/06/2014 | 6 | June | 2014 |
| Montana | None | 1545.0 | $5.00 | $12.00 | $18,540.00 | $- | $18,540.00 | $4,635.00 | $13,905.00 | 01/06/2014 | 6 | June | 2014 |
| Montana | None | 2470.0 | $5.00 | $15.00 | $37,050.00 | $- | $37,050.00 | $24,700.00 | $12,350.00 | 01/06/2014 | 6 | June | 2014 |
| Montana | None | 2665.5 | $5.00 | $125.00 | $3,33,187.50 | $- | $3,33,187.50 | $3,19,860.00 | $13,327.50 | 01/07/2014 | 7 | July | 2014 |
| Montana | None | 958.0 | $5.00 | $300.00 | $2,87,400.00 | $- | $2,87,400.00 | $2,39,500.00 | $47,900.00 | 01/08/2014 | 8 | August | 2014 |
| Montana | None | 2146.0 | $5.00 | $7.00 | $15,022.00 | $- | $15,022.00 | $10,730.00 | $4,292.00 | 01/09/2014 | 9 | September | 2014 |
| Montana | None | 345.0 | $5.00 | $125.00 | $43,125.00 | $- | $43,125.00 | $41,400.00 | $1,725.00 | 01/10/2013 | 10 | October | 2013 |
| Montana | None | 615.0 | $5.00 | $15.00 | $9,225.00 | $- | $9,225.00 | $6,150.00 | $3,075.00 | 01/12/2014 | 12 | December | 2014 |
| Paseo | None | 292.0 | $10.00 | $20.00 | $5,840.00 | $- | $5,840.00 | $2,920.00 | $2,920.00 | 01/02/2014 | 2 | February | 2014 |
| Paseo | None | 974.0 | $10.00 | $15.00 | $14,610.00 | $- | $14,610.00 | $9,740.00 | $4,870.00 | 01/02/2014 | 2 | February | 2014 |
| Paseo | None | 2518.0 | $10.00 | $12.00 | $30,216.00 | $- | $30,216.00 | $7,554.00 | $22,662.00 | 01/06/2014 | 6 | June | 2014 |
| Paseo | None | 1006.0 | $10.00 | $350.00 | $3,52,100.00 | $- | $3,52,100.00 | $2,61,560.00 | $90,540.00 | 01/06/2014 | 6 | June | 2014 |
| Paseo | None | 367.0 | $10.00 | $12.00 | $4,404.00 | $- | $4,404.00 | $1,101.00 | $3,303.00 | 01/07/2014 | 7 | July | 2014 |
| Paseo | None | 883.0 | $10.00 | $7.00 | $6,181.00 | $- | $6,181.00 | $4,415.00 | $1,766.00 | 01/08/2014 | 8 | August | 2014 |
| Paseo | None | 549.0 | $10.00 | $15.00 | $8,235.00 | $- | $8,235.00 | $5,490.00 | $2,745.00 | 01/09/2013 | 9 | September | 2013 |
| Paseo | None | 788.0 | $10.00 | $300.00 | $2,36,400.00 | $- | $2,36,400.00 | $1,97,000.00 | $39,400.00 | 01/09/2013 | 9 | September | 2013 |
| Paseo | None | 2472.0 | $10.00 | $15.00 | $37,080.00 | $- | $37,080.00 | $24,720.00 | $12,360.00 | 01/09/2014 | 9 | September | 2014 |
| Paseo | None | 1143.0 | $10.00 | $7.00 | $8,001.00 | $- | $8,001.00 | $5,715.00 | $2,286.00 | 01/10/2014 | 10 | October | 2014 |
| Paseo | None | 1725.0 | $10.00 | $350.00 | $6,03,750.00 | $- | $6,03,750.00 | $4,48,500.00 | $1,55,250.00 | 01/11/2013 | 11 | November | 2013 |
| Paseo | None | 912.0 | $10.00 | $12.00 | $10,944.00 | $- | $10,944.00 | $2,736.00 | $8,208.00 | 01/11/2013 | 11 | November | 2013 |

Showing 1 to 28 of 700 entries, 16 total columns

```
 $ Month.Name         : chr  " January " " January " " June " " June " ...
 $ Year               : Factor w/ 2 levels "2013","2014": 2 2 2 2 2 2 2 2 2 2 ...
> View(MWhitney_KWallis.data)
> 
```

**Fig. 11.4** Example of CSV data imported and stored as an R object in RStudio

**# Step 3—Conduct Mann–Whitney U test (used for Categorical/Ordinal variable).**

With the example dataset stored and the targeted variables in categorical/ordinal scale, we can proceed to analyze the data which we have stored as **MWhitney_ KWallis.data** in R (see: Fig. 11.4).

As defined earlier in the Introduction section (Sect. 11.1):

- **Mann–Whitney U** test or statistics compares the *median or distribution* between two groups of an independent variable against a target dependent variable.
- The targeted independent variable must be an "ordinal" data type.
- The targeted dependent variable must be an "ordinal or continuous" data type.

To demonstrate the Mann–Whitney U test using the **wilcox.test( )** method in R:

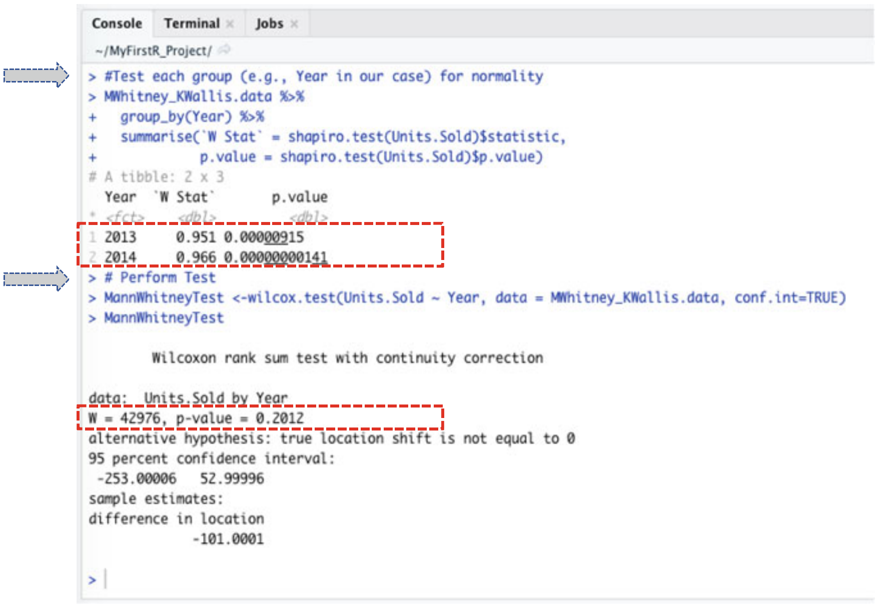
- We will test whether the median or distribution of the "**Year**" variable (independent variable with two levels: 2013 and 2014) differ based on the "**Units.Sold**" (dependent variable) in the data.

The syntax and code to conducting the above test (Mann–Whitney) in R is as shown in the code below and represented in Fig. 11.3 (Step 3, Lines 30 to 40).

```
# Step 3 - Conduct Mann Whitney tests (categorical/Ordinal)

#Test each group (e.g., Year in our case) for normality
MWhitney_KWallis.data %>%
  group_by(Year) %>%
  summarise(`W Stat` = shapiro.test(Units.Sold)$statistic,
            p.value = shapiro.test(Units.Sold)$p.value)


# Perform Test
MannWhitneyTest <-wilcox.test(Units.Sold ~ Year, data =
MWhitney_KWallis.data, conf.int=TRUE)
MannWhitneyTest
```

As shown in the code above, we first conducted a data normality test (see Chap. 3) (estimated acceptable p-value > 0.05) by considering each group of the "**Year**" variable before conducting the Mann–Whitney U test. This was done in order to confirm that the dataset does not meet the assumption of normality usually attributed to the Mann–Whitney U test (a non-parametric test), where: neither the 2013 (W=0.951, p=0.00000915) nor the 2014 (W=0.966, p=0.00000000141) groups of the Year variable were normally distributed, otherwise the user would have preferably conducted the Independent sample t-test (parametric equivalent of the Mann–Whitney U) in the event that the data appear to be normally distributed.

Once the user have successfully run the codes provided above (Step 3, Lines 30 to 40, Fig. 11.3), the user will be presented with the results of the Mann–Whitney method in the Console as shown in Fig. 11.5.



```
> #Test each group (e.g., Year in our case) for normality
> MWhitney_KWallis.data %>%
+   group_by(Year) %>%
+   summarise(`W Stat` = shapiro.test(Units.Sold)$statistic,
+             p.value = shapiro.test(Units.Sold)$p.value)
# A tibble: 2 x 3
  Year  `W Stat`       p.value
* <fct>    <dbl>         <dbl>
1 2013     0.951 0.00000915
2 2014     0.966 0.00000000141
> # Perform Test
> MannWhitneyTest <-wilcox.test(Units.Sold ~ Year, data = MWhitney_KWallis.data, conf.int=TRUE)
> MannWhitneyTest

	Wilcoxon rank sum test with continuity correction

data:  Units.Sold by Year
W = 42976, p-value = 0.2012
alternative hypothesis: true location shift is not equal to 0
95 percent confidence interval:
 -253.00006   52.99996
sample estimates:
difference in location
             -101.0001

> |
```

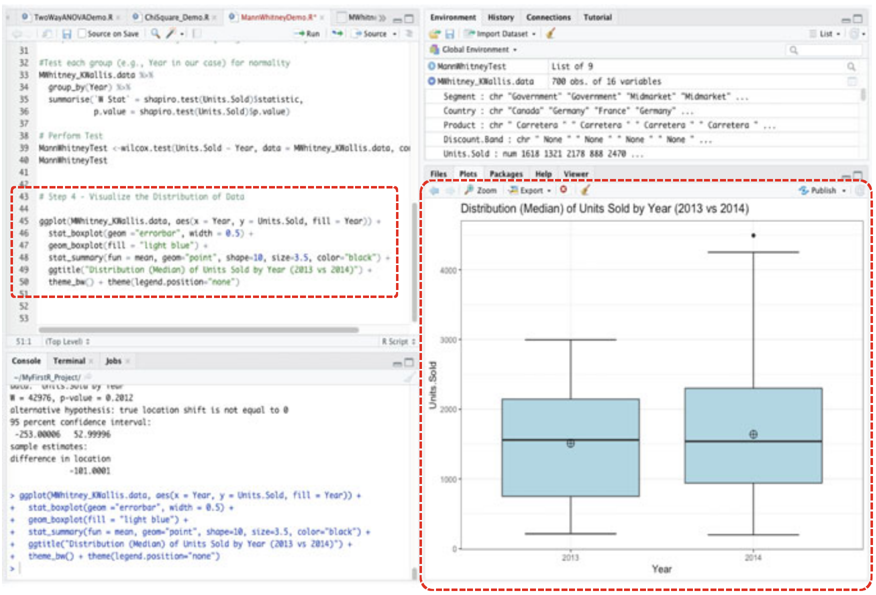
**Fig. 11.5** Results of Mann–Whitney U test displayed in the Console in R

To describe the test of assumptions, in Fig. 11.5, we conducted a normality test by considering the two groups (2013, 2014) of the "**Year**" variable against the "**Units.Sold**" variable using the **shapiro.test( )** method or function in R. As highlighted in the figure (Fig. 11.5), the result of the assumption test shows that the dataset taking into account the two groups of the variable was not normally distributed (where a significant level is considered values whereby $p > 0.05$). Therefore, we can assume that the data meets the condition to perform the Mann–Whitney U test with Group A (2013) showing a normality test statistic of `W=0.951, p-value=0.00000915`, and Group B (2014) showing `W=0.966, p-value=0.00000000141`, respectively.

Consequentially, we proceeded to perform the Mann–Whitney U test for the independent variable "**Year**" (with two levels or group) against the dependent variable "**Units.Sold**" as contained in the dataset "sample-csv-file-for-testing" (**Sample CSV Files**) that we stored as an R object named "**MWhitney_KWallis.data**" in R. Accordingly, the result of the Mann–Whitney U statistics was stored as an R object we named or defined as `MannWhitneyTest` (see: Fig. 11.5) which the authors will subsequently discuss in detail in Step 5 in this section.

# **Step 4**—Plot and visualize the data distribution and results.

In Fig. 11.6 (Step 4, Lines 43 to 50), the authors made use of the **ggplot( )** function in R to display a boxplot of the distribution between the two groups (2013, 2014)



```
#Test each group (e.g., Year in our case) for normality
MWhitney_KWallis.data %>%
  group_by(Year) %>%
  summarise(`W Stat` = shapiro.test(Units.Sold)$statistic,
            p.value = shapiro.test(Units.Sold)$p.value)

# Perform Test
MannWhitneyTest <-wilcox.test(Units.Sold ~ Year, data = MWhitney_KWallis.data, co
MannWhitneyTest


# Step 4 - Visualize the Distribution of Data

ggplot(MWhitney_KWallis.data, aes(x = Year, y = Units.Sold, fill = Year)) +
  stat_boxplot(geom ="errorbar", width = 0.5) +
  geom_boxplot(fill = "light blue") +
  stat_summary(fun = mean, geom="point", shape=10, size=3.5, color="black") +
  ggtitle("Distribution (Median) of Units Sold by Year (2013 vs 2014)") +
  theme_bw() + theme(legend.position="none")
```

```
W = 42976, p-value = 0.2012
alternative hypothesis: true location shift is not equal to 0
95 percent confidence interval:
 -253.00006   52.99996
sample estimates:
difference in location
             -101.0001

> ggplot(MWhitney_KWallis.data, aes(x = Year, y = Units.Sold, fill = Year)) +
+   stat_boxplot(geom ="errorbar", width = 0.5) +
+   geom_boxplot(fill = "light blue") +
+   stat_summary(fun = mean, geom="point", shape=10, size=3.5, color="black") +
+   ggtitle("Distribution (Median) of Units Sold by Year (2013 vs 2014)") +
+   theme_bw() + theme(legend.position="none")
>
```

```
MannWhitneyTest        List of 9
MWhitney_KWallis.data  700 obs. of 16 variables
  Segment : chr "Government" "Government" "Midmarket" "Midmarket" ...
  Country : chr "Canada" "Germany" "France" "Germany" ...
  Product : chr " Carretera " " Carretera " " Carretera " " Carretera " ...
  Discount.Band : chr " None " " None " " None " " None " ...
  Units.Sold : num 1618 1321 2178 888 2470 ...
```

Fig. 11.6 Plot representing the distribution of the two groups of Year variable broken down by Units.Sold using the ggplot() function in R

of the "**Year**" variable against the **"Units.Sold**" as contained in the stored data "**MWhitney_KWallis.data**".

The syntax and code used to plot and visualize the distribution is as shown in the code below, and the resultant plot is represented in Fig. 11.6.

```
# Step 4 - Visualize the Distribution of Data
ggplot(MWhitney_KWallis.data, aes(x = Year, y = Units.Sold, fill = Year)) +
  stat_boxplot(geom ="errorbar", width = 0.5) +
  geom_boxplot(fill = "light blue") +
  stat_summary(fun = mean, geom="point", shape=10, size=3.5, color="black")
+
  ggtitle("Distribution (Median) of Units Sold by Year (2013 vs 2014)") +
  theme_bw() + theme(legend.position="none")
```

**# Step 5**—Results Interpretation (Mann–Whitney U).

The final step for the Mann–Whitney U test and analysis is to interpret and understand the results of the test/method.

By default, the hypothesis for conducting the test (Mann–Whitney) considering the analyzed variables "**Year**" and "**Units.Sold**" (see: Fig. 11.4) is;

- **($H_1$)** *IF* the p-value of the test is less than or equal to 0.05 ($p \leq 0.05$), *THEN* we can assume that there is a difference in the distribution of the two groups of the “Year” variable (2014, 2014) taking into account the “Units.Sold”. Thus, the *median* of the two groups of population (2013, 2014) are statistically different (varies).

- **($H_0$)** *ELSE IF* the p-value is greater than 0.05 (p > 0.05) *THEN* we can say that there is no difference in the *median* of the two groups. Thus, the two independent groups (2013, 2014) are homogeneous and have the same distribution (stochastically equal) taking into account the “Units.Sold”.

```
> MannWhitneyTest

Wilcoxon rank sum test with continuity correction
data:  Units.Sold by Year
W = 42976, p-value = 0.2012
alternative hypothesis: true location shift is not equal to 0
95 percent confidence interval:
 -253.00006   52.99996
sample estimates:
difference in location
             -101.0001
```

As shown in the results presented above (see: Fig. 11.5), the meaning of the **Mann–Whitney U** test statistics or output can be explained as a list containing the following:

- **Statistics**: `W (U-statistics) = 42976` which represents the value of the distribution test.
- **p-value**: `p-value = 0.2012` is the significance levels of the test.

Statistically, we can see from the result that the p-value (`p-value = 0.2012`) is greater than the stated significance level (i.e., $p \leq 0.05$). Therefore, we reject the $H_1$ and accept $H_0$ by concluding that there is no significant difference (i.e., groups distribution are stochastically equal) between the two groups (“**2013**” and “**2014**”) of the “Year” variable taking into account the “Units.Sold”.

## 11.3 Kruskal–Wallis H Test in R

The *Kruskal–Wallis* H test is an extension of the Mann–Whitney U test. Statistically, the same assumptions or test criteria apply for both tests (Mann–Whitney and Kruskal–Wallis) except that with the Kruskal–Wallis H test, the targeted independent variable must have *more than two groups* or *categories* (i.e., minimum of three or more levels). Therefore, the test is applied by the researchers to test and compare the hypothesis that the *k* (nth) groups (minimum of three) in a data sample have been obtained or drawn from the same population. It is noteworthy to mention that the Kruskal–Wallis test is regarded as an alternative (non-parametric version) to the One-way ANOVA (le Cessie, 2020; Ortega, 2023). Thus, the test (Kruskal–Wallis) is used or applied when assumptions such as the data normality have not been met or the sample size is too small to conduct the parametric test (One-way ANOVA).

Just like the Mann–Whitney U test, by default, the hypothesis for testing whether there is a *difference in the median* of the *k* (nth) groups (minimum of three or more) of an independent data (ordinal) against the dependent (usually ordinal or continuous) variable is; *IF* the p-value of the test result (Kruskal–Wallis H) is less than or equal to 0.05 ($p \leq 0.05$), *THEN* we assume that at least one of the groups (of the *k* (nth) categories or levels) (see Chap. 6, Sect. 6.2.8) being analyzed comes from a population with a different distribution, and therefore, we can then further perform a multiple comparison (Post-Hoc) test to determine where the significant difference may lie across the data. In other words, we can assume that the *median* of the groups of population, *k* (nth), in the data sample are statistically different (varies), and that this is not by chance ($H_1$). *ELSE IF* the p-value is greater than 0.05 ($p > 0.05$) *THEN* we can say that there is no difference in the *median* of the *k* (nth) groups (three or more). Thus, the *k* (nth) independent groups are homogeneous and have the same distribution (i.e., are stochastically equal), and any difference observed could only occur by chance, and therefore in this scenario, there will be no need to further perform a multiple comparison (Post-Hoc) test ($H_0$).

Here, the authors will be demonstrating to the readers how to perform the Kruskal–Wallis H test using the **kruskal.test()** function in R. We will do this using the steps outlined earlier in Fig. 11.1.

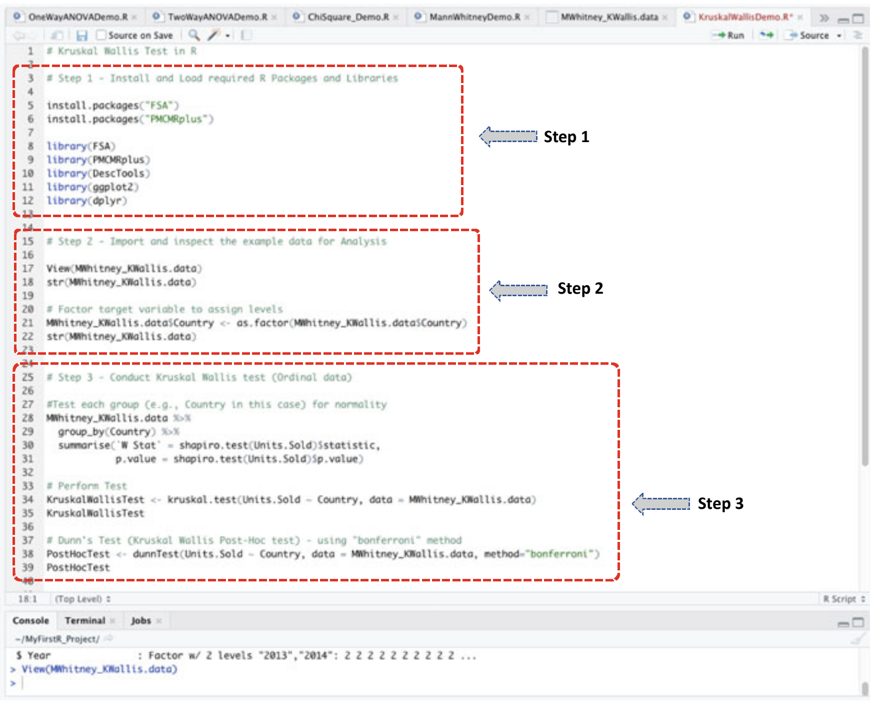
To begin, **Create a new RScript** and name it "**KruskalWallisDemo**" or any name of your choice.

Also, we will continue to use the same example dataset "sample-csv-file-for-testing" (**Sample CSV Files**) that we have stored earlier as an R object named "**MWhitney_KWallis.data**" to illustrate the Kruskal–Wallis H test. ***The users can also access and download the example dataset via the following link: https://doi.org/https://doi.org/10.6084/m9.figshare.24728073.

# **Step 1**—Install and Load the required R Packages and Libraries

**Install** and **Load** the following *R packages* and *libraries* (Fig. 11.7, Step1, Lines 3 to 12) that will be used to call the different R functions, data manipulations, and graphical visualizations for the Kruskal–Wallis H test.

The syntax to install and load the necessary R packages and Libraries are as follows:



```
1  # Kruskal Wallis Test in R
2
3  # Step 1 - Install and Load required R Packages and Libraries
4
5  install.packages("FSA")
6  install.packages("PMCMRplus")
7
8  library(FSA)
9  library(PMCMRplus)
10 library(DescTools)
11 library(ggplot2)
12 library(dplyr)
13
14
15 # Step 2 - Import and inspect the example data for Analysis
16
17 View(MWhitney_KWallis.data)
18 str(MWhitney_KWallis.data)
19
20 # Factor target variable to assign levels
21 MWhitney_KWallis.data$Country <- as.factor(MWhitney_KWallis.data$Country)
22 str(MWhitney_KWallis.data)
23
24
25 # Step 3 - Conduct Kruskal Wallis test (Ordinal data)
26
27 #Test each group (e.g., Country in this case) for normality
28 MWhitney_KWallis.data %>%
29   group_by(Country) %>%
30   summarise(`W Stat` = shapiro.test(Units.Sold)$statistic,
31             p.value = shapiro.test(Units.Sold)$p.value)
32
33 # Perform Test
34 KruskalWallisTest <- kruskal.test(Units.Sold ~ Country, data = MWhitney_KWallis.data)
35 KruskalWallisTest
36
37 # Dunn's Test (Kruskal Wallis Post-Hoc test) - using "bonferroni" method
38 PostHocTest <- dunnTest(Units.Sold ~ Country, data = MWhitney_KWallis.data, method="bonferroni")
39 PostHocTest
40
```

```
$ Year                : Factor w/ 2 levels "2013","2014": 2 2 2 2 2 2 2 2 2 2 ...
> View(MWhitney_KWallis.data)
> 
```

**Fig. 11.7** Steps to conducting Kruskal–Wallis H test in R

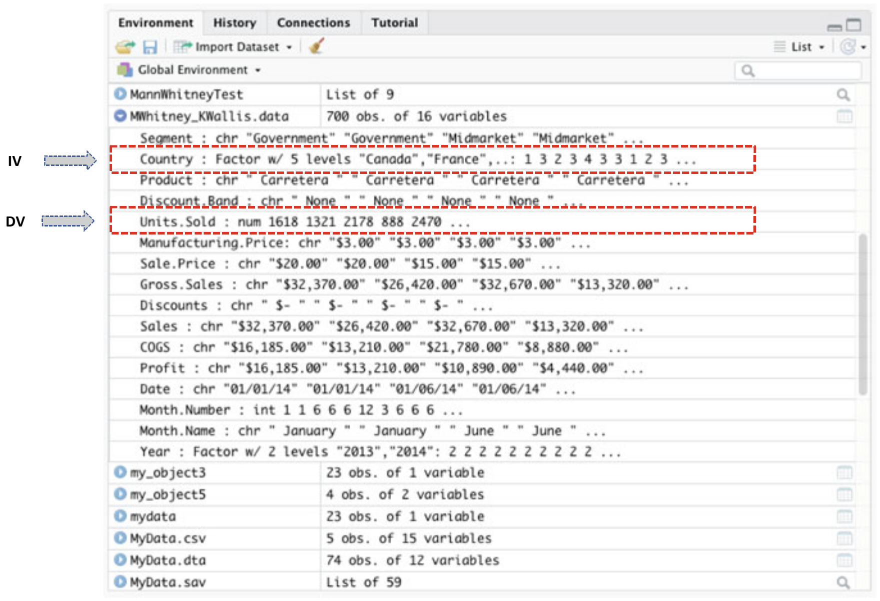
```
install.packages("FSA")
install.packages("PMCMRplus")


library(FSA)
library(PMCMRplus)
library(DescTools)
library(ggplot2)
library(dplyr)
```

*****Note**: as you can see in the code above (see highlighted part) and in Fig. 11.7, we only installed the additional R packages `"FSA"` and `"PMCMRplus"` that will be required to perform the Kruskal–Wallis H test, as we have previously installed the other required R packages in R in the previous section or example (see Sect. 11.2). Therefore, we only needed to just load the libraries for the already installed packages (i.e., `DescTools, ggplot2, dplyr`) (see: Lines 10 to 12, Fig. 11.7). ***Note: the users may need to install or re-install the above packages, if necessary, for instance, in the event they have not practiced the previous example in the previous section (Sect. 11.2), or have directly visited this particular section. Please refer to Chap. 2 (Sect. 2.6) on how to install R packages or a refresher on the topic.

**# Step 2—Import or Inspect the example dataset for Analysis.**

As shown in Fig. 11.7 (Step 2, Lines 15 to 22), since we have already imported and stored the example dataset "sample-csv-file-for-testing" (**Sample CSV Files**) as an R object we named "**MWhitney_KWallis.data**", we only need to view or inspect the data to make sure we have the variables want to analyze listed there (see: Code below and Fig. 11.8).

```
View(MWhitney_KWallis.data)
str(MWhitney_KWallis.data)

# Factor target variable to assign levels
MWhitney_KWallis.data$Country <-as.factor(MWhitney_KWallis.data$Country)
str(MWhitney_KWallis.data)
```



```
MannWhitneyTest          List of 9
MWhitney_KWallis.data    700 obs. of 16 variables
   Segment : chr "Government" "Government" "Midmarket" "Midmarket" ...
   Country : Factor w/ 5 levels "Canada","France",..: 1 3 2 3 4 3 3 1 2 3 ...
   Product : chr " Carretera " " Carretera " " Carretera " " Carretera " ...
   Discount.Band : chr " None " " None " " None " " None " ...
   Units.Sold : num 1618 1321 2178 888 2470 ...
   Manufacturing.Price: chr "$3.00" "$3.00" "$3.00" "$3.00" ...
   Sale.Price : chr "$20.00" "$20.00" "$15.00" "$15.00" ...
   Gross.Sales : chr "$32,370.00" "$26,420.00" "$32,670.00" "$13,320.00" ...
   Discounts : chr " $- " " $- " " $- " " $- " ...
   Sales : chr "$32,370.00" "$26,420.00" "$32,670.00" "$13,320.00" ...
   COGS : chr "$16,185.00" "$13,210.00" "$21,780.00" "$8,880.00" ...
   Profit : chr "$16,185.00" "$13,210.00" "$10,890.00" "$4,440.00" ...
   Date : chr "01/01/14" "01/01/14" "01/06/14" "01/06/14" ...
   Month.Number : int 1 1 6 6 6 12 3 6 6 6 ...
   Month.Name : chr " January " " January " " June " " June " ...
   Year : Factor w/ 2 levels "2013","2014": 2 2 2 2 2 2 2 2 2 2 ...
my_object3               23 obs. of 1 variable
my_object5               4 obs. of 2 variables
mydata                   23 obs. of 1 variable
MyData.csv               5 obs. of 15 variables
MyData.dta               74 obs. of 12 variables
MyData.sav               List of 59
```

**Fig. 11.8** Example of suitable variables for conducting the Kruskal–Wallis H test displayed in R (IV = Independent Variable, DV = Dependent Variable)

**Note**: if the user have directly visited this specific section or have exited and re-opened RStudio, then, they may need to use the following code below to upload and re-attach the data from their local machine or computer, as the case may be:

```
MWhitney_KWallis.data <- read.csv(file.choose())
attach(MWhitney_KWallis.data)
View(MWhitney_KWallis.data)
str(MWhitney_KWallis.data)
```

Once the user have successfully loaded, inspected, and completed the data conversion (see Step 2, Lines 15 to 22, Fig. 11.7); you will see in the Environment Tab or Console that the variable named "Country" has been factored with 5 levels or groups (see: highlighted part in Fig. 11.8) as we will be using this variable (Country) to illustrate the Kruskal–Wallis H test (i.e., that requires minimum of three levels of an independent variable as a condition to conduct the test).

**# Step 3—Conduct Kruskal–Wallis H test (Ordinal data).**

With all the necessary conditions and data format met, we can proceed to analyze the selected variables as highlighted in Fig. 11.8.

As defined earlier in the Introduction section (Sect. 11.1):

- **Kruskal–Wallis H** test or statistics compares the *median or distribution* between three or more groups of an independent variable against a targeted dependent variable.
- The targeted independent variable must be an "ordinal" data type.
- The targeted dependent variable must be an "ordinal or continuous" data type.

To demonstrate to the readers how to conduct the Kruskal–Wallis H test by using the **kruskal.test( )** and **dunnTest( )** method or functions in R:

- We will test whether the *median* or distribution of the "**Country**" (independent variable with 5 levels) differ based on the "**Units.Sold**" (dependent variable)—see Fig. 11.8.

The syntax for conducting the above test (Kruskal–Wallis) in R is as shown in the code below, and as represented in Fig. 11.7 (Step 3, Lines 25 to 39).

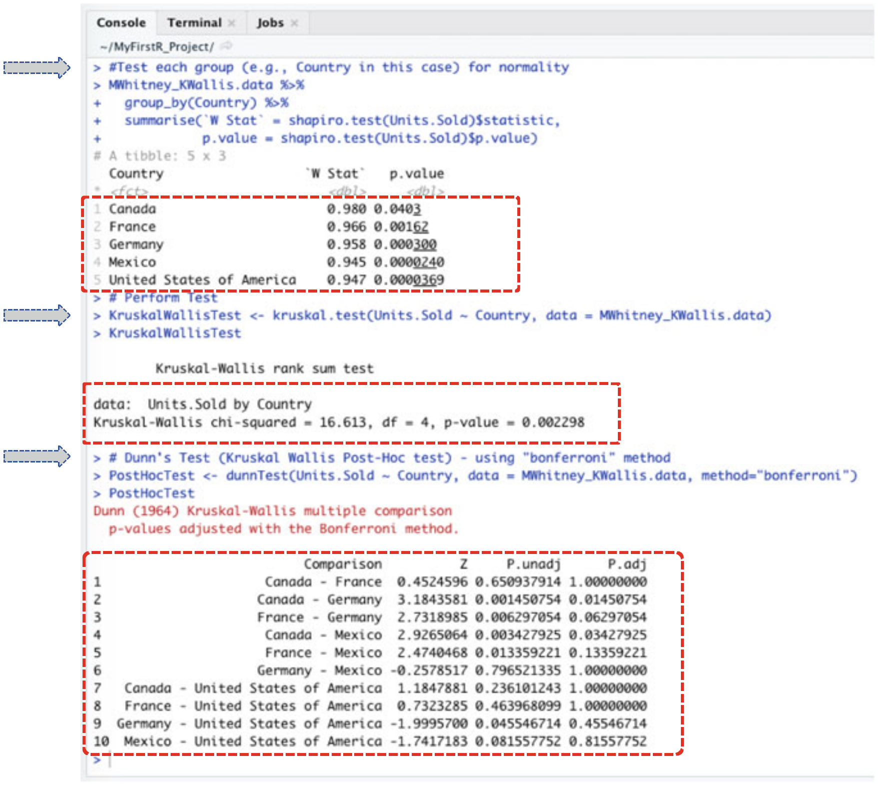
```
# Step 3 - Conduct Kruskal Wallis test (Ordinal data)

#Test each group (e.g., Country in this case) for normality
MWhitney_KWallis.data %>%
  group_by(Country) %>%
  summarise(`W Stat` = shapiro.test(Units.Sold)$statistic,
            p.value = shapiro.test(Units.Sold)$p.value)


# Perform Test
KruskalWallisTest <- kruskal.test(Units.Sold ~ Country, data =
MWhitney_KWallis.data)
KruskalWallisTest


# Dunn's Test (Kruskal Wallis Post-Hoc test) - using "bonferroni" method
PostHocTest <- dunnTest(Units.Sold ~ Country, data =
MWhitney_KWallis.data, method="bonferroni")
PostHocTest
```

*****Note**: As defined in the code above; the authors first conducted a normality test by considering each group (five groups) of the "**Country**" variable before conducting the Kruskal–Wallis H test. This was done in order to confirm that the data does not meet the assumption of normality which is commonly a prerequisite to performing the Kruskal–Wallis H test (non-parametric test) (see: Chaps. 3 and 4).

Once the user have successfully run the codes (Step 3, Lines 25 to 39, Fig. 11.7), they will be presented with the results of the assumption test and method in the Console as represented in Fig. 11.9.

As presented in Fig. 11.9, we conducted a normality test considering the five groups of the "**Country**" variable (Canada, France, Germany, Mexico, United States of America) by taking into account the "**Units.Sold**" using the **shapiro.test( )** function in R. As highlighted in the figure (Fig. 11.9), the result of the assumption test shows that the dataset considering the five groups of the "Country" variable against the "Units.Sold" was not normally distributed (whereby significant level is considered values where $p > 0.05$). Therefore, we assume that the data or targeted variables met the condition to perform the Kruskal–Wallis test with Group A (Canada): showing a normality test statistic of `W=0.980, p-value=0.0403`; Group B (France): `W=0.966, p-value=0.00162`; Group C (Germany): `W=0.958, p-value=0.000300`; Group D (Mexico): `W=0.945, p-value=0.0000240`, and Group E (United States of America): `W=0.947, p-value=0.0000369`, respectively.

Therefore, we proceeded to conduct the Kruskal–Wallis H test considering the independent variable "Country" (with five groups) against the dependent variable "Units.Sold" as contained in the example dataset (MWhitney_KWallis.data). Consequentially, we also performed a post-hoc test using the **DunnTest( )** function in R



```
~/MyFirstR_Project/
> #Test each group (e.g., Country in this case) for normality
> MWhitney_KWallis.data %>%
+   group_by(Country) %>%
+   summarise(`W Stat` = shapiro.test(Units.Sold)$statistic,
+             p.value = shapiro.test(Units.Sold)$p.value)
# A tibble: 5 x 3
  Country                     `W Stat`   p.value
  <fct>                        <dbl>      <dbl>
1 Canada                       0.980 0.0403
2 France                       0.966 0.00162
3 Germany                      0.958 0.000300
4 Mexico                       0.945 0.0000240
5 United States of America     0.947 0.0000369
> # Perform Test
> KruskalWallisTest <- kruskal.test(Units.Sold ~ Country, data = MWhitney_KWallis.data)
> KruskalWallisTest

        Kruskal-Wallis rank sum test

data:  Units.Sold by Country
Kruskal-Wallis chi-squared = 16.613, df = 4, p-value = 0.002298

> # Dunn's Test (Kruskal Wallis Post-Hoc test) - using "bonferroni" method
> PostHocTest <- dunnTest(Units.Sold ~ Country, data = MWhitney_KWallis.data, method="bonferroni")
> PostHocTest
Dunn (1964) Kruskal-Wallis multiple comparison
  p-values adjusted with the Bonferroni method.

                       Comparison          Z      P.unadj      P.adj
1                  Canada - France  0.4524596 0.650937914 1.00000000
2                 Canada - Germany  3.1843581 0.001450754 0.01450754
3                 France - Germany  2.7318985 0.006297054 0.06297054
4                  Canada - Mexico  2.9265064 0.003427925 0.03427925
5                  France - Mexico  2.4740468 0.013359221 0.13359221
6                 Germany - Mexico -0.2578517 0.796521335 1.00000000
7   Canada - United States of America  1.1847881 0.236101243 1.00000000
8   France - United States of America  0.7323285 0.463968099 1.00000000
9  Germany - United States of America -1.9995700 0.045546714 0.45546714
10  Mexico - United States of America -1.7417183 0.081557752 0.81557752
>
```

**Fig. 11.9** Results of Kruskal–Wallis H test and Post-Hoc test displayed in the Console in R

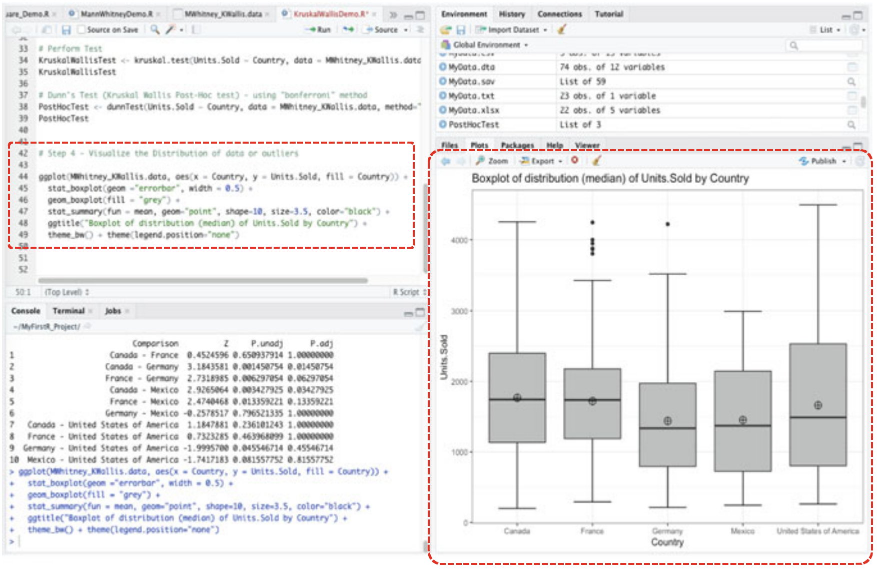
adjusted with the "bonferroni" method. This is due to the fact that the test (Kruskal–Wallis) results came out significant ($p \leq 0.05$) as we will discuss in detail in Step 5 (Results Interpretation).

Accordingly, the results of the Kruskal–Wallis H test and statistics were stored as an R object we called `KruskalWallisTest,` and the post-hoc test stored as `PostHocTest`, respectively (see: Fig. 11.9).

# **Step 4**—Plot and visualize the data distribution (outliers) and results.

In Fig. 11.10 (Step 4, Lines 42 to 49); we utilized the **ggplot( )** function in R to display a boxplot of the distribution (outliers) for the five groups of the "**Country**" variable (i.e., Canada, France, Germany, Mexico, United States of America) plotted against the **"Units.Sold**". As shown in the figure (Fig. 11.10), the difference in the distribution also confirms the significant difference we found in the *H* test statistics (Step 3) as explained in detail in the next Step 5.

The syntax and used to plot or visualize the distribution of the data or outliers is as shown in the code below, and the resultant chart is represented in Fig. 11.10.


```
# Perform Test
KruskalWallisTest <- kruskal.test(Units.Sold ~ Country, data = MWhitney_KWallis.data
KruskalWallisTest

# Dunn's Test (Kruskal Wallis Post-Hoc test) - using "bonferroni" method
PostHocTest <- dunnTest(Units.Sold ~ Country, data = MWhitney_KWallis.data, method="
PostHocTest


# Step 4 - Visualize the Distribution of data or outliers

ggplot(MWhitney_KWallis.data, aes(x = Country, y = Units.Sold, fill = Country)) +
  stat_boxplot(geom ="errorbar", width = 0.5) +
  geom_boxplot(fill = "grey") +
  stat_summary(fun = mean, geom="point", shape=10, size=3.5, color="black") +
  ggtitle("Boxplot of distribution (median) of Units.Sold by Country") +
  theme_bw() + theme(legend.position="none")
```

```
                      Comparison          Z     P.unadj      P.adj
1                Canada - France  0.4524596 0.650937914 1.00000000
2               Canada - Germany  3.1843581 0.001450754 0.01450754
3               France - Germany  2.7318985 0.006297054 0.06297054
4                Canada - Mexico  2.9265064 0.003427925 0.03427925
5                France - Mexico  2.4740468 0.013359221 0.13359221
6               Germany - Mexico -0.2578517 0.796521335 1.00000000
7   Canada - United States of America  1.1847881 0.236101243 1.00000000
8   France - United States of America  0.7323285 0.463968099 1.00000000
9  Germany - United States of America -1.9995700 0.045546714 0.45546714
10  Mexico - United States of America -1.7417183 0.081557752 0.81557752
> ggplot(MWhitney_KWallis.data, aes(x = Country, y = Units.Sold, fill = Country)) +
+   stat_boxplot(geom ="errorbar", width = 0.5) +
+   geom_boxplot(fill = "grey") +
+   stat_summary(fun = mean, geom="point", shape=10, size=3.5, color="black") +
+   ggtitle("Boxplot of distribution (median) of Units.Sold by Country") +
+   theme_bw() + theme(legend.position="none")
> 
```

```
MyData.dta     74 obs. of 12 variables
MyData.sav     List of 59
MyData.txt     23 obs. of 1 variable
MyData.xlsx    22 obs. of 5 variables
PostHocTest    List of 3
```

**Fig. 11.10** Plot representing the distribution of the five groups of the independent variable broken down by Country using the ggplot() function in R

```
# Step 4 - Visualize the Distribution of data or outliers
ggplot(MWhitney_KWallis.data, aes(x = Country, y = Units.Sold, fill =
Country)) +
  stat_boxplot(geom ="errorbar", width = 0.5) +
  geom_boxplot(fill = "grey") +
  stat_summary(fun = mean, geom="point", shape=10, size=3.5, color="black")+
  ggtitle("Boxplot of distribution (median) of Units.Sold by Country") +
  theme_bw() + theme(legend.position="none")
```

**# Step 5**—Results Interpretation (Kruskal–Wallis H).

The final step for the Kruskal–Wallis test and analysis is to interpret and understand the results of the test.

By default, the hypothesis for conducting the test (Kruskal–Wallis) considering the two variables "**Country**" and "**Units.Sold**" (see: Figs. 11.8 and 11.9) is:

- **($H_1$)** *IF* the p-value of the test is less than or equal to 0.05 ($p \leq 0.05$), *THEN* we can assume that there is a difference in the distribution of the groups (Canada, France, Germany, Mexico, United States of America) of the "Country" variable taking into account the dependent variable "Units.Sold". Thus, the *median* of the individual group of population (Canada, France, Germany, Mexico, United States of America) are statistically different (varies). Hence, we would consider to further perform a multiple comparison (Post-Hoc) test to determine where the significant differences may lie across the data or groups or variables.

- **($H_0$)** *ELSE IF* the p-value is greater than 0.05 ($p > 0.05$) *THEN* we can conclude that there is no difference in the *median* of the five groups of the independent variable taking into account the dependent variable "Units.Sold". Thus, the five groups of the independent variable (Canada, France, Germany, Mexico, United States of America) are homogeneous and have the same distribution (i.e., are stochastically equal). And, if this was the case, then we do not need to further conduct a post-hoc test.

```
> KruskalWallisTest

Kruskal-Wallis rank sum test

data:  Units.Sold by Country

Kruskal-Wallis chi-squared = 16.613, df = 4, p-value = 0.002298
```

As shown in the results of the test presented above (see: Fig. 11.9); the meaning of the **Kruskal–Wallis H** test statistic or output can be explained as a list containing the following:

- **Statistics**: `X`$^2$ `(H-statistics) = 16.613` which represents the value of the distribution test.
- **Degrees of freedom**: `df = 4` is the degree of freedom for the *k* (nth) groups of the independent variable.
- **p-value**: `p-value = 0.002298` is the significance level of the test.

Statistically, we can see from the reported result that the p-value is less than the stated significance level (significance, $p \leq 0.05$). Therefore, we reject $H_0$ and accept $H_1$ by concluding that there is a significant difference between the five groups of the "Country" variable (Canada, France, Germany, Mexico, United States of America) taking into account the "Units.Sold".

Consequently, having found a significant difference for the analyzed variable or group of countries (`p-value=0.002298`), we do not know which one or where among the countries the differences may lie. Therefore, a post-hoc (multiple comparison) test needs to be conducted, in this case, in order to establish this fact or variations.

```
> PostHocTest
Dunn (1964) Kruskal-Wallis multiple comparison
  p-values adjusted with the Bonferroni method.

                         Comparison          Z      P.unadj      P.adj
1                  Canada - France  0.4524596 0.650937914 1.00000000
2                 Canada - Germany  3.1843581 0.001450754 0.01450754
3                 France - Germany  2.7318985 0.006297054 0.06297054
4                  Canada - Mexico  2.9265064 0.003427925 0.03427925
5                  France - Mexico  2.4740468 0.013359221 0.13359221
6                 Germany - Mexico -0.2578517 0.796521335 1.00000000
7   Canada - United States of America  1.1847881 0.236101243 1.00000000
8   France - United States of America  0.7323285 0.463968099 1.00000000
9  Germany - United States of America -1.9995700 0.045546714 0.45546714
10  Mexico - United States of America -1.7417183 0.081557752 0.81557752
```

As gathered in the above results of the post-hoc (multiple comparison) test using the **DunnTest( )** method adjusted with the "Bonferroni" method in R (see Fig. 11.9); we can now see where among the individual countries (Canada, France, Germany, Mexico, United States of America) the statistical differences we observed lies. For example, we can see that the difference in distribution was exceptionally observed for **`Canada-Germany`** `(Z=3.1843581, P.unadj=0.001450754, P.adj=0.01450754;` **`Canada-Mexico`** `(Z=2.9265064, P.unadj=0.003427925, P.adj=0.03427925).` **`France-Germany`** was also slightly significant with `Z=2.7318985, P.unadj=0.006297054, P.adj =0.06297054`, respectively.

## 11.4 Summary

In this chapter, the authors explained in detail and practically demonstrated to the readers how to conduct the two most commonly used types of *non-parametric* (or distribution free) tests (Mann–Whitney U and Kruskal–Wallis H) used by the researchers to compare the *median* of "non-normally" distributed data samples in R. In Sect. 11.2, we illustrated how to conduct the Mann–Whitney U test. While in Sect. 11.3 we looked at how to perform the Kruskal–Wallis H test using R.

Also, the chapter covered how to graphically plot the median or distribution (outliers) of the two types of tests (Mann–Whitney U and Kruskal–Wallis H), and then discussed in detail how to interpret and understand the results of the tests in R.

In summary, the main contents covered in this chapter includes:

- Mann–Whitney (U-Statistics) test is a statistical test of hypothesis used to compare the *distribution (in median)* of data samples that are represented in "two independent comparison groups" (usually in ordinal form) and an ordinal or continuous dependent variable.
- Kruskal–Wallis (H-Statistics) test is, on the other hand, applied to compare the *distribution (in median)* of data samples that are represented in "three or more

independent comparison groups" (ordinal form) and an ordinal or continuous dependent variable.
- Mann–Whitney U test is the non-parametric version or alternative (equivalent) to the Independent Sample *t*-test.
- Kruskal–Wallis H test is the non-parametric version or alternative (equivalent) to the One-way ANOVA test.

When choosing whether to conduct a Mann–Whitney U test or Kruskal–Wallis H test? The researcher or data analyst should:

- Perform the "*Mann–Whitney U*" test if the *two groups* come from a single *independently* sampled population, and the distribution of the data sample has been statistically measured or determined to be non-normally distributed.
- Perform the "*Kruskal–Wallis H*" test if the targeted independent variable has *more than two groups* (i.e., minimum of three or more categories), comes from or is drawn from a single *independently* sampled population, and the distribution of the data sample has been statistically measured or determined to be non-normally distributed.
- Perform a post-hoc test (a multiple comparison test) if the result of the Kruskal–Wallis H statistics has shown or appeared to be significant (i.e., $p \leq 0.05$). This is done in order to determine where the significant differences among the groups "$k$ ($n^{th}$)" (minimum of three groups of the independent variable) may lie across the data.

## References

Frey, B. B. (2018). Kruskal–Wallis Test. In *The SAGE Encyclopedia of Educational Research, Measurement, and Evaluation.* SAGE Publications, Inc. https://doi.org/10.4135/9781506326139.n377.

Kruskal, W. H., & Wallis, W. A. (1952). Use of Ranks in One-Criterion Variance Analysis. In *Source: Journal of the American Statistical Association* (Vol. 47, Issue 260).

le Cessie, S., Goeman, J. J., & Dekkers, O. M. (2020). Who is afraid of non-normal data? Choosing between parametric and non-parametric tests. In *European journal of endocrinology* (Vol. 182, Issue 2, pp. E1–E3). NLM (Medline). https://doi.org/10.1530/EJE-19-0922.

MacFarland, T. W., Yates, J. M., MacFarland, T. W., & Yates, J. M. (2016). Mann–Whitney U Test. In *Introduction to Nonparametric Statistics for the Biological Sciences Using R* (pp. 103–132). Springer International Publishing. https://doi.org/10.1007/978-3-319-30634-6_4.

Mann, H. B., & Whitney, D. R. (1947). On a test of whether one of two random variables is stochastically larger than the other. *The Annals of Mathematical Statistics, 18*(1), 50–60. https://doi.org/10.1214/aoms/1177730491

McKight, P. E., & Najab, J. (2010). Kruskal-Wallis Test. In *The Corsini Encyclopedia of Psychology* (pp. 1–1). John Wiley & Sons, Inc. https://doi.org/10.1002/9780470479216.corpsy0491.

McKnight, P. E., & Najab, J. (2010). Mann-Whitney U Test. In *The Corsini Encyclopedia of Psychology* (pp. 1–1). John Wiley & Sons, Inc. https://doi.org/10.1002/9780470479216.corpsy0524.

Nachar, N. (2008). The Mann-Whitney U: A Test for Assessing Whether Two Independent Samples Come from the Same Distribution. In *Tutorials in Quantitative Methods for Psychology* (Vol. 4, Issue 1).

Okoye, K., Arrona-Palacios, A., Camacho-Zuñiga, C., Achem, J. A. G., Escamilla, J., & Hosseini, S. (2022). Towards teaching analytics: a contextual model for analysis of students' evaluation of teaching through text mining and machine learning classification. *Education and Information Technologies 2021 27:3*, *27*(3), 3891–3933. https://doi.org/10.1007/S10639-021-10751-5.

Ortega, C. (2023). *Kruskal-Wallis test: What it is, advantages and how it is performed*. QuestionPro. https://www.questionpro.com/blog/es/prueba-de-kruskal-wallis/.

Ostertagová, E., Ostertag, O., & Kováč, J. (2014). Methodology and application of the Kruskal-Wallis test. *Applied Mechanics and Materials, 611*, 115–120. https://doi.org/10.4028/www.scientific.net/AMM.611.115

Ramtin, S. (2023). How to choose appropriate bivariate test. In A. E. M. Eltorai, J. A. Bakal, S. F. DeFroda, & B. D. Owens (Eds.), *Translational Sports Medicine* (pp. 145–149). Academic Press. https://doi.org/10.1016/B978-0-323-91259-4.00115-6.

Vargha, A., & Delaney, H. D. (1998). The Kruskal-Wallis Test and Stochastic Homogeneity. *Journal of Educational and Behavioral Statistics, 23*(2), 170–192. https://doi.org/10.3102/10769986023002170

# Chapter 12
# Correlation Tests in R: Pearson Cor, Kendall's Tau, and Spearman's Rho

## 12.1 Introduction

*Correlation* (Cor) is a statistical procedure or method used by researchers or the data analysts to evaluate the *strength* or *degree* of relationship between two variables (continuous or categorical) (Privitera, 2023; Schober & Schwarte, 2018). Statistically, the correlation test can be defined as a "bivariate analysis" that measures the strength of association or relationship between two variables or datasets and the direction of the relationship (see Chap. 6, Sect. 6.2.9). The result of the test (usually for linearity or strength of association) between the datasets or data points (depending on the type of correlation method being used or applied and usually determined through the *p*-values: where $p \leq 0.05$) means that a high correlation statistics indicates that the variables or data being measured have a strong relationship between each other. On the other hand, a weak correlation ($p > 0.05$) signifies that the variables are barely (insignificantly) related or associated.

Thus, with correlated datasets, it is assumed that a change in the magnitude of one variable is statistically associated with a change in the magnitude of another variable that it is being measured against, be it in the same direction (positive correlation) or in the opposite direction (negative correlation) (Akoglu, 2018; Privitera, 2023; Schober & Schwarte, 2018).

According to Akoglu (2018), the correlation (relationship, association) between the two specified variables is denoted by the letter $r$ and quantified through a number, that varies between $-1$ and $+1$ (denoting the negative and positive correlations, respectively). Whereby, a value of zero (0) implies that there is no correlation between the variables, and a value of one (1) denotes an absolute (perfect) correlation. Therefore, whereas $r$ represents the direction of the correlation, a positive $r$ signifies that the measured variables are *certainly* (positively) related, while a negative $r$ signifies that the measured variables are *inversely* (negatively) related. Statistically, the strength of the correlation increases both from 0 to $+1$, and from 0 to $-1$, respectively (Akoglu, 2018).

K. Okoye and S. Hosseini, *R Programming*,
https://doi.org/10.1007/978-981-97-3385-9_12

There are three main types of correlation analysis commonly applied by the researchers, in theory. These are (i) Pearson product–moment correlation, (ii) Kendall's tau correlation, and (iii) Spearman's rho correlation (Akoglu, 2018; Brossart et al., 2018; Hauke & Kossowski, 2011; Puth et al., 2014; Schober & Schwarte, 2018; Wang et al., 2019; Zar, 2014).

*Pearson correlation* (also known as Pearson product–moment correlation coefficient) is described as a parametric test that measures the strength of linear association (linear trend) that exists between two continuous variables. Statistically, the method (Pearson correlation, denoted by $r$) draws a "line of best fit" through the two datasets or variables by establishing how far away the two data points are to the drawn line (model) of best fit.

Mathematically, to apply the Pearson's statistics by measuring the two quantities or variables $X$ and $Y$ on each of $N$ individuals in order to produce a data set of $X_1$, $Y_1, \ldots, X_N, Y_N$ (Puth et al., 2014), the formula to calculate the correlation coefficient is given as:

$$\text{Cor}\,(r) = \frac{N\sum xy - \left(\sum x\right)\left(\sum y\right)}{\sqrt{\left[N\sum x^2 - \left(\sum x\right)^2\right]\left[N\sum y^2 - \left(\sum y\right)^2\right]}}$$

whereby

| | |
|---|---|
| $N$ | the number of pairs of scores |
| $\Sigma xy$ | the sum of the products of paired scores |
| $\Sigma x$ | the sum of $x$ scores |
| $\Sigma y$ | the sum of $y$ scores |
| $\Sigma x^2$ | the sum of squared $x$ scores |
| $\Sigma y^2$ | the sum of squared $y$ scores |

Just like many of the other existing types of *parametric* procedures or statistical methods (see Chap. 4), the Pearson's product–moment correlation coefficient requires the assumption that the relationship between the variables is *linear* and is measured on an interval (continuous) scale. Thus, the researchers or data analysts must check that the following below assumptions are met before applying or using the Pearson correlation.

**Pearson's Correlation Assumptions**

- Independence: the drawn dataset or sample must be independent to each other.
- Linearity: the two tested variables should be linearly related to each other, e.g., when plotted in a graph should result in a moderately straight line.
- Normality: the dataset must be normally distributed, i.e., should produce a bell-shaped graph when the means of the samples are plotted.
- Homoscedasticity or equality of variances must be present.

Furthermore, on the other hand, *Kendall's tau* correlation (also known as Kendall rank correlation coefficient) is a non-parametric test (i.e., an alternative to Pearson's correlation) mainly used by the researchers to measure the *strength of dependence* between two *categorical* or *ordinal* variables. According to Couso et al. (2018), the method (Kendall's tau) can be applied as an efficient and robust way of identifying monotone relationships between two data sequences, although when applied to digital data (e.g., discrete or discontinuous format), the high number of ties could produce inconsistent results due to quantization.

Theoretically, the Kendall's tau ($\tau$) statistics symbolizes the degree of agreement between two specified "ordinal" variables by indicating how similarly the two variables order a set of individuals or data points (Brossart et al., 2018). Thus, mathematically, the following formula is used to calculate the value of Kendall's tau statistics or rank correlation coefficient:

$$\text{Kendall's tau } (\tau) = \frac{C - D}{C + D} \text{ or } \frac{n_c - n_d}{\frac{1}{2}n(n-1)}$$

whereby

$n_c$ number of concordant, i.e., ordered in the same way.
$n_d$ Number of discordant, i.e., ordered differently.

With the Kendall's tau statistic, commonly calculated through pairwise comparison; a value of $\tau_{(X,Y)} = +1$ means that the data points for the two (ordinal) variables ($X$ and $Y$) are ordered in exactly the same way, i.e., occupies the same rank position. While on the other hand, a value of $\tau_{(X,Y)} = -1$ implies that the data points for the two variables are ordered in exactly the opposite way, with one data point occupying the first rank in one variable and the last rank in the other variable. Accordingly, a value of $\tau_{(X,Y)} = 0$ indicates that there is no relationship in the way or order that the two variables are ranked considering the data points, thus, are independent (Brossart et al., 2018).

In the same vein or similar manner, just like the Kendall's tau correlation, *Spearman's rho* correlation (also known as Spearman rank correlation coefficient) is another type of non-parametric (i.e., alternative to Pearson correlation) test used by the researchers to measure the degree of association between two (ordinal) variables. The method can also be applied for interval or ratio datasets provided the datasets are found to be distribution-free. Mathematically, the following formula is used to calculate the value of the Spearman's rho statistics or rank correlation coefficient:

$$\text{Spearman's rho } (\rho) = 1 - \frac{6\sum\left(d_i^2\right)}{n\left(n^2 - 1\right)}$$

whereby

$n$ number of data points of the two variables ($x$ and $y$).
$d_i$ rank difference of element "$n$", i.e., difference between the corresponding statistics of order of $x - y$.

The only difference between the *Spearman's rho* versus *Kendall's tau* method is that while the Spearman's rho ($\rho$) statistics or results are calculated through the "ordinary least squares", the Kendall's tau ($\tau$) statistics is calculated through the "pairwise comparison" of all the data points (Brossart et al., 2018). Thus, whilst the Kendall's tau ($\tau$) statistics are based on "concordant and discordant pairs", the Spearman's rho ($\rho$) statistics are based on "deviations".

It is also noteworthy to mention that Spearman's rho ($\rho$) method is much more sensitive to error and handling discrepancies in data samples than the Kendall's tau ($\tau$) method, which, on the other hand, are more accurate with smaller sample sizes than the Spearman's rho ($\rho$).

In any case, a lot of the time the interpretations of the two methods (Kendall's tau and Spearman's rho) are very similar, thus, tend to invariably lead to the same inferences or statistical results.

Also, unlike Pearson correlation, both methods (Kendall's tau and Spearman's rho) do not require the available data or sample to meet the assumption that the relationship between the considered variables is linear (i.e., when plotted does not necessarily need to result in a moderately straight line), or normally distributed (i.e., distribution-free), nor does it require the measurement scale of the variables to be represented on a continuous or interval scale.

Table 12.1 is a summary of the differences and similarities between the Pearson cor, Kendall's tau, and Spearman's rho Correlation tests including the conditions that are required to perform the different tests, which the authors will be demonstrating using R in the next sections (Sect. 12.2 and 12.3) of this chapter.

In the next sections of this chapter (Sects. 12.2 and 12.3), the authors will be demonstrating to the readers how to conduct the Pearson cor, Kendall's tau, and Spearman's rho correlation tests in R, harmoniously. We will illustrate the different steps to performing the three types of tests in R using the following steps outlined in Fig. 12.1.

## 12.2 Pearson Correlation Test in R

*Pearson correlation* measures the strength of linear association (correlation) that exists between two "continuous" variables. Thus, it calculates the effect of change (be it positive or negative) in one variable when the other variable changes.

By default, the hypothesis for testing whether there is a *correlation* (measure of linearity or association) between the two given set of (continuous) variables is; *IF* the *p*-value of the test is less than or equal to 0.05 ($p \leq 0.05$), *THEN* we assume that there is a statistically significant strong relationship between the two analyzed variables and that this is not by chance ($H_1$). *ELSE IF* the *p*-value is greater than 0.05 ($p > 0.05$) *THEN* we can conclude that there is no significant relationship between the two variables, and any observed association could only have occurred by chance ($H_0$).

**Table 12.1** Differences and similarities between the Pearsoncor, Kendall's tau, and Spearman's rho correlation tests and assumptions

| Pearson | Kendall's tau | Spearman's rho |
|---|---|---|
| Data sample should be independently drawn from the population | Data sample should be independently drawn from the population | Data sample should be independently drawn from the population |
| Used for continuous (intervalor ratio) datasets | Used for categorical (ranked or ordinal) datasets. Although can also be applied to intervalor ratio datasets | Used for categorical (ranked or ordinal) datasets. Although can also be applied to intervalor ratio datasets |
| Data sample or observations must be normally distributed, i.e., bell-shaped | Data samples are distribution-free, thus, are not normally distributed, i.e., skewed | Data samples are distribution-free, thus, are not normally distributed, i.e., skewed |
| Calculated by measuring the "average weight" of the two variables (i.e., covariance of the two variables divided by the product of their standard deviations) | Calculated through the "pairwise comparison" of the data points based on concordant and discordant pairs | Calculated through "ordinary least squares" based on deviations |
| Described as parametric test for linearity or relationship between two variables | Non-parametric test for strength of dependence between two variables | Non-parametric test to measure the degree of association between two variables |

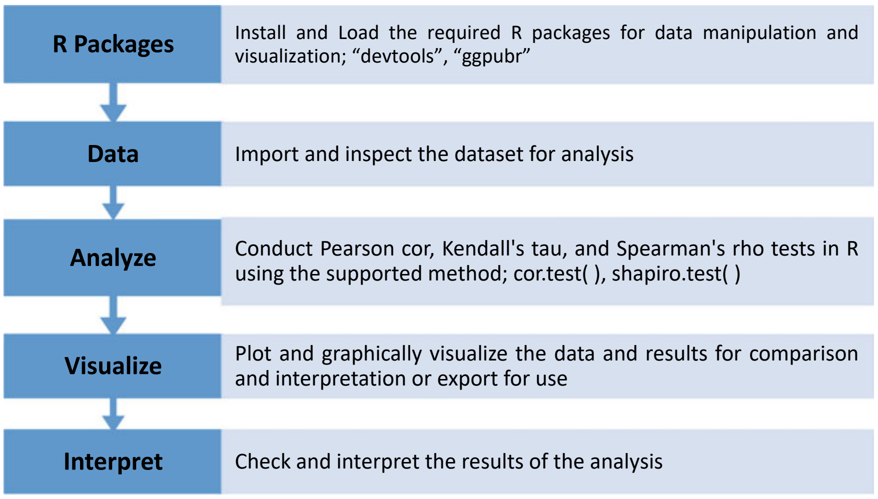
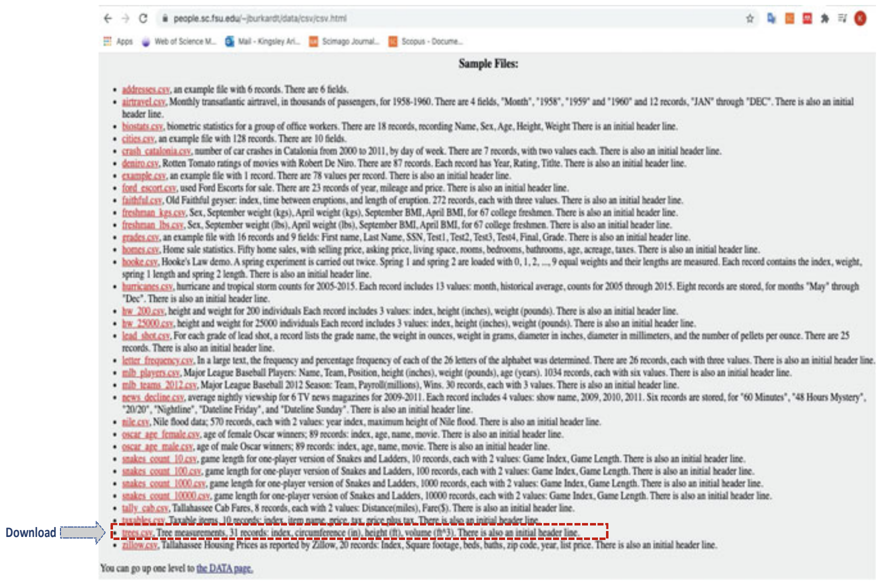

R Packages
Install and Load the required R packages for data manipulation and visualization; "devtools", "ggpubr"
Data
Import and inspect the dataset for analysis
Analyze
Conduct Pearson cor, Kendall's tau, and Spearman's rho tests in R using the supported method; cor.test( ), shapiro.test( )
Visualize
Plot and graphically visualize the data and results for comparison and interpretation or export for use
Interpret
Check and interpret the results of the analysis

**Fig. 12.1** Steps to conducting the Pearson cor, Kendall's tau, and Spearman's rho correlation tests in R



**Sample Files:**

- addresses.csv, an example file with 6 records. There are 6 fields.
- airtravel.csv, Monthly transatlantic airtravel, in thousands of passengers, for 1958-1960. There are 4 fields, "Month", "1958", "1959" and "1960" and 12 records, "JAN" through "DEC". There is also an initial header line.
- biostats.csv, biometric statistics for a group of office workers. There are 18 records, recording Name, Sex, Age, Height, Weight There is an initial header line.
- cities.csv, an example file with 128 records. There are 10 fields.
- crash_catalonia.csv, number of car crashes in Catalonia from 2000 to 2011, by day of week. There are 7 records, with two values each. There is also an initial header line.
- deniro.csv, Rotten Tomato ratings of movies with Robert De Niro. There are 87 records. Each record has Year, Rating, Titlte. There is also an initial header line.
- example.csv, an example file with 1 record. There are 78 values per record. There is also an initial header line.
- ford_escort.csv, used Ford Escorts for sale. There are 23 records of year, mileage and price. There is also an initial header line.
- faithful.csv, Old Faithful geyser: index, time between eruptions, and length of eruption. 272 records, each with three values. There is also an initial header line.
- freshman_kgs.csv, Sex, September weight (kgs), April weight (kgs), September BMI, April BMI, for 67 college freshmen. There is also an initial header line.
- freshman_lbs.csv, Sex, September weight (lbs), April weight (lbs), September BMI, April BMI, for 67 college freshmen. There is also an initial header line.
- grades.csv, an example file with 16 records and 9 fields: First name, Last Name, SSN, Test1, Test2, Test3, Test4, Final, Grade. There is also an initial header line.
- homes.csv, Home sale statistics. Fifty home sales, with selling price, asking price, living space, rooms, bedrooms, bathrooms, age, acreage, taxes. There is also an initial header line.
- hooke.csv, Hooke's Law demo. A spring experiment is carried out twice. Spring 1 and spring 2 are loaded with 0, 1, 2, ..., 9 equal weights and their lengths are measured. Each record contains the index, weight, spring 1 length and spring 2 length. There is also an initial header line.
- hurricanes.csv, hurricane and tropical storm counts for 2005-2015. Each record includes 13 values: month, historical average, counts for 2005 through 2015. Eight records are stored, for months "May" through "Dec". There is also an initial header line.
- hw_200.csv, height and weight for 200 individuals Each record includes 3 values: index, height (inches), weight (pounds). There is also an initial header line.
- hw_25000.csv, height and weight for 25000 individuals Each record includes 3 values: index, height (inches), weight (pounds). There is also an initial header line.
- lead_shot.csv, For each grade of lead shot, a record lists the grade name, the weight in ounces, weight in grams, diameter in inches, diameter in millimeters, and the number of pellets per ounce. There are 25 records. There is also an initial header line.
- letter_frequency.csv, In a large text, the frequency and percentage frequency of each of the 26 letters of the alphabet was determined. There are 26 records, each with three values. There is also an initial header line.
- mlb_players.csv, Major League Baseball Players: Name, Team, Position, height (inches), weight (pounds), age (years). 1034 records, each with six values. There is also an initial header line.
- mlb_teams_2012.csv, Major League Baseball 2012 Season: Team, Payroll(millions), Wins. 30 records, each with 3 values. There is also an initial header line.
- news_decline.csv, average nightly viewship for 6 TV news magazines for 2009-2011. Each record includes 4 values: show name, 2009, 2010, 2011. Six records are stored, for "60 Minutes", "48 Hours Mystery", "20/20", "Nightline", "Dateline Friday", and "Dateline Sunday". There is also an initial header line.
- nile.csv, Nile flood data; 570 records, each with 2 values: year index, maximum height of Nile flood. There is also an initial header line.
- oscar_age_female.csv, age of female Oscar winners; 89 records: index, age, name, movie. There is also an initial header line.
- oscar_age_male.csv, age of male Oscar winners; 89 records: index, age, name, movie. There is also an initial header line.
- snakes_count_10.csv, game length for one-player version of Snakes and Ladders, 10 records, each with 2 values: Game Index, Game Length. There is also an initial header line.
- snakes_count_100.csv, game length for one-player version of Snakes and Ladders, 100 records, each with 2 values: Game Index, Game Length. There is also an initial header line.
- snakes_count_1000.csv, game length for one-player version of Snakes and Ladders, 1000 records, each with 2 values: Game Index, Game Length. There is also an initial header line.
- snakes_count_10000.csv, game length for one-player version of Snakes and Ladders, 10000 records, each with 2 values: Game Index, Game Length. There is also an initial header line.
- tally_cab.csv, Tallahassee Cab Fares, 8 records, each with 2 values: Distance(miles), Fare($). There is also an initial header line.
- taxables.csv, Taxable items, 10 records: index, item name, price, tax, price plus tax. There is also an initial header line.
- trees.csv, Tree measurements, 31 records: index, circumference (in), height (ft), volume (ft^3). There is also an initial header line.
- zillow.csv, Tallahassee Housing Prices as reported by Zillow, 20 records: Index, Square footage, beds, baths, zip code, year, list price. There is also an initial header line.

You can go up one level to the DATA page.

**Fig. 12.2** Example of CSV file download (*Source* https://people.sc.fsu.edu/~jburkardt/data/csv/csv.html)

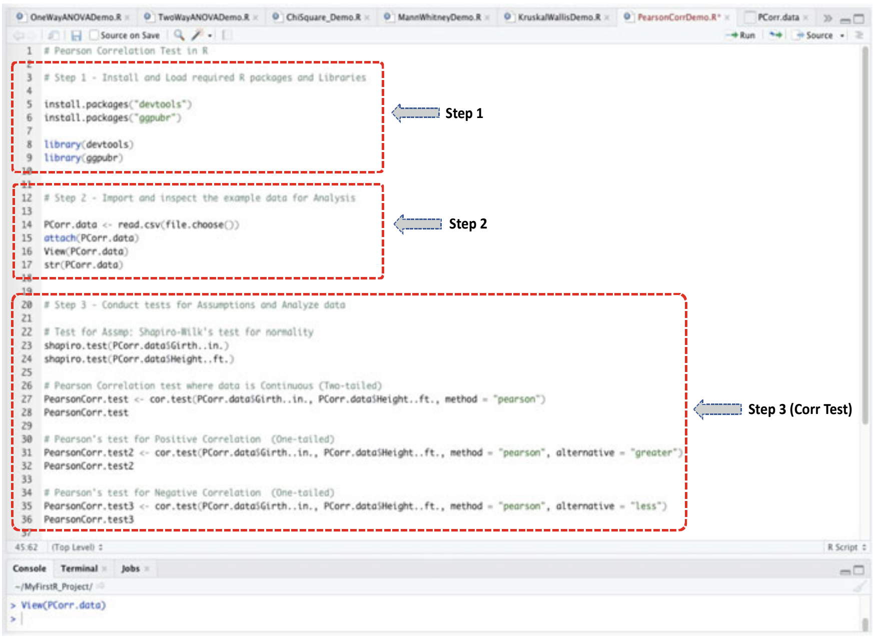
Here, the authors will demonstrate to the readers how to conduct the Pearson correlation test in R using the **cor.test( )** function in R. We will do this using the steps outlined in Fig. 12.1.

To begin, **Open RStudio** and **Create a new or Open an existing project**. Once the user has the RStudio and an R Project opened, **Create a new R Script** and name it "**PearsonCorrDemo**" or any name the user may preferentially choose (see Chap. 1 and 2 if the user needs to refresh on how to do this step).

Now, we are going to download an example file or dataset that we will use to demonstrate the Pearson correlation test (the users are welcome to use any dataset or format if they wish to do so).

As shown in Fig. 12.2, download the example CSV dataset named "**trees.csv**" via the following source: https://people.sc.fsu.edu/~jburkardt/data/csv/csv.html and save the file on the users' local machine or computer. *** The users can also access the list of example datasets used in this book at the following repository (https://doi.org/10.6084/m9.figshare.24728073) to download the example CSV file.

Once the user has successfully downloaded and saved the example file (trees.csv) on the computer, we can proceed to conduct the Pearson Correlation test in R.



```
1  # Pearson Correlation Test in R
2
3  # Step 1 - Install and Load required R packages and Libraries
4
5  install.packages("devtools")
6  install.packages("ggpubr")
7
8  library(devtools)
9  library(ggpubr)
10
11
12 # Step 2 - Import and inspect the example data for Analysis
13
14 PCorr.data <- read.csv(file.choose())
15 attach(PCorr.data)
16 View(PCorr.data)
17 str(PCorr.data)
18
19
20 # Step 3 - Conduct tests for Assumptions and Analyze data
21
22 # Test for Assmp: Shapiro-Wilk's test for normality
23 shapiro.test(PCorr.data$Girth..in.)
24 shapiro.test(PCorr.data$Height..ft.)
25
26 # Pearson Correlation test where data is Continuous (Two-tailed)
27 PearsonCorr.test <- cor.test(PCorr.data$Girth..in., PCorr.data$Height..ft., method = "pearson")
28 PearsonCorr.test
29
30 # Pearson's test for Positive Correlation  (One-tailed)
31 PearsonCorr.test2 <- cor.test(PCorr.data$Girth..in., PCorr.data$Height..ft., method = "pearson", alternative = "greater")
32 PearsonCorr.test2
33
34 # Pearson's test for Negative Correlation  (One-tailed)
35 PearsonCorr.test3 <- cor.test(PCorr.data$Girth..in., PCorr.data$Height..ft., method = "pearson", alternative = "less")
36 PearsonCorr.test3
37
```

```
> View(PCorr.data)
> 
```

**Fig. 12.3** Steps to conducting Pearson correlation test in R

## # Step 1—Install and Load the Required R Packages and Libraries

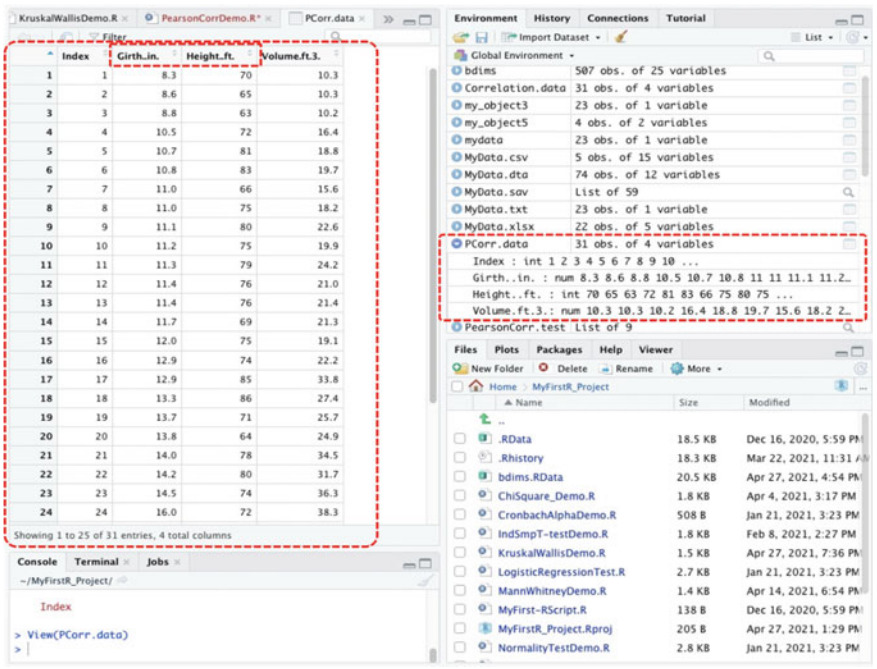
**Install** and **Load** the following *R packages* and *libraries* (see Fig. 12.3, Step 1, Lines 3–9) that will be used to call the different R functions, data manipulations, and graphical visualizations for the Pearson Correlation test.

The syntax and code to install and load the required R packages and libraries are as follows:

```
install.packages("devtools")
install.packages("ggpubr")


library(devtools)
library(ggpubr)
```

## # Step 2—Import and Inspect the Example Dataset for Pearson Correlation Analysis

As illustrated in Step 2 in Fig. 12.3 (Lines 12–17), import the dataset named "**trees.csv**" that we downloaded earlier and store this as an R object named "**PCorr.data**" in R (the users are welcome to use any name they may choose if they wish to do so).



| | Index | Girth..in. | Height..ft. | Volume.ft.3. |
|---|---|---|---|---|
| 1 | 1 | 8.3 | 70 | 10.3 |
| 2 | 2 | 8.6 | 65 | 10.3 |
| 3 | 3 | 8.8 | 63 | 10.2 |
| 4 | 4 | 10.5 | 72 | 16.4 |
| 5 | 5 | 10.7 | 81 | 18.8 |
| 6 | 6 | 10.8 | 83 | 19.7 |
| 7 | 7 | 11.0 | 66 | 15.6 |
| 8 | 8 | 11.0 | 75 | 18.2 |
| 9 | 9 | 11.1 | 80 | 22.6 |
| 10 | 10 | 11.2 | 75 | 19.9 |
| 11 | 11 | 11.3 | 79 | 24.2 |
| 12 | 12 | 11.4 | 76 | 21.0 |
| 13 | 13 | 11.4 | 76 | 21.4 |
| 14 | 14 | 11.7 | 69 | 21.3 |
| 15 | 15 | 12.0 | 75 | 19.1 |
| 16 | 16 | 12.9 | 74 | 22.2 |
| 17 | 17 | 12.9 | 85 | 33.8 |
| 18 | 18 | 13.3 | 86 | 27.4 |
| 19 | 19 | 13.7 | 71 | 25.7 |
| 20 | 20 | 13.8 | 64 | 24.9 |
| 21 | 21 | 14.0 | 78 | 34.5 |
| 22 | 22 | 14.2 | 80 | 31.7 |
| 23 | 23 | 14.5 | 74 | 36.3 |
| 24 | 24 | 16.0 | 72 | 38.3 |

Showing 1 to 25 of 31 entries, 4 total columns

```
~/MyFirstR_Project/

    Index

> View(PCorr.data)
>
```

```
bdims              507 obs. of 25 variables
Correlation.data   31 obs. of 4 variables
my_object3         23 obs. of 1 variable
my_object5         4 obs. of 2 variables
mydata             23 obs. of 1 variable
MyData.csv         5 obs. of 15 variables
MyData.dta         74 obs. of 12 variables
MyData.sav         List of 59
MyData.txt         23 obs. of 1 variable
MyData.xlsx        22 obs. of 5 variables
PCorr.data         31 obs. of 4 variables
   Index : int 1 2 3 4 5 6 7 8 9 10 ...
   Girth..in. : num 8.3 8.6 8.8 10.5 10.7 10.8 11 11 11.1 11.2…
   Height..ft. : int 70 65 63 72 81 83 66 75 80 75 ...
   Volume.ft.3.: num 10.3 10.3 10.2 16.4 18.8 19.7 15.6 18.2 2…
PearsonCorr.test   List of 9
```

| Name | Size | Modified |
|---|---|---|
| .. | | |
| .RData | 18.5 KB | Dec 16, 2020, 5:59 PM |
| .Rhistory | 18.3 KB | Mar 22, 2021, 11:31 AM |
| bdims.RData | 20.5 KB | Apr 27, 2021, 4:54 PM |
| ChiSquare_Demo.R | 1.8 KB | Apr 4, 2021, 3:17 PM |
| CronbachAlphaDemo.R | 508 B | Jan 21, 2021, 3:23 PM |
| IndSmpT-testDemo.R | 1.8 KB | Feb 8, 2021, 2:27 PM |
| KruskalWallisDemo.R | 1.5 KB | Apr 27, 2021, 7:36 PM |
| LogisticRegressionTest.R | 2.7 KB | Jan 21, 2021, 3:23 PM |
| MannWhitneyDemo.R | 1.4 KB | Apr 14, 2021, 6:54 PM |
| MyFirst-RScript.R | 138 B | Dec 16, 2020, 5:59 PM |
| MyFirstR_Project.Rproj | 205 B | Apr 27, 2021, 1:29 PM |
| NormalityTestDemo.R | 2.8 KB | Jan 21, 2021, 3:23 PM |

**Fig. 12.4** Example of CSV dataset imported and stored as R object in R

Once the user has successfully imported the dataset, you will be able to view the details of the **trees.csv** dataset as shown in Fig. 12.4 with 31 observations and 4 variables in the data sample.

The syntax and code to import and save the data in R is shown below:

```
PCorr.data <- read.csv(file.choose())
attach(PCorr.data)
View(PCorr.data)
str(PCorr.data)
```

## # Step 3—Conduct Tests for Assumptions and Analyze Data

Now that we have successfully imported the example dataset and stored this in an R object we called "**PCorr.data**", we can proceed to analyze the data.

As defined in Fig. 12.3 (Step 3, Lines 20–36), first we will conduct the tests of assumptions (data normality) (see: Lines 22–24) as discussed earlier in Sect. 12.1 by using the **shapiro.test( )** method, and then perform the **Pearson Correlation** test if all the necessary conditions to conduct the test are met using the **cor.test( )** function in R, respectively (see: Fig. 12.3, Step 3, Lines 26–36).

Also, as defined earlier in the Introduction section (Sect. 12.1);

- **Pearson's correlation** statistics checks whether there exists a *linear relationship* between two independently sampled variables or data.
- The targeted variables must be continuous data type.

To illustrate the above defined tests using the example dataset we stored as "**PCorr.data**" in R (see: highlighted columns in Fig. 12.4):

1. We will test whether there exists a relationship (correlation) between the **Girth..in.** and **Height..ft.** variables of the trees example data? (**two-tailed test**).
2. Also, we will check whether the correlation (if there exist any) is a *positive* or *negative* (direction) correlation? (**one-tailed test**).

The syntax and code to performing the above tests in R is as shown in the codes below (see: Fig. 12.3, Step 3, Lines 20–36):

```
# Test for Assmp: Shapiro-Wilk's test for normality
  shapiro.test(PCorr.data$Girth..in.)
  shapiro.test(PCorr.data$Height..ft.)


# Pearson Correlation test where data is Continuous (Two-tailed)
PearsonCorr.test <- cor.test(PCorr.data$Girth..in.,
PCorr.data$Height..ft., method = "pearson")
PearsonCorr.test


# Pearson's test for Positive Correlation (One-tailed)
PearsonCorr.test2 <- cor.test(PCorr.data$Girth..in.,
PCorr.data$Height..ft., method = "pearson", alternative = "greater")
PearsonCorr.test2


# Pearson's test for Negative Correlation (One-tailed)
PearsonCorr.test3 <- cor.test(PCorr.data$Girth..in.,
PCorr.data$Height..ft., method = "pearson", alternative = "less")
PearsonCorr.test3
```

## Useful Tips

- The users should always use the `alternative = "greater"` and `alternative = "less"` options to specify the "positive" and "negative" (direction) correlation tests (one-tailed), respectively.

Once the user has successfully run the codes as defined in the **Step 3** in Fig. 12.3 (Lines 20–36); they will be presented with the results of the "tests for assumptions" and the "Pearson Correlation" tests in the Console as shown in Fig. 12.5a and b, respectively.

In Fig. 12.5a, we conducted the test for assumption (data normality) necessary for the Pearson correlation test or parametric methods. This is done in order to determine if the targeted variables (i.e., **Girth..in.** and **Height..ft.**) are fitting and valid for the test (Pearson correlation, a parametric test) (see Chap. 4).

As highlighted in the figure (Fig. 12.5a); we can see that the *normality test* by using the Shapiro–Wilk's method **shapiro.test( ),** where we assume a value of $p > 0.05$ is normal, shows that the distribution of the two variables (**Girth..in.** and **Height..ft.**) are normal, with **Girth..in.** variable showing a significant value of `p-value=0.08893 (W=0.94117)` and **Height..ft.** showing significant value of `p-value=0.4034 (W=0.96545)`, respectively.

Therefore, with the necessary conditions met, we proceeded to conduct the "Pearson Correlation" as defined in the Step 3 (Fig. 12.3) and the results reported in Fig. 12.5b.

As shown in Fig. 12.5b, the authors performed the Pearson's correlation tests by considering the two variables (**Girth..in.** and **Height..ft.**). We stored the results of the tests in an R objects named "`PearsonCorr.test`" for the **two-tailed** analysis, and "`PearsonCorr.test2`" and "`PearsonCorr.test3`" for the **one-tailed** analysis, respectively.

## # Step 4—Plot and Visualize Correlation Between the Targeted Variables

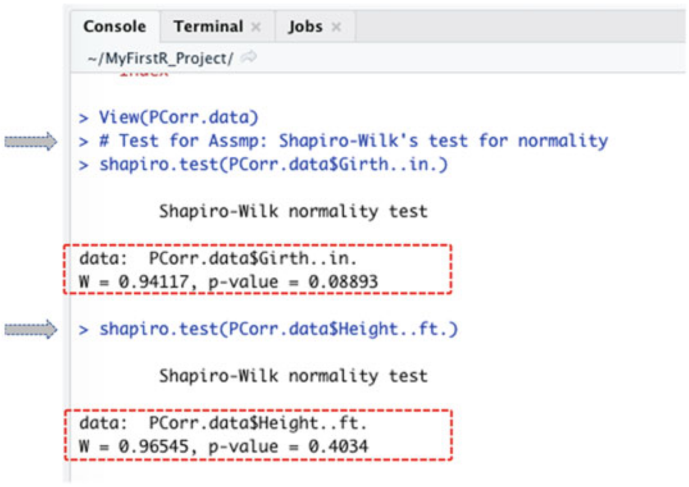
Another great way to check whether there is a relationship (correlation) between the two specified variables is by plotting them as graph. By so doing, the users will be able to visualize the "linear line" between the variables.

As described in Fig. 12.6 (Step 4, Lines 39–45) and the resultant scatterplot in the same figure (Fig. 12.6); the authors applied the **ggscatter( )** function in R to visualize the relationship between the two variables "**Girth..in.**" and "**Height..ft.**" as contained in the example dataset we stored as "**PCorr.data**" in R.

The syntax and code used to plot the graph is as shown below, and the chart or scatterplot represented in Fig. 12.6.

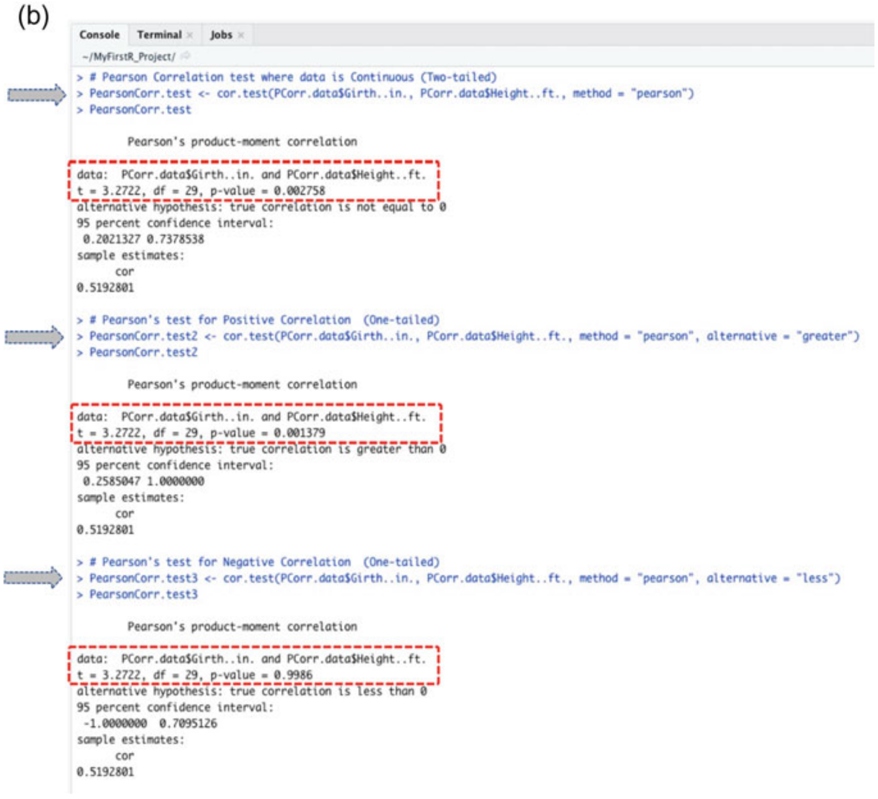
```
# Step 4 - Visualize Correlation between the two variables

ggscatter(PCorr.data, x = "Girth..in.", y = "Height..ft.",

          add = "reg.line", conf.int = TRUE,

          cor.coef = TRUE, cor.method = "pearson",

          xlab = "Girth (inches)", ylab = "Height (ft)",

          main = "Correlation between Tree Girth and Height")

)
```

(a)



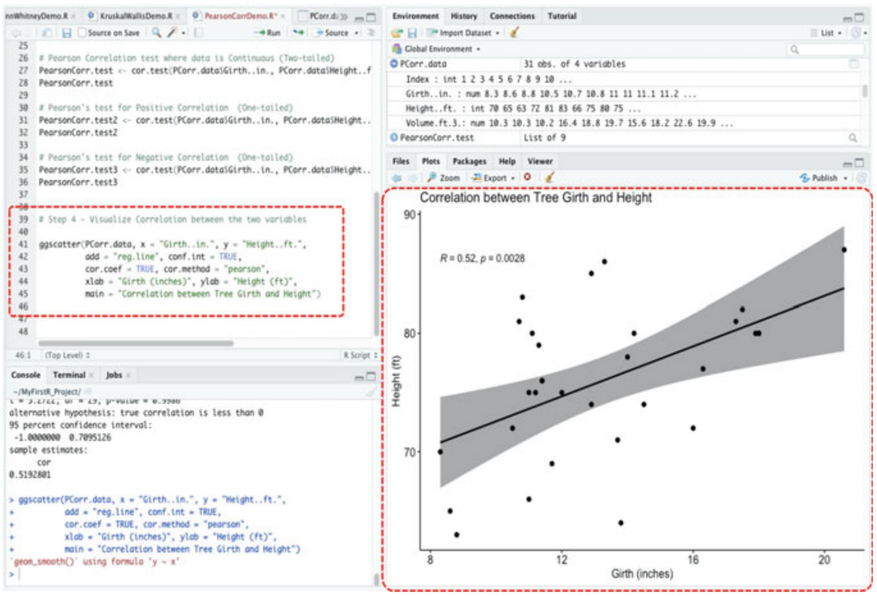
```
> View(PCorr.data)
> # Test for Assmp: Shapiro-Wilk's test for normality
> shapiro.test(PCorr.data$Girth..in.)

        Shapiro-Wilk normality test

data:  PCorr.data$Girth..in.
W = 0.94117, p-value = 0.08893

> shapiro.test(PCorr.data$Height..ft.)

        Shapiro-Wilk normality test

data:  PCorr.data$Height..ft.
W = 0.96545, p-value = 0.4034
```

(b)



```
> # Pearson Correlation test where data is Continuous (Two-tailed)
> PearsonCorr.test <- cor.test(PCorr.data$Girth..in., PCorr.data$Height..ft., method = "pearson")
> PearsonCorr.test

        Pearson's product-moment correlation

data:  PCorr.data$Girth..in. and PCorr.data$Height..ft.
t = 3.2722, df = 29, p-value = 0.002758
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.2021327 0.7378538
sample estimates:
      cor
0.5192801

> # Pearson's test for Positive Correlation  (One-tailed)
> PearsonCorr.test2 <- cor.test(PCorr.data$Girth..in., PCorr.data$Height..ft., method = "pearson", alternative = "greater")
> PearsonCorr.test2

        Pearson's product-moment correlation

data:  PCorr.data$Girth..in. and PCorr.data$Height..ft.
t = 3.2722, df = 29, p-value = 0.001379
alternative hypothesis: true correlation is greater than 0
95 percent confidence interval:
 0.2585047 1.0000000
sample estimates:
      cor
0.5192801

> # Pearson's test for Negative Correlation  (One-tailed)
> PearsonCorr.test3 <- cor.test(PCorr.data$Girth..in., PCorr.data$Height..ft., method = "pearson", alternative = "less")
> PearsonCorr.test3

        Pearson's product-moment correlation

data:  PCorr.data$Girth..in. and PCorr.data$Height..ft.
t = 3.2722, df = 29, p-value = 0.9986
alternative hypothesis: true correlation is less than 0
95 percent confidence interval:
 -1.0000000  0.7095126
sample estimates:
      cor
0.5192801
```

**Fig. 12.5** **a** Results of test for data normality displayed in the Console in R. **b** Results of Pearson correlation tests displayed in the Console in R



```
# Pearson Correlation test where data is Continuous (Two-tailed)
PearsonCorr.test <- cor.test(PCorr.data$Girth..in., PCorr.data$Height..f
PearsonCorr.test

# Pearson's test for Positive Correlation  (One-tailed)
PearsonCorr.test2 <- cor.test(PCorr.data$Girth..in., PCorr.data$Height..
PearsonCorr.test2

# Pearson's test for Negative Correlation  (One-tailed)
PearsonCorr.test3 <- cor.test(PCorr.data$Girth..in., PCorr.data$Height..
PearsonCorr.test3


# Step 4 - Visualize Correlation between the two variables

ggscatter(PCorr.data, x = "Girth..in.", y = "Height..ft.",
          add = "reg.line", conf.int = TRUE,
          cor.coef = TRUE, cor.method = "pearson",
          xlab = "Girth (inches)", ylab = "Height (ft)",
          main = "Correlation between Tree Girth and Height")
```

```
alternative hypothesis: true correlation is less than 0
95 percent confidence interval:
 -1.0000000  0.7095126
sample estimates:
      cor
0.5192801

> ggscatter(PCorr.data, x = "Girth..in.", y = "Height..ft.",
+           add = "reg.line", conf.int = TRUE,
+           cor.coef = TRUE, cor.method = "pearson",
+           xlab = "Girth (inches)", ylab = "Height (ft)",
+           main = "Correlation between Tree Girth and Height")
`geom_smooth()` using formula 'y ~ x'
> 
```

```
PCorr.data          31 obs. of 4 variables
  Index : int 1 2 3 4 5 6 7 8 9 10 ...
  Girth..in. : num 8.3 8.6 8.8 10.5 10.7 10.8 11 11 11.1 11.2 ...
  Height..ft. : int 70 65 63 72 81 83 66 75 80 75 ...
  Volume.ft.3.: num 10.3 10.3 10.2 16.4 18.8 19.7 15.6 18.2 22.6 19.9 ...
PearsonCorr.test    List of 9
```

**Fig. 12.6** Plot representing correlation (relationship) between two variables in R using the ggscatter( ) function

## # Step 5—Results Interpretation (Pearson Correlation)

The final step in the Pearson's correlation analysis is to interpret and understand the result of the test.

By default, the hypothesis for conducting the test (Pearson Correlation) by considering the two continuous variables "**Girth..in.**" and "**Height..ft.**" (see: Fig. 12.5b) is as follows;

### Two-Tailed Pearson Correlation

- **($H_1$)** *IF* the *p*-value of the test is less than or equal to 0.05 ($p \leq 0.05$), *THEN* we can assume that there is a correlation between the two variables (**Girth..in.** and **Height..ft.**). Thus, the population correlation coefficient ($\rho$) $\neq 0$. Meaning that the population correlation coefficient is not 0, therefore, we can assume that a non-zero correlation exist between the "**Girth..in.**" and "**Height..ft.**" variables.
- **($H_0$)** *ELSE IF* the *p*-value is greater than 0.05 ($p > 0.05$) THEN we can say that there is no correlation between the two variables. Therefore, $\rho = 0$. Meaning that the population correlation coefficient is 0, therefore, there is no association (correlation) between the two variables.

## One-Tailed Pearson Correlation

- **($H_1$)** *IF* the *p*-value of the test is less than or equal to 0.05 ($p \leq 0.05$), *THEN* we can statistically assume that either $\rho > 0$, i.e., the population correlation coefficient is greater than 0, thus, a *positive* correlation may exist.
  OR
  $\rho < 0$, i.e., the population correlation coefficient is less than 0, thus, a *negative* correlation may exist between the two variables (**Girth..in.** and **Height..ft.**).
- **($H_0$)** *ELSE IF* the *p*-value is greater than 0.05 ($p > 0.05$) THEN we can conclude that there is no correlation between the two variables. Therefore, $\rho = 0$. Meaning that the population correlation coefficient is 0, thus, there is no association (correlation) between the two variables.

```
> PearsonCorr.test <- cor.test(PCorr.data$Girth..in.,
PCorr.data$Height..ft., method = "pearson")
> PearsonCorr.test

     Pearson's product-moment correlation

data:  PCorr.data$Girth..in. and PCorr.data$Height..ft.
t = 3.2722, df = 29, p-value = 0.002758
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.2021327 0.7378538
sample estimates:
      cor
0.5192801)
```

As shown in the above result and gathered in the outcome of the Pearson correlation (two-tailed) test for the example dataset (**PCorr.data**) represented in Fig. 12.5b; the meaning of the results of the **cor.test( )** method we applied by testing the relationship between the **Girth..in.** and **Height..ft**. variables (stored in an R object we called "`PearsonCorr.test`") can be explained as a list containing the following:

- **Statistics**: `t = 3.2722` that denotes the value of the Pearson correlation statistics.
- **Parameter**: `df = 29` which signifies the degrees of freedom for the test statistics.
- ***p*-value**: `p-value = 0.002758` is the *p*-value (significance levels) of the test.
- **Confidence interval**: `Conf.Int(95%, 0.2021327 0.7378538)` represents the confidence interval for the correlation assumed to be appropriate to the specified alternative hypothesis.
- **Sample estimates**: `cor = 0.5192801` is the value of the population correlation coefficient ($\rho$).

Statistically, the *p*-value of the Pearson correlation test (`PearsonCorr.test`) we conducted is `p = 0.002758` (see Fig. 12.5b). As we can see, the value is significantly less than the scientifically acceptable significance levels ($p \leq 0.05$). Therefore, we reject the $H_0$ and accept $H_1$ by concluding that there is a significant relationship (correlation) between the two sets of variables (**Girth..in.** and **Height..ft.**) in the dataset (**two-tailed test**).

Furthermore, as shown in the next results of the Pearson correlation test presented below and in Fig. 12.5b, done for the "one-tailed" correlation tests, therein;

- We also checked whether the correlation, if any? (in this example case, yes—see result of the correlation described above) is a "positive" or "negative" (direction) correlation, respectively. The results of this particular test (one-tailed) were stored in R objects we called "`PearsonCorr.test2`" and "`PearsonCorr.test3`", respectively.

```
> > PearsonCorr.test2 <- cor.test(PCorr.data$Girth..in.,
PCorr.data$Height..ft., method = "pearson", alternative = "greater")
> PearsonCorr.test2

      Pearson's product-moment correlation

data:  PCorr.data$Girth..in. and PCorr.data$Height..ft.
t = 3.2722, df = 29, p-value = 0.001379
alternative hypothesis: true correlation is greater than 0
95 percent confidence interval:
 0.2585047 1.0000000
sample estimates:
      cor
0.5192801
```

```
> PearsonCorr.test3 <- cor.test(PCorr.data$Girth..in.,
PCorr.data$Height..ft., method = "pearson", alternative = "less")
> PearsonCorr.test3

      Pearson's product-moment correlation

data:  PCorr.data$Girth..in. and PCorr.data$Height..ft.
t = 3.2722, df = 29, p-value = 0.9986
alternative hypothesis: true correlation is less than 0
95 percent confidence interval:
 -1.0000000  0.7095126
sample estimates:
      cor
0.5192801
```

As reported in the above results of the "one-tailed" tests for *positive correlation* (`PearsonCorr.test2, p=0.001379`), and *negative correlation* (`PearsonCorr.test3, p=0.9986`); we can see based on the *p*-values of the "direction test" as it is called (significant levels, $p \leq 0.05$); that the correlation we found between the two variables "**Girth..in**" and "**Height..ft.**" (`two-tailed, PearsonCorr.test, p=0.002758`) (see Fig. 12.5b) was a "positive" directed correlation or association (PearsonCorr.test2, p=0.001379).

## 12.3 Kendall's Tau and Spearman's Rho Correlation Tests in R

*Kendall's tau* and *Spearman's rho* correlation (non-parametric equivalents or alternatives to the Pearson correlation) measures the strength of dependence or degree of association between two categorical or ordinal variables. In this statistical settings, the methods are used when the dataset the researcher or data analyst wants to investigate or analyze violates the assumptions of the parametric counterpart (Pearson), e.g., non-normally distributed data samples or existence of ordinal data type, etc.

Just like Pearson correlation test, the methods (*Kendall's tau* and *Spearman's rho*) also can be used to calculate the level of change (be it positive or negative) in one variable when another variable changes.

By default, the hypothesis for testing whether there is *correlation* (measure of strength of dependence or degree of association) between the two specified set of (categorical or ordinal) variables is; *IF* the *p*-value of the test is less than or equal to 0.05 ($p \leq 0.05$), *THEN* we can assume that there is a statistically significant strong dependence or association between the two analyzed variables, and that this is not by chance ($H_1$). *ELSE IF* the *p*-value is greater than 0.05 ($p > 0.05$) THEN we can say that there is no significant dependency or association between the two variables, and any observed dependency or association could only occur by chance ($H_0$).

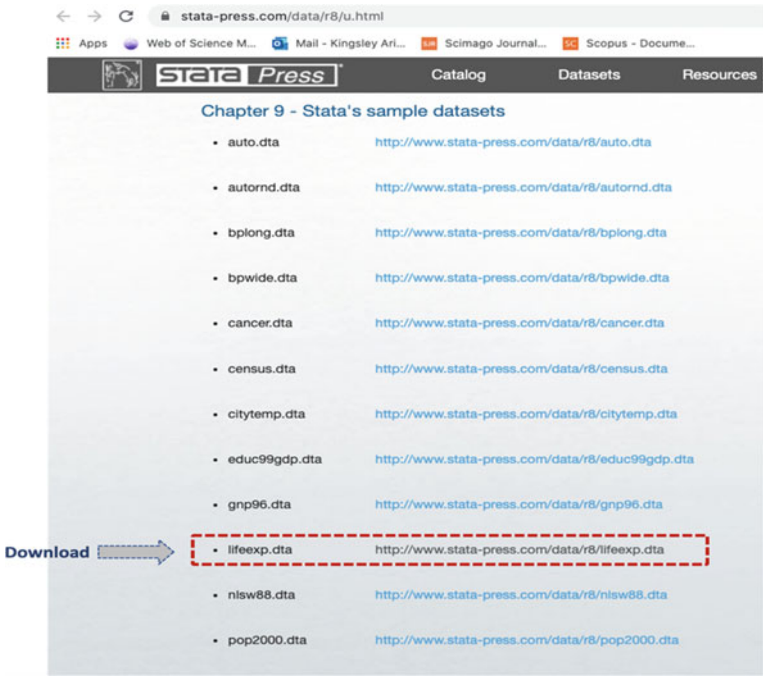
Here, the authors will demonstrate how to conduct the *Kendall's tau* and *Spearman's rho* correlation tests in R using the **cor.test**( ) function. We will do this following the same steps we have outlined in Fig. 12.1.

To start, **Create a new R Script** and name it "**Tau.Rho.Demo**" or any name the user may preferably choose.

Now, let's proceed to download an example dataset or file that we will use to demonstrate the two tests (*Kendall's tau* and *Spearman's rho*) (*** the users are welcome to use any dataset they may want to use provided the dataset are in the right format and type, and they can follow the example codes provided by the authors accordingly).

As shown in Fig. 12.7, download the example **.dta** dataset named "**lifeexp.dta**" through the following source: https://www.stata-press.com/data/r8/u.html and save the file on the computer or local machine (*** the example file can also be downloaded via the following repository by the authors: https://doi.org/10.6084/m9.figshare.24728073).

Once the user has successfully downloaded and saved the example file on the computer, we can proceed to conduct the *Kendall's tau* and *Spearman's rho* correlation tests using R.



Chapter 9 - Stata's sample datasets

| | |
|---|---|
| auto.dta | http://www.stata-press.com/data/r8/auto.dta |
| autornd.dta | http://www.stata-press.com/data/r8/autornd.dta |
| bplong.dta | http://www.stata-press.com/data/r8/bplong.dta |
| bpwide.dta | http://www.stata-press.com/data/r8/bpwide.dta |
| cancer.dta | http://www.stata-press.com/data/r8/cancer.dta |
| census.dta | http://www.stata-press.com/data/r8/census.dta |
| citytemp.dta | http://www.stata-press.com/data/r8/citytemp.dta |
| educ99gdp.dta | http://www.stata-press.com/data/r8/educ99gdp.dta |
| gnp96.dta | http://www.stata-press.com/data/r8/gnp96.dta |
| lifeexp.dta | http://www.stata-press.com/data/r8/lifeexp.dta |
| nlsw88.dta | http://www.stata-press.com/data/r8/nlsw88.dta |
| pop2000.dta | http://www.stata-press.com/data/r8/pop2000.dta |

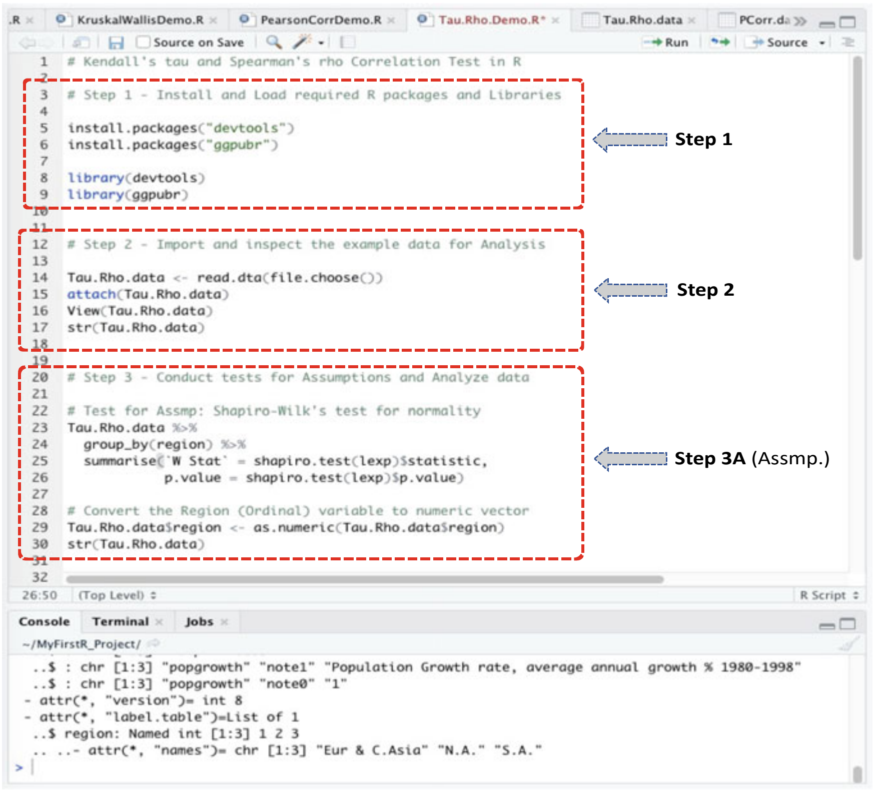
**Fig. 12.7** Example of (.dta) sample file download (*Source* https://www.stata-press.com/data/r8/u.html)

## # Step 1—Install and Load the Required R Packages and Libraries

**Install** and **Load** the following *R packages* and *libraries* (see Fig. 12.8, Step 1, Lines 3–9) that will be used to call the different R functions, data manipulations, and graphical visualizations for the *Kendall's tau* and *Spearman's rho* Correlation tests.

The syntax and code to install and load the R packages and libraries are as follows: (***Note: if the reader have practiced and implemented the previous example in Sect. 12.2, then you may not need to re-install the following R packages again. New readers that may have directly visited this section will need to install and load the following packages and libraries as described below.)

```
install.packages("devtools")
install.packages("ggpubr")

library(devtools)
library(ggpubr)
```



```
1  # Kendall's tau and Spearman's rho Correlation Test in R
2
3  # Step 1 - Install and Load required R packages and Libraries
4
5  install.packages("devtools")
6  install.packages("ggpubr")
7
8  library(devtools)
9  library(ggpubr)
10
11
12 # Step 2 - Import and inspect the example data for Analysis
13
14 Tau.Rho.data <- read.dta(file.choose())
15 attach(Tau.Rho.data)
16 View(Tau.Rho.data)
17 str(Tau.Rho.data)
18
19
20 # Step 3 - Conduct tests for Assumptions and Analyze data
21
22 # Test for Assmp: Shapiro-Wilk's test for normality
23 Tau.Rho.data %>%
24   group_by(region) %>%
25   summarise(`W Stat` = shapiro.test(lexp)$statistic,
26             p.value = shapiro.test(lexp)$p.value)
27
28 # Convert the Region (Ordinal) variable to numeric vector
29 Tau.Rho.data$region <- as.numeric(Tau.Rho.data$region)
30 str(Tau.Rho.data)
31
32
```

```
  ..$ : chr [1:3] "popgrowth" "note1" "Population Growth rate, average annual growth % 1980-1998"
  ..$ : chr [1:3] "popgrowth" "note0" "1"
 - attr(*, "version")= int 8
 - attr(*, "label.table")=List of 1
  ..$ region: Named int [1:3] 1 2 3
  .. ..- attr(*, "names")= chr [1:3] "Eur & C.Asia" "N.A." "S.A."
>
```

**Fig. 12.8** Steps used for conducting *Kendall's tau* and *Spearman's rho* correlation tests in R

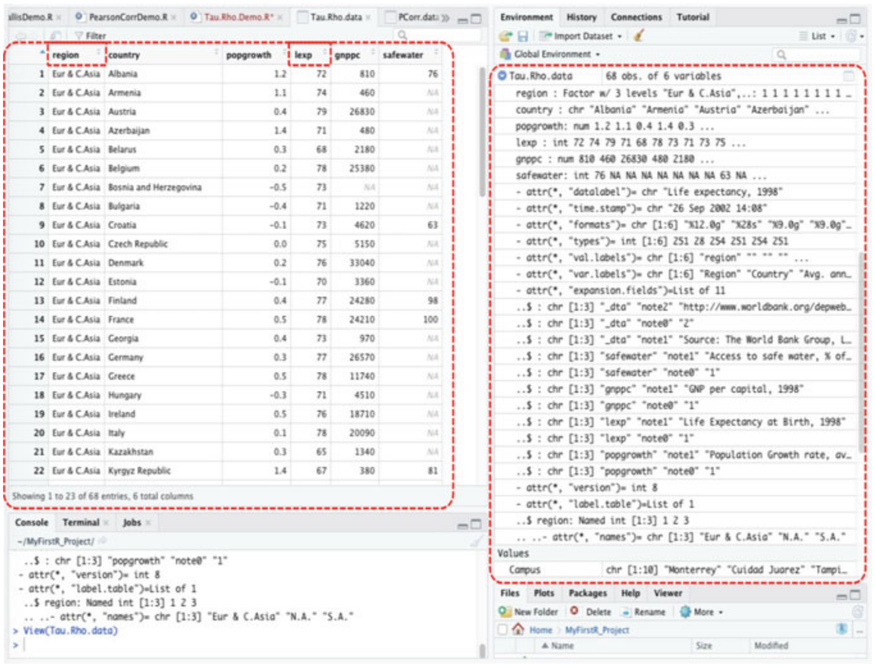
## # Step 2—Import and Inspect the Example Dataset for Correlation Analysis

As defined in Step 2 in Fig. 12.8 (Lines 12–17); import the dataset named "**lifeexp.dta**" that we downloaded earlier, and store this in an R object named "**Tau.Rho.data**" in R (the users are welcome to use any name of choice if they wish to do so).

Once the user has successfully imported the example dataset, they will be able to view the details of the dataset (**lifeexp.dta**) as shown in Fig. 12.9 with 68 observations and 6 variables in the data sample.

The syntax and code for importing and attaching the file in R are as shown below:

```
Tau.Rho.data <- read.dta(file.choose())

attach(Tau.Rho.data)

View(Tau.Rho.data)

str(Tau.Rho.data)
```



| | region | country | popgrowth | lexp | gnppc | safewater |
|---|---|---|---|---|---|---|
| 1 | Eur & C.Asia | Albania | 1.2 | 72 | 810 | 76 |
| 2 | Eur & C.Asia | Armenia | 1.1 | 74 | 460 | NA |
| 3 | Eur & C.Asia | Austria | 0.4 | 79 | 26830 | NA |
| 4 | Eur & C.Asia | Azerbaijan | 1.4 | 71 | 480 | NA |
| 5 | Eur & C.Asia | Belarus | 0.3 | 68 | 2180 | NA |
| 6 | Eur & C.Asia | Belgium | 0.2 | 78 | 25380 | NA |
| 7 | Eur & C.Asia | Bosnia and Herzegovina | -0.5 | 73 | NA | NA |
| 8 | Eur & C.Asia | Bulgaria | -0.4 | 71 | 1220 | NA |
| 9 | Eur & C.Asia | Croatia | -0.1 | 73 | 4620 | 63 |
| 10 | Eur & C.Asia | Czech Republic | 0.0 | 75 | 5150 | NA |
| 11 | Eur & C.Asia | Denmark | 0.2 | 76 | 33040 | NA |
| 12 | Eur & C.Asia | Estonia | -0.1 | 70 | 3360 | NA |
| 13 | Eur & C.Asia | Finland | 0.4 | 77 | 24280 | 98 |
| 14 | Eur & C.Asia | France | 0.5 | 78 | 24210 | 100 |
| 15 | Eur & C.Asia | Georgia | 0.4 | 73 | 970 | NA |
| 16 | Eur & C.Asia | Germany | 0.3 | 77 | 26570 | NA |
| 17 | Eur & C.Asia | Greece | 0.5 | 78 | 11740 | NA |
| 18 | Eur & C.Asia | Hungary | -0.3 | 71 | 4510 | NA |
| 19 | Eur & C.Asia | Ireland | 0.5 | 76 | 18710 | NA |
| 20 | Eur & C.Asia | Italy | 0.1 | 78 | 20090 | NA |
| 21 | Eur & C.Asia | Kazakhstan | 0.3 | 65 | 1340 | NA |
| 22 | Eur & C.Asia | Kyrgyz Republic | 1.4 | 67 | 380 | 81 |

Showing 1 to 23 of 68 entries, 6 total columns

```
~/MyFirstR_Project/
  ..$ : chr [1:3] "popgrowth" "note0" "1"
 - attr(*, "version")= int 8
 - attr(*, "label.table")=List of 1
  ..$ region: Named int [1:3] 1 2 3
  .. ..- attr(*, "names")= chr [1:3] "Eur & C.Asia" "N.A." "S.A."
> View(Tau.Rho.data)
>
```

```
Tau.Rho.data        68 obs. of 6 variables
   region : Factor w/ 3 levels "Eur & C.Asia",..: 1 1 1 1 1 1 1 1 …
   country : chr "Albania" "Armenia" "Austria" "Azerbaijan" ...
   popgrowth: num 1.2 1.1 0.4 1.4 0.3 ...
   lexp : int 72 74 79 71 68 78 73 71 73 75 ...
   gnppc : num 810 460 26830 480 2180 ...
   safewater: int 76 NA NA NA NA NA NA NA 63 NA ...
   - attr(*, "datalabel")= chr "Life expectancy, 1998"
   - attr(*, "time.stamp")= chr "26 Sep 2002 14:08"
   - attr(*, "formats")= chr [1:6] "%12.0g" "%28s" "%9.0g" "%9.0g"…
   - attr(*, "types")= int [1:6] 251 28 254 251 254 251
   - attr(*, "val.labels")= chr [1:6] "region" "" "" "" ...
   - attr(*, "var.labels")= chr [1:6] "Region" "Country" "Avg. ann…
   - attr(*, "expansion.fields")=List of 11
   ..$ : chr [1:3] "_dta" "note2" "http://www.worldbank.org/depweb…
   ..$ : chr [1:3] "_dta" "note0" "2"
   ..$ : chr [1:3] "_dta" "note1" "Source: The World Bank Group, L…
   ..$ : chr [1:3] "safewater" "note1" "Access to safe water, % of…
   ..$ : chr [1:3] "safewater" "note0" "1"
   ..$ : chr [1:3] "gnppc" "note1" "GNP per capital, 1998"
   ..$ : chr [1:3] "gnppc" "note0" "1"
   ..$ : chr [1:3] "lexp" "note1" "Life Expectancy at Birth, 1998"
   ..$ : chr [1:3] "lexp" "note0" "1"
   ..$ : chr [1:3] "popgrowth" "note1" "Population Growth rate, av…
   ..$ : chr [1:3] "popgrowth" "note0" "1"
   - attr(*, "version")= int 8
   - attr(*, "label.table")=List of 1
   ..$ region: Named int [1:3] 1 2 3
   .. ..- attr(*, "names")= chr [1:3] "Eur & C.Asia" "N.A." "S.A."
Values
  Campus              chr [1:10] "Monterrey" "Cuidad Juarez" "Tampi…
```

**Fig. 12.9** Example of a .dta dataset imported and stored as an R object in R

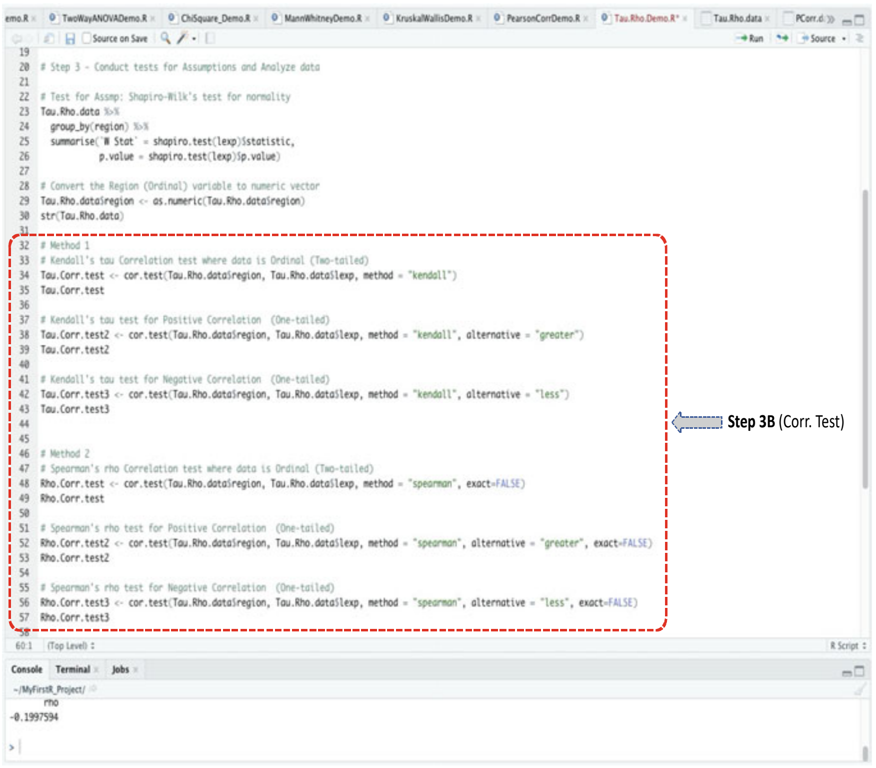
## # Step 3—Conduct Tests for Assumptions and Analyze Data

Now that we have imported the example dataset and stored this in an R object we named “**Tau.Rho.data**”, we can proceed to analyze the data.

As defined in Step 3A in Fig. 12.8 (Lines 20–30), we will first conduct the test of assumptions (e.g., data normality, and factorization of ordinal data type, etc.), and then perform the *Kendall’s tau* and *Spearman’s rho* tests (Step 3B, Fig. 12.10, Lines 32–57), if all the necessary conditions are met, by using the **cor.test( )** function in R.

As defined earlier in the Introduction section (Sect. 12.1);

- The **Kendall’s tau** and **Spearman’s rho** correlation statistics checks whether there exists a *dependency* or *association* between two independently sampled variables.
- The targeted variables should be categorical or ordinal data type.



```
19
20 # Step 3 - Conduct tests for Assumptions and Analyze data
21
22 # Test for Assmp: Shapiro-Wilk's test for normality
23 Tau.Rho.data %>%
24   group_by(region) %>%
25   summarise(`W Stat` = shapiro.test(lexp)$statistic,
26             p.value = shapiro.test(lexp)$p.value)
27
28 # Convert the Region (Ordinal) variable to numeric vector
29 Tau.Rho.data$region <- as.numeric(Tau.Rho.data$region)
30 str(Tau.Rho.data)
31
32 # Method 1
33 # Kendall's tau Correlation test where data is Ordinal (Two-tailed)
34 Tau.Corr.test <- cor.test(Tau.Rho.data$region, Tau.Rho.data$lexp, method = "kendall")
35 Tau.Corr.test
36
37 # Kendall's tau test for Positive Correlation  (One-tailed)
38 Tau.Corr.test2 <- cor.test(Tau.Rho.data$region, Tau.Rho.data$lexp, method = "kendall", alternative = "greater")
39 Tau.Corr.test2
40
41 # Kendall's tau test for Negative Correlation  (One-tailed)
42 Tau.Corr.test3 <- cor.test(Tau.Rho.data$region, Tau.Rho.data$lexp, method = "kendall", alternative = "less")
43 Tau.Corr.test3
44
45
46 # Method 2
47 # Spearman's rho Correlation test where data is Ordinal (Two-tailed)
48 Rho.Corr.test <- cor.test(Tau.Rho.data$region, Tau.Rho.data$lexp, method = "spearman", exact=FALSE)
49 Rho.Corr.test
50
51 # Spearman's rho test for Positive Correlation  (One-tailed)
52 Rho.Corr.test2 <- cor.test(Tau.Rho.data$region, Tau.Rho.data$lexp, method = "spearman", alternative = "greater", exact=FALSE)
53 Rho.Corr.test2
54
55 # Spearman's rho test for Negative Correlation  (One-tailed)
56 Rho.Corr.test3 <- cor.test(Tau.Rho.data$region, Tau.Rho.data$lexp, method = "spearman", alternative = "less", exact=FALSE)
57 Rho.Corr.test3
```

```
       rho
-0.1997594
>
```

**Fig. 12.10** Conducting *Kendall's tau* and *Spearman's rho* correlation tests in R

To illustrate the two tests (*Kendall's tau* and *Spearman's rho*) using the example dataset we stored as **"Tau.Rho.data"** in R (see: highlighted columns in Fig. 12.9):

1. We will test whether there exists a dependency or association (correlation) between the "**region**" and "**lexp**" variables in the example (Tau.Rho.data) life expectancy data (**two-tailed test**).
2. Then, we will also test whether the correlation (if there exist any) is a *positive* or *negative* (direction) correlation (**one-tailed test**).

The syntax to performing the above tests in R is as shown in the codes provided and described below (see: Fig. 12.10, Step 3B, Lines 32–57):

```
# Test for Assmp: Shapiro-Wilk's test for normality
Tau.Rho.data %>%
  group_by(region) %>%
  summarise(`W Stat` = shapiro.test(lexp)$statistic,
            p.value = shapiro.test(lexp)$p.value)

# Convert the Region (Ordinal) variable to numeric vector
Tau.Rho.data$region <- as.numeric(Tau.Rho.data$region)
str(Tau.Rho.data)

# Method 1
# Kendall's tau Correlation test where data is Ordinal (Two-tailed)
Tau.Corr.test <- cor.test(Tau.Rho.data$region, Tau.Rho.data$lexp, method
= "kendall")
Tau.Corr.test

# Kendall's tau test for Positive Correlation  (One-tailed)
Tau.Corr.test2   <-   cor.test(Tau.Rho.data$region,   Tau.Rho.data$lexp,
method = "kendall", alternative = "greater")
Tau.Corr.test2

# Kendall's tau test for Negative Correlation  (One-tailed)
Tau.Corr.test3   <-   cor.test(Tau.Rho.data$region,   Tau.Rho.data$lexp,
method = "kendall", alternative = "less")
Tau.Corr.test3


# Method 2
# Spearman's rho Correlation test where data is Ordinal (Two-tailed)
Rho.Corr.test <- cor.test(Tau.Rho.data$region, Tau.Rho.data$lexp, method
= "spearman", exact=FALSE)
Rho.Corr.test

# Spearman's rho test for Positive Correlation  (One-tailed)
Rho.Corr.test2   <-   cor.test(Tau.Rho.data$region,   Tau.Rho.data$lexp,
method = "spearman", alternative = "greater", exact=FALSE)
Rho.Corr.test2

# Spearman's rho test for Negative Correlation  (One-tailed)
Rho.Corr.test3   <-   cor.test(Tau.Rho.data$region,   Tau.Rho.data$lexp,
method = "spearman", alternative = "less", exact=FALSE)
Rho.Corr.test3
```

## Useful Tips and Information

- The users should always use the `alternative = "greater"` and `alternative = "less"` options to specify the "positive" and "negative" (direction) correlation analysis (i.e., for one-tailed test), respectively.
- Another important task the authors conducted which the users may need to do (depending on the readily available dataset) prior to performing the tests (Kendall or Spearman) was to factorize the targeted ordinal data type (e.g., region) into a numeric format (see: Fig. 12.8, Lines 28–30) before applying the **cor.test( )** function or methods.

*****Note**: For *Spearman's rho* test (Method 2), we included the R code `exact=FALSE` in the **cor.test( )** function (see: Fig. 12.10, Lines 48, 52, and 56). This was done in order to handle the error "Cannot compute exact *p*-value with ties" when running the method (Method 2—see Fig. 12.10). This is owing to the fact that the *Spearman's rho* method is much more sensitive to error and handling discrepancies in data samples than the *Kendall's tau* method, as we explained and pointed out earlier in Sect. 12.1).

Once the user has successfully run the set of codes and analysis as defined in **Steps 3A and 3B** (Figs. 12.8 and 12.10, Lines 20–57), they will be presented with the results of the "tests for assumptions", followed by the "*Kendall's tau*" (method 1) test, and then "*Spearman's rho*" (method 2) tests in the Console in R as shown in Figs. 12.11a, b, and c, respectively.

Consequentially, in Fig. 12.11a, the authors performed the test for assumption (data normality) for the *Kendall's tau* and *Spearman's rho* correlation analysis in order to determine if the selected or targeted variables "**region**" and "**lexp**" are suitable for conducting the two tests.

As highlighted in the figure (Fig. 12.11a), we can see that the *normality test* using the Shapiro–Wilk's method or function—**shapiro.test( )** (where we assume a value of *p > 0.05* is normal) shows that the distribution of the two variables was not normally distributed, with *p*-values of the "**region**" variable (with three ranked groups) when analyzed against the "**lexp**" variable showing to be mostly non-normal values (p≤0.05) whereby the values of `p-value=0.0203` (W=0.938) for "Eur & C.Asia", `p-value=0.0538` (W=0.878) for "N.A", and `p-value=0.308 (W=0.914)` for "S.A", respectively. Therefore, we assume that the dataset or analyzed variables are not normally distributed, and a *distributed-free* method such as the *Kendall's tau* and *Spearman's rho* correlation analysis will be suitable for analyzing the data sample.

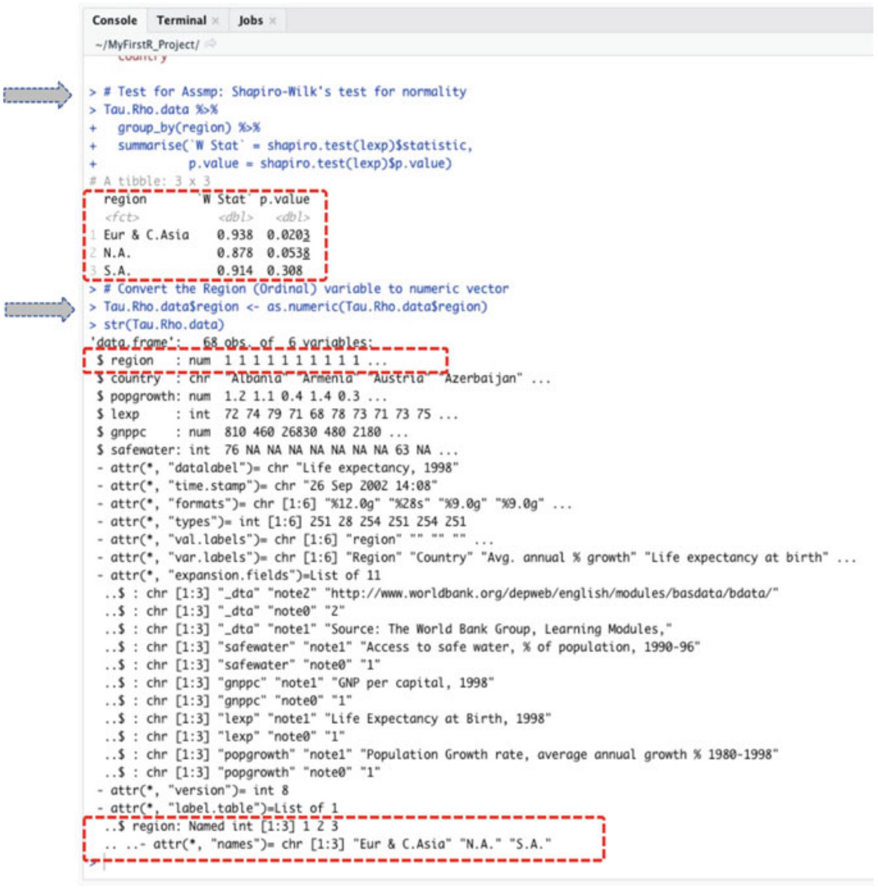
Thus, we proceed to conduct the "*Kendall's tau*" and "*Spearman's rho*" correlation analysis as defined in Step 3B (Fig. 12.10, Lines 32–57) and the results are as presented in Figs. 12.11b and c, respectively.

As shown in Figs. 12.11b, c, the authors performed the *Kendall's tau* and *Spearman's rho* tests by considering the two variables **"region"** and "**lexp"** in the example data (stored as Tau.Rho.data in R).

- The results of the *Kendall's tau* tests were stored in an R object we named "`Tau.Corr.test`" for the **two-tailed** analysis, and then

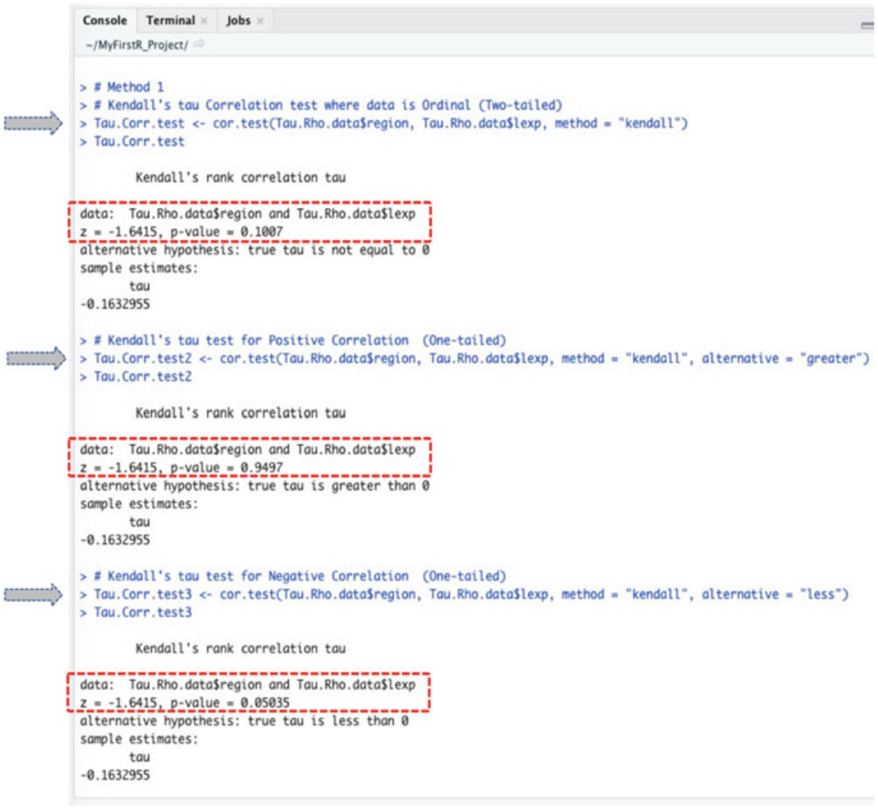
"`Tau.Corr.test2`" and "`Tau.Corr.test3`" for the **one-tailed** analysis, respectively.

- Accordingly, we stored the results of the *Spearman's rho* tests in R objects we called "`Rho.Corr.test`" for the **two-tailed** analysis, and then "`Rho.Corr.test2`" and "`Rho.Corr.test3`" for the **one-tailed** analysis, respectively.



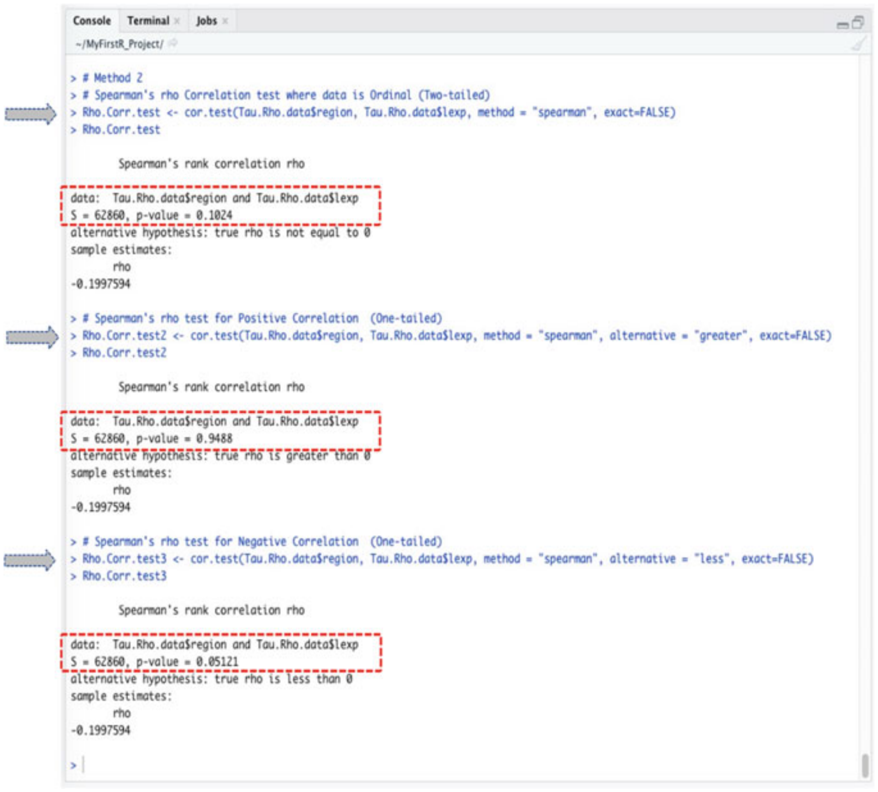
```
> # Test for Assmp: Shapiro-Wilk's test for normality
> Tau.Rho.data %>%
+   group_by(region) %>%
+   summarise(`W Stat` = shapiro.test(lexp)$statistic,
+             p.value = shapiro.test(lexp)$p.value)
# A tibble: 3 x 3
  region      `W Stat` p.value
  <fct>          <dbl>   <dbl>
1 Eur & C.Asia   0.938  0.0203
2 N.A.           0.878  0.0538
3 S.A.           0.914  0.308
> # Convert the Region (Ordinal) variable to numeric vector
> Tau.Rho.data$region <- as.numeric(Tau.Rho.data$region)
> str(Tau.Rho.data)
'data.frame':	68 obs. of  6 variables:
 $ region   : num  1 1 1 1 1 1 1 1 1 1 ...
 $ country  : chr  "Albania" "Armenia" "Austria" "Azerbaijan" ...
 $ popgrowth: num  1.2 1.1 0.4 1.4 0.3 ...
 $ lexp     : int  72 74 79 71 68 78 73 71 73 75 ...
 $ gnppc    : num  810 460 26830 480 2180 ...
 $ safewater: int  76 NA NA NA NA NA NA NA 63 NA ...
 - attr(*, "datalabel")= chr "Life expectancy, 1998"
 - attr(*, "time.stamp")= chr "26 Sep 2002 14:08"
 - attr(*, "formats")= chr [1:6] "%12.0g" "%28s" "%9.0g" "%9.0g" ...
 - attr(*, "types")= int [1:6] 251 28 254 251 254 251
 - attr(*, "val.labels")= chr [1:6] "region" "" "" "" ...
 - attr(*, "var.labels")= chr [1:6] "Region" "Country" "Avg. annual % growth" "Life expectancy at birth" ...
 - attr(*, "expansion.fields")=List of 11
  ..$ : chr [1:3] "_dta" "note2" "http://www.worldbank.org/depweb/english/modules/basdata/bdata/"
  ..$ : chr [1:3] "_dta" "note0" "2"
  ..$ : chr [1:3] "_dta" "note1" "Source: The World Bank Group, Learning Modules,"
  ..$ : chr [1:3] "safewater" "note1" "Access to safe water, % of population, 1990-96"
  ..$ : chr [1:3] "safewater" "note0" "1"
  ..$ : chr [1:3] "gnppc" "note1" "GNP per capital, 1998"
  ..$ : chr [1:3] "gnppc" "note0" "1"
  ..$ : chr [1:3] "lexp" "note1" "Life Expectancy at Birth, 1998"
  ..$ : chr [1:3] "lexp" "note0" "1"
  ..$ : chr [1:3] "popgrowth" "note1" "Population Growth rate, average annual growth % 1980-1998"
  ..$ : chr [1:3] "popgrowth" "note0" "1"
 - attr(*, "version")= int 8
 - attr(*, "label.table")=List of 1
  ..$ region: Named int [1:3] 1 2 3
  .. ..- attr(*, "names")= chr [1:3] "Eur & C.Asia" "N.A." "S.A."
>
```

**Fig. 12.11** **a** Results of test for data normality and factorization displayed in the Console in R. **b** Results of *Kendall's tau* correlation tests displayed in the Console in R. **c** Results of *Spearman's rho* correlation tests displayed in the Console in R



```
> # Method 1
> # Kendall's tau Correlation test where data is Ordinal (Two-tailed)
> Tau.Corr.test <- cor.test(Tau.Rho.data$region, Tau.Rho.data$lexp, method = "kendall")
> Tau.Corr.test

	Kendall's rank correlation tau

data:  Tau.Rho.data$region and Tau.Rho.data$lexp
z = -1.6415, p-value = 0.1007
alternative hypothesis: true tau is not equal to 0
sample estimates:
       tau
-0.1632955

> # Kendall's tau test for Positive Correlation  (One-tailed)
> Tau.Corr.test2 <- cor.test(Tau.Rho.data$region, Tau.Rho.data$lexp, method = "kendall", alternative = "greater")
> Tau.Corr.test2

	Kendall's rank correlation tau

data:  Tau.Rho.data$region and Tau.Rho.data$lexp
z = -1.6415, p-value = 0.9497
alternative hypothesis: true tau is greater than 0
sample estimates:
       tau
-0.1632955

> # Kendall's tau test for Negative Correlation  (One-tailed)
> Tau.Corr.test3 <- cor.test(Tau.Rho.data$region, Tau.Rho.data$lexp, method = "kendall", alternative = "less")
> Tau.Corr.test3

	Kendall's rank correlation tau

data:  Tau.Rho.data$region and Tau.Rho.data$lexp
z = -1.6415, p-value = 0.05035
alternative hypothesis: true tau is less than 0
sample estimates:
       tau
-0.1632955
```

**Fig. 12.11** (continued)



```
> # Method 2
> # Spearman's rho Correlation test where data is Ordinal (Two-tailed)
> Rho.Corr.test <- cor.test(Tau.Rho.data$region, Tau.Rho.data$lexp, method = "spearman", exact=FALSE)
> Rho.Corr.test

	Spearman's rank correlation rho

data:  Tau.Rho.data$region and Tau.Rho.data$lexp
S = 62860, p-value = 0.1024
alternative hypothesis: true rho is not equal to 0
sample estimates:
       rho
-0.1997594

> # Spearman's rho test for Positive Correlation  (One-tailed)
> Rho.Corr.test2 <- cor.test(Tau.Rho.data$region, Tau.Rho.data$lexp, method = "spearman", alternative = "greater", exact=FALSE)
> Rho.Corr.test2

	Spearman's rank correlation rho

data:  Tau.Rho.data$region and Tau.Rho.data$lexp
S = 62860, p-value = 0.9488
alternative hypothesis: true rho is greater than 0
sample estimates:
       rho
-0.1997594

> # Spearman's rho test for Negative Correlation  (One-tailed)
> Rho.Corr.test3 <- cor.test(Tau.Rho.data$region, Tau.Rho.data$lexp, method = "spearman", alternative = "less", exact=FALSE)
> Rho.Corr.test3

	Spearman's rank correlation rho

data:  Tau.Rho.data$region and Tau.Rho.data$lexp
S = 62860, p-value = 0.05121
alternative hypothesis: true rho is less than 0
sample estimates:
       rho
-0.1997594

> 
```

**Fig. 12.11** (continued)

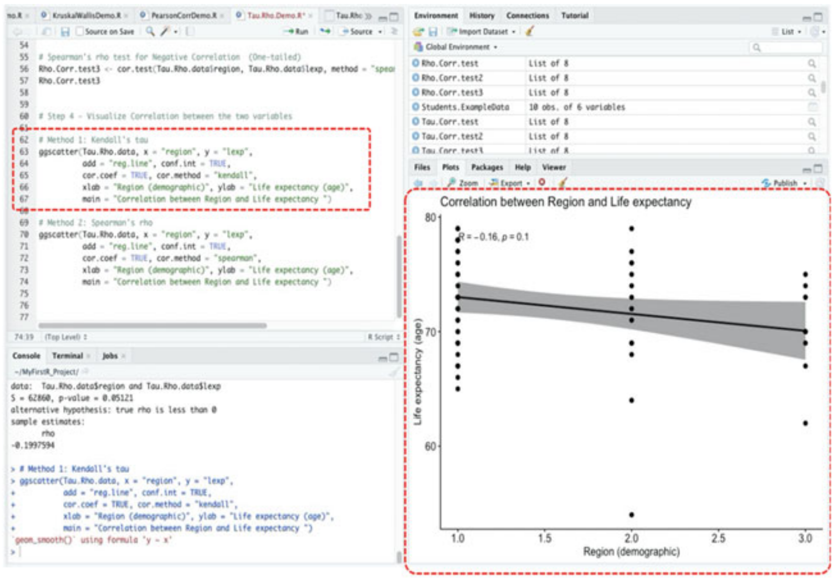
## # Step 4—Plot and Visualize Correlation Between the Variables

As previously illustrated earlier in Sect. 12.2, another way to check whether there is association or relationship (correlation) between two variables is by plotting them as graph. By so doing, the researcher or data analyst are able to visualize the *linear line* (correlation) between the two analyzed variables.

As represented in Figs. 12.12a, b (see Step 4, Lines 60–74) and the resultant scatterplots in the same figures (Fig. 12.12a, b); the authors utilized the **ggscatter( )** function to visualize the association or linearity between the two variables "**region**" and "**lexp**" as contained in the example data we stored as "**Tau.Rho.data**" in R.

The syntax and code we used to plot the graphs for both the Kendall's tau (method 1) and Spearman's rho (method 2) correlation is as shown in the codes below, and the resultant charts are represented in Figs. 12.12a and b, respectively.

(a)

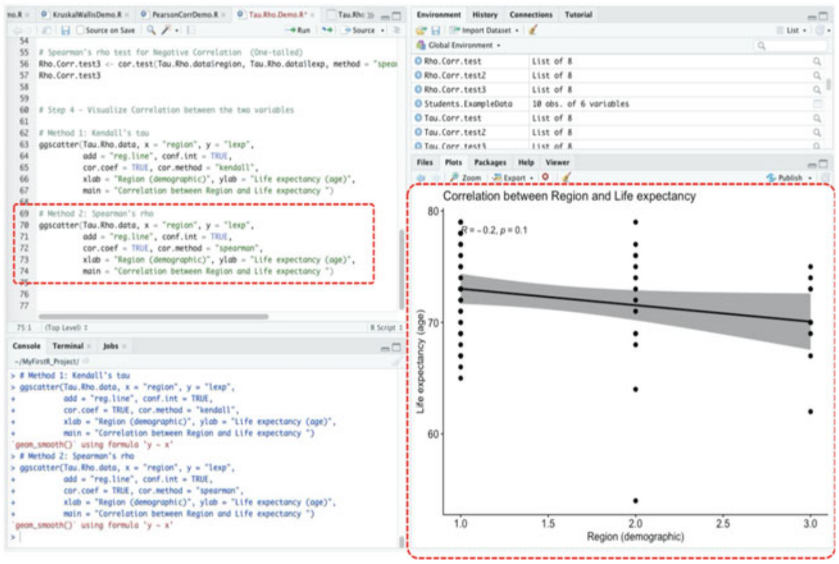


```
# Spearman's rho test for Negative Correlation  (One-tailed)
Rho.Corr.test3 <- cor.test(Tau.Rho.data$region, Tau.Rho.data$lexp, method = "spear
Rho.Corr.test3


# Step 4 - Visualize Correlation between the two variables

# Method 1: Kendall's tau
ggscatter(Tau.Rho.data, x = "region", y = "lexp",
          add = "reg.line", conf.int = TRUE,
          cor.coef = TRUE, cor.method = "kendall",
          xlab = "Region (demographic)", ylab = "Life expectancy (age)",
          main = "Correlation between Region and Life expectancy ")

# Method 2: Spearman's rho
ggscatter(Tau.Rho.data, x = "region", y = "lexp",
          add = "reg.line", conf.int = TRUE,
          cor.coef = TRUE, cor.method = "spearman",
          xlab = "Region (demographic)", ylab = "Life expectancy (age)",
          main = "Correlation between Region and Life expectancy ")
```

```
data:  Tau.Rho.data$region and Tau.Rho.data$lexp
S = 62860, p-value = 0.05121
alternative hypothesis: true rho is less than 0
sample estimates:
       rho
-0.1997594

> # Method 1: Kendall's tau
> ggscatter(Tau.Rho.data, x = "region", y = "lexp",
+           add = "reg.line", conf.int = TRUE,
+           cor.coef = TRUE, cor.method = "kendall",
+           xlab = "Region (demographic)", ylab = "Life expectancy (age)",
+           main = "Correlation between Region and Life expectancy ")
`geom_smooth()` using formula 'y ~ x'
>
```

```
Rho.Corr.test          List of 8
Rho.Corr.test2         List of 8
Rho.Corr.test3         List of 8
Students.ExampleData   10 obs. of 6 variables
Tau.Corr.test          List of 8
Tau.Corr.test2         List of 8
Tau.Corr.test3         List of 8
```

(b)



```
# Spearman's rho test for Negative Correlation  (One-tailed)
Rho.Corr.test3 <- cor.test(Tau.Rho.data$region, Tau.Rho.data$lexp, method = "spear
Rho.Corr.test3


# Step 4 - Visualize Correlation between the two variables

# Method 1: Kendall's tau
ggscatter(Tau.Rho.data, x = "region", y = "lexp",
          add = "reg.line", conf.int = TRUE,
          cor.coef = TRUE, cor.method = "kendall",
          xlab = "Region (demographic)", ylab = "Life expectancy (age)",
          main = "Correlation between Region and Life expectancy ")

# Method 2: Spearman's rho
ggscatter(Tau.Rho.data, x = "region", y = "lexp",
          add = "reg.line", conf.int = TRUE,
          cor.coef = TRUE, cor.method = "spearman",
          xlab = "Region (demographic)", ylab = "Life expectancy (age)",
          main = "Correlation between Region and Life expectancy ")
```

```
> # Method 1: Kendall's tau
> ggscatter(Tau.Rho.data, x = "region", y = "lexp",
+           add = "reg.line", conf.int = TRUE,
+           cor.coef = TRUE, cor.method = "kendall",
+           xlab = "Region (demographic)", ylab = "Life expectancy (age)",
+           main = "Correlation between Region and Life expectancy ")
`geom_smooth()` using formula 'y ~ x'
> # Method 2: Spearman's rho
> ggscatter(Tau.Rho.data, x = "region", y = "lexp",
+           add = "reg.line", conf.int = TRUE,
+           cor.coef = TRUE, cor.method = "spearman",
+           xlab = "Region (demographic)", ylab = "Life expectancy (age)",
+           main = "Correlation between Region and Life expectancy ")
`geom_smooth()` using formula 'y ~ x'
>
```

```
Rho.Corr.test          List of 8
Rho.Corr.test2         List of 8
Rho.Corr.test3         List of 8
Students.ExampleData   10 obs. of 6 variables
Tau.Corr.test          List of 8
Tau.Corr.test2         List of 8
Tau.Corr.test3         List of 8
```

**Fig. 12.12** **a** Plot for *Kendall’s tau* correlation (test for dependency) between two variables in R using the ggscatter() function. **b** Plot for *Spearman’s rho* correlation (test for association) between two variables in R using the ggscatter() function

```
# Step 4 - Visualize Correlation between the two variables

# Method 1: Kendall's tau
ggscatter(Tau.Rho.data, x = "region", y = "lexp",
          add = "reg.line", conf.int = TRUE,
          cor.coef = TRUE, cor.method = "kendall",
          xlab = "Region (demographic)", ylab = "Life expectancy (age)",
          main = "Correlation between Region and Life expectancy ")


# Method 2: Spearman's rho
ggscatter(Tau.Rho.data, x = "region", y = "lexp",
          add = "reg.line", conf.int = TRUE,
          cor.coef = TRUE, cor.method = "spearman",
          xlab = "Region (demographic)", ylab = "Life expectancy (age)",
          main = "Correlation between Region and Life expectancy ")
```

**# Step 5—Results Interpretation (*Kendall's Tau* and *Spearman's Rho*)**

The final step for the *Kendall's tau* (method 1) and *Spearman's rho* (method 2) correlation analysis is to interpret and understand the results of the tests.

By default, the hypothesis for conducting the tests (*Kendall's tau* and *Spearman's rho*) by considering the analyzed variables "**region**" and "**lexp**" in this particular example (see: Fig. 12.11b, c) is;

**Two-Tailed Kendall's Tau and Spearman's Rho Correlation Test**

- **($H_1$)** *IF* the *p*-value of the tests is less than or equal to 0.05 ($p \leq 0.05$), *THEN* we can assume that there is a dependency or association between the two variables (**region** and **lexp**). Thus, the population correlation coefficient ($\rho$) $\neq$ 0. Meaning that the population correlation coefficient is not 0, and consequently, we can assume that a non-zero correlation exist between the "**region**" and "**lexp**" variables.
- **($H_0$)** *ELSE IF* the *p*-value is greater than 0.05 ($p > 0.05$) THEN we can assume that there is no correlation (association) between the two variables. Thus, $\rho = 0$. Meaning that the population correlation coefficient is 0, and therefore, there is no association (correlation) between the two variables.

**One-Tailed Kendall's Tau and Spearman's Rho Correlation Test**

- **($H_1$)** *IF* the *p*-value of the test is less than or equal to 0.05 ($p \leq 0.05$), *THEN* we can statistically assume that the value of $\rho > 0$, i.e., the population correlation coefficient is greater than 0, thus, a positive correlation exist between the two analyzed variables.

  OR

  $\rho < 0$, i.e., the population correlation coefficient is less than 0, thus, a negative correlation exist between the two variables (**region** and **lexp**).

- **($H_0$)** *ELSE IF* the *p*-value is greater than 0.05 ($p > 0.05$) *THEN* we can conclude that there is no correlation between the two variables. Thus, $\rho = 0$. Meaning that the population correlation coefficient is 0, and therefore, there is no association (correlation) between the two variables.

```
> Tau.Corr.test <- cor.test(Tau.Rho.data$region, Tau.Rho.data$lexp,
method = "kendall")
> Tau.Corr.test

     Kendall's rank correlation tau

data:  Tau.Rho.data$region and Tau.Rho.data$lexp
z = -1.6415, p-value = 0.1007
alternative hypothesis: true tau is not equal to 0
sample estimates:
       tau
-0.1632955)
```

```
> Rho.Corr.test <- cor.test(Tau.Rho.data$region, Tau.Rho.data$lexp,
method = "spearman", exact=FALSE)
> Rho.Corr.test
     Spearman's rank correlation rho
data:  Tau.Rho.data$region and Tau.Rho.data$lexp
S = 62860, p-value = 0.1024
alternative hypothesis: true rho is not equal to 0
sample estimates:
       rho
-0.1997594)
```

As shown in the results above which is the outcome of the *Kendall's tau* (method 1) and *Spearman's rho* (method 2) correlation analysis (Two-tailed) for the example dataset (**Tau.Rho.data**) that we have reported in Fig. 12.11b, c; the meaning of the results of the **cor.test( )** method or function that we implemented to test the association or dependency between the **region** and **lexp** variables (stored as R objects "`Tau.Corr.test`" and "`Tau.Corr.test`") can be explained as a list containing the following:

**Method 1: Kendall's Tau**

- **Statistics**: `z = −1.6415` denotes the value of the Kendall's tau correlation analysis.
- ***p*-value**: `p-value = 0.1007` is the *p*-value (significance level) of the test.
- **Sample estimates**: `tau = −0.1632955` is the value of the population correlation coefficient.

**Method 2: Spearman's Rho**

- **Statistics**: `s = 62860` signifies the value of the Spearman's rho correlation analysis.
- ***p*-value**: `p-value = 0.1024` is the *p*-value (significance level) of the test.
- **Sample estimates**: `rho = −0.1997594` is the value of the population correlation coefficient.

Statistically, we can see that the *p*-value of both tests, i.e., the *Kendall's tau* (`Tau.Corr.test, z=-1.6415, p=0.1007`, method 1) and *Spearman's rho* (`Rho.Corr.test, s=62860, p = 0.1024`, method 2) correlation analysis (Two-tailed) are conventionally the same (p=0.1) and greater than the stated or scientifically acceptable significance levels ($p \leq 0.05$). Therefore, we reject the $H_1$ and accept $H_0$ by supposedly concluding that there is no dependency or association (correlation) between the two sets of analyzed variables (**region** and **lexp**) in the example data (**two-tailed test**).

Also, as shown in the next results reported below, and in Figs. 12.11b, c for the "one-tailed" correlation tests:

- We checked whether the correlation, if any? (in this case, no) may be a *positive* or *negative* (direction) correlation by considering the outcomes or output of the *Kendall's tau* and *Spearman's rho* tests, respectively.

**Method 1: Kendall's Tau Test for Positive or Negative Correlation (One-Tailed)**

```
> Tau.Corr.test2 <- cor.test(Tau.Rho.data$region, Tau.Rho.data$lexp,
method = "kendall", alternative = "greater")
> Tau.Corr.test2

      Kendall's rank correlation tau

data:  Tau.Rho.data$region and Tau.Rho.data$lexp
z = -1.6415, p-value = 0.9497
alternative hypothesis: true tau is greater than 0
sample estimates:
       tau
-0.1632955
```

```
> > Tau.Corr.test3 <- cor.test(Tau.Rho.data$region,
Tau.Rho.data$lexp, method = "kendall", alternative = "less")
> Tau.Corr.test3

      Kendall's rank correlation tau

data:  Tau.Rho.data$region and Tau.Rho.data$lexp
z = -1.6415, p-value = 0.05035
alternative hypothesis: true tau is less than 0
sample estimates:
       tau
-0.1632955)
```

**Method 2: Spearman's Rho Test for Positive or Negative Correlation (One-Tailed)**

```
> Rho.Corr.test2 <- cor.test(Tau.Rho.data$region, Tau.Rho.data$lexp,
method = "spearman", alternative = "greater", exact=FALSE)
> Rho.Corr.test2

      Spearman's rank correlation rho

data:  Tau.Rho.data$region and Tau.Rho.data$lexp
S = 62860, p-value = 0.9488
alternative hypothesis: true rho is greater than 0
sample estimates:
       rho
-0.1997594
```

```
> Rho.Corr.test3 <- cor.test(Tau.Rho.data$region, Tau.Rho.data$lexp,
method = "spearman", alternative = "less", exact=FALSE)
> Rho.Corr.test3

      Spearman's rank correlation rho

data:  Tau.Rho.data$region and Tau.Rho.data$lexp
S = 62860, p-value = 0.05121
alternative hypothesis: true rho is less than 0
sample estimates:
       rho
-0.1997594)
```

As gathered in the above results for the "one-tailed" test for *positive* and *negative* correlation (direction test) for the *Kendall's tau* (method 1) and *Spearman's rho* (method 2) tests; we can see that the results of the direction test (one-tailed) based on the *p*-values or estimated significance levels, i.e., $p \leq 0.05$, show that there is a negatively directed correlation between the targeted variables (`Tau.Corr.test3, p = 0.05035`) and (`Rho.Corr.test3, p = 0.05121`), respectively. Indeed, this is also reflected in the outcomes of the **two-tailed** test results (see Fig. 12.11b, c), therein we found that the sample estimates or population correlation coefficient ($\rho$) is less than 0, (i.e. Kendall tau, $\rho = -0.1632955$) and (Spearman rho, $\rho = -0.1997594$), and thus, it can be said in addition to the fact that there was no correction or association between the two analyzed variables (**region** and **lexp**), that a negatively directed correlation exists between the two variables (**region** and **lexp**).

## 12.4 Summary

In this chapter, the authors covered and demonstrated to the readers how to conduct the three main types of Correlational Analysis in R. This includes the practical illustration of how to perform the Pearson cor, Kendall's tau, and Spearman's rho correlation tests using R.

We illustrated how to conduct the *Pearson correlation* test, also known as the Pearson product–moment correlation coefficient in Sect. 12.2. While in Sect. 12.3, the chapter covered how to perform the *Kendall's tau* and *Spearman's rho* correlation tests.

Also, the chapter covered in each of the above sections (Sects. 12.2 and 12.3) how to graphically plot or visualize the correlation between two specified variables and/or the results of the correlational analysis. The content of the chapter also discussed in detail how to interpret and understand the results of the three main tests (Pearson, Kendall's tau, and Spearman's rho) in R.

In summary, the main contents covered in this chapter include:

- *Pearson correlation* (also known as Pearson Product–moment correlation coefficient) is a parametric procedure or statistical test of hypothesis used to compare the relationship that exists (linearity) between two sets of continuous (usually normally distributed) variables.
- *Kendall's tau* (also known as Kendall rank correlation coefficient) is described as a non-parametric procedure (distribution-free) or statistical test of hypothesis applied by the researchers to measure the strength of dependence or association between two categorical or ordinal variable types.
- *Spearman's rho* (also known as Spearman rank correlation coefficient) is equally described as non-parametric procedure (distribution-free) or statistical test of hypothesis applied by the researchers to measure the degree of association between two categorical or ordinal variable types.
- Both the *Kendall's tau* and *Spearman's rho* correlation tests are considered as the non-parametric versions or alternative to the *Pearson's correlation* test.

When choosing whether to conduct a Pearson, Kendall tau, or Spearman's rho correlation tests? The researcher or data analyst should:

- Perform the "*Pearson correlation*" if the targeted variables come from an independently sampled population, are normally distributed, in continuous data format, and shows or presents to be linearly related when plotted.
- Perform the "*Kendall's tau* or *Spearman's rho*" tests if the targeted variables come from an independently sampled population, are distribution-free (i.e., non-normally distributed), and in categorical or ordinal data format. Although it is noteworthy to mention that the two tests (i.e., Kendall's and Spearman's) can also be applied for discrete or interval datasets, as long as the dataset being analyzed has violated the test of assumptions such as data normality or homoscedasticity.
- In any case (be it Pearson, Kendall's tau, or Spearman's rho); the researchers or data analyst can perform a "one-tailed" correlational analysis to determine

the direction test (positive or negative) of the linear relationship or association/dependency (if there exist any) between the analyzed variables.

## References

Akoglu, H. (2018). User's guide to correlation coefficients. In *Turkish journal of emergency medicine* (Vol. 18, Issue 3, pp. 91–93). Emergency Medicine Association of Turkey. https://doi.org/10.1016/j.tjem.2018.08.001

Brossart, D. F., Laird, V. C., & Armstrong, T. W. (2018). Interpreting Kendall's Tau and Tau-U for single-case experimental designs. *Cogent Psychology, 5*(1), 1518687. https://doi.org/10.1080/23311908.2018.1518687

Couso, I., Strauss, O., & Saulnier, H. (2018). Kendall's rank correlation on quantized data: An interval-valued approach. *Fuzzy Sets and Systems, 343*, 50–64. https://doi.org/10.1016/j.fss.2017.09.003

Hauke, J., & Kossowski, T. (2011). Comparison of values of Pearson's and Spearman's correlation coefficient on the same sets of data. *Quaestiones Geographicae*, *30*(2), 87–93. https://repozytorium.amu.edu.pl/handle/10593/15580

Privitera, G. J. (2023). *Statistics for the behavioral sciences* (4th ed.). SAGE Publications, Inc. https://us.sagepub.com/en-us/nam/statistics-for-the-behavioral-sciences/book265576#contents

Puth, M. T., Neuhäuser, M., & Ruxton, G. D. (2014). Effective use of Pearson's product-moment correlation coefficient. In *Animal behaviour* (Vol. 93, pp. 183–189). Academic Press. https://doi.org/10.1016/j.anbehav.2014.05.003

Schober, P., & Schwarte, L. A. (2018). Correlation coefficients: Appropriate use and interpretation. *Anesthesia and Analgesia, 126*(5), 1763–1768. https://doi.org/10.1213/ANE.0000000000002864

Wang, B., Wang, R., & Wang, Y. (2019). Compatible matrices of Spearman's rank correlation. *Statistics and Probability Letters, 151*, 67–72. https://doi.org/10.1016/j.spl.2019.03.015

Zar, J. H. (2014). Spearman rank correlation: Overview. In *Wiley StatsRef: Statistics reference online*. Wiley. https://doi.org/10.1002/9781118445112.stat05964

# Chapter 13
# Wilcoxon Statistics in R: Signed-Rank Test and Rank-Sum Test

## 13.1 Introduction

The Wilcoxon test, which is of two types (i) *Wilcoxon Signed-rank* and (ii) *Wilcoxon Rank-sum* (Wilcoxon, 1945), is a non-parametric test and alternative version of the *t*-test (Rey & Neuhäuser, 2011). The test is mostly applied by the researchers to compare two samples by testing whether the median values of the data or variables differ significantly from each other. The resultant models assume that the data comes from two matched, or dependent populations, following the same distribution through time or place (Hayes, 2023). The test (Wilcoxon) can be applied to test the hypothesis that the *median* of a symmetrical distribution equals a given constant. And, as the name implies and as with the many other types of non-parametric tests that we have also previously covered in this book (see Chaps. 4, 6 and 12); this *distribution-free* test is based on ranks (Rey & Neuhäuser, 2011). It is expected that the independent variable in a Wilcoxon test is dichotomous, while the dependent variable is a continuous variable whose measurement is at least ordinal.

The main types and features or summary of the Wilcoxon test include (Hayes, 2023):

- The Wilcoxon test compares two paired or independent groups of variable and comes in two versions depending on the data groupings or scenario: (i) the rank-sum test and (ii) signed-rank test.
- The aim of the tests is to determine if two or more sets of pairs are different from one another in a statistically significant manner.
- Both tests (whether rank-sum or signed-rank) assume that the pairs in the data sample come from the same dependent populations.
- Unlike t-test that calculates the *mean difference* of two variables, the Wilcoxon test is used to calculate the *median difference* between two variables.

The "signed-ranked" version of the Wilcoxon test is calculated based on differences in the samples' median scores but in addition to it taking into account the signs of the differences, thus, takes into consideration the magnitudes of the observed differences. As the non-parametric equivalent of the *paired t-test*, the signed-rank can be used as an alternative to the t-test when the population data does not follow a normal distribution.

On the other hand, the Wilcoxon "rank-sum" test is often used as the non-parametric version or alternative to the *independent* or *two-sample t-test*.

Thus, the Wilcoxon rank-sum test is used to compare two independent samples, while Wilcoxon signed-rank test is used to compare two related samples.

The value of z in a Wilcoxon test is calculated with the following formula:

$$Z_T = \frac{T - \mu_T}{\sigma_T}$$

where:

- T = the sum of values from calculating the ranges of differences in the sample.

In the next sections of this chapter (Sects. 13.2 and 13.3); the authors will be demonstrating to the readers how to conduct the two main types of the Wilcoxon test (Signed-rank and Rank-Sum) in R. We will explain and illustrate the different steps and functions that are used to perform the test (Wilcoxon) in R by following the outlined steps in Fig. 13.1.

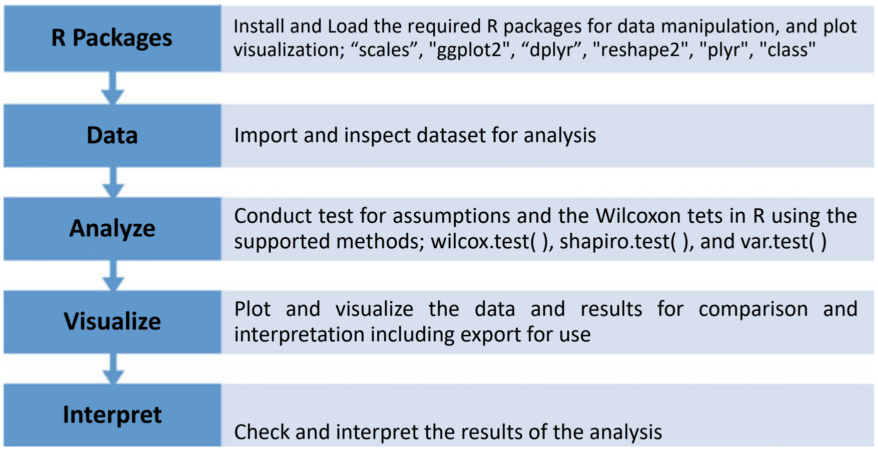

R Packages
Install and Load the required R packages for data manipulation, and plot visualization; "scales", "ggplot2", "dplyr", "reshape2", "plyr", "class"
Data
Import and inspect dataset for analysis
Analyze
Conduct test for assumptions and the Wilcoxon tets in R using the supported methods; wilcox.test( ), shapiro.test( ), and var.test( )
Visualize
Plot and visualize the data and results for comparison and interpretation including export for use
Interpret
Check and interpret the results of the analysis

**Fig. 13.1** Steps to conducting Wilcoxon tests in R

## 13.2 Signed-Rank Wilcoxon Test in R

*Signed-Rank Wilcoxon* test is used by the researchers to determine the median difference between two sets of data. It is used when the researchers or data analysts are interested in knowing the *difference in median* between two measures in a sample (e.g., pre and post tests, or before and after test).

By default, the hypothesis for testing whether there is a *difference in median* of the two (paired) data samples in Signed-Rank Wilcoxon tests is; *IF* the p-value is less than or equal to 0.05 ($p \leq 0.05$), *THEN* we assume that the median of the two sets of data or group of variables are statistically different and that this is not by chance ($H_1$), *ELSE IF* the p-value is greater than 0.05 ($p > 0.05$) *THEN* we can presume that there is no difference in the median of the two groups and any potential difference could only occur by chance ($H_0$).

The authors will demonstrate to the readers how to perform the Signed-Rank Wilcoxon tests used for two paired samples in R using the **wilcox.test( )**, **shapiro.test( )** and **var.test( )** functions.

As defined in the previous section (Sect. 13.1), we will do this using the steps outlined in Fig. 13.1.

To begin, **Open RStudio** and **Create a new or open an existing project**. Once the user has the RStudio and an R Project opened, **Create a new R Script** and name it "**Signed-Rank-Wilcoxon**" or any name the user may preferably choose (see Chaps. 1 and 2 if the user requires to refresh on how to do these steps).

Now, let's download an example data that we will use to demonstrate the two types of the Wilcoxon tests (Signed-Rank and Rank-Sum) in R. ***Note: the users are welcome to use any existing data or format they may wish to use for this illustration or analysis***. The example datasets the authors have used here are only for illustration purposes (users can see Chap. 2 for a step-by-step guide on how to work with different data types and format in R).

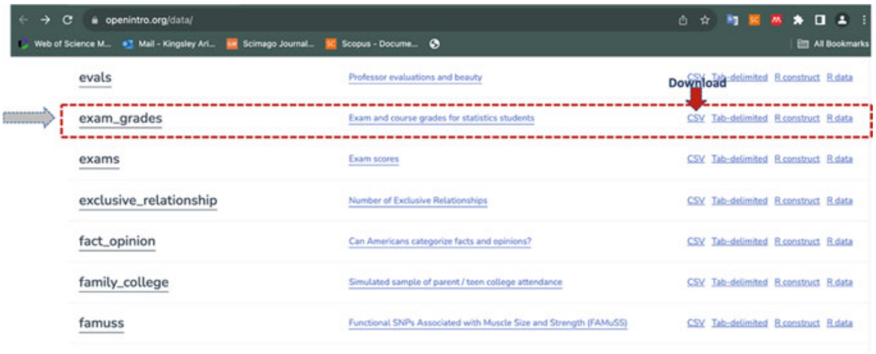
As shown in Fig. 13.2, download the example file named "**exam_grades.csv**" from the following source (https://www.openintro.org/data/) and save it on the local machine or computer. ***Note: the readers can also visit the following repository (https://doi.org/https://doi.org/10.6084/m9.figshare.24728073) where the authors have uploaded all the example files used in this book to directly access and download the file.

Once the user has downloaded the example file (exam_grades.csv) and saved this on the local machine or computer, we can proceed to conduct the first Wilcoxon test (*Signed-Rank Wilcoxon)* in R.

### # Step 1—Load the Required R Packages and Libraries

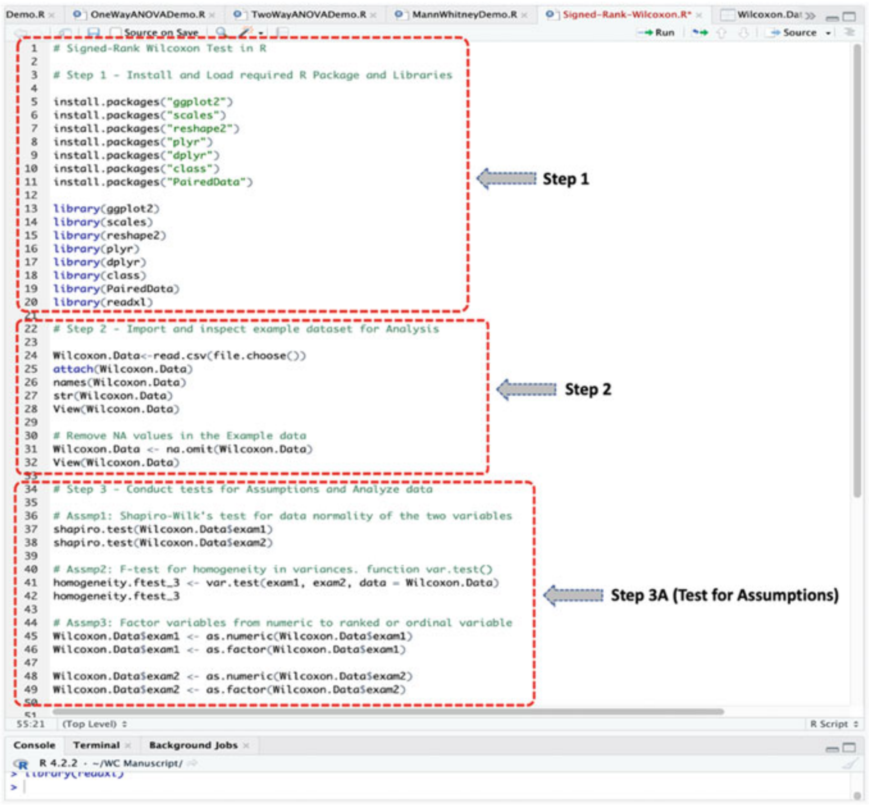
**Install** and **Load** the following *R packages* and *libraries* (see Fig. 13.3, Step1, Lines 3–20) that will be used to call and run the different R functions, data manipulations, and graphical visualizations for the Signed-Rank Wilcoxon analysis.

The code and syntax to install and load the required R packages are as follows (Fig. 13.3, Step 1, Lines 3–20):



| | | | | | |
|---|---|---|---|---|---|
| evals | Professor evaluations and beauty | CSV | Tab-delimited | R construct | R data |
| exam_grades | Exam and course grades for statistics students | CSV | Tab-delimited | R construct | R data |
| exams | Exam scores | CSV | Tab-delimited | R construct | R data |
| exclusive_relationship | Number of Exclusive Relationships | CSV | Tab-delimited | R construct | R data |
| fact_opinion | Can Americans categorize facts and opinions? | CSV | Tab-delimited | R construct | R data |
| family_college | Simulated sample of parent / teen college attendance | CSV | Tab-delimited | R construct | R data |
| famuss | Functional SNPs Associated with Muscle Size and Strength (FAMuSS) | CSV | Tab-delimited | R construct | R data |

**Fig. 13.2** Example of csv data download for Wilcoxon test. *Source* https://www.openintro.org/data/



```
# Signed-Rank Wilcoxon Test in R

# Step 1 - Install and Load required R Package and Libraries

install.packages("ggplot2")
install.packages("scales")
install.packages("reshape2")
install.packages("plyr")
install.packages("dplyr")
install.packages("class")
install.packages("PairedData")

library(ggplot2)
library(scales)
library(reshape2)
library(plyr)
library(dplyr)
library(class)
library(PairedData)
library(readxl)

# Step 2 - Import and inspect example dataset for Analysis

Wilcoxon.Data<-read.csv(file.choose())
attach(Wilcoxon.Data)
names(Wilcoxon.Data)
str(Wilcoxon.Data)
View(Wilcoxon.Data)

# Remove NA values in the Example data
Wilcoxon.Data <- na.omit(Wilcoxon.Data)
View(Wilcoxon.Data)

# Step 3 - Conduct tests for Assumptions and Analyze data

# Assmp1: Shapiro-Wilk's test for data normality of the two variables
shapiro.test(Wilcoxon.Data$exam1)
shapiro.test(Wilcoxon.Data$exam2)

# Assmp2: F-test for homogeneity in variances. function var.test()
homogeneity.ftest_3 <- var.test(exam1, exam2, data = Wilcoxon.Data)
homogeneity.ftest_3

# Assmp3: Factor variables from numeric to ranked or ordinal variable
Wilcoxon.Data$exam1 <- as.numeric(Wilcoxon.Data$exam1)
Wilcoxon.Data$exam1 <- as.factor(Wilcoxon.Data$exam1)

Wilcoxon.Data$exam2 <- as.numeric(Wilcoxon.Data$exam2)
Wilcoxon.Data$exam2 <- as.factor(Wilcoxon.Data$exam2)
```

```
> library(readxl)
>
```

**Fig. 13.3** Steps to conducting Wilcoxon test in R

```
install.packages("ggplot2")
install.packages("scales")
install.packages("reshape2")
install.packages("plyr")
install.packages("dplyr")
install.packages("class")
install.packages("PairedData")
```

```
library(ggplot2)
library(scales)
library(reshape2)
library(plyr)
library(dplyr)
library(class)
library(PairedData)
library(readxl)
```

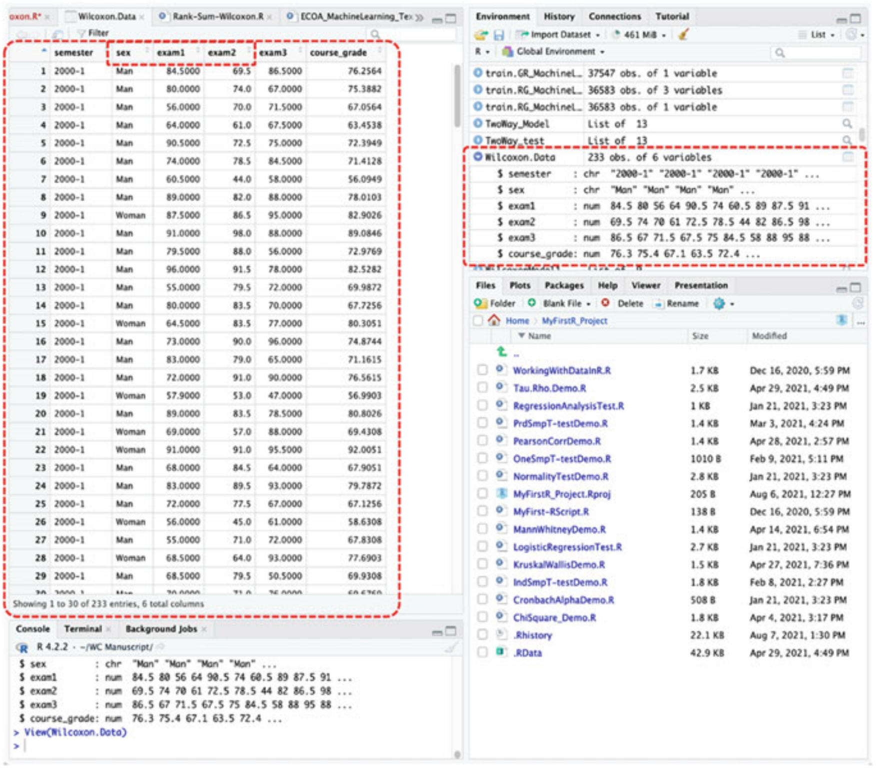
**# Step 2—Import and Inspect the Example Dataset for Wilcoxon Analysis**

As illustrated in Fig. 13.3 (Step 2, Lines 22–32), import the dataset named "**exam_grades.csv**" that we have downloaded earlier, and store this in an R object named "**Wilcoxon.Data**" (remember the users can use any name of they may preferably choose if they wish to do so).

Once the user has successfully imported the dataset, they will be able to view the details of the example dataset (**exam_grades.csv**) stored as R object we named "**Wilcoxon.Data**" in R as shown and highlighted in Fig. 13.4 with 233 observations and 6 variables (column) in the data sample.

```
Wilcoxon.Data <- read.csv(file.choose())
attach(Wilcoxon.Data)
names(Wilcoxon.Data)
str(Wilcoxon.Data)
View(Wilcoxon.Data)
```

***Note: a good scientific practice when working with datasets both in R or for research purposes is to clean up the dataset for use, e.g., by removing the NA or empty cells or values to ensure an accurate and reliable calculation or computation. For example, as shown in Step 2 in Fig. 13.3 (see Lines 30 to 32), the authors have used the following syntax and code to "remove the NA values" in the stored dataset (Wilcoxon.Data).



| | semester | sex | exam1 | exam2 | exam3 | course_grade |
|---|---|---|---|---|---|---|
| 1 | 2000-1 | Man | 84.5000 | 69.5 | 86.5000 | 76.2564 |
| 2 | 2000-1 | Man | 80.0000 | 74.0 | 67.0000 | 75.3882 |
| 3 | 2000-1 | Man | 56.0000 | 70.0 | 71.5000 | 67.0564 |
| 4 | 2000-1 | Man | 64.0000 | 61.0 | 67.5000 | 63.4538 |
| 5 | 2000-1 | Man | 90.5000 | 72.5 | 75.0000 | 72.3949 |
| 6 | 2000-1 | Man | 74.0000 | 78.5 | 84.5000 | 71.4128 |
| 7 | 2000-1 | Man | 60.5000 | 44.0 | 58.0000 | 56.0949 |
| 8 | 2000-1 | Man | 89.0000 | 82.0 | 88.0000 | 78.0103 |
| 9 | 2000-1 | Woman | 87.5000 | 86.5 | 95.0000 | 82.9026 |
| 10 | 2000-1 | Man | 91.0000 | 98.0 | 88.0000 | 89.0846 |
| 11 | 2000-1 | Man | 79.5000 | 88.0 | 56.0000 | 72.9769 |
| 12 | 2000-1 | Man | 96.0000 | 91.5 | 78.0000 | 82.5282 |
| 13 | 2000-1 | Man | 55.0000 | 79.5 | 72.0000 | 69.9872 |
| 14 | 2000-1 | Man | 80.0000 | 83.5 | 70.0000 | 67.7256 |
| 15 | 2000-1 | Woman | 64.5000 | 83.5 | 77.0000 | 80.3051 |
| 16 | 2000-1 | Man | 73.0000 | 90.0 | 96.0000 | 74.8744 |
| 17 | 2000-1 | Man | 83.0000 | 79.0 | 65.0000 | 71.1615 |
| 18 | 2000-1 | Man | 72.0000 | 91.0 | 90.0000 | 76.5615 |
| 19 | 2000-1 | Woman | 57.9000 | 53.0 | 47.0000 | 56.9903 |
| 20 | 2000-1 | Man | 89.0000 | 83.5 | 78.5000 | 80.8026 |
| 21 | 2000-1 | Woman | 69.0000 | 57.0 | 88.0000 | 69.4308 |
| 22 | 2000-1 | Woman | 91.0000 | 91.0 | 95.5000 | 92.0051 |
| 23 | 2000-1 | Man | 68.0000 | 84.5 | 64.0000 | 67.9051 |
| 24 | 2000-1 | Man | 83.0000 | 89.5 | 93.0000 | 79.7872 |
| 25 | 2000-1 | Man | 72.0000 | 77.5 | 67.0000 | 67.1256 |
| 26 | 2000-1 | Woman | 56.0000 | 45.0 | 61.0000 | 58.6308 |
| 27 | 2000-1 | Man | 55.0000 | 71.0 | 72.0000 | 67.8308 |
| 28 | 2000-1 | Woman | 68.5000 | 64.0 | 93.0000 | 77.6903 |
| 29 | 2000-1 | Man | 68.5000 | 79.5 | 50.5000 | 69.9308 |

Showing 1 to 30 of 233 entries, 6 total columns

```
train.GR_MachineL… 37547 obs. of 1 variable
train.RG_MachineL… 36583 obs. of 3 variables
train.RG_MachineL… 36583 obs. of 1 variable
TwoWay_Model       List of  13
TwoWay_test        List of  13
Wilcoxon.Data      233 obs. of 6 variables
   $ semester    : chr  "2000-1" "2000-1" "2000-1" "2000-1" ...
   $ sex         : chr  "Man" "Man" "Man" "Man" ...
   $ exam1       : num  84.5 80 56 64 90.5 74 60.5 89 87.5 91 ...
   $ exam2       : num  69.5 74 70 61 72.5 78.5 44 82 86.5 98 ...
   $ exam3       : num  86.5 67 71.5 67.5 75 84.5 58 88 95 88 ...
   $ course_grade: num  76.3 75.4 67.1 63.5 72.4 ...
```

| Name | Size | Modified |
|---|---|---|
| .. | | |
| WorkingWithDataInR.R | 1.7 KB | Dec 16, 2020, 5:59 PM |
| Tau.Rho.Demo.R | 2.5 KB | Apr 29, 2021, 4:49 PM |
| RegressionAnalysisTest.R | 1 KB | Jan 21, 2021, 3:23 PM |
| PrdSmpT-testDemo.R | 1.4 KB | Mar 3, 2021, 4:24 PM |
| PearsonCorrDemo.R | 1.4 KB | Apr 28, 2021, 2:57 PM |
| OneSmpT-testDemo.R | 1010 B | Feb 9, 2021, 5:11 PM |
| NormalityTestDemo.R | 2.8 KB | Jan 21, 2021, 3:23 PM |
| MyFirstR_Project.Rproj | 205 B | Aug 6, 2021, 12:27 PM |
| MyFirst-RScript.R | 138 B | Dec 16, 2020, 5:59 PM |
| MannWhitneyDemo.R | 1.4 KB | Apr 14, 2021, 6:54 PM |
| LogisticRegressionTest.R | 2.7 KB | Jan 21, 2021, 3:23 PM |
| KruskalWallisDemo.R | 1.5 KB | Apr 27, 2021, 7:36 PM |
| IndSmpT-testDemo.R | 1.8 KB | Feb 8, 2021, 2:27 PM |
| CronbachAlphaDemo.R | 508 B | Jan 21, 2021, 3:23 PM |
| ChiSquare_Demo.R | 1.8 KB | Apr 4, 2021, 3:17 PM |
| .Rhistory | 22.1 KB | Aug 7, 2021, 1:30 PM |
| .RData | 42.9 KB | Apr 29, 2021, 4:49 PM |

```
R 4.2.2 · ~/WC Manuscript/
 $ sex         : chr  "Man" "Man" "Man" "Man" ...
 $ exam1       : num  84.5 80 56 64 90.5 74 60.5 89 87.5 91 ...
 $ exam2       : num  69.5 74 70 61 72.5 78.5 44 82 86.5 98 ...
 $ exam3       : num  86.5 67 71.5 67.5 75 84.5 58 88 95 88 ...
 $ course_grade: num  76.3 75.4 67.1 63.5 72.4 ...
> View(Wilcoxon.Data)
>
```

**Fig. 13.4** Example dataset.csv imported and stored as an R object in R

```
# Remove NA values in the Example data
Wilcoxon.Data <- na.omit (Wilcoxon.Data)
View(Wilcoxon.Data)
```

***Now, when you use the view function `View(Wilcoxon.Data)` to visualize the example dataset again, you will notice that the program has removed the row that contain the "NA" value under the "exam1" variable. Consequentially, the user will also notice in the Environment Tab that there are now a total of "232 observations" and 6 variables (column) in the cleaned data sample (Wilcoxon.Data).

## # Step 3—Conduct Tests for Assumptions and Analyze Data

As shown in Fig. 13.5 (Step 3A, Lines 34–49), we first conducted the various necessary tests of assumptions (e.g., data normality and homogeneity of variance) for the selected items or variables (i.e., "**exam1**" and "**exam2**"—see Fig. 13.4) in R before proceeding to perform the main analysis (Signed-Rank Wilcoxon test—Step 3B). The assumption test done in Step 3A (Fig. 13.4) is to ensure that the data does not meet (violates) the data normality or homogeneity of variance condition, which are prerequisite to carrying out the non-parametric tests such as the Wilcoxon test.

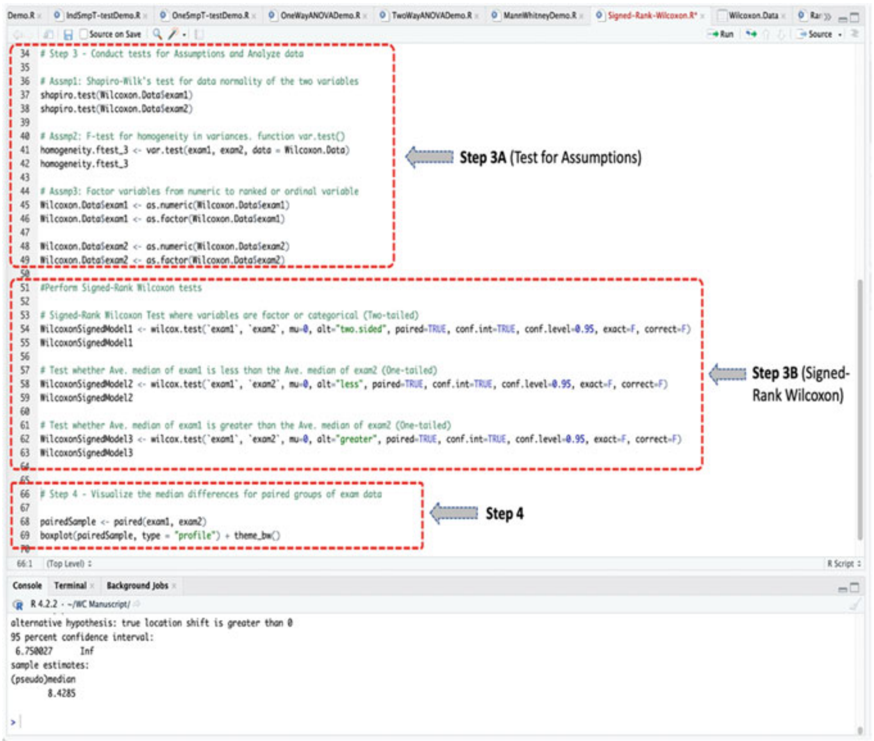
The test of assumptions (Step 3A, Fig. 13.5) and Signed-Rank Wilcoxon test defined in Step B (see Fig. 13.5, Lines 51–63) is done by using the **shapiro.test( ), var.test( )**, and **wilcox.test( )** functions in R.

As defined in the Introduction section (Sect. 13.1);

- **Signed-Rank Wilcoxon test** statistics compares the median for two sets of data from a single population but analyzed at different time intervals (e.g., pre and post test, before and after, etc.).
- The targeted variables must be measured in ranked or ordinal scale. Thus, it is assumed that the independent variable in a Wilcoxon test is dichotomous, and the dependent variable is a continuous variable whose measurement is at least ordinal.

To illustrate the Signed-Rank Wilcoxon test using the example dataset we stored as "**Wilcoxon.Data**" (see: highlighted columns in Fig. 13.4) we will:

1. Test whether the median of the grades for the "**exam1**" variable is *equal to* the median of the "**exam2**" variable in the dataset? **(two-tailed test)**.



```
34  # Step 3 - Conduct tests for Assumptions and Analyze data
35
36  # Assmp1: Shapiro-Wilk's test for data normality of the two variables
37  shapiro.test(Wilcoxon.Data$exam1)
38  shapiro.test(Wilcoxon.Data$exam2)
39
40  # Assmp2: F-test for homogeneity in variances. function var.test()
41  homogeneity.ftest_3 <- var.test(exam1, exam2, data = Wilcoxon.Data)
42  homogeneity.ftest_3
43
44  # Assmp3: Factor variables from numeric to ranked or ordinal variable
45  Wilcoxon.Data$exam1 <- as.numeric(Wilcoxon.Data$exam1)
46  Wilcoxon.Data$exam1 <- as.factor(Wilcoxon.Data$exam1)
47
48  Wilcoxon.Data$exam2 <- as.numeric(Wilcoxon.Data$exam2)
49  Wilcoxon.Data$exam2 <- as.factor(Wilcoxon.Data$exam2)
50
51  #Perform Signed-Rank Wilcoxon tests
52
53  # Signed-Rank Wilcoxon Test where variables are factor or categorical (Two-tailed)
54  WilcoxonSignedModel1 <- wilcox.test(`exam1`, `exam2`, mu=0, alt="two.sided", paired=TRUE, conf.int=TRUE, conf.level=0.95, exact=F, correct=F)
55  WilcoxonSignedModel1
56
57  # Test whether Ave. median of exam1 is less than the Ave. median of exam2 (One-tailed)
58  WilcoxonSignedModel2 <- wilcox.test(`exam1`, `exam2`, mu=0, alt="less", paired=TRUE, conf.int=TRUE, conf.level=0.95, exact=F, correct=F)
59  WilcoxonSignedModel2
60
61  # Test whether Ave. median of exam1 is greater than the Ave. median of exam2 (One-tailed)
62  WilcoxonSignedModel3 <- wilcox.test(`exam1`, `exam2`, mu=0, alt="greater", paired=TRUE, conf.int=TRUE, conf.level=0.95, exact=F, correct=F)
63  WilcoxonSignedModel3
64
65
66  # Step 4 - Visualize the median differences for paired groups of exam data
67
68  pairedSample <- paired(exam1, exam2)
69  boxplot(pairedSample, type = "profile") + theme_bw()
70
```

```
alternative hypothesis: true location shift is greater than 0
95 percent confidence interval:
 6.750027      Inf
sample estimates:
(pseudo)median
       8.4285

>
```

**Fig. 13.5** Conducting signed-rank Wilcoxon test in R

2. Test whether the median of the "**exam1**" grades is *less than* the median of the "**exam2**"? **(one-tailed test)**.
3. Test whether the median of the "**exam1**" grades is *greater than* the median of the "**exam2**"? **(one-tailed test)**.

Accordingly, the syntax to performing the above-listed tests in R (see Fig. 13.5, Steps 3A and 3B, Lines 34–63) are as shown in the codes below:

```
# Assmp1: Shapiro-Wilk's test for data normality of the two variables
shapiro.test(Wilcoxon.Data$exam1)
shapiro.test(Wilcoxon.Data$exam2)

# Assmp2: F-test for homogeneity in variances. function var.test()
homogeneity.ftest_3 <- var.test(exam1, exam2, data = Wilcoxon.Data)
homogeneity.ftest_3

# Assmp3: Factor variables from numeric to ranked or ordinal variable
Wilcoxon.Data$exam1 <- as.numeric(Wilcoxon.Data$exam1)
Wilcoxon.Data$exam1 <- as.factor(Wilcoxon.Data$exam1)

Wilcoxon.Data$exam2 <- as.numeric(Wilcoxon.Data$exam2)
Wilcoxon.Data$exam2 <- as.factor(Wilcoxon.Data$exam2)
```

**#Perform Signed-Rank Wilcoxon tests**

```
# Signed-Rank Wilcoxon Test where variables are factor or categorical (Two-tailed)
WilcoxonSignedModel1 <- wilcox.test(`exam1`, `exam2`, mu=0, alt="two.sided",
paired=TRUE, conf.int=TRUE, conf.level=0.95, exact=F, correct=F)
WilcoxonSignedModel1

# Test whether Ave. median of exam1 is less than the Ave. median of exam2 (One-tailed)
WilcoxonSignedModel2 <- wilcox.test(`exam1`, `exam2`, mu=0, alt="less", paired=TRUE,
conf.int=TRUE, conf.level=0.95, exact=F, correct=F)
WilcoxonSignedModel2

# Test whether Ave. median of exam1 is greater than the Ave. median of exam2 (One-tailed)
WilcoxonSignedModel3 <- wilcox.test(`exam1`, `exam2`, mu=0, alt="greater",
paired=TRUE, conf.int=TRUE, conf.level=0.95, exact=F, correct=F)
WilcoxonSignedModel3
```

**Useful Tip:**

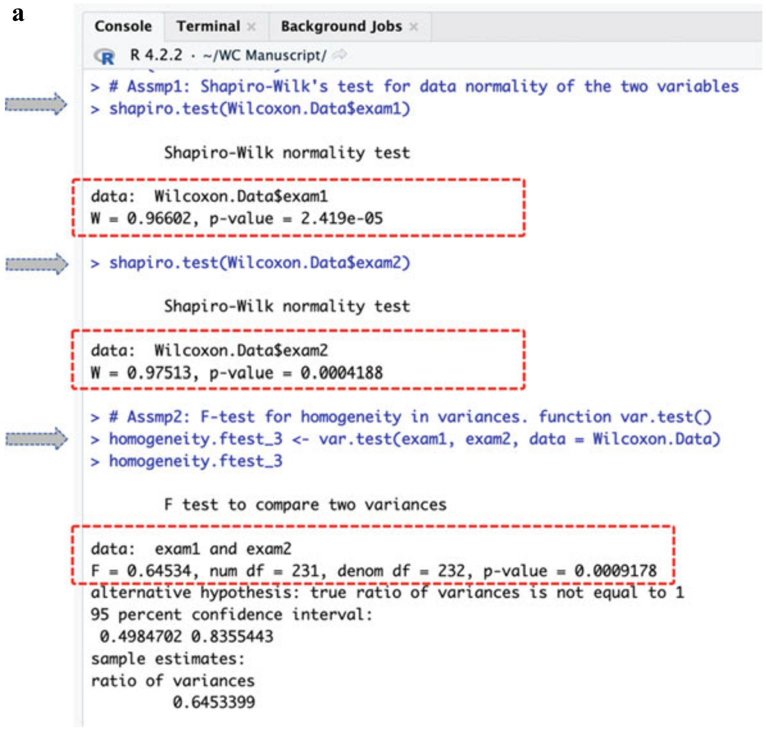
- As described in the codes and figure above (Fig. 13.5), the users are always required to specify the `paired = TRUE` option when conducting the Signed-Rank Wilcoxon test, which represents as the alternative (non-parametric equivalent) to the Paired Sample *t*-test.
- Use the `alt = "less"` and `alt = "greater"` options to specify a "one-tailed" *t*-test.

Once the user has successfully run the codes defined in **Steps 3A** and **3B** (Lines 34–63) in Fig. 13.5, they will be presented with the results of the "tests for assumptions" (Step 3A) and the "Signed-Rank Wilcoxon test" (Step 3B) in the Console as shown in Figs. 13.6a and b, respectively.

In Fig. 13.6a which represents as the result or outcome of Step 3A (see: Fig. 13.5), we conducted the different necessary assumptions tests for the Wilcoxon test in order to determine if the available dataset and variables are valid to perform the test.

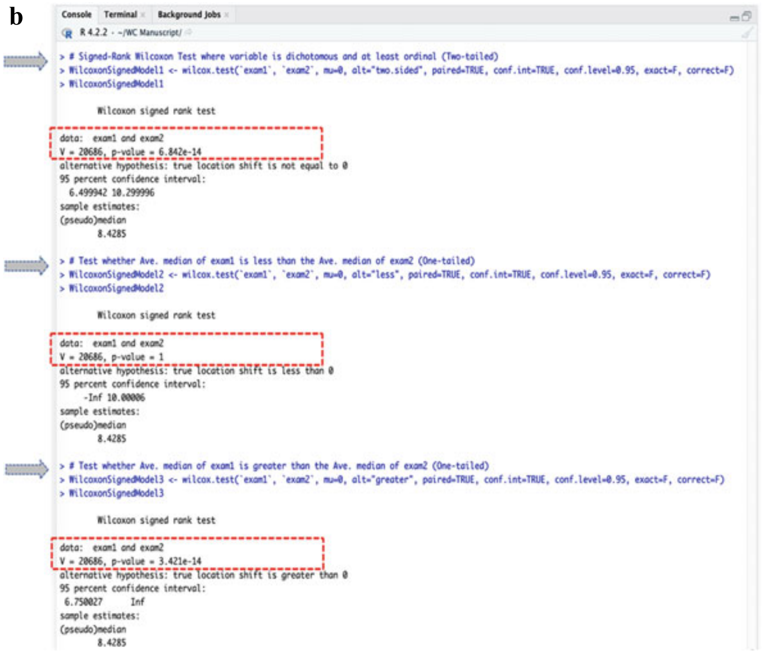
As highlighted in the figure (Fig. 13.6a), the **normality test** (Assmp1) by using the Shapiro–Wilk's method **shapiro.test( )** whereby we hypothetically assume that a

a



```
> # Assmp1: Shapiro-Wilk's test for data normality of the two variables
> shapiro.test(Wilcoxon.Data$exam1)

	Shapiro-Wilk normality test

data:  Wilcoxon.Data$exam1
W = 0.96602, p-value = 2.419e-05

> shapiro.test(Wilcoxon.Data$exam2)

	Shapiro-Wilk normality test

data:  Wilcoxon.Data$exam2
W = 0.97513, p-value = 0.0004188

> # Assmp2: F-test for homogeneity in variances. function var.test()
> homogeneity.ftest_3 <- var.test(exam1, exam2, data = Wilcoxon.Data)
> homogeneity.ftest_3

	F test to compare two variances

data:  exam1 and exam2
F = 0.64534, num df = 231, denom df = 232, p-value = 0.0009178
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
 0.4984702 0.8355443
sample estimates:
ratio of variances
         0.6453399
```

**Fig. 13.6** **a** Results of data normality and homogeneity of variance test displayed in the console in R. **b** Results of signed-rank Wilcoxon test displayed in Console in R b



```
> # Signed-Rank Wilcoxon Test where variable is dichotomous and at least ordinal (Two-tailed)
> WilcoxonSignedModel1 <- wilcox.test(`exam1`, `exam2`, mu=0, alt="two.sided", paired=TRUE, conf.int=TRUE, conf.level=0.95, exact=F, correct=F)
> WilcoxonSignedModel1

	Wilcoxon signed rank test

data:  exam1 and exam2
V = 20686, p-value = 6.842e-14
alternative hypothesis: true location shift is not equal to 0
95 percent confidence interval:
  6.499942 10.299996
sample estimates:
(pseudo)median
        8.4285

> # Test whether Ave. median of exam1 is less than the Ave. median of exam2 (One-tailed)
> WilcoxonSignedModel2 <- wilcox.test(`exam1`, `exam2`, mu=0, alt="less", paired=TRUE, conf.int=TRUE, conf.level=0.95, exact=F, correct=F)
> WilcoxonSignedModel2

	Wilcoxon signed rank test

data:  exam1 and exam2
V = 20686, p-value = 1
alternative hypothesis: true location shift is less than 0
95 percent confidence interval:
     -Inf 10.00006
sample estimates:
(pseudo)median
        8.4285

> # Test whether Ave. median of exam1 is greater than the Ave. median of exam2 (One-tailed)
> WilcoxonSignedModel3 <- wilcox.test(`exam1`, `exam2`, mu=0, alt="greater", paired=TRUE, conf.int=TRUE, conf.level=0.95, exact=F, correct=F)
> WilcoxonSignedModel3

	Wilcoxon signed rank test

data:  exam1 and exam2
V = 20686, p-value = 3.421e-14
alternative hypothesis: true location shift is greater than 0
95 percent confidence interval:
 6.750027      Inf
sample estimates:
(pseudo)median
        8.4285
```

**Fig. 13.6** (continued)

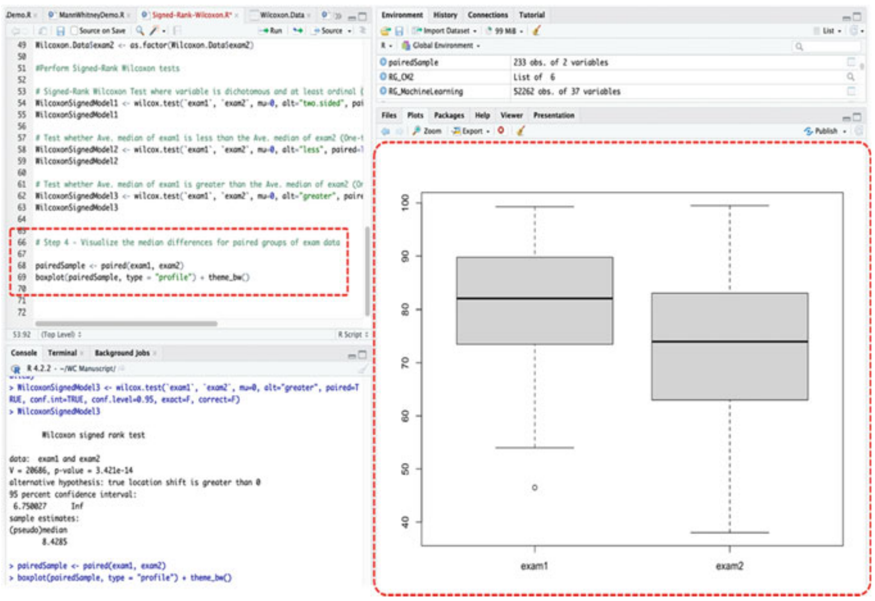
score of *p-value > 0.05* is normal, shows that the distribution of the two sets of data or variables (i.e., `"exam1" where W=0.96602, p-value=2.419e-05`, and `"exam2" where W=097513, p-value=0.0004188)` are not normality distributed.

Also, the **homogeneity of variance** test (Assmp2) for the two variables (`exam1` and `exam2`) using the **var.test( )** method, whereby we assume that a value of $p > 0.05$ indicates equality in variance, shows that there is difference in the variance for the two variables (`exam1`, `exam2`) with `p-value=0.0009178` and `F=0.64534`.

Thus, we presume that the data normality and assumption of equality in variance are not met, and proceed to conduct the Signed-rank Wilcoxon test.

Consequently, Fig. 13.6b is the result of the Signed-Rank Wilcoxon test and statistics as described in Step 3B in Fig. 13.5 (Lines 51–63).

As reported in Fig. 13.6b, we conducted the Signed-Rank Wilcoxon test by testing the median differences for the two target variables (`exam1`, `exam2`). The results of the test were stored in an R object we defined as `"WilcoxonSignedModel1"` for the **two-tailed** analysis, and `"WilcoxonSignedModel2"` and `"WilcoxonSignedModel3"` for the **one-tailed** analysis, respectively. The meaning of the results is discussed in detail in Step 5 in this section.



```
49 Wilcoxon.Data$exam2 <- as.factor(Wilcoxon.Data$exam2)
50
51 #Perform Signed-Rank Wilcoxon tests
52
53 # Signed-Rank Wilcoxon Test where variable is dichotomous and at least ordinal (
54 WilcoxonSignedModel1 <- wilcox.test(`exam1`, `exam2`, mu=0, alt="two.sided", pai
55 WilcoxonSignedModel1
56
57 # Test whether Ave. median of exam1 is less than the Ave. median of exam2 (One-t
58 WilcoxonSignedModel2 <- wilcox.test(`exam1`, `exam2`, mu=0, alt="less", paired=1
59 WilcoxonSignedModel2
60
61 # Test whether Ave. median of exam1 is greater than the Ave. median of exam2 (Or
62 WilcoxonSignedModel3 <- wilcox.test(`exam1`, `exam2`, mu=0, alt="greater", paire
63 WilcoxonSignedModel3
64
65
66 # Step 4 - Visualize the median differences for paired groups of exam data
67
68 pairedSample <- paired(exam1, exam2)
69 boxplot(pairedSample, type = "profile") + theme_bw()
70
71
72
```

```
> WilcoxonSignedModel3 <- wilcox.test(`exam1`, `exam2`, mu=0, alt="greater", paired=T
RUE, conf.int=TRUE, conf.level=0.95, exact=F, correct=F)
> WilcoxonSignedModel3

        Wilcoxon signed rank test

data:  exam1 and exam2
V = 20686, p-value = 3.421e-14
alternative hypothesis: true location shift is greater than 0
95 percent confidence interval:
 6.750027      Inf
sample estimates:
(pseudo)median
        8.4285

> pairedSample <- paired(exam1, exam2)
> boxplot(pairedSample, type = "profile") + theme_bw()
```

Fig. 13.7 Plot for median difference for two paired group of variables in R

## # Step 4—Plot and Visualize the Mean Differences for the Two Paired Variables

Another good way to determine the median differences between two variables is to graphically plot or represent it. As defined in Step 4 in Fig. 13.5 (Lines 66–69) and the resultant chart represented in Fig. 13.7; the authors graphically represented or visualized the median between the two paired groups of variables (`exam1`, `exam2`) in the example dataset "**Wilcoxon.Data**" by plotting them using the **paired( )** and **boxplot( )** functions in R.

The code used to plot the median of the two variables (`exam1`, `exam2`) is as shown below, and the resultant graph is as presented in Fig. 13.7.

```
# Visualize median differences for paired groups of data
pairedSample <- paired(exam1, exam2)
boxplot(pairedSample, type = "profile") + theme_bw()
```

## # Step 5—Results' Interpretation for Signed-Rank Wilcoxon Test

The final step for the "Signed-Rank Wilcoxon" test statistics is to interpret and understand the results of the analysis.

By default, the hypothesis for conducting the test (Signed-Rank Wilcoxon) is; *IF* the p-value is less than or equal to 0.05 ($p \leq 0.05$), *THEN* we assume that the median of the two set of variables or analyzed data are statistically different and not by chance ($H_1$), *ELSE IF* the p-value is greater than 0.05 ($p > 0.05$) *THEN* we can say that there is no difference in the median of the two sets of data ($H_0$).

```
> WilcoxonSignedModel1
        Wilcoxon signed rank test
data:  exam1 and exam2
V = 20686, p-value = 6.842e-14
alternative hypothesis: true location shift is not equal to 0
95 percent confidence interval:
  6.499942 10.299996
sample estimates:
(pseudo)median
      8.4285
```

As reported in the above statistics and outcome of the Wilcoxon test (Signed-Rank)—Figs. 13.6b; the meaning of the results of the test by using the **wilcox.test( )** function in R can be explained as a list containing the following:

- **Statistics**: `v = 20686` which denotes the value of the *t*-test analysis.
- **p-value**: `p-value = 6.842e-14` is the significance level of the test.
- **Confidence interval:** `Conf.Int(95%, 6.499942 10.299996)` represents the confidence interval for the median assumed to be appropriate to the specified alternative hypothesis.
- **Sample estimates:** `(pseudo)media = 8.4285` is the estimated median of the two groups of variables, e.g., compared by considering the two variables (`exam1`, `exam2`).

Accordingly, the outcome of the **wilcox.test( )** function used for the **Two-tailed** Singed-Rank test or model (`WilcoxonSignedModel1`) shows that there exist a difference in the median between the two sets of analyzed variable (`exam1`, `exam2`). The p-value for the "**Two-tailed**" test was statistically found to be `p=6.842e-14 (V=20686)`, which is significantly less than the scientifically accepted levels ($p \leq 0.05$). Therefore, we conclude that there is a significant difference between the median of the two sets of exam grades (`exam1`, `exam2`) across the analyzed period or data.

Furthermore, as gathered in the results for the **one-tailed** Signed-Rank tests described below (see Fig. 13.6b);

- In our analysis, we also checked whether the median of the "**exam1**" variable is *less than* the median of the "**exam2**" (`WilcoxonSignedModel2`).
- Then checked whether the median of the "**exam1**" is *greater than* the median of the "**exam2**" (`WilcoxonSignedModel3`).

```
> WilcoxonSignedModel2

      Wilcoxon signed rank test

data:  exam1 and exam2
V = 20686, p-value = 1
alternative hypothesis: true location shift is less than 0
95 percent confidence interval:
     -Inf 10.00006
sample estimates:
(pseudo)median
             8.4285
```

```
> WilcoxonSignedModel3

      Wilcoxon signed rank test

data:  exam1 and exam2
V = 20686, p-value = 3.421e-14
alternative hypothesis: true location shift is greater than 0
95 percent confidence interval:
 6.750027      Inf
sample estimates:
(pseudo)median
        8.4285
```

In the results of the **one-tailed** tests presented above, we can see that when we analyzed whether the median of the "**exam1**" variable is *less than* the median of the "**exam2**" that there was no significant difference (`WilcoxonSignedModel2, V=20686, p=1`). But when we analyzed whether the median of the "**exam1**" is *greater than* the median of the "**exam2**" there was a significant difference (`WilcoxonSignedModel3, V=20686, p=3.421e-14`) (i.e., p $\leq$ 0.05). Therefore, it can be said from the results of the one-tailed Wilcoxon tests that the median of the "**exam1**" is *greater* and *not less* than the median of the "**exam2**" grades which was statistically differenced (sample estimates) by margin of `8.4285` (pseudo median of the differences) presented in both tests (one-tailed).

Therefore, in summary, we can statistically say that there was a significant difference or variation in the *median* of the grades or targeted variables (exam1 and exam2) across the periods of the exams based on the example dataset that we stored as "**Wilxoxon.Data**" in R. For example, this result suggests a change in the grades of the students or participants across those periods. Moreover, the average median of the "**exam1**" represents to be *greater* (and *not less*) than the median of the "**exam2**" in the analyzed data.

## 13.3 Rank-Sum Wilcoxon Test in R

*Rank-Sum Wilcoxon test* (also referred to as the non-parametric equivalent or alternative to the "Unpaired" or "Two-sample" or "Independent-sample" t-test) is used when the dataset the researcher or data analysts wants to analyze are of two types or sample, and are statistically *independent*. In essence, the "Rank-Sum Wilcoxon test" is used to compare the median of two independent groups of variables or data samples. As the non-parametric equivalent of the Independent sample t-test, it (Sum-Rank) allows the researchers to *compare the median* of two distinctive sets of data randomly drawn from two different populations, when the dataset in question violates the necessary conditions to perform the Independent sample t-test or is said to contain outliers.

By default, the hypothesis for testing whether there is a *difference in median* of the two (independent) data samples is; *IF* the p-value is less than or equal to 0.05 ($p \leq 0.05$), *THEN* we assume that the median of the two sets of data or groups of population in the sample are statistically different and that this is not by chance ($H_1$), *ELSE IF* the p-value is greater than 0.05 ($p > 0.05$) *THEN* we can presume that there is no difference in the median of the two groups of data and any difference observed could only occur by chance ($H_0$).

In this section, the authors will demonstrate to the readers how to conduct the Rank-Sum (unpaired sample) Wilcoxon test in R using the **shapiro.test( ), var.test( )** and **wilcox.test( )** functions in R.

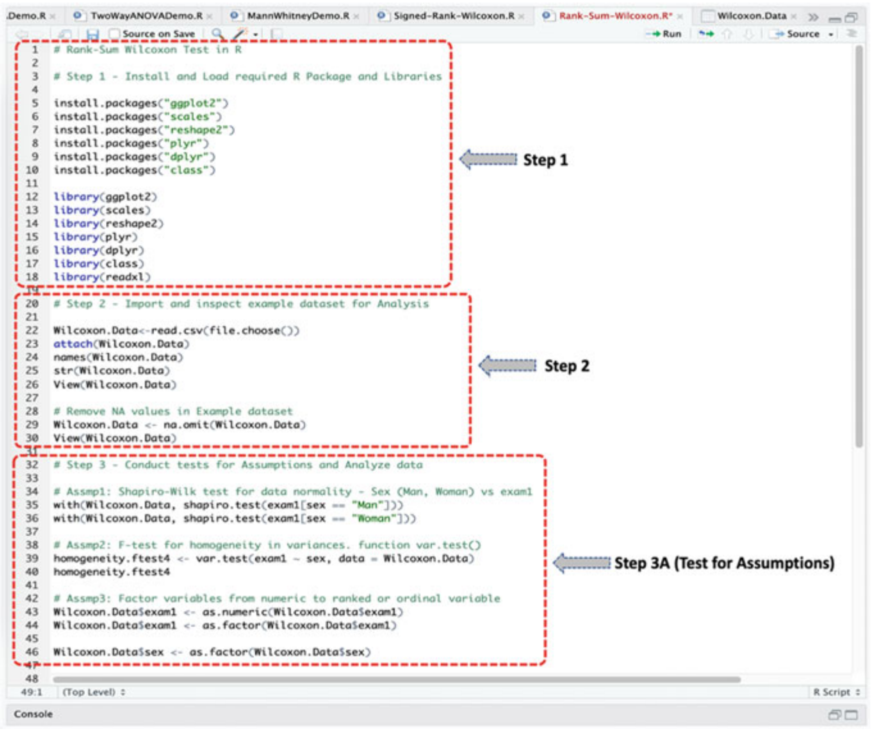
As defined earlier in the previous section (Sect. 13.1), we will do this by using the computational steps outlined in Fig. 13.1.

To begin, **Create a new or open an existing R project**. Once the user has the RStudio and an R Project opened, **Create a new R Script** and name it "**Sum-Rank-Wilcoxon**" or any name the user may preferably choose (see Chaps. 1 and 2 for step-by-step guide on how to do these steps if required).

Next, we will continue to use the example dataset (**exam_grades.csv**) (see Figs. 13.2 and 13.4) that we downloaded earlier and stored as R object we named or defined as "Wilcoxon.Data" in R to illustrate the Rank-Sum Wilcoxon test. ***Note: the readers can refer to the following repository (https://doi.org/https://doi.org/10.6084/m9.figshare.24728073) where the authors have uploaded all the example files used in this book or if they have not practiced the previous example the authors presented in Sect. 13.2.

### # Step 1—Install and Load the Required R Packages

Since we have previously installed the necessary R packages in our previous example in Sect. 13.2, we do not necessarily need to install the different R packages again, rather we just need to "load" the libraries for the necessary packages (see Lines 12 to 18, Step 1, Fig. 13.8). However, if the user has directly visited this section or has exited or not practiced the previous example in Sect. 13.2, then they would need to "Install and load" the necessary R packages as defined in Lines 5–18 in Step 1 (Fig. 13.8).



```
# Rank-Sum Wilcoxon Test in R

# Step 1 - Install and Load required R Package and Libraries

install.packages("ggplot2")
install.packages("scales")
install.packages("reshape2")
install.packages("plyr")
install.packages("dplyr")
install.packages("class")

library(ggplot2)
library(scales)
library(reshape2)
library(plyr)
library(dplyr)
library(class)
library(readxl)

# Step 2 - Import and inspect example dataset for Analysis

Wilcoxon.Data<-read.csv(file.choose())
attach(Wilcoxon.Data)
names(Wilcoxon.Data)
str(Wilcoxon.Data)
View(Wilcoxon.Data)

# Remove NA values in Example dataset
Wilcoxon.Data <- na.omit(Wilcoxon.Data)
View(Wilcoxon.Data)

# Step 3 - Conduct tests for Assumptions and Analyze data

# Assmp1: Shapiro-Wilk test for data normality - Sex (Man, Woman) vs exam1
with(Wilcoxon.Data, shapiro.test(exam1[sex == "Man"]))
with(Wilcoxon.Data, shapiro.test(exam1[sex == "Woman"]))

# Assmp2: F-test for homogeneity in variances. function var.test()
homogeneity.ftest4 <- var.test(exam1 ~ sex, data = Wilcoxon.Data)
homogeneity.ftest4

# Assmp3: Factor variables from numeric to ranked or ordinal variable
Wilcoxon.Data$exam1 <- as.numeric(Wilcoxon.Data$exam1)
Wilcoxon.Data$exam1 <- as.factor(Wilcoxon.Data$exam1)

Wilcoxon.Data$sex <- as.factor(Wilcoxon.Data$sex)
```

**Fig. 13.8** Steps to conducting Rank-Sum Wilcoxon test in R

Depending on the users case scenario, **Install** or **Load** the following *R packages* and *libraries* (Fig. 13.8, Step 1), that we will be using to call the different R functions, data manipulations, and graphical visualizations for the Rank-Sum Wilcoxon test.

The code and syntax to install and load the required R packages are as follows:

```
install.packages("ggplot2")
install.packages("scales")
install.packages("reshape2")
install.packages("plyr")
install.packages("dplyr")
install.packages("class")

library(ggplot2)
library(scales)
library(reshape2)
library(plyr)
library(dplyr)
library(class)
library(readxl)
```

## # Step 2—Import and Inspect Example Dataset for Analysis

As illustrated in Step 2 in Fig. 13.8 (Lines 20–30); import and/or load the dataset named "**exam_grades.csv**" which we have previously downloaded and stored as an R object we named "**Wilcoxon.Data**" in R (see Figs. 13.2 and 13.4).

Depending on whether the user has continued in our previous example, or has exited or visited this particular section; the code to import and/or attach the example dataset is provided below:

```
Wilcoxon.Data <- read.csv(file.choose())
attach(Wilcoxon.Data)
names(Wilcoxon.Data)
str(Wilcoxon.Data)
View(Wilcoxon.Data)

# Remove NA values in Example dataset
Wilcoxon.Data <- na.omit(Wilcoxon.Data)
View(Wilcoxon.Data)
```

Once the user has successfully imported or loaded the dataset, you will be able to view the details of the dataset (exam_grades.csv) stored as R object "**Wilcoxon.Data**" in the R environment as shown in Figs. 13.2 and 13.4, respectively, with 233 observations and 6 variables (column) in the data sample.

## # Step 3—Conduct Tests for Assumptions and Analyze Data

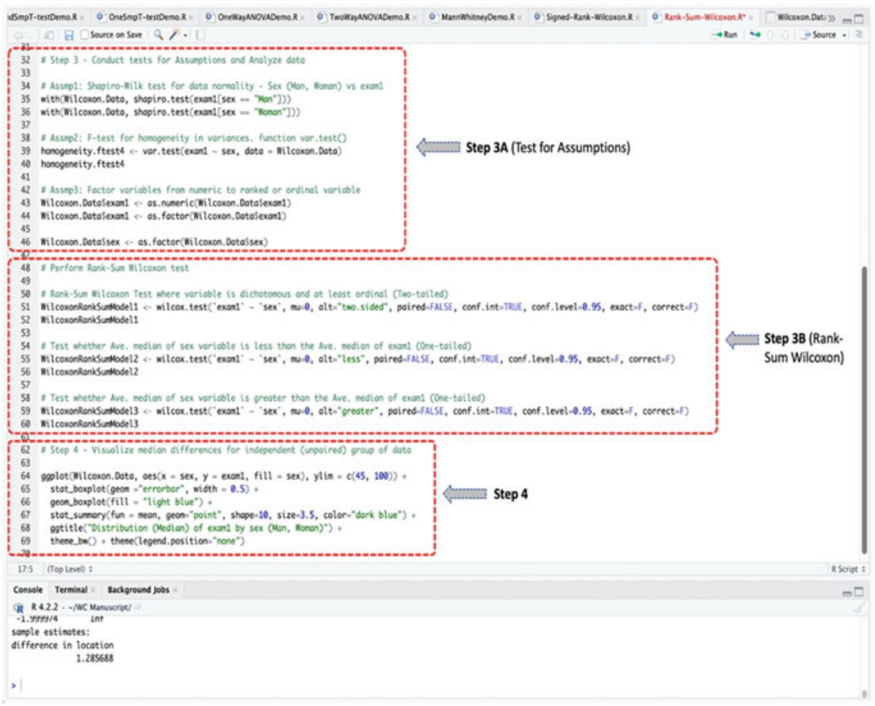
Now that we have imported the dataset and/or loaded it ready for analysis (Wilcoxon.Data), we can proceed to analyze the data.

As defined in Step 3A in Fig. 13.8 (Lines 32–46), we first conducted the various necessary tests of assumptions (e.g., data normality and homogeneity of variance) for the chosen items or variables (i.e., "**sex**" and "**exam1**"—see Fig. 13.4) in R before proceeding to perform the analysis (Rank-Sum Wilcoxon test – Step 3B). The assumption tests we performed in Step 3A (Fig. 13.8) are to ensure that the data does not meet or violate the data normality or homogeneity of variance condition which are preconditions to carrying out the non-parametric tests, such as the Wilcoxon test.

The test of assumptions (Step 3A, Fig. 13.8) and Rank-Sum Wilcoxon test described in Step B (see Fig. 13.9, Step 3B, Lines 48–60) is performed by using the **shapiro.test( ), var.test(** ), and **wilcox.test(** ) functions in R.

As defined earlier in the Introduction section (Sect. 13.1);

- **Rank-Sum Wilxocon test** compares the *median* for two independently sampled groups from two different population whereby the two variables under consideration are independent of each other.
- The targeted grouping "independent" variable (***x***) is often a dichotomous or binary type, while the ***y*** variable is continuous with at least ordinal scale measurement.


Step 3A (Test for Assumptions)
Step 3B (Rank-Sum Wilcoxon)
Step 4
Console
Terminal
Background Jobs
R 4.2.2 · ~/WC Manuscript/

```
32  # Step 3 - Conduct tests for Assumptions and Analyze data
33
34  # Assmp1: Shapiro-Wilk test for data normality - Sex (Man, Woman) vs exam1
35  with(Wilcoxon.Data, shapiro.test(exam1[sex == "Man"]))
36  with(Wilcoxon.Data, shapiro.test(exam1[sex == "Woman"]))
37
38  # Assmp2: F-test for homogeneity in variances. function var.test()
39  homogeneity.ftest4 <- var.test(exam1 ~ sex, data = Wilcoxon.Data)
40  homogeneity.ftest4
41
42  # Assmp3: Factor variables from numeric to ranked or ordinal variable
43  Wilcoxon.Data$exam1 <- as.numeric(Wilcoxon.Data$exam1)
44  Wilcoxon.Data$exam1 <- as.factor(Wilcoxon.Data$exam1)
45
46  Wilcoxon.Data$sex <- as.factor(Wilcoxon.Data$sex)
47
48  # Perform Rank-Sum Wilcoxon test
49
50  # Rank-Sum Wilcoxon Test where variable is dichotomous and at least ordinal (Two-tailed)
51  WilcoxonRankSumModel1 <- wilcox.test(`exam1` ~ `sex`, mu=0, alt="two.sided", paired=FALSE, conf.int=TRUE, conf.level=0.95, exact=F, correct=F)
52  WilcoxonRankSumModel1
53
54  # Test whether Ave. median of sex variable is less than the Ave. median of exam1 (One-tailed)
55  WilcoxonRankSumModel2 <- wilcox.test(`exam1` ~ `sex`, mu=0, alt="less", paired=FALSE, conf.int=TRUE, conf.level=0.95, exact=F, correct=F)
56  WilcoxonRankSumModel2
57
58  # Test whether Ave. median of sex variable is greater than the Ave. median of exam1 (One-tailed)
59  WilcoxonRankSumModel3 <- wilcox.test(`exam1` ~ `sex`, mu=0, alt="greater", paired=FALSE, conf.int=TRUE, conf.level=0.95, exact=F, correct=F)
60  WilcoxonRankSumModel3
61
62  # Step 4 - Visualize median differences for independent (unpaired) group of data
63
64  ggplot(Wilcoxon.Data, aes(x = sex, y = exam1, fill = sex), ylim = c(45, 100)) +
65    stat_boxplot(geom ="errorbar", width = 0.5) +
66    geom_boxplot(fill = "light blue") +
67    stat_summary(fun = mean, geom="point", shape=10, size=3.5, color="dark blue") +
68    ggtitle("Distribution (Median) of exam1 by sex (Man, Woman)") +
69    theme_bw() + theme(legend.position="none")
```

```
 -1.999974        Inf
sample estimates:
difference in location
            1.285688

> 
```

**Fig. 13.9** Conducting Rank-Sum (Unpaired) Wilcoxon Test in R

To illustrate this test (Rank-Sum) using the example dataset "**Wilxocon.Data"** in R (see: highlighted columns in Fig. 13.4) we will:

1. Test whether the median of the **group A** (Man) of the "**sex**" variable is *equal to* the mean of the **group B** (Woman) of the "**sex**" variable considering the "**exam1**" period in the data? (**two-tailed test)**
2. Also, we will check whether the *median* of the **group A** (Man) is *less than* the median of **group B** (Woman) in the "**sex**" variable? **(one-tailed test)**
3. Then we will check whether the *median* of **group A** (Man) is *greater than* the median of **group B** (Woman) in the "**sex**" variable? **(one-tailed test)**

The syntax and code to perform the above tests in R (see Fig. 13.9, Steps 3A and 3B, Lines 32–60) is as shown in the codes provided below:

## Test of Assumption (Step 3A):

```
# Assmp1: Shapiro-Wilk test for data normality - Sex (Man, Woman) vs exam1
with(Wilcoxon.Data, shapiro.test(exam1[sex == "Man"]))
with(Wilcoxon.Data, shapiro.test(exam1[sex == "Woman"]))

# Assmp2: F-test for homogeneity in variances. function var.test()
homogeneity.ftest4 <- var.test(exam1 ~ sex, data = Wilcoxon.Data)
homogeneity.ftest4

# Assmp3: Factor variables from numeric to ranked or ordinal variable
Wilcoxon.Data$exam1 <- as.numeric(Wilcoxon.Data$exam1)
Wilcoxon.Data$exam1 <- as.factor(Wilcoxon.Data$exam1)
Wilcoxon.Data$sex <- as.factor(Wilcoxon.Data$sex)
```

## Perform Rank-Sum Wilcoxon Test (Step 3B):

```
# Rank-Sum Wilcoxon Test where variable is dichotomous and at least ordinal (Two-tailed)
WilcoxonRankSumModel1 <- wilcox.test(`exam1` ~ `sex`, mu=0, alt="two.sided", paired=FALSE, conf.int=TRUE, conf.level=0.95, exact=F, correct=F)
WilcoxonRankSumModel1

# Test whether Ave. median of sex variable is less than the Ave. median of exam1 (One-tailed)
WilcoxonRankSumModel2 <- wilcox.test(`exam1` ~ `sex`, mu=0, alt="less", paired=FALSE, conf.int=TRUE, conf.level=0.95, exact=F, correct=F)
WilcoxonRankSumModel2

# Test whether Ave. median of sex variable is greater than the Ave. median of exam1 (One-tailed)
WilcoxonRankSumModel3 <- wilcox.test(`exam1` ~ `sex`, mu=0, alt="greater", paired=FALSE, conf.int=TRUE, conf.level=0.95, exact=F, correct=F)
WilcoxonRankSumModel3
```

**Useful Tips**

- As shown and illustrated in the code and figure above (Fig. 13.9), the users must always use the `paired = FALSE` option to specify the Rank-Sum (unpaired) Wilcoxon test.
- Whereas the `paired = TRUE` option is used to specify the Signed-Rank (paired) Wilcoxon test (which the authors have covered in the previous Sect. 13.2).
- Users need to use the `exact = F` and `correct = F` options to specify unequal variances for both the Signed-Rank and Rank-Sum Wilcoxon test or analysis.
- Then use the `alt = "less"` and `alt = "greater"` option to specify a "one-tailed" Wilcoxon analysis.

Accordingly, once the user has successfully run the codes as defined in **Steps 3A** and **3B** (see Fig. 13.9, Lines 32–60), they will be presented with the results of the "tests for assumptions" (Step 3A) and the "Rank-Sum Wilcoxon test" in the Console as shown in Figs. 13.10a and b, respectively.

In Fig. 13.10a, which represents the outcome of the Step 3A (see Figs. 13.8 and 13.9); we conducted the different tests for assumptions for the Rank-Sum Wilcoxon test in order to determine if the analyzed dataset is fitting and valid for the test (Independent Samples t-test). In other words, whether the dataset in question does not meet the criteria to perform the parametric test or version (i.e., Independent Sample t-test).

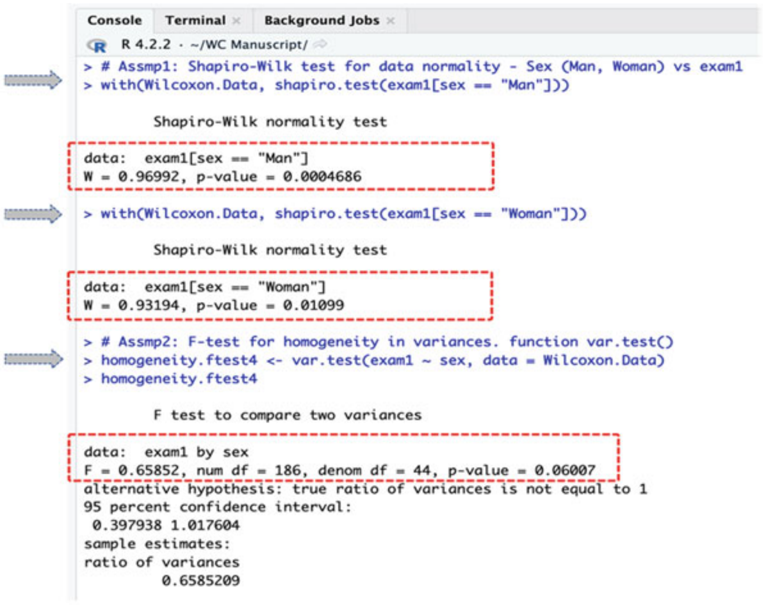
As highlighted in the figure (Fig. 13.10a), we can see that the *normality test* (Assmp1) by using the **shapiro.test( )** method (where we assume a value of $p > 0.05$ is normal) shows that the distribution of the two groups of the independent variable (i.e., **Man** and **Woman** of the "**sex**") are not normality distributed when analyzed against the "**exam1**" variable, which is the second targeted variable in our analysis, whereby (exam1[sex == "Man"], p-value=0.0004686, W=0.96992) and (`exam1[sex == "Woman"], p-value=0.01099, W=0.93194`), respectively.

Also, in the second assumption test (Assmp2, Fig. 13.10a); we tested the *homogeneity of variance* for the two targeted variables (**exam1** ~ **sex**) using the **var.test( )** method, whereby we assume that a value of $p > 0.05$ indicates that "equality in variance" is met. Consequentially, as highlighted in Fig. 13.10a, we note that there are slightly differences (i.e., close to equality of variance being met) in the variance for the two sets of analyzed variables with `p=0.06007.` and `F = 0.65852`, respectively.

Thus, with no data normality met and the slight differences in homogeneity of variance, we can proceed to conduct the "Rank-Sum Wilcoxon test" as defined in Step 3B (Fig. 13.9) and the results represented in Fig. 13.10b.

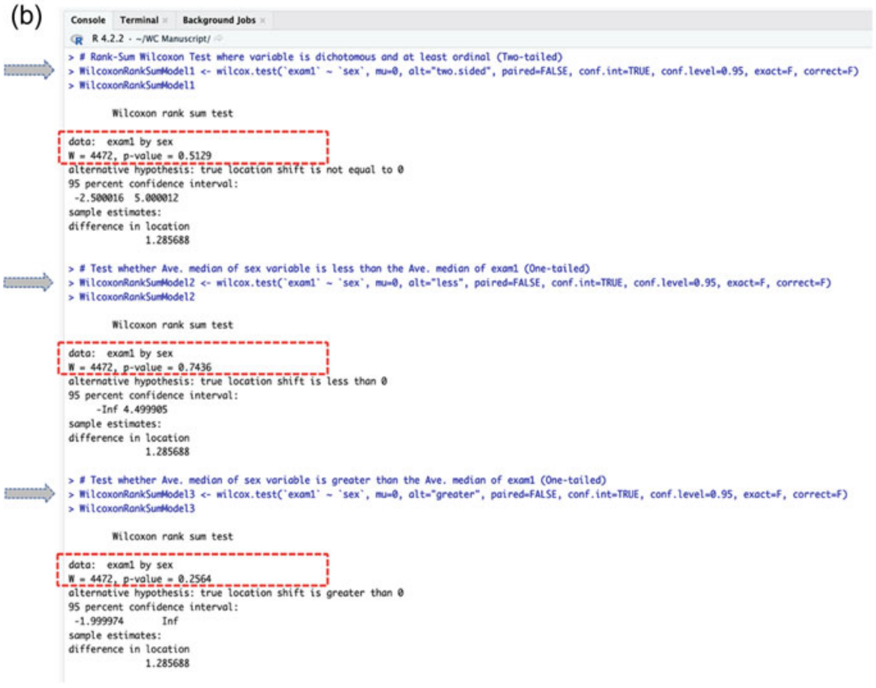
As gathered the results presented in Fig. 13.10b, we conducted the Rank-Sum (Two-sample) Wilcoxon test by considering the two variables (**exam1** ~ **sex**). The results of the tests are stored in R object we named or defined as "`WilcoxonRankSumModel1`" for the **two-tailed** analysis, and "`WilcoxonRankSumModel2`" and "`WilcoxonRankSumModel3`" for the **one-tailed** analysis, respectively. The meaning of the results is interpreted in detail in Step 5 described in this section.

(a)



```
> # Assmp1: Shapiro-Wilk test for data normality - Sex (Man, Woman) vs exam1
> with(Wilcoxon.Data, shapiro.test(exam1[sex == "Man"]))

        Shapiro-Wilk normality test

data:  exam1[sex == "Man"]
W = 0.96992, p-value = 0.0004686

> with(Wilcoxon.Data, shapiro.test(exam1[sex == "Woman"]))

        Shapiro-Wilk normality test

data:  exam1[sex == "Woman"]
W = 0.93194, p-value = 0.01099

> # Assmp2: F-test for homogeneity in variances. function var.test()
> homogeneity.ftest4 <- var.test(exam1 ~ sex, data = Wilcoxon.Data)
> homogeneity.ftest4

        F test to compare two variances

data:  exam1 by sex
F = 0.65852, num df = 186, denom df = 44, p-value = 0.06007
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
 0.397938 1.017604
sample estimates:
ratio of variances
         0.6585209
```

(b)



```
> # Rank-Sum Wilcoxon Test where variable is dichotomous and at least ordinal (Two-tailed)
> WilcoxonRankSumModel1 <- wilcox.test(`exam1` ~ `sex`, mu=0, alt="two.sided", paired=FALSE, conf.int=TRUE, conf.level=0.95, exact=F, correct=F)
> WilcoxonRankSumModel1

        Wilcoxon rank sum test

data:  exam1 by sex
W = 4472, p-value = 0.5129
alternative hypothesis: true location shift is not equal to 0
95 percent confidence interval:
 -2.500016  5.000012
sample estimates:
difference in location
              1.285688

> # Test whether Ave. median of sex variable is less than the Ave. median of exam1 (One-tailed)
> WilcoxonRankSumModel2 <- wilcox.test(`exam1` ~ `sex`, mu=0, alt="less", paired=FALSE, conf.int=TRUE, conf.level=0.95, exact=F, correct=F)
> WilcoxonRankSumModel2

        Wilcoxon rank sum test

data:  exam1 by sex
W = 4472, p-value = 0.7436
alternative hypothesis: true location shift is less than 0
95 percent confidence interval:
     -Inf 4.499905
sample estimates:
difference in location
              1.285688

> # Test whether Ave. median of sex variable is greater than the Ave. median of exam1 (One-tailed)
> WilcoxonRankSumModel3 <- wilcox.test(`exam1` ~ `sex`, mu=0, alt="greater", paired=FALSE, conf.int=TRUE, conf.level=0.95, exact=F, correct=F)
> WilcoxonRankSumModel3

        Wilcoxon rank sum test

data:  exam1 by sex
W = 4472, p-value = 0.2564
alternative hypothesis: true location shift is greater than 0
95 percent confidence interval:
 -1.999974       Inf
sample estimates:
difference in location
              1.285688
```

**Fig. 13.10** **a** Results of data normality and homogeneity of variance tests displayed in the console in R. **b** Result of rank-sum (two-sample or unpaired) Wilcoxon test in R

## # Step 4—Plot and Visualize the Median Differences for Two (Independent) Variables

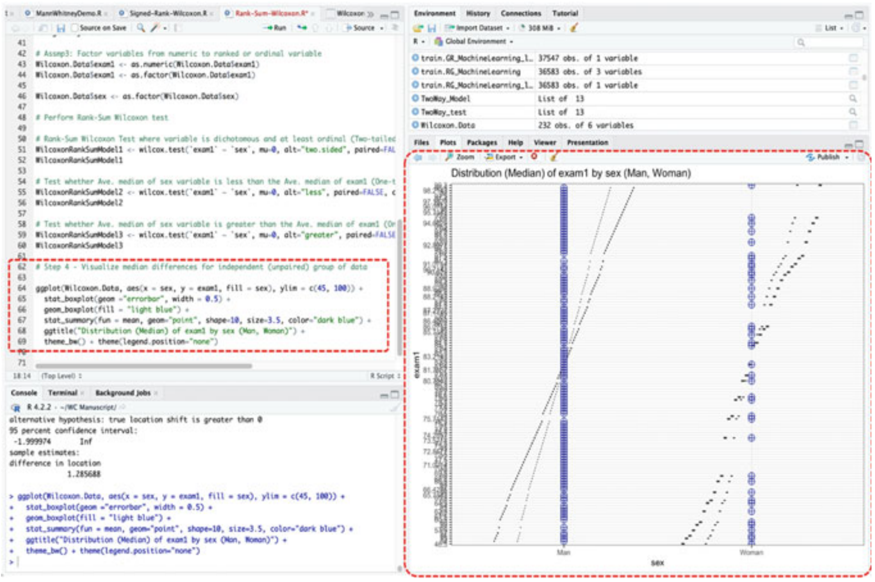
As illustrated previously in Sect. 13.2, another great way to check whether there is a difference in median of the two independent group of variables (e.g., sex and exam1) is by plotting them as graph. By so doing, the users will be able to visualize the difference in the median (if any) between the two data sample.

To do this, in Step 4 in Fig. 13.9 (Lines 62–69) and the resultant chart represented in Fig. 13.11; the authors used the **ggplot( )** function in R to visualize the *median* between the two groups of variables "**sex**" (**Man, Woman**) by taking into account or plotting it against the second variable **"exam1**" in the example data "**Wilcoxon.Data**".

The code and syntax used in plotting the median for the two variables is shown in the code below, and the resultant chart is as represented in Fig. 13.11.

## # Visualize Median Difference for Two Independent (Unpaired) Variables

```
ggplot(Wilcoxon.Data, aes(x = sex, y = exam1, fill = sex), ylim = c(45, 100)) +
  stat_boxplot(geom ="errorbar", width = 0.5) +
  geom_boxplot(fill = "light blue") +
  stat_summary(fun = mean, geom="point", shape=10, size=3.5, color="dark blue") +
  ggtitle("Distribution (Median) of exam1 by sex (Man, Woman)") +
  theme_bw() + theme(legend.position="none")
```



```
41
42  # Assmp3: Factor variables from numeric to ranked or ordinal variable
43  Wilcoxon.Data$exam1 <- as.numeric(Wilcoxon.Data$exam1)
44  Wilcoxon.Data$exam1 <- as.factor(Wilcoxon.Data$exam1)
45
46  Wilcoxon.Data$sex <- as.factor(Wilcoxon.Data$sex)
47
48  # Perform Rank-Sum Wilcoxon test
49
50  # Rank-Sum Wilcoxon Test where variable is dichotomous and at least ordinal (Two-tailed
51  WilcoxonRankSumModel1 <- wilcox.test(`exam1` ~ `sex`, mu=0, alt="two.sided", paired=FAL
52  WilcoxonRankSumModel1
53
54  # Test whether Ave. median of sex variable is less than the Ave. median of exam1 (One-t
55  WilcoxonRankSumModel2 <- wilcox.test(`exam1` ~ `sex`, mu=0, alt="less", paired=FALSE, c
56  WilcoxonRankSumModel2
57
58  # Test whether Ave. median of sex variable is greater than the Ave. median of exam1 (Or
59  WilcoxonRankSumModel3 <- wilcox.test(`exam1` ~ `sex`, mu=0, alt="greater", paired=FALSE
60  WilcoxonRankSumModel3
61
62  # Step 4 - Visualize median differences for independent (unpaired) group of data
63
64  ggplot(Wilcoxon.Data, aes(x = sex, y = exam1, fill = sex), ylim = c(45, 100)) +
65    stat_boxplot(geom ="errorbar", width = 0.5) +
66    geom_boxplot(fill = "light blue") +
67    stat_summary(fun = mean, geom="point", shape=10, size=3.5, color="dark blue") +
68    ggtitle("Distribution (Median) of exam1 by sex (Man, Woman)") +
69    theme_bw() + theme(legend.position="none")
70
71
```

```
alternative hypothesis: true location shift is greater than 0
95 percent confidence interval:
 -1.999974       Inf
sample estimates:
difference in location
              1.285688

> ggplot(Wilcoxon.Data, aes(x = sex, y = exam1, fill = sex), ylim = c(45, 100)) +
+   stat_boxplot(geom ="errorbar", width = 0.5) +
+   geom_boxplot(fill = "light blue") +
+   stat_summary(fun = mean, geom="point", shape=10, size=3.5, color="dark blue") +
+   ggtitle("Distribution (Median) of exam1 by sex (Man, Woman)") +
+   theme_bw() + theme(legend.position="none")
>
```

| | |
|---|---|
| train.GR_MachineLearning_1… | 37547 obs. of 1 variable |
| train.RG_MachineLearning | 36583 obs. of 3 variables |
| train.RG_MachineLearning_1… | 36583 obs. of 1 variable |
| TwoWay_Model | List of 13 |
| TwoWay_test | List of 13 |
| Wilcoxon.Data | 232 obs. of 6 variables |

**Fig. 13.11** Plotting the median differences for two sets of independent variables in R using the ggplot() function

## # Step 5—Results' Interpretation (Sum-Rank Wilcoxon Test)

The final step for the Sum-Rank Wilcoxon test or statistics is to interpret and understand the results of the test.

By default, the hypothesis for conducting the test (Sum-Rank Wilcoxon) is; *IF* the p-value is less than or equal to 0.05 ($p \leq 0.05$), *THEN* we assume that the median of the two sets of data or groups of variables is statistically different and that this is not by chance ($H_1$), *ELSE IF* the p-value is greater than 0.05 ($p > 0.05$) THEN we can conclude that there is no difference in the median of the two sets of variable and that any difference observed may only occur by chance ($H_0$).

```
> WilcoxonRankSumModel1 <- wilcox.test(`exam1` ~ `sex`, mu=0,
alt="two.sided", paired=FALSE, conf.int=TRUE, conf.level=0.95, exact=F,
correct=F)
> WilcoxonRankSumModel1

   Wilcoxon rank sum test

data:  exam1 by sex
W = 4472, p-value = 0.5129
alternative hypothesis: true location shift is not equal to 0
95 percent confidence interval:
 -2.500016  5.000012
sample estimates:
difference in location
              1.285688%
```

As shown in the results of the test presented above or the outcome of the Rank-Sum Wilcoxon test (`WilcoxonRankSumModel1`) (see (Fig. 13.10b); the meaning of the results of the Rank-Sum test for the **two-tailed** (**exam1 ~ sex**) by using the **wilcox.test( )** function in R can be explained as a list containing the following:

- **Statistics**: `w = 4472`, which denotes the value of the Rank-Sum test statistics.
- **p-value**: `p-value = 0.5129` is the p-value or significance level of the test.
- **Confidence interval:** `Conf.Int(95%, -2.500016 5.000012)` represents the confidence interval for the median assumed to be appropriate to the specified alternative hypothesis.
- **Sample estimates:** `1.285688` is the median difference in location between the two groups of data or population that is compared considering the two variables (exam1 ~ sex).

As seen in the results described above, statistically it can be said that the p-value of the Rank-Sum Wilcoxon test (`WilcoxonRankSumModel1`), i.e., the **two-tailed** analysis, is `p=0.5129`. As we can see, the p-value of the test is greater than the scientifically acceptable significance levels ($p \leq 0.05$). Therefore, we can statistically conclude that there is no statistically significant difference between the medians of the two groups of data (Man vs Woman) of the "**sex**" variable taking into account the "**exam1**" grades of the students or participants in the analyzed data. In other words, the *median* of the **group A (Man)** of the "**sex**" variable is *equal to* the median of **group B (Woman)** of the "**sex**" when analyzed against the "**exam1**" grades (**two-tailed test).**

Furthermore, as shown in the next results for the "one-tailed" Rank_Sum tests presented below and as reported in Fig. 13.10b.

– We also checked whether the median of the **group A** (i.e., Man) is *less than* the median of the **group B** (Woman) of the "**sex**" variable (`WilcoxonRankSumModel2`), and
– Whether the median of the **group A** (Man) is *greater than* the median of **group B** (Woman) of the "**sex**" variable (`WilcoxonRankSumModel3`), respectively.

```
> WilcoxonRankSumModel2 <- wilcox.test(`exam1` ~ `sex`, mu=0,
alt="less", paired=FALSE, conf.int=TRUE, conf.level=0.95, exact=F,
correct=F)
> WilcoxonRankSumModel2

      Wilcoxon rank sum test

data:  exam1 by sex
W = 4472, p-value = 0.7436
alternative hypothesis: true location shift is less than 0
95 percent confidence interval:
 -Inf  4.499905
sample estimates:
difference in location
                  1.285688
```

```
> WilcoxonRankSumModel3 <- wilcox.test(`exam1` ~ `sex`, mu=0,
alt="greater", paired=FALSE, conf.int=TRUE, conf.level=0.95, exact=F,
correct=F)
> WilcoxonRankSumModel3

      Wilcoxon rank sum test

data:  exam1 by sex
W = 4472, p-value = 0.2564
alternative hypothesis: true location shift is greater than 0
95 percent confidence interval:
  -1.999974 Inf
sample estimates:
difference in location
                  1.285688
```

As gathered in the above results of the one-tailed tests (`WilcoxonRankSumModel2, where W=4472, p=0.7436`; and `WilcoxonRankSumModel3, where W=4472, p=0.2564`), we can see that there are no significant differences in the medians of the two groups of data (i.e., the value of p-value is above the threshold of p $\leq$ 0.05, and the `difference in location (sample estimates) = 1.285688`), as also evidenced in the result of the “two-tailed test (see Fig. 13.10b). Therefore, we can statistically conclude that there are no differences in the *median* of the scores for the group of Men vs Women in the “exam1” variable.

## 13.4 Summary

In this final chapter of the book and PART II, the authors illustrated to the readers how to conduct the two main types of Wilcoxon test (Signed-Rank and Rank_Sum) in R. In Sect. 13.2, we explained and practically illustrated how to perform the Signed-rank (paired) Wilcoxon test. While Sect. 13.3 covers how to conduct the Rank-Sum (unpaired or two-sample) Wilcoxon test.

The chapter also covered how to graphically plot the median of the different analyzed variables in the data and/or results of the Wilcoxon tests. In addition, it discussed in detail how to interpret and understand the results of the two main types of the Wilcoxon tests (Signed-Rank and Rank_Sum) in R.

To summarize the contents of this chapter:

- *Wilcoxon test* is one of the inferential (non-parametric) statistics that are used for hypothesis testing, and for determining if there are any significant differences (where applicable) between the *medians* of two independent groups of data or paired variables in a given data sample.

When choosing whether to conduct the ”Signed-Rank“ or ”Rank-Sum“ Wilcoxon test? The researcher or data analyst should:

- Perform “*Signed-Rank (paired) Wilcoxon test”* if the targeted group or variable comes from a single population but is analyzed at different time intervals (e.g., *pre* and *post* tests, before and after an intervention or treatment for the same group of people, place, or thing, etc). It is also important to mention that this test (Signed-Rank) is done when the dataset in question violates the necessary conditions or assumptions for conducting the Paired-Sample t-test. Thus, is (Signed-Rank) considered as the non-parametric alternative to the parametric Paired-sample t-test.
- Perform the “*Rank-Sum (independent or two sample) Wilcoxon test”* if the groups of data to be analyzed come from *two different populations* (i.e., are statistically independent). For example, two different categories of people, thing or place (gender: male and female, state: on and off, region: north and south, etc.). In addition, it is also important to mention that the test (Rank-Sum) is done when the dataset in question violates the necessary conditions or assumptions for conducting the Independent Sample t-test. Thus, is (Rank-Sum) considered as the non-parametric alternative to the parametric Independent-sample t-test.

Finally, the users will need to perform the "*two-tailed*" test or statistics, if they only want to determine whether the median of the two data samples is different from one another. Whereas, on the other hand, they will require to perform a "*one-tailed test*" if their goal includes also to determine whether the median of a specific sample or variable is *less than* or *greater than* the median of another variable, as the case may be.

## References

Hayes, A. (2023). *Wilcoxon Test: Definition in Statistics, Types, and Calculation.* https://www.investopedia.com/terms/w/wilcoxon-test.asp.

Rey, D., & Neuhäuser, M. (2011). Wilcoxon-Signed-Rank Test. In M. Lovric (Ed.), *International Encyclopedia of Statistical Science* (pp. 1658–1659). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-04898-2_616.

Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics Bulletin, 1*(6), 80–83. https://doi.org/10.2307/3001968

# Epilogue and Conclusion

## Achievements of This Book

First and foremost, the goal of this book is to provide a textbook and manual for R statistics applied for academic research, data analytics, and computer programming.

It provides information about the different types of statistical data analysis and methods, and the best scenarios for use of each case in R.

It gives a hands-on step-by-step practical guide on how to identify and perform the different parametric and non-parametric procedures mostly used in social science research. This includes a description of the different conditions or assumptions that are necessary to perform the different statistical tests, and how to understand the results of the various methods.

This book also covers the different types of data formats and sources, and how to test for the reliability and validity of the available datasets e.g for research.

Different research experiments, case scenarios, and examples are explained in this book.

It is the first book to provide a comprehensive description and step-by-step practical hands-on guide on how to carry out the different types of statistical analysis in R particularly for research purposes with examples. Ranging from how to import and store datasets in R as objects, how to code and call the different R methods and functions for manipulating the datasets or objects, factorization, and vectorization, to better reasoning, interpretation, and storage of the results for future use, and graphical visualizations and representations.

Bringing in perspectives both from the authors' background in the fields of Computer Science and Education, to the several workshops delivered on Research Methods, Statistical Data Analysis and Data Collection topics, to the hands-on practical implementation of the statistical methods in the form of scientific research studies and experiments; the insightful analysis and practical guidelines provided by the authors in this book serves as a Reference Source Material for both the

Educational community (Teachers, PhD Students, Researchers, Libraries) and Industrial experts (Statisticians, Professionals, programmers, Software Developers, Data Analysts) who are interested in using R for Statistical Data Analysis or Research purposes.

Thus, the congruence of Statistics and Computer Programming for Research.

# Index