Chapter 13
Managing Data Frames

A data frame is the most common way of storing data in R and, generally, is the data structure most often used for data analyses. Under the hood, a data frame is a list of equal-length vectors. Each element of the list can be thought of as a column and the length of each element of the list is the number of rows. As a result, data frames can store different classes of objects in each column (i.e. numeric, character, factor). In essence, the easiest way to think of a data frame is as an Excel worksheet that contains columns of different types of data but are all of equal length rows. In this chapter I will illustrate how to create data frames, add additional elements to pre-existing data frames, add attributes to data frames, and subset data frames.

13.1 Creating Data Frames

Data frames are usually created by reading in a dataset using `read.table()` or `read.csv()`; this will be covered in the importing and scraping data chapters. However, data frames can also be created explicitly with the `data.frame()` function or they can be coerced from other types of objects like lists. In this case I'll create a simple data frame `df` and assess its basic structure:

```
df <- data.frame(col1 = 1:3,
                 col2 = c("this", "is", "text"),
                 col3 = c(TRUE, FALSE, TRUE),
                 col4 = c(2.5, 4.2, pi))

# assess the structure of a data frame
str(df)
## 'data.frame':    3 obs. of  4 variables:
##  $ col1: int  1 2 3
##  $ col2: Factor w/ 3 levels "is","text","this": 3 1 2
##  $ col3: logi  TRUE FALSE TRUE
##  $ col4: num  2.5 4.2 3.14
```


B.C. Boehmke, *Data Wrangling with R*, Use R!, DOI 10.1007/978-3-319-45599-0_13

```
# number of rows
nrow(df)
## [1] 3

# number of columns
ncol(df)
## [1] 4
```

Note how `col2` in `df` was converted to a column of factors. This is because there is a default setting in `data.frame()` that converts character columns to factors. We can turn this off by setting the `stringsAsFactors = FALSE` argument:

```
df <- data.frame(col1 = 1:3,
                 col2 = c("this", "is", "text"),
                 col3 = c(TRUE, FALSE, TRUE),
                 col4 = c(2.5, 4.2, pi),
                 stringsAsFactors = FALSE)

# note how col2 now is of a character class
str(df)
## 'data.frame':    3 obs. of  4 variables:
##  $ col1: int  1 2 3
##  $ col2: chr  "this" "is" "text"
##  $ col3: logi  TRUE FALSE TRUE
##  $ col4: num  2.5 4.2 3.14
```

We can also convert pre-existing structures to a data frame. The following illustrates how we can turn multiple vectors, a list, or a matrix into a data frame:

```
v1 <- 1:3
v2 <-c("this", "is", "text")
v3 <- c(TRUE, FALSE, TRUE)

# convert same length vectors to a data frame using data.frame()
data.frame(col1 = v1, col2 = v2, col3 = v3)
##   col1 col2  col3
## 1    1 this  TRUE
## 2    2   is FALSE
## 3    3 text  TRUE

# convert a list to a data frame using as.data.frame()
l <- list(item1 = 1:3,
          item2 = c("this", "is", "text"),
          item3 = c(2.5, 4.2, 5.1))
l
## $item1
## [1] 1 2 3
##
## $item2
## [1] "this" "is"   "text"
##
## $item3
## [1] 2.5 4.2 5.1
```

```
as.data.frame(l)
##   item1 item2 item3
## 1     1  this   2.5
## 2     2    is   4.2
## 3     3  text   5.1

# convert a matrix to a data frame using as.data.frame()
m1 <- matrix(1:12, nrow = 4, ncol = 3)
m1
##      [,1] [,2] [,3]
## [1,]    1    5    9
## [2,]    2    6   10
## [3,]    3    7   11
## [4,]    4    8   12

as.data.frame(m1)
##   V1 V2 V3
## 1  1  5  9
## 2  2  6 10
## 3  3  7 11
## 4  4  8 12
```

13.2 Adding On To Data Frames

We can leverage the `cbind()` function for adding columns to a data frame. Note that one of the objects being combined must already be a data frame otherwise `cbind()` could produce a matrix.

```
df
##   col1 col2  col3     col4
## 1    1 this  TRUE 2.500000
## 2    2   is FALSE 4.200000
## 3    3 text  TRUE 3.141593

# add a new column
v4 <- c("A", "B", "C")
cbind(df, v4)
##   col1 col2  col3     col4 v4
## 1    1 this  TRUE 2.500000  A
## 2    2   is FALSE 4.200000  B
## 3    3 text  TRUE 3.141593  C
```

We can also use the `rbind()` function to add data frame rows together. However, severe caution should be taken because this can cause changes in the classes of the columns. For instance, our data frame `df` currently consists of an integer, character, logical, and numeric variables.

```
df
##   col1 col2  col3     col4
## 1    1 this  TRUE 2.500000
## 2    2   is FALSE 4.200000
```

```
## 3    3 text  TRUE 3.141593
str(df)
## 'data.frame':    3 obs. of  4 variables:
##  $ col1: int  1 2 3
##  $ col2: chr  "this" "is" "text"
##  $ col3: logi  TRUE FALSE TRUE
##  $ col4: num  2.5 4.2 3.14
```

If we attempt to add a row using `rbind()` and `c()` it converts all columns to a character class. This is because all elements in the vector created by `c()` must be of the same class so they are all coerced to the character class which then coerces all the variables in the data frame to the character class.

```
df2 <- rbind(df, c(4, "R", F, 1.1))
df2
##   col1 col2  col3             col4
## 1    1 this  TRUE              2.5
## 2    2   is FALSE              4.2
## 3    3 text  TRUE 3.14159265358979
## 4    4    R FALSE              1.1
str(df2)
## 'data.frame':    4 obs. of  4 variables:
##  $ col1: chr  "1" "2" "3" "4"
##  $ col2: chr  "this" "is" "text" "R"
##  $ col3: chr  "TRUE" "FALSE" "TRUE" "FALSE"
##  $ col4: chr  "2.5" "4.2" "3.14159265358979" "1.1"
```

To add rows appropriately, we need to convert the items being added to a data frame and make sure the columns are the same class as the original data frame.

```
adding_df <- data.frame(col1 = 4,
                        col2 = "R",
                        col3 = FALSE,
                        col4 = 1.1,
                        stringsAsFactors = FALSE)

df3 <- rbind(df, adding_df)
df3
##   col1 col2  col3     col4
## 1    1 this  TRUE 2.500000
## 2    2   is FALSE 4.200000
## 3    3 text  TRUE 3.141593
## 4    4    R FALSE 1.100000
str(df3)
## 'data.frame':    4 obs. of  4 variables:
##  $ col1: num  1 2 3 4
##  $ col2: chr  "this" "is" "text" "R"
##  $ col3: logi  TRUE FALSE TRUE FALSE
##  $ col4: num  2.5 4.2 3.14 1.1
```

There are better ways to join data frames together than to use `cbind()` and `rbind()`. These are covered later on in the transforming your data with dplyr chapter.

13.3 Adding Attributes to Data Frames

Similar to matrices, data frames will have a dimension attribute. In addition, data frames can also have additional attributes such as row names, column names, and comments. We can illustrate with data frame `df`.

```
# basic data frame
df
##   col1 col2  col3     col4
## 1    1 this  TRUE 2.500000
## 2    2   is FALSE 4.200000
## 3    3 text  TRUE 3.141593
dim(df)
## [1] 3 4
attributes(df)
## $names
## [1] "col1" "col2" "col3" "col4"
##
## $row.names
## [1] 1 2 3
##
## $class
## [1] "data.frame"
```

Currently `df` does not have row names but we can add them with `rownames()`:

```
# add row names
rownames(df) <- c("row1", "row2", "row3")
df
##      col1 col2  col3     col4
## row1    1 this  TRUE 2.500000
## row2    2   is FALSE 4.200000
## row3    3 text  TRUE 3.141593
attributes(df)
## $names
## [1] "col1" "col2" "col3" "col4"
##
## $row.names
## [1] "row1" "row2" "row3"
##
## $class
## [1] "data.frame"
```

We can also change the existing column names by using `colnames()` or `names()`:

```
# add/change column names with colnames()
colnames(df) <- c("col_1", "col_2", "col_3", "col_4")
df
##      col_1 col_2 col_3    col_4
## row1     1  this  TRUE 2.500000
## row2     2    is FALSE 4.200000
## row3     3  text  TRUE 3.141593
attributes(df)
## $names
## [1] "col_1" "col_2" "col_3" "col_4"
##
## $row.names
## [1] "row1" "row2" "row3"
##
## $class
## [1] "data.frame"

# add/change column names with names()
names(df) <- c("col.1", "col.2", "col.3", "col.4")
df
##      col.1 col.2 col.3    col.4
## row1     1  this  TRUE 2.500000
## row2     2    is FALSE 4.200000
## row3     3  text  TRUE 3.141593
attributes(df)
## $names
## [1] "col.1" "col.2" "col.3" "col.4"
##
## $row.names
## [1] "row1" "row2" "row3"
##
## $class
## [1] "data.frame"
```

Lastly, just like vectors, lists, and matrices, we can add a comment to a data frame without affecting how it operates.

```
# adding a comment attribute
comment(df) <- "adding a comment to a data frame"
attributes(df)
## $names
## [1] "col.1" "col.2" "col.3" "col.4"
##
## $row.names
## [1] "row1" "row2" "row3"
##
## $class
## [1] "data.frame"
##
## $comment
## [1] "adding a comment to a data frame"
```

13.4 Subsetting Data Frames

Data frames possess the characteristics of both lists and matrices: if you subset with a single vector, they behave like lists and will return the selected columns with all rows; if you subset with two vectors, they behave like matrices and can be subset by row and column:

```
df
##      col.1 col.2 col.3    col.4
## row1     1  this  TRUE 2.500000
## row2     2    is FALSE 4.200000
## row3     3  text  TRUE 3.141593

# subsetting by row numbers
df[2:3, ]
##      col.1 col.2 col.3    col.4
## row2     2    is FALSE 4.200000
## row3     3  text  TRUE 3.141593

# subsetting by row names
df[c("row2", "row3"), ]
##      col.1 col.2 col.3    col.4
## row2     2    is FALSE 4.200000
## row3     3  text  TRUE 3.141593

# subsetting columns like a list
df[c("col.2", "col.4")]
##      col.2    col.4
## row1  this 2.500000
## row2    is 4.200000
## row3  text 3.141593

# subsetting columns like a matrix
df[ , c("col.2", "col.4")]
##      col.2    col.4
## row1  this 2.500000
## row2    is 4.200000
## row3  text 3.141593

# subset for both rows and columns
df[1:2, c(1, 3)]
##      col.1 col.3
## row1     1  TRUE
## row2     2 FALSE

# use a vector to subset
v <- c(1, 2, 4)
df[ , v]
##      col.1 col.2    col.4
## row1     1  this 2.500000
## row2     2    is 4.200000
## row3     3  text 3.141593
```

Note that subsetting data frames with the `[` operator will simplify the results to the lowest possible dimension. To avoid this you can introduce the `drop = FALSE` argument:

```
# simplifying results in a named vector
df[, 2]
## [1] "this" "is"   "text"

# preserving results in a 3x1 data frame
df[, 2, drop = FALSE]
##      col.2
## row1  this
## row2    is
## row3  text
```