# Chapter 18 Functions

R is a functional programming language, meaning that everything you do is basically built on functions. However, moving beyond simply *using* pre-built functions to *writing* your own functions is when your capabilities really start to take off and your code development/writing takes on a new level of efficiency. Functions allow you to reduce code duplication by automating a generalized task to be applied recursively. Whenever you catch yourself repeating a function or copy-and-pasting code there is a good change that you should write a function to eliminate the redundancies.

Unfortunately, due to their abstractness, grasping the idea of writing functions (let alone writing them well) can take some time. However, in this chapter I will provide you with the basic knowledge of how functions operate in R to get you started on the right path. To do this, I cover the general components of functions, specifying function arguments, scoping and evaluation rules, managing function outputs, handling invalid parameters, and saving and sourcing functions for reuse. This will provide you with the required knowledge to start building your own functions. Lastly, I offer some additional resources that will help you learn more about functions in R.

### **18.1** Function Components

With the exception of primitive functions all R functions have three parts:

- body (): the code inside the function
- formals (): the list of arguments used to call the function
- environment(): the mapping of the location(s) of the function's variables

For example, let's build a function that calculates the present value (PV) of a single future sum. The equation for a single sum PV is:

$$PV = FV / (1+r)^n$$

where FV is future value, r is the interest rate, and n is the number of periods. In the function that follows the body of the function includes the equation

 $FV / (1+r)^{n}$ 

and then rounding the output to two decimals. The formals (or arguments) required for the function include FV, r, and n. And the environment shows that function operates in the global environment.

```
PV <- function(FV, r, n) {
        PV <- FV / (1 + r)^n
        round(PV, 2)
}
body(PV)
## {
      PV <- FV / (1 + r)^n
##
      round(PV, 2)
##
## }
formals(PV)
## $FV
##
##
## $r
##
##
## $n
environment(PV)
## <environment: R GlobalEnv>
```

# 18.2 Arguments

To perform the PV() function we can call the arguments in different ways.

```
# using argument names
PV(FV = 1000, r = .08, n = 5)
## [1] 680.58
# same as above but without using names (aka "positional matching")
PV(1000, .08, 5)
## [1] 680.58
# if using names you can change the order
PV(r = .08, FV = 1000, n = 5)
## [1] 680.58
# if not using names you must insert arguments in proper order
# in this e.g. the function assumes FV = .08, r = 1000, and n = 5
PV(.08, 1000, 5)
## [1] 0
```

Note that when building a function you can also set default values for arguments. In our original PV() we did not provide any default values so if we do not supply all the argument parameters an error will be returned. However, if we set default values then the function will use the stated default if any parameters are missing:

# 18.3 Scoping Rules

Scoping refers to the set of rules a programming language uses to lookup the value for variables and/or symbols. The following illustrates the basic concept behind the lexical scoping rules that R follows.

A function will first look inside the function to identify all the variables being called. If all variables exist then their is no additional search required to identify variables.

```
PV1 <- function() {
    FV <- 1000
    r <- .08
    n <- 5
    FV / (1 + r)^n
}
PV1()
## [1] 680.5832</pre>
```

However, if a variable does not exist within the function, R will look one level up to see if the variable exists.

```
\# the FV variable is outside the function environment FV <- 1000
```

```
PV2 <- function() {
    r <- .08
    n <- 5
    FV / (1 + r)^n
}
PV2()
## [1] 680.5832</pre>
```

This same concept applies if you have functions embedded within functions:

```
FV <- 1000
PV3 <- function() {
    r <- .08
    n <- 5
    denominator <- function() {
        (1 + r)^n
      }
      FV/denominator()
}
PV3()
## [1] 680.5832</pre>
```

This also applies for functions in which some arguments are called but not all variables used in the body are identified as arguments:

```
# n is specified within the function
PV4 <- function(FV, r) {
       n <- 5
        FV / (1 + r)^n
}
PV4(1000, .08)
## [1] 680.5832
# n is specified within the function and
# r is specified outside the function
r <- 0.08
PV5 <- function(FV) {
       n <- 5
        FV / (1 + r)^n
}
PV5(1000)
## [1] 680.5832
```

#### **18.4 Lazy Evaluation**

R functions perform "lazy" evaluation in which arguments are only evaluated if required in the body of the function.

# **18.5** Returning Multiple Outputs from a Function

If a function performs multiple tasks and therefore has multiple results to report then we have to include the c() function inside the function to display all the results. If you do not include the c() function then the function output will only return the last expression:

```
bad <- function(x, y) {
        2 * x + y
        x + 2 * y
        2 * x + 2 * y
        х / у
}
bad(1, 2)
## [1] 0.5
qood <- function(x, y) {</pre>
        output1 <- 2 * x + y
        output2 <- x + 2 * y
        output3 <- 2 * x + 2 * y
        output4 <- x / y
        c(output1, output2, output3, output4)
}
good(1, 2)
## [1] 4.0 5.0 6.0 0.5
```

Furthermore, when we have a function which performs multiple tasks (i.e. computes multiple computations) then it is often useful to save the results in a list.

```
good list <- function(x, y) {</pre>
        output1 <- 2 * x + y
        output2 <- x + 2 * y
        output3 <- 2 * x + 2 * y
        output4 <- x / y
        c(list(Output1 = output1, Output2 = output2,
               Output3 = output3, Output4 = output4))
}
good list(1, 2)
## $Output1
## [1] 4
##
## $Output2
## [1] 5
##
## $Output3
## [1] 6
##
## $Output4
## [1] 0.5
```

# **18.6 Dealing with Invalid Parameters**

For functions that will be used again, and especially for those used by someone other than the creator of the function, it is good to check the validity of arguments within the function. One way to do this is to use the stop() function. The following uses an if() statement to check if the class of each argument is numeric. If one or more arguments are not numeric then the stop() function will be triggered to provide a meaningful message to the user.

```
PV <- function(FV, r, n) {
    if(!is.numeric(FV) | !is.numeric(r) | !is.numeric(n)){
        stop('This function only works for numeric inputs!\n',
            'You have provided objects of the following classes:\n',
            'FV: ', class(FV), '\n',
            'r: ', class(r), '\n',
            'n: ', class(n))
    }
    PV <- FV / (1 + r)^n
    round(PV, 2)
}</pre>
```

```
FV("1000", 0.08, "5")
## Error in FV("1000", 0.08, "5"): This function only works for numeric inputs!
## You have provided objects of the following classes:
## FV: character
## r: numeric
## n: character
```

Another concern is dealing with missing or NA values. Lets say you wanted to perform the PV() function on a vector of potential future values. The function as is will output NA in place of any missing values in the FV input vector. If you want to remove the missing values then you can incorporate the na.rm parameter in the function arguments along with an if statement to remove missing values if na.rm = TRUE.

```
# vector of future value inputs
fv <- c(800, 900, NA, 1100, NA)
# original PV() function will return NAs
PV(fv, .08, 5)
## [1] 544.47 612.52
                      NA 748.64
                                      NA
# add na.rm argument
PV <- function (FV, r, n, na.rm = FALSE) {
        if(!is.numeric(FV) | !is.numeric(r) | !is.numeric(n)) {
                stop('This function only works for numeric inputs!\n',
                 'You have provided objects of the following classes:\n',
                      'FV: ', class(FV), '\n',
                      'r: ', class(r), '\n',
                     'n: ', class(n))
        }
        if(na.rm == TRUE) {
                FV <- FV[!is.na(FV)]</pre>
        }
        PV <- FV / (1 + r)^n
        round (PV, 2)
}
# setting na.rm = TRUE argument eliminates NA outputs
PV(fv, 0.08, 5, na.rm = TRUE)
## [1] 544.47 612.52 748.64
```

# 18.7 Saving and Sourcing Functions

If you want to save a function to be used at other times and within other scripts there are two main ways to do this. One way is to build a package which I do not cover in this book but is discussed in more details in Hadley Wickhams R Packages book, which is openly available at http://r-pkgs.had.co.nz/. Another option, and the one discussed here, is to save the function in a script. For example, we can save a script that contains the PV() function and save this script as PV.R.

01PVR x	-	Environment History Git	
Source on Save 9 2 • 1	-+ Run Source	• 🞯 🕞 📑 Import Dataset • 📢	( ) ( ) Ust-
	(m)){ c inputstvn', llowing classes:\n',	Clobal Environment &	
	8 South	Files Plots Packages Help	Viewer a
	- A progr		
Console ~/Desktop/Personal/R Vocabulary/R-Vocabulary/ P	-	Learning R Online	RStudio IDE Support
R is free software and comes with ABSOLUTELY NO WARRANTY.		CRAN Task Views	RStudio Tip of the Day
You are welcome to redistribute it under certain conditions.		Cettion Help with B	RStudio Products
Natural language support but running in an English locale		Manuals	
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.		An Introduction to R Writing R Extensions	The R Language Definition R Installation and Administration
		R Data Import/Export	R Internals
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.		Reference	

Now, if we are working in a fresh script you'll see that we have no objects and functions in our working environment:

9 Untited1 =		Environment History Git	
Source on Save Several	Run 🐲 🔿 Source +	🞯 📄 🖙 Import Dataset - 🥑	@ = Ust
1		Clobal Environment •	(Q,
1:1 (Top Level) ÷	R Script ‡		
Console -/Desktop/Personal/R Vocabulary/R-Vocabulary/ /0	-0	Files Plots Packages Help	Viewer
R version 3.2.2 (2015-08-14) "Fire Safety" Copyright (C) 2015 The R Foundation for Statistical Computing Platform: 246 64-cond-domin3.4 & (64-hit)		Home • (Ted is Tape)	Q.
R is free software and comes with ABSOLUTELY NO WARRANTY.		R Resources	RStudio
Type 'license()' or 'licence()' for distribution details.		Learning R Online	RStudio IDE Support
Natural language support but supping is on faulish langle		R on StackOverflow	RStudio Tip of the Day
interest surgards address and surger surgers and surgers		Getting Help with R	RStudio Products
R is a collaborative project with many contributors.			
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.		Manuals	
R is a collaborative project with many contributors. Type 'contributors()' for more information and 'citation()' on the to tilt & Ar & Rockages in publications. Type 'demo()' for some demos, 'help()' for an-line help, or 'help.start()' for an HML browser interface to help. Type 'd()' to quit R.		Manuals An Introduction to R Writing R Extensions	The R Language Definition R Installation and Administration
R is a collaborative project with many contributors. Type 'contributors()' for more information and 'citation()' on none to cite R or R pockages in publications. Type 'amen()' for some demos, 'help()' for an-line help, or 'help.star()' (or an HTML browser interface to help. Type 'a()' to quit R.		Manuals An Introduction to R Writing R Extensions R Data Import/Export	The R Language Definition R Installation and Administration R Internals

If we want to use the PV function in this new script we can simply read in the function by sourcing the script using source("PV.R"). Now, you'll notice that we have the PV() function in our global environment and can use it as normal. Note that if you are working in a different directory then where the PV.R file is located you'll need to include the proper path to access the relevant directory.

P] Unoticed1 ≈		Environment History Git		
🗅 🗇 🔒 🗋 Source on Save 🛛 💁 🖉 + 📋	📑 Run 📑 📑 Source 🔹	📑 😭 📑 Import Dataset • 🔮	C	List-
1		Global Environment •	(Q,	
		Functions		
		PV function (FV, r, n, no.rm = FALSE)		.E
11 (Ten Jawi) :	R Script =			
Console -/Desktop/Personal/R Vocabulary/R-Vocabulary/ /0	-0	Files Plots Packages Help	Viewer	-
Console -/Desktop/Personal/R Vocabulary/R-Vocabulary/ /0 Copyright (C) 2015 The R Foundation for Statistical Computing	-0	Files Plots Packages Help	Viewer	-
Console -/Desktop/Mrrsonal/R Vacabulary/R-Vacabulary/ /0 Copyright (C) 2015 The R Foundation for Statistical Computing Platform: x86_64-opple-damin13.4.0 (64-bit)	=0	Files Plots Packages Help	Viewer Q.	-0
Console -/Okakap/Persanal/R Vocabulary/R-Vocabulary/ /© Copyright (C) 2015 The R Foundation for Statistical Computing Platform: x86.64-opple-domini3.4.0 (64-bit) R is free software and comes with ABSOUTILY NO NARBANY. You are welcome to realistribute it under certain conditions. Type 'llcense()' or 'llcence()' for distribution details. Natural language support but running in an English locale	-0	Files Plots Packages Help Packages Help Mome - Tradie Trace R R Resources Learning R Online CRAN Task Views	Viewer RStudio RStudio IDE Support RStudio Tip of the Day	-
Computer -[Dealking)/Personal/A Vocabulary/ / D Copyright (C) 2825 The R Toundation for Statistical Computing Plotform: x86.64-apple-dommin3.4.0 (64-bit) R is free software and comes with ASSQUITLY NO MARSANTY. You are welcome to redistribute it under certain conditions. Type 'licensec') or 'licensec') for distribution details. Natural language support but running in an English locale R is a collaborative project with many contributors.	-0	Files Plots Packages Help	Viewer RStudio RStudio IDE Support RStudio Tip of the Day RStudio Takkages	
Conside -/Deckhop/Personal/R Vocabulary/R-Vocabulary/ /0 Copyright (C) 2815 The R Foundation for Statistical Computing Platform: #86_decglet_downial, 4.8 (64-bit) R is free software and comes with ABSOLUTELY NO MARSANY. Tou 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	-0	Files         Plots         Packages         Help           Image: State of the state	Vewer RStudio DE Support RStudio Tad by Annual RStudio Products	-0
Compute -(Desking)/Personal/A Vocabulary/ A-Vocabulary/ AD Copyright (C) 2815 The R Foundation For Statistical Computing Platform: x86.64-apple-dommin3.4.8 (64-bit) R is free software and comes with ABSOUTCI VN NABBANY. You are welcome to redistribute it under certain conditions. Type 'licenseC' or 'licenceC' for distribution details. Natural language support but running in an inglish locale R is a collaborative project with many contributors. Type 'contributors()' for more information and 'citation()' on how to cit R For R pockages in publications.	-0	Rise         Packages         Help           Image: Strategy         Image: Strategy         Image: Strategy           Home • (mage: Strategy)         Image: Strategy         Image: Strategy           Image: Strategy         Image: Strategy         Image: Strategy         Image: Strategy           Image: Strategy         Image: Strategy         Image: Strategy         Image: Strategy         Image: Strategy           Image: Strategy         Image: Strategy         Image: Strategy         Image: Strategy         Image: Strategy         Image: Strategy           Image: Strategy<	Vewer RStudio RStudio IDE Support RStudio Products RStudio Products	-
Console -/Deaking/Personal/R Vocabulary/R-Vocabulary/ 20 Copyright (C) 2815 The R Toundation for Statistical Computing Platform: #86.de-opple-downial.4.8 (66-hit) R is free software and comes with ABSOLUTELY NO RABBANY. Tou are swelcome to realistribute it under cartain 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 citle R or R pockages in publications.	<b>D</b> _	Ries         Packages         Help           Image: Second Se	Viewer RStudio DE Support RStudio IDE Support RStudio Products RStudio Products	-0
Compile -(Desking)/Personal/N Vocabulary/ PS-Vocabulary/ PS Copyright (C) 2815 The R Foundation for Statistical Computing Platform: x86.64-opple-domini3.4.0 (64-bit) R is free software and comes with A850UTLIV NO KARBANY. You are welcome to radistribute 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 home to cite R or R packages in publications. Type 'deemo()' for some deemos, 'help()' for on-line help, or 'help.start()' for an iNLB browser interface to help.	-0	Files         Packages         Help           Image: Second S	Vewer RStudio RStudio IDE Support RStudio Tajo of the Day RStudio Products	-0
Conside -/Deskapp/Hrvsmal/K Vocabulary/M-Vocabulary/ /20 Copyright (C) 2015 The R Toundation for Statistical Computing Platform: 386,de-opple-domini14.24 (64-bit) R is free software and comes with ABSOLUTLY NO RABOUNT. Tou are sellower to redistribute it under cartain conditions. Type 'losses'/or 'literect'/ for distribution destills. Notural language support but running in an English locale R is a collaborative project with many contributors. Type 'contributory' for more information and 'clibition()' on hom to clib R or R pockapps in publications. Type 'down'r for some desor, 'help()' for on-line help, or 'help.start()' for an HTML browser interface to help. Type 'down'r built R.	-D	Rise         Packages         Help           Image: Second Se	Verwer	-
Conside -/Okakap/Persand/R Vocahalary/R-Vocahalary/ /0 Copyright (C) 2815 The R Foundation for Statistical Computing Platform: x86.64-opple-domini3.4-0 (64-bit) R is free software and comes with ASSOUTIL VO NARRANY. You are selcome to realistribute it under certain conditions. Type 'losens()' or 'licerec()' for distribution details. Natural language support but running in an English locale R is a collaborative project with many contributors. Type 'controlutory()' for more information and 'citation()' on how to cite R or R pockages in publications. Type 'deem()' for some demos, 'help()' for on-line help, or 'help.start()' for an INB browser interface to help. Type 'de	-D	Files         Packages         Help           Image: Second S	Vewer RStudio RStudio IDL Support RStudio Tip of the Day RStudio Products The R Language Definition R Installation and Administration	-
Conside -/Deckado/Fersand/K Vocabulary/R-Vocabulary/ 20 Copyright (C) 2825 The R Toundation for Statistical Computing Platform: 386,de-opple-domini14.3 4 (64-bit) R is free software and comes with ABSOLUTLY NO RABOUNT. Tou are selcome to indistribute it under cartain conditions. Type 'losses'/or 'literact' for distribution destills. Notural language support but running in an English locale R is a collaborative project with many contributors. Type 'contributory' for some information and 'clibition()' on hom to clib R or R pockages in publications. Type 'down't por some dease, 'help()' for on :Inhelp, or 'help.start()' for an HTML browser interface to help. Type 'down't on the R. St	5a	Files         Packages         Help           Image: Second S	Viewer © RStudio RStudio IDE Support RStudio Tip of the Day RStudio Products RStudio Products The R translation and Administration R Internals	-
Conside -/Okakap/Persand/R Vocahalary/R-Vocahalary/ /0 Copyright (C) 2815 The R Foundation for Statistical Computing Platform: x86.64-opte-domini3.4.0 (64-bit) R is free software and comes with ASSOUTILY NO MARSANY. You are welcome to realistribute it under certain conditions. Type 'locare()' or 'licerece()' for distribution details. Natural language support but running in an English locale R is a collaborative project with may contributors. Type 'onterprotect()' for one information and 'citation()' on how to cite R or R packages in publications. Type 'demo()' for an HTML browser interface to help. Type 'd()' to guit R. > Supre('TYP, R') > PV(1080, 188, 5) [] 688.58	-D	Rise         Packages         Help           Image: Second Se	Viewer © RStudio RStudio IDL Support RStudio Tip of the Day RStudio Products RStudio Products The R Language Definition R Installation and Administration R Internals	-

# 18.8 Additional Resources

Functions are a fundamental building block of R and writing functions is a core activity of an R programmer. It represents the key step of the transition from a mere "user" to a developer who creates new functionality for R. As a result, its important to turn your existing, informal knowledge of functions into a rigorous understanding of what functions are and how they work. A few additional resources that can help you get to the next step of understanding functions include:

- Hadley Wickham's Advanced R book
- Roger Peng's R Programming for Data Science book
- DataCamp's Intermediate R course
- Coursera's R Programming course