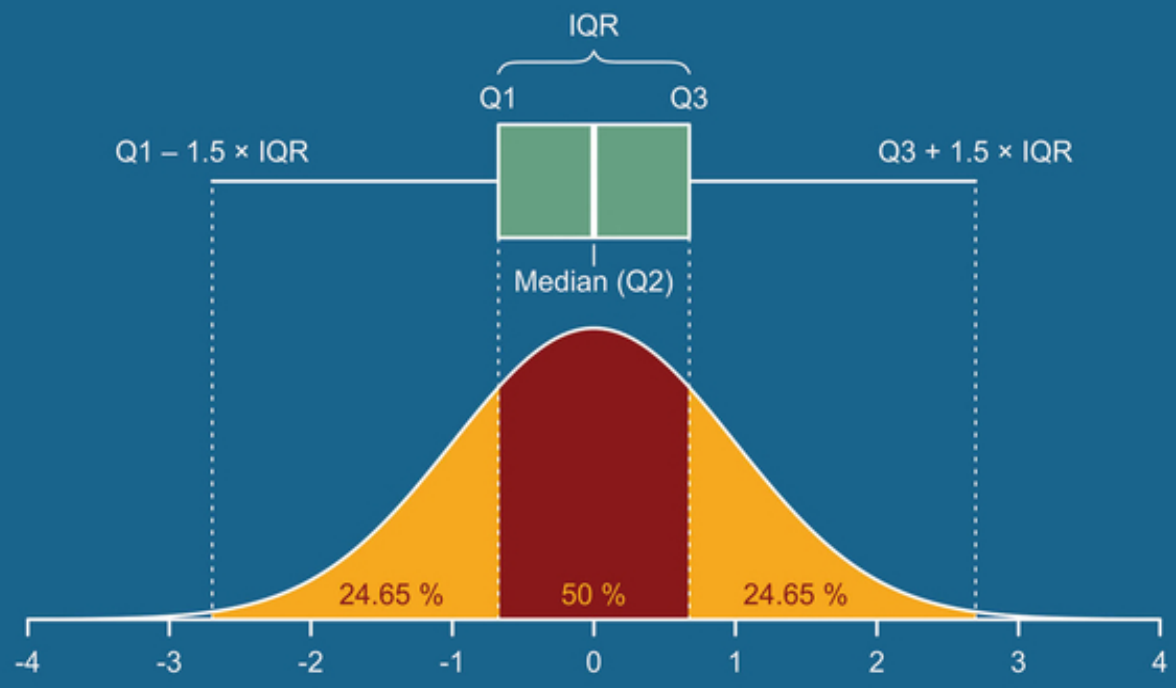
MARTIN BADER
SEBASTIAN LEUZINGER

R-TICULATE

A BEGINNER'S GUIDE TO DATA ANALYSIS FOR NATURAL SCIENTISTS



with website

WILEY

R-ticulate

R-ticulate

A Beginner's Guide to Data Analysis for Natural Scientists

Martin Bader
Department of Forestry and Wood Technology
Linnaeus University
Växjö, Sweden

Sebastian Leuzinger
School of Science
Auckland University of Technology
Auckland, New Zealand

WILEY

Published by John Wiley & Sons, Inc., Hoboken, New Jersey.
Published simultaneously in Canada.

Library of Congress Cataloging-in-Publication Data

Names: Bader, Martin (Professor), author. | Leuzinger, Sebastian, author.
Title: R-ticulate : a beginner's guide to data analysis for natural scientists / Martin Bader and Sebastian Leuzinger.
Description: Hoboken, New Jersey : Wiley, [2024] | Includes index.
Identifiers: LCCN 2024001120 (print) | LCCN 2024001121 (ebook) | ISBN 9781119717997 (cloth) | ISBN 9781119718000 (adobe pdf) | ISBN 9781119718024 (epub)
Subjects: LCSH: Science–Data processing. | Science–Statistical methods. | R (Computer program language)
Classification: LCC Q183.9 .B33 2024 (print) | LCC Q183.9 (ebook) | DDC 502.85/5133–dc23/eng20240215
LC record available at https://lccn.loc.gov/2024001120
LC ebook record available at https://lccn.loc.gov/2024001121

Cover Design: Wiley
Cover Image: © Martin Bader

Set in 9.5/12.5pt STIXTwoText by Straive, Chennai, India

Contents

Foreword

It has been over 30 years since Robert Gentleman and I began work on what would eventually become "R". At the time, we had rather modest goals; we thought that we might be able to use the software to support our own teaching and perhaps some research.

The development of the concept of free software and the GNU system in particular was attractive to us from both philosophical and practical viewpoints. Our decision to release R as free software using the GNU license created by Richard Stallman was fateful. To our surprise, the software seemed of interest to a large number of talented developers who quickly became collaborators. The work of this group was instrumental in creating a platform that appealed to both users and developers.

Despite its now venerable age, R remains a useful tool. Thousands of add-ons have been created that have moved R into areas of application that we could not have envisaged when we started work on it. This book is an example of this kind of use. It shows that R is suitable as a companion software for those students who major in subjects other than statistics and mathematics.

If you put R to use, please do so in the spirit that it was created. Collaboration is a very powerful tool, and freely sharing the results of your work creates benefits for all, including you.

January 2024

Dr Ross Ihaka
Co-founder of R, Auckland

You can't have too many stats books – at least that seems to be the message from my office book shelves. For a mathematical subject, statistics is surprisingly philosophical and multi-faceted – rich enough to justify many views and presentations. And then there's the issue of how to implement the chosen analysis. Here, Bader and Leuzinger provide a non-technical, reader-friendly introduction to linear model analysis using R. An attractive feature of the book is the extensive use of explanatory boxes and the demonstration of key ideas (e.g. pseudoreplication) through simple simulations. The 12 chapters provide an accessible introduction to basic statistical principles, all while providing the keener student with a glimpse of the more advanced methods. The authors' approachable and dynamic style makes this textbook a gem for students and teachers who are seeking quick access to modern statistical methods.

February 2024

Prof. Andy Hector
University of Oxford, UK

Preface

The way statistics is taught in undergraduate university courses for natural scientists has changed enormously over the past two decades. On the one hand, the use of a software accompanying both theory and practical sessions has become the standard, facilitating access to an unprecedented and ever-increasing wealth of statistical methods. On the other hand, the ease with which some of these more complex methods can be applied has meant that the theoretical understanding of what is going on under the hood has suffered. In teaching undergraduate statistics over many years, we have found that there is a danger of skipping some of the basics and heading straight to the strategy of 'googling – copying code – trying to apply it to my data', or, more recently, letting AI do the heavy lifting. This may result in an uncritical and sometimes incorrect use of statistical techniques. We seriously doubt that one can ever understand variance modelling or even a simple non-parametric test without a fundamental comprehension of the basic concepts of distributions, test statistics, but also more broadly applicable ideas such as the principle of parsimony. In our teaching experience, we found that there is a gap between introductory texts that have encyclopaedic character and lack conciseness and the myriad of more specialised texts that are clearly out of reach for an undergraduate student.

Our aim was to produce a surmountable, paced, 12-chapter text which suits the typical 12 week (1 semester) layout of many undergraduate courses. Just like in our classroom teaching, we use easy-to-understand language to explain key concepts and strive to strike a balance between the practical application of statistical models and a conceptual understanding of the theoretical foundations. The content of the chapters can be used selectively, and we deliberately go deeper where we deem detail is useful. As such, the text may in part also suit postgraduate students of the natural sciences. Our philosophy generally suits the curious student's attitude of 'wanting to know why', but we make sure we summarise the key concepts for those who are after a simple 'recipe' to get there. Margin texts and dedicated boxes will help with this. We demonstrate the use of base R as well as that of more recent developments (tidyverse) as we firmly believe that both have their place, and taking the best of both worlds makes most sense. Exercises and datasets are available online to save space and keep the volume of the text 'light', particularly for those who prefer to work with a hard copy. We hope this text will help you to *R-ticulate* yourself well in a data-driven world where statistical analysis and modelling skills are ever more important and in high demand with employers, far beyond the natural sciences.

January 2024

Martin Bader
Växjö, Sweden

Sebastian Leuzinger
Auckland, New Zealand

About the Companion Website

This book is complemented by a companion website.

www.wiley.com/go/Bader

The companion website provides all datasets used in the book, as well as exercises with solutions for each chapter.

1

Hypotheses, Variables, Data

In a world where information circulates at unprecedented speed and almost exclusively via the internet, often on indiscriminate platforms, it has become increasingly difficult to distinguish between *fact* and *fake*, between *true* and *false*, and between *evidence* and *opinion*. One admittedly non-spectacular, yet indispensable way to confidently plough our way through the jungle of those dichotomies is learning and understanding the basics of statistical thinking. Thinking statistically needs training, as it is not intuitive. Humans are particularly bad at 'collecting' data. For example, the perception of whether we had a 'warm' or 'cold' winter will depend on a wealth of subjective factors such as how much time someone has spent outdoors, and likely shows little correlation with the actual average temperature of that particular winter. Similarly, people perceive risks in life in an utterly 'non-statistical' way. While, statistically, the largest risks for early death in most western countries are sugar intake and lack of exercise, we often perceive the risk of an airplane crashing or a great white shark attack as much more threatening to our lives. In fact, the mentioned risks are four to five orders of magnitude (about 100,000 times) apart!

Human perception is a notoriously bad statistician

In this introductory chapter, we will set the foundations for 'statistical thinking', and the most basic statistical skills, which are the pillars that scientific thinking in a broader sense rests on. From our own experience, confusion at a later stage of someone's scientific career is often caused by a lack of knowledge of the fundamentals of statistical principles. For example, we often get approached by students asking us for statistical help, but then the initial conversation shows that the student is not clear about how many variables they are looking at, what their nature is, which of them might be a response or a predictor variable, what the unit of replication is, and so forth. Confusion can also originate following data collection in the absence of a sound scientific hypothesis, or from the absence of a sound study design. Pretty much everything is bound to go wrong from there if these foundations are not laid in time. The purpose of this chapter is therefore to introduce a basic statistical vocabulary, including the formation of a good scientific hypothesis, and how it relates to your data. This also includes training ourselves to dissect and categorise the datasets we encounter. We need to clarify the number of variables we are looking at, as well as their nature and purpose in the dataset.

Alongside all this, we will slowly ease you into the use of the statistical software 'R'. Despite this adding an element of complexity, it is useful to learn the theory at the same time as the use of a software. We recommend that you replicate our examples on your computer to get a hands-on experience.

R-ticulate: A Beginner's Guide to Data Analysis for Natural Scientists, First Edition.
Martin Bader and Sebastian Leuzinger.

Companion website: www.wiley.com/go/Bader

1.1 Occam's Razor

The principle of parsimony (or Occam's razor) is an extremely useful one that should accompany us not only in our statistical thinking, but also generally in all our scientific work. The idea goes back to William of Ockham, a medieval philosopher, who articulated the superiority of a simple as opposed to a more complex explanation for a phenomenon, given equal explanatory power. An illustrative example is the heliocentric model (the earth rotating around the sun) versus the geocentric model (which has the Earth in the centre of the solar system). Both are (or were) used to explain the trajectories of objects in the sky. It turns out that the heliocentric model is far simpler with equal or superior explanatory power, and is thus preferable. The search for the simplest possible explanation has been at the origin of many scientific discoveries, and it is useful to let this principle guide us throughout all stages of planning and conducting our data collection and analysis. The iconic phrase 'It is in vain to do with more what can be done with fewer' is applicable to the various stages of scientific research process:

'It is in vain to do with more what can be done with fewer'

(1) Planning stage. What is our hypothesis? What data do we need to answer the question? The clearer our (scientific) question is formulated, the easier it is to decide what kind of data we need to collect. 'De-cluttering' the relationship between the question we want to answer and the data we collect can save us an enormous amount of time and frustration.
(2) Data cleansing and organising stage. Remove any unnecessary element in our data, use minimal (but unique) nomenclature, both for variable names and values. A well organised and simplified dataset forces us to gain a much better understanding of it.
(3) Analysis stage. Use the minimum number of explanatory variables (and their interactions) that best explains the patterns in your response variables. Every additional explanatory variable will explain some variation in the response by pure chance. We will get to know statistical tools that help us decide on the most *parsimonious* model, i.e. the smallest set of explanatory variables that explain the most variation in the response variable(s).
(4) Presentation stage. Once we are ready to convey our (statistical) findings, we again use 'Occam's razor' to reduce and 'distill' our texts, figures, and tables. This means the removal of any frills that do not serve to make our results more comprehensible. Often, information (for instance, graphs) can be condensed, white space can be minimised or used for insets. In short, we maximise the ratio of information conveyed over the space used (see Chapter 6).

Maximise $\left\{\frac{\text{Information}}{\text{Space, Time}}\right\}$ and $\left\{\frac{\text{Explanatory power}}{\text{\# of explanatory variables}}\right\}$

We will revisit most of the aforementioned strategies in chapters to follow. The guiding principle can be extended to many more areas of science not covered in this book, such as scientific writing and oral presentations.

1.2 Scientific Hypotheses

Sometimes we are confronted with complete datasets that we obtain from somewhere, which we then have to analyse. Other times, we pose the scientific question ourselves, and we then set out to collect the data. In both cases, we need to be absolutely clear what question we

can or want to answer, as this question is linked with our *scientific hypothesis*. A scientific hypothesis is a well-founded assumption that must be testable. For example, 'The majority of consumers under 20 years of age prefer to pay cashless', serves as a scientific hypothesis, while 'Young people prefer to pay by card' lacks the level of precision required for a scientific hypothesis that we can test, but it may serve as an idea that could lead to a scientific hypothesis. A mismatch between an early (often vague) idea of what we want to research, the resulting hypothesis, and the data we set out to collect, can lead to much confusion.

Be absolutely clear what question you want to answer, and how it could be turned into a hypothesis!

Say, we are interested in biodiversity changes in rock pools on a rocky shore along a tidal gradient. This initial idea could lead to various hypotheses, which would require different data collections. If our hypothesis is 'The closer the rock pools to the high tide mark, the fewer species per pool', we would want to count the number of species per rock pool. However, if our hypothesis is 'species A, B, and C become more abundant moving from the low to the high tide mark, while species D, E, and F become less abundant', then we would count the number of individuals per species and rock pool. The hypothesis 'Biodiversity decreases along a tidal gradient' may not be precise enough to decide exactly what kind of data need to be collected. It is always advisable to have a clear idea of the units we are using (also referred to as the 'metric', or the 'reference metric' to refer to the denominator of the metric only). For example, if we count the number of species *per pool*, we will get a fundamentally different outcome compared to if we had counted the number of species *per cubic metre of water*, or *per square metre of rock pool surface*. In some instances, the choice of the reference metric can completely change the meaning of the data: while the 'cases of Covid-19 per 1000 people' in a country can increase from one week to another, the 'cases of Covid-19 per 1000 administered Covid tests' can decrease at the same time. These examples illustrate how quickly confusion can arise, which can lead to misinformation, but also how easily a statistically uninformed audience can deliberately be misled. Having a crystal clear idea of what the research question is, what hypothesis we test, and what data we need to collect is therefore pivotal not only for our own work, but also to scrutinise and criticise data or findings that are presented to us.

What units are my variables in? What reference metrics am I using?

The aforementioned considerations are important, but so far they are purely theoretical. To really understand and 'experience' these issues, we need to 'leave the classroom' and get our hands dirty!

1.3 The Choice of a Software

This topic can be delicate, as people's beliefs, experiences, preferences, and even commercial interests play a role, such that the choice of a statistical software is ultimately a very subjective decision. We will spare you with all the arguments that could be used to advertise R. It is important to note that the R project is entirely non-commercial, so the software is absolutely free. R is easy to download on all major operating systems. We additionally recommend downloading the user interface 'RStudio', which is also free for non-commercial individual use. With easily accessible online help, there is not much more to instruct here than telling you to install R (by searching for 'R software' and then clicking your way through), and installing 'RStudio' in the same way. You should then always open 'RStudio' rather than R, as 'RStudio' will run R in the background.

1.3.1 First Steps in R

Open 'RStudio'. Go to file -> New file -> R Script (basically just a text file with your code). Figure 1.1 illustrates what RStudio will look like. Focus on the left two windows, the top one

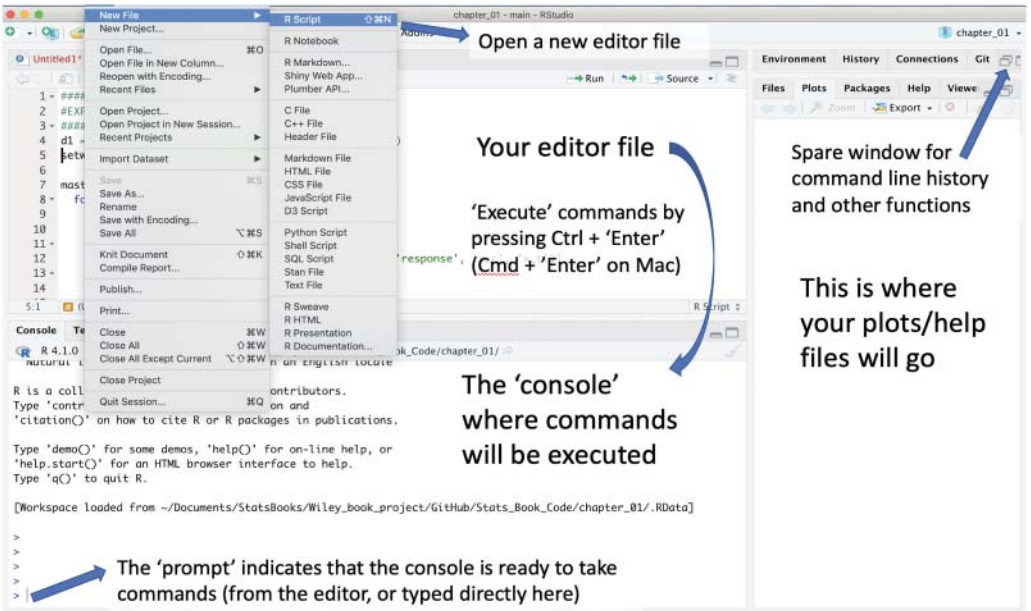


Figure 1.1 RStudio, showing the editor file (top left), the console (bottom left), and two extra windows to the right for plots, help, etc. (the top one has been collapsed here).

Send code from the editor to the console by pressing CTRL + Enter (PC) or Command + Enter (Mac)

is the editor (where your code will go), and the bottom one is the so-called 'console', where code is evaluated. That means that at the top, you can write whatever you like, R will not care until you *evaluate* or *run* it, by sending the code to the console. To illustrate this, type '1 + 1' into the top window (the editor), and press enter. You will see that nothing happens, except that the cursor jumps onto the next line, just like in a text document. Do the same thing in the console – and you will see the result (2) appear, and the prompt ('>') pops back, awaiting the next task. We will see what the '[1]' that you see in front of the result means later. As a beginner, you will often make typos, forget brackets, etc., causing R to get stuck or hung up, which you notice by the missing prompt sign in the console. Instead, you usually see a 'plus' sign in the console because R awaits further input (e.g. missing brackets). Often, it is difficult to figure out where things have gone off the rails. If this happens, simply hit the ESCAPE key, which forces R to terminate the incomplete code and the prompt should pop back up.

Mathematical operators:
* times
/ divided by
() brackets
^ to the power of
sqrt() square root
exp() exponent
log() logarithm

The best way to become familiar with the use of R is to simply play around. Much of what follows sounds more complicated than you will find once you get your hands 'dirty'. Start with playing in the console, for instance, by typing mathematical operations. You can do the same thing in the editor of course, and then 'send' your commands to the console to be evaluated. For example, type '10-7' into the editor, and then simultaneously press 'Ctrl' and 'enter' ('Cmd' and 'enter' on Mac). The cursor will stay up in the editor window, but you will see the result in the console, and you are ready for the next command to be evaluated. See margin for a number of mathematical operators that R can interpret and you can play with, e.g.:

```
> ## R as a calculator
> 1 + 2 * 3
[1] 7
```

```
> 2 * 5^2 - 10 * 5
[1] 0
> 4 * sin(pi / 2)
[1] 4
> 0 / 0  #not defined (Not a Number)
[1] NaN
```

R is also a powerful calculator!

The hashtag (#) allows you to leave comments in your code – anything after the hashtag is ignored by R!

So when should you type into the editor, and when directly into the console? Whatever you type into the console is not saved, but code that is typed into the editor file will be saved. So use the console to try things out, and maintain a clean editor file with code that works, and that you want to save. It is advisable to make sure the entire code runs if you send it down to the console, i.e. you do not receive any error messages. Here is a list of tips and tricks that make your start with RStudio easier:

- If you want to evaluate several lines of code that you have written in the editor, highlight the part of code you would like to evaluate, then press 'Ctrl' and 'enter' (simultaneously, henceforth referred to as *evaluating* or *running* code); partial code evaluation also works when you highlight the desired bit of code with the cursor, i.e. you can execute a command that is wrapped inside one or more commands. Very helpful when studying complex code examples!
- If you press 'Ctrl' and 'enter' with no code highlighted, the whole line that the cursor sits in will be evaluated (regardless of the exact position of the cursor).
- If you would like to leave a comment that R should ignore, place a hashtag in front of it, e.g. 7*52 # days per year. R will then ignore any characters after the hashtag in that particular line, so it will calculate 7×52, but ignore the comment 'days per year'.
- If you are typing directly into the console, use the 'up' arrow to retrieve previously typed lines. This saves you a lot of typing!
- When you close RStudio, you will be asked which files you want to save. It is critical to save the editor file (essentially a series of instructions on what R is to do with your raw data). The 'workspace' contains the actual objects that are created, resulting in a relatively large file size. Saving the 'workspace' is much less important, as you will be able to recreate it easily with your saved code (as an editor file a.k.a. R script, usually with the suffix .R). It is only worthwhile saving the workspace if you have run some models or algorithms that took substantial computing time. In these cases, opening a saved workspace allows you to pick up things where you have left them.

Use the 'up' arrow to retrieve code in the console

1.4 Variables

The ability to develop a fundamental understanding of what variables we encounter in a dataset is at the onset of any data analysis. For example, if we cannot answer with confidence how many variables we are looking at, what distribution their values follow, whether they are likely predictor or response variables etc., then we are in for a lot of trouble. Some of the following points might appear trivial, but from our experience, it is important to go through these notions in detail in order to build on solid grounds.

1.4.1 Variable Names and Values

Whenever you come across data for the first time, try to see them in variable names and their values. This is a bit like sorting out a pile of matches – get all the heads (variable

names) lined up with their wooden ends pointing downwards (their values). We could, for example, have three variables in a dataset: age, height, and weight (those are the variable names). If we have four observations (values), those might be [41, 38, 73, 29] for the variable 'age', [178, 163, 159, 181] for the variable 'height', and [82, 78, 59, 92] for the variable 'weight'. It seems simple and easy to see those three matchsticks line up in front of our imaginary eye.

Use '=' or '<-' to assign values to objects. They are equivalent. The shortcut 'alt' + 'dash' (minus key also works) should be used to efficiently type the assignment 'arrow'

To make this a little less theoretical, it is best to set up these variables in R:

```
> ## Assigning variables
> age <- c(41, 38, 73, 29)
> height = c(178, 163, 159, 181)
> weight <- c(82, 78, 59, 92)
```

The `c` is an R function (see Box 1.1), just like `sqrt` used earlier. It stands for 'concatenate', so we are asking R to arrange the values in the parenthesis in the form of a variable. Note that the '=' sign is equivalent to the frequently used 'assignment arrow' ('<-'). We can now collate those variables into a dataset. R calls this a 'data frame':

Use '`data.frame`' to create a dataset from variables

```
> d1 <- data.frame(age, height, weight)
> d1 # show the dataset
  age height weight
1  41    178     82
2  38    163     78
3  73    159     59
4  29    181     92
```

When assigning objects as earlier, note that the names (age, height, weight, d1) can be chosen freely. We use 'd1', meaning 'dataset 1', as often we make changes and end up with a modified dataset, which we can then call 'd2'. Also note that unless you subsequently type the object name (e.g. 'd1') into the console, you will not see the dataset, as it is simply assigned to the object name you have chosen. Make sure you do not overwrite existing function names such as `data` or `df`, as tempting as it may be. The command `data` (used without any argument) shows the built-in datasets and those available in loaded R packages. It is also used to load data from packages (that do not come with the base installation) into the current R session. The command `df` is the density function of the F-distribution. R will not get corrupted when you assign an object to an existing function name, but such stunts may entail some funny behaviour down the track.

Before assigning an object to a name, make sure the name is available in R and not already used as a function name or in-built object

Box 1.1 Functions in R

The syntax of R is simple in the sense that whatever you do, you are mostly calling functions, i.e. you type the name of the function (e.g. `round`), open a parenthesis, type the arguments that the function requires separated by commas, and close the parenthesis. For example, the function `round` requires 2 arguments: (i) a number or an entire variable whose values are to be rounded, and (ii) the number of decimals desired. In the round function, the second argument has a default value (zero), i.e. if you do not specify it, R will leave zero decimals to the number(s) you want to round. RStudio will display what