# Statistical Programming with Case Studies in R

Brian Muse[1] and Charles Carter[2]

August 5, 2016



[1]muse_william@columbusstate.edu
[2]carter_charles@columbusstate.edu

# Contents

# Preface

this is the preface!

# Chapter 1

# An Introduction to the Research Process

this is chapter one!

# Chapter 2

# Computer-Assisted Analysis

Why do we need computer assisted analysis? In order to answer this question, we need to consider both what a computer does, and what a programer does.

A computer computes, or more elegantly, a computer is just a computational machine, but an immensely powerful one. To illustrate its power, assume that we have in our employ a number of super mathematicians. Assume that one mathematician can perform one computation a second, 60 computations a minute, 60 minutes an hour, for 2000 hours a year, and that our mathematician is never tired, sick, in meetings, or needs professional time. How many computations can our mathematician do in a year?

```
#computations per second
number.of.computations <- 1
#computations per minute
number.of.computations <- number.of.computations * 60
#computations per hour
number.of.computations <- number.of.computations * 60
#computations per year
number.of.computations <- number.of.computations * 2000
# our mathematician is good for
number.of.computations

## [1] 7200000
```

Now, let's say that our computer is just an ordinary machine, nothing special, and that the processor runs at a clock speed of 2 gHz, or two billion computations a second. How many mathamaticians do we need to equal the speed of one ordinary computer?

```
    number.of.mathematicians <- 2000000000 / number.of.computations
    number.of.mathematicians

## [1] 277.7778
```

As you can see, it takes 277.7777778 mathematicians working for a year to perform the same number of computations that a computer can do in one second! This is amazing power, and leads us to the second question, what does a programmer do? What a programmer does is ... think. A computer is dumb, all it knows how to do is compute. A human, otherwise known as a programmer, tells the computer what to compute. Currently, we have at our fingertips incredible amounts of data. Ordinary computers have storage capacity of 100 gigabytes, and storage capacities of 1000 gigabytes are relatively common. Research computers can store terabytes ($10^{12}$ bytes), petabytes ($10^{15}$ bytes), and exabytes ($10^{18}$ bytes) of data. For these amounts of data, computer assisted analysis is an absolute prerequisite.

This chapter introduces the R programming language, perhaps the most widely used computer assisted analytical software in the world.

**Exercise:** Name two or three commonly used technologies for data analytics.



## 2.1 The R programming language

### 2.1.1 Invoking and quitting R

This chapter assumes that you have R up and running. If you do not, please read appendix A on page 11. Details of invocation differ according to the operating system you use. This tutorial assumes that you use some current version of Microsoft Windows. If you have a shortcut on our desktop, you invoke R by opening the shortcut. You can also invoke R via the start menu. If you choose to use an integrated development (IDE), you invoke R by opening your IDE, see appendix B on page 13. If you open R program on Windows, you will see the R graphical user interface (GUI), a bare bones command interpreter, shown in figure 2.1. It's also possible to invoke R on the command line interface (the DOS or Powershell prompt), but we will not cover this.

You quit R by typing the command `q()` at the R prompt. You will see an alert box asking you whether you wish to save. You should always save your image unless you have a good reason not to. See figure 2.2. You may also give the command `q('yes')` to directly save the image, or `q('no')` to quit without saving the image. On Windows, if you save the image, R creates a file in your working directory named `.RData`. You can invoke R by clicking on this link in the normal fashion, and your previous workspace will be restored. See figure 2.3.

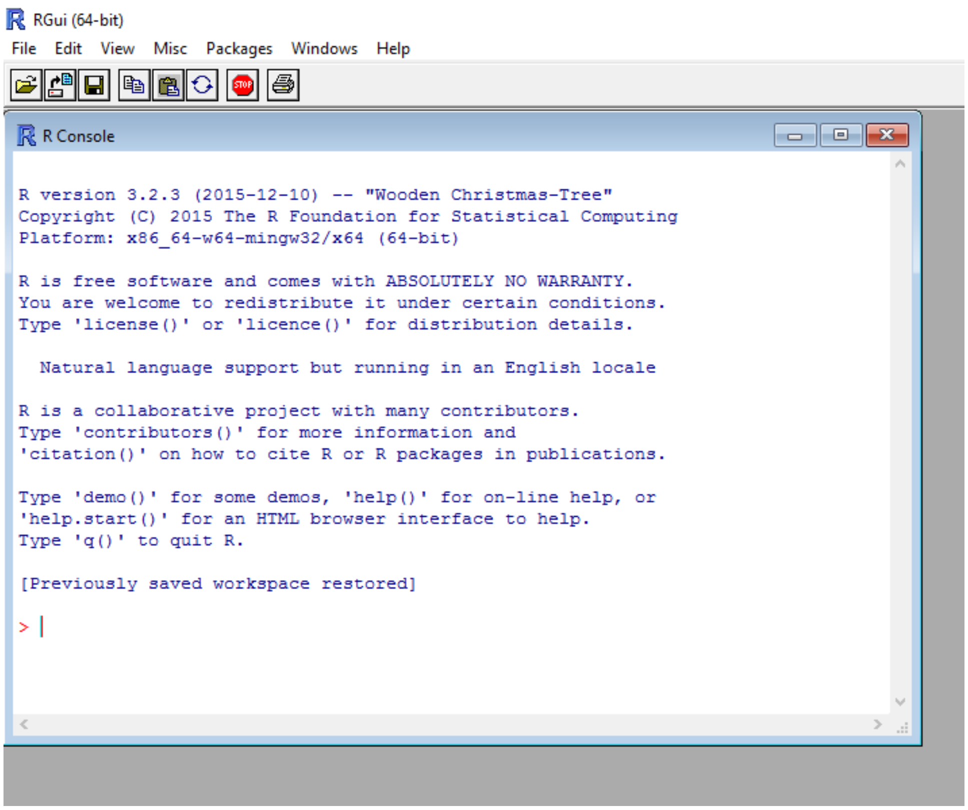


Figure 2.1: R GUI

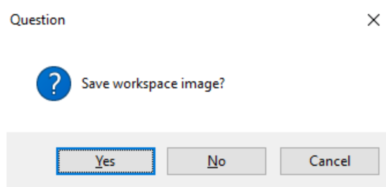


Figure 2.2: R quit alert

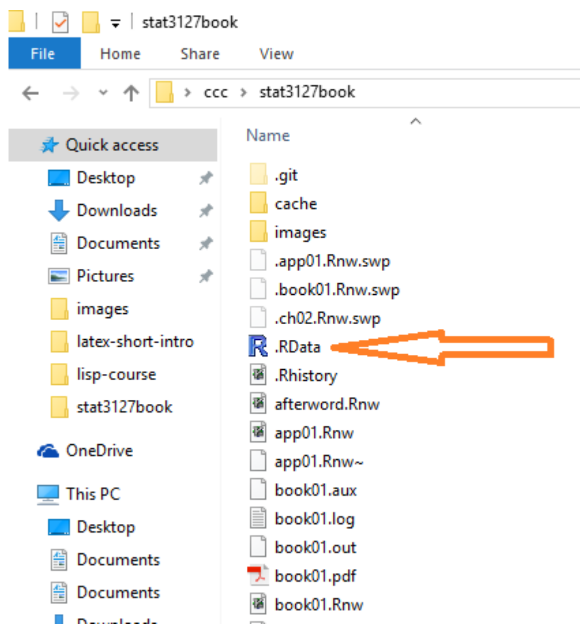


Figure 2.3: R.data file

**Exercise:** Invoke R in the usual manner, then quit R saving the workspace, and invoke R by clicking on the `.RData` file icon.

### 2.1.2 Directory and file operations

We will begin a tour of R with directory and file operations. These allow you to use R as a command shell, which may come in handy occasionally, however, you will mostly use your operating system's file and directory manipulation facilities. Here are the commands you should run.

```
#get help for getwd()
?getwd
#list the current working directory
getwd()
#create a new child directory named 'testing'
dir.create("testing")
```

```
#change to the testing child directory
setwd("testing")
#create a numeric vector
num.vec <- c(1,2,3,4,5)
#print the num.vec
num.vec
#save the vector as a CSV file
write.csv(num.vec, "my-num-vec.csv")
#copy the file
file.copy("my-num-vec.csv", "another-num-vec.csv")
#list the directory contents
dir()
#delete the files you just created
file.remove("my-num-vec.csv")
file.remove("another-num-vec.csv")
#check to see if it's still there
dir()
#return to the parent directory
setwd("..")
#see what unlink does
?unlink
#remove the testing directory
unlink("testing")
#is it still there?
dir("testing")

```

Let's see how this works.



```
#list the current working directory
getwd()

## [1] "C:/Users/ccc/stat3127book"

#create a new child directory named 'testing'
dir.create("testing")
#change to the testing child directory
setwd("testing")
#create a numeric vector
num.vec <- c(1,2,3,4,5)
#print the num.vec
num.vec

## [1] 1 2 3 4 5

#save the vector as a CSV file
write.csv(num.vec, "my-num-vec.csv")
#copy the file
file.copy("my-num-vec.csv", "another-num-vec.csv")

## [1] TRUE

#list the directory contents
dir()
```

```
## [1] "another-num-vec.csv" "my-num-vec.csv"

#delete the files you just created
file.remove("my-num-vec.csv")

## [1] TRUE

file.remove("another-num-vec.csv")

## [1] TRUE

#check to see if it's still there
dir()

## character(0)

#return to the parent directory
setwd("..")
#remove the testing directory
unlink("testing")
#is it still there?
dir("testing")

## character(0)
```



### 2.1.3 Data types

### 2.1.4 Statistical operations

## 2.2 R packages

### 2.2.1 CRAN

### 2.2.2 knitr

### 2.2.3 ggplot2

### 2.2.4 dplyr

## 2.3 Example analyses

# Appendix A

# Installing and Running R()

this is appendix one!

Appendix B

# R Integrated Development Environments

this is appendix one part b!

# Afterword

this is the afterword!