Getting Started with R

Introduction

We are going to use R as a tool to explore ideas we've discussed in class. Through a series of questions, you will get the opportunity to use R to explore various statistical ideas and concepts. While R is a coding intense language, the assignments will be structured so that they encourage the exploration and understanding of statistical concepts and they don't merely turn into coding exercises.

However, it is important that you have a basic understanding of this software and its language. Exposure to a programming language (whether it's R, SAS, etc.) is valuable in many ways, in that it provides a good skill set for the job market and encourages the development and sophistication of logical thinking and expression. With that in mind, the main purpose for these assignments is to 1) introduce you all to a software package that you might encounter again (or at least benefit from knowing), and to 2) explore ideas (and extensions to those ideas) we've discussed in class.

Downloading Introduction Document(s)

Given that we have different levels of experience using R, this handout—in conjunction with "A (Very) Short Introduction to R" (Torfs & Brauer, 2014)—is meant to give a general introduction to the software and functions we'll be using in this class. To get started, download the Torfs & Brauer introduction:

1. Go to http://www.r-project.org/.
2. Click on the "Other" link under "Documentation" on the left side. (The links under "Documentation" provide valuable resources that are free to access and use!)
3. Click on the "contributed documentation" link in the third bullet point.
4. In the "Short Documents and Reference Cards" section, click on the pdf link for "A (very) short Introduction to R." (This section of contributed documents also has various reference cards. These are handy when you just can't remember the name of the function you want to use!)

Now you're ready to get started! While there are various introductory documents, this one is (very) short and has "ToDo" tasks that let you practice what they are talking about. (For a more comprehensive introduction, please see the document "An Introduction to R" (Venables, Smith & R Core Team, 2019) located under the Manuals link under "Documentation" on R home page.)

Directions

As you read the Torfs & Brauer introduction, try to complete the "ToDo" tasks in R by yourself. Use the information provided below as needed for extra support, comparisons, extra tidbits, etc. This assignment is not graded for accuracy, but it will be beneficial for future assignments that are graded.

Section 2.1: Install R

The site located at http://www.r-project.org/ provides links for downloads as well as documentation of manuals, FAQs and many other resources.

1. Go to http://www.r-project.org/.
2. Click on the CRAN link under "Download" on the left side and choose a CRAN mirror to download. (The 0-Cloud mirror or any mirror from the United States should suffice.)
3. In the box labeled Download and Install R, click on the link appropriate for the type of operating system you use.

Section 2.2: Install RStudio

Although this program is not necessary to use R, it provides a better user interface.

1. Go to http://www.rstudio.com/.
2. Click on the Products tab and choose "RStudio" under "Open Source".
3. Scroll down to find the download link for RStudio Desktop.

A short video with installation instructions for R and RStudio can be found here: https://rconnect.math.montana.edu/InstallDemo/.

Section 2.4: Working Directory

```
# Comments, reminders, etc. can be typed after the pound sign

# Specify the location where you want everything saved, etc.
setwd("/Users/staceyhancock/Documents/Repos/staceyhancock/stat501")
```

Section 2.5: Libraries

Install the `BHH2` package using the command:

```
install.packages("BHH2")
```

This can also be accomplished by going to the Packages tab, clicking "Install", and searching for the package name, or by navigating to "Tools" -> "Install Packages…".

Once the package is installed, we can load it into our R session:

```
library("BHH2")
```

You only need to install a package once, but you need to load it into your session each time you start a new session.

Bonus: Import a CSV Data File

```
test <- read.csv(file = Survey_Data.csv)
## Error in read.table(file = file, header = header, sep = sep, quote = quote, : object 'Survey_Data.cs
```

Oh no! I forgot to put quotation marks around the file name.

```
test <- read.csv(file = "Survey_Data.csv")
```

If your file is in a different location than the working directory, you will need to specify where R needs to go to find your data file by either using a full pathname, or a relative pathname to direct R where to find the data file from the current working directory.

Double Bonus: There are two options for using the drop-down menu in RStudio to browse your directory and select a file:

1. Go to "File" -> "Import Dataset" -> "From Text (base)".
2. Click on the spreadsheet icon under the "Environment" tab and click "From Text (base)".

Notes:

- Using the drop-down menu is an easier way to read external data files into R, and the code is provided in the "Code Preview" box. Saving the code used to import the file is critical for reproducing what you did!
- Though .csv files sometimes open in Excel by default, they are not classified as "Excel" files in R (.xls or .xlsx).
- Note that `test` can be changed to any name you want to assign your dataset.

Section 3.1: Calculator

```
(2014 - 2013) / (2014 - 1982) * 100
## [1] 3.125
```

Section 3.2: Workspace

```
startdiff <- (2014 - 2013)
borndiff <- (2014 - 1982)
prop <- startdiff / borndiff
percent <- prop * 100

percent
## [1] 3.125

rm(list = ls())
```

Section 3.3: Scalars, vectors and matrices

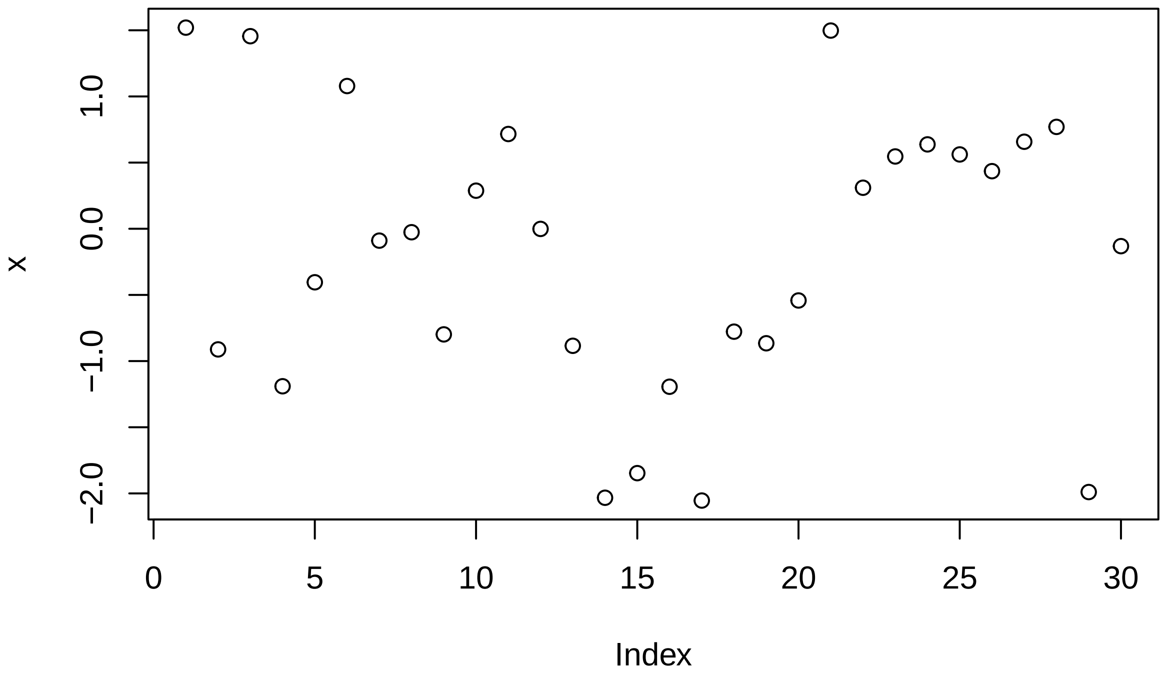
CAUTION! Their footnote about `=` vs. `<-` is accurate for some but not all situations. In general, I recommend using `<-` to make assignments in R to avoid having an unanticipated error.

Section 3.4: Functions

```
a <- c(4, 5, 8, 11)

sum(a)
## [1] 28
```

Section 3.5: Plots

```
x <- rnorm(30)

x
##  [1]  1.519928219 -0.911596961  1.455305087 -1.190284706 -0.404520562
##  [6]  1.078583219 -0.089628390 -0.026158452 -0.798207159  0.287846454
## [11]  0.716091350 -0.001238568 -0.884388968 -2.032869138 -1.846877910
## [16] -1.193784357 -2.053909840 -0.777574705 -0.865439972 -0.541812891
## [21]  1.497861813  0.310090487  0.546392799  0.637748266  0.561806311
## [26]  0.435251395  0.657570666  0.769877813 -1.989786062 -0.131347812

plot(x)
```



Section 4: Help and Documentation

Get help with the `sqrt` function by typing one of the lines below:

```
help(sqrt)
?sqrt
```

Section 5: Scripts

Type the following code into new script file (File -> New File -> R Script):

```
x <- rnorm(100)
x
plot(x)
```

Ready to run it? Highlight code and press “Run” or “CTRL/Enter” (PC) or “Command/Enter” (Mac).

Section 6.2: Matrices

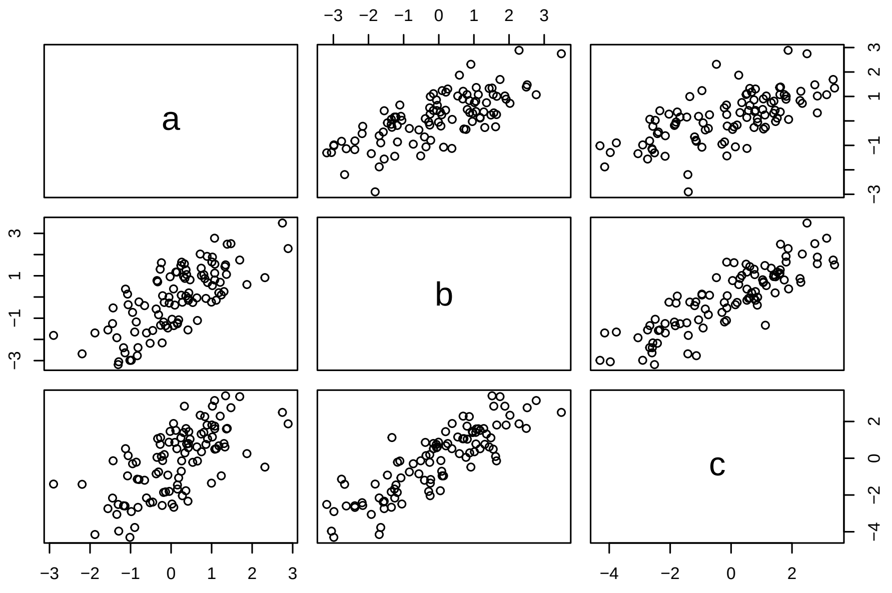
```
P <- seq(from = 31, to = 60, by = 1)

# Do we get the same matrix if we specify ncol instead of nrow?
Q <- matrix(P, nrow = 6)
Z <- matrix(P, ncol = 5)

Q
##      [,1] [,2] [,3] [,4] [,5]
## [1,]   31   37   43   49   55
## [2,]   32   38   44   50   56
## [3,]   33   39   45   51   57
## [4,]   34   40   46   52   58
## [5,]   35   41   47   53   59
## [6,]   36   42   48   54   60
z
## Error in eval(expr, envir, enclos): object 'z' not found
```

Oh no! I forgot R is case sensitive.

```
Z
##      [,1] [,2] [,3] [,4] [,5]
## [1,]   31   37   43   49   55
## [2,]   32   38   44   50   56
## [3,]   33   39   45   51   57
## [4,]   34   40   46   52   58
## [5,]   35   41   47   53   59
## [6,]   36   42   48   54   60

Q[4, 5]
## [1] 58

sum(Q)
## [1] 1365

mean(Q)
## [1] 45.5
```

Section 6.3: Data Frames

```
x1 <- rnorm(100)
x2 <- rnorm(100)
x3 <- rnorm(100)
t <- data.frame(a = x1, b = x1 + x2, c = x1 + x2 + x3)
plot(t)
```



```
sd(t)
## Error in is.data.frame(x): 'list' object cannot be coerced to type 'double'

sd(t$a)
## [1] 1.020093
```

Hmm. . . what is the difference between `sd(t)` and `sd(t$a)`?

Section 7: Graphics

This tutorial explores graphics functions in base R. It is a VERY brief intro to graphics—R can do so much more! We will also explore creating data visualizations through the `ggplot2` package (contained in the `tidyverse` package).

Try running the code below in your R session:

```
plot(t$a, type = "l", ylim = range(t))
lines(t$b, type = "s", lwd = 2, col = rgb(.3, .5, .3, .9))
points(t$c, pch = 20, cex = 4, col = rgb(0, 0, 1, .3))

?pch
example(pch)

# Wait a second! What does example() do?

??lwd

# Hmm...why did we use "??" instead of "?"

??cex
?rgb
?par
```

Section 8: Reading and Writing Data Files

```
a <- c(1, 2, 4, 8, 16, 32)

# Let's take a shortcut so we don't have to type all the numbers.
new <- data.frame(a, g = 2 * a, x = 3 * a)

new
##    a  g  x
## 1  1  2  3
## 2  2  4  6
## 3  4  8 12
## 4  8 16 24
## 5 16 32 48
## 6 32 64 96

write.table(new, file = "tst1.txt", rownames = FALSE)
## Error in write.table(new, file = "tst1.txt", rownames = FALSE): unused argument (rownames = FALSE)
```

Oops! I forgot the period in `row.names`.

Check out where this goes:

```
write.table(new, file = "tst1.txt", row.names = FALSE)
```

Let's read it back in:

```
new2 <- read.table(file = "tst1.txt", header = TRUE)

# Version 1 of new2
new2
##    a  g  x
## 1  1  2  3
## 2  2  4  6
## 3  4  8 12
## 4  8 16 24
## 5 16 32 48
## 6 32 64 96

# Let's modify the variable g
new2$g <- new2$g * 5

# Version 2 of new2
new2$g
## [1]  10  20  40  80 160 320

new2
##    a   g  x
## 1  1  10  3
## 2  2  20  6
## 3  4  40 12
## 4  8  80 24
## 5 16 160 48
## 6 32 320 96
```

Check out where this goes:

```
write.table(new2, "tst2.txt", row.names = FALSE)
```

Section 9: Not Available Data

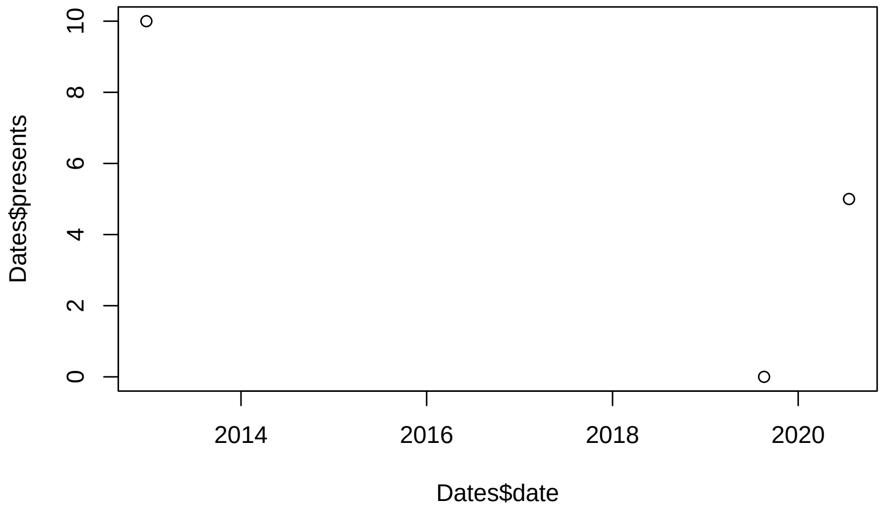
```
sqrt(mean(rnorm(100)))
## Warning in sqrt(mean(rnorm(100))): NaNs produced
## [1] NaN
```

Why did this error occur? Explore to find out!

Section 10.2: Dates

This is a boring plot! Can you make it more interesting?

```
Dates <- data.frame(date = strptime(c("20190820", "20121225", "20200719"),
                                    format = "%Y%m%d"),
                    presents = c(0, 10, 5))

Dates
##         date presents
## 1 2019-08-20        0
## 2 2012-12-25       10
## 3 2020-07-19        5

plot(Dates$date, Dates$presents)
```



Section 11.2: For-Loop

```
vec <- seq(from = 1, to = 100, by = 1)
s <- c()
for (i in 1:100) {
  if (vec[i] < 5 | vec[i] > 90) {
    s[i] <- vec[i] * 10
  } else {
    s[i] <- vec[i] * .10
  }
}

s
##   [1]   10.0   20.0   30.0   40.0    0.5    0.6    0.7    0.8    0.9    1.0
##  [11]    1.1    1.2    1.3    1.4    1.5    1.6    1.7    1.8    1.9    2.0
##  [21]    2.1    2.2    2.3    2.4    2.5    2.6    2.7    2.8    2.9    3.0
##  [31]    3.1    3.2    3.3    3.4    3.5    3.6    3.7    3.8    3.9    4.0
##  [41]    4.1    4.2    4.3    4.4    4.5    4.6    4.7    4.8    4.9    5.0
##  [51]    5.1    5.2    5.3    5.4    5.5    5.6    5.7    5.8    5.9    6.0
##  [61]    6.1    6.2    6.3    6.4    6.5    6.6    6.7    6.8    6.9    7.0
##  [71]    7.1    7.2    7.3    7.4    7.5    7.6    7.7    7.8    7.9    8.0
##  [81]    8.1    8.2    8.3    8.4    8.5    8.6    8.7    8.8    8.9    9.0
##  [91]  910.0  920.0  930.0  940.0  950.0  960.0  970.0  980.0  990.0 1000.0
```

Section 11.3: Writing Your Own Functions

```
ToDo <- function(low, high) {
  vec <- seq(from = low, to = high, by = 1)
  s <- c()
  for (i in low:length(vec))
  {
    if (vec[i] < low + 4 | vec[i] > high - 10)

    # What is (low+4) & (high-10) doing?

      {
        s[i] <- vec[i] * 10
      } else {
      s[i] <- vec[i] * .10
    }
  }
  s
}

ToDo(1, 100)
##   [1]   10.0   20.0   30.0   40.0    0.5    0.6    0.7    0.8    0.9    1.0
##  [11]    1.1    1.2    1.3    1.4    1.5    1.6    1.7    1.8    1.9    2.0
## [21]    2.1    2.2    2.3    2.4    2.5    2.6    2.7    2.8    2.9    3.0
## [31]    3.1    3.2    3.3    3.4    3.5    3.6    3.7    3.8    3.9    4.0
## [41]    4.1    4.2    4.3    4.4    4.5    4.6    4.7    4.8    4.9    5.0
## [51]    5.1    5.2    5.3    5.4    5.5    5.6    5.7    5.8    5.9    6.0
## [61]    6.1    6.2    6.3    6.4    6.5    6.6    6.7    6.8    6.9    7.0
## [71]    7.1    7.2    7.3    7.4    7.5    7.6    7.7    7.8    7.9    8.0
## [81]    8.1    8.2    8.3    8.4    8.5    8.6    8.7    8.8    8.9    9.0
## [91]  910.0  920.0  930.0  940.0  950.0  960.0  970.0  980.0  990.0 1000.0
```

Acknowledgements

This handout is adapted from Dr. Jennifer Green's "Getting Started with R" activity (M/STAT 501, Fall 2019).