Intro to R for High Schoolers

Scott LaForest

2020-02-22

Contents

Chapter 1

Prerequisites

We will have to install a few things before we get started using R to help us understand math and statistics.

1.1 Install R

Download and install the most recent version of R. Install R. If you are running the most updated version of OSX you should be fine to select the first download file, version 3.6.2.

1.2 Install RStudio

RStudion is the most popular integrated development environment (IDE) for R. Download RStudio. Again, if you are running the most updated version of OSX you should be fine to download the most recent version of RStudio.

That's it, now you are ready to follow along with the walkthrough.

Chapter 2

Walkthrough

2.1 What R you talking about?

R is a programming language that is popular among mathematicians, scientists, staticians, and many other professional careers that need to analyze and understand large amounts of data. R and Python are the two most popluar languages for working with data. If you plan on going into a STEM field or plan to do any research you will most likely need to have an understanding of how to program in one of these two languages. Now is a great time to learn how to analyze data programmitically.

2.2 Walkthrough

2.2.1 R Basics

We will start by playing around on the R Console. The console is in the lower left portion of the screen and should have a line that is blank except for a `>` symbol. Place your cursor next to the `>` to start typing inputs for R to evaluate. R can be used as a basic calculator. Try typing some mathematical expressions then press enter and R will evaluate them.

Follow along by typing the following commands into the R console.

```
3 + 4
```

```
## [1] 7
```

```
1/3
```

```
## [1] 0.3333333
```

```
2.5 * 3
```

```
## [1] 7.5
```

Nothing too impressive yet but that's not all R can do. Think of a mathematical function and try evaluating an expression.

```
sqrt(64)
```

```
## [1] 8
```

```
cos(60)
```

```
## [1] -0.952413
```

Oops, we just calculated the cos of 60 radians! Let's convert radians to degrees.

```
cos(60*pi/180)
```

```
## [1] 0.5
```

So far we've seen how R can function as a calculator but R can do so much more. Before we start working we need one more concept that will help us in our calculations. The programming concept of variables. A variable in programming acts as a bucket to hold a piece of data for later use. The benefit of using a variable is that we can set it equal to a value and use that value as many times as we want.

```
# This line is a comment and will not be interpreted by R
# We use comments to help explain what our code is doing, whatever comes after the # i
# The syntax for a variable is as follows var_name <- some_value
a <- 5
# Notice when you run this chucnk nothing prints out.
```

Now we can use `a` as many times as we want.

```
a
```

```
## [1] 5
```

```
a*3
```

```
## [1] 15
```

```
b <- a+3
```

```
b
```

```
## [1] 8
```

That should be enough R for now. Lets move on to linear regression.

2.2.2 Linear Regression with MTcars

Follow along via the R console. Our goal with the walkthrough is to look at correlation between variables and use linear regression to predict or infer meaning in the data. Let's start by looking at all the datasets that come installed in R. The code below should open another tab with a list of available datasets.

```
data()
```

We will start by looking at some relationships between variables in the mtcars dataset. Try the following code to learn more about the dataset.

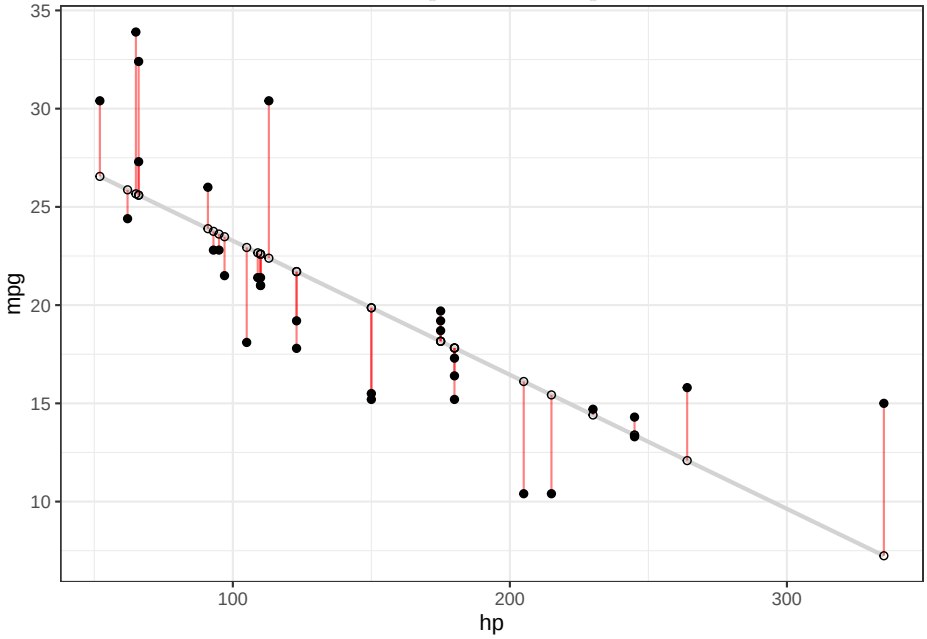
```
?mtcars
```

Notice that we get the definitionn of each of the variables included in the data set as well as the number of rows and variables. However, we do not get any examples of the data. Now let's take a look at a subset of the data.

```
head(mtcars)
```

```
##                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

Now we will try to select a variable to predict the mpg of a car. Before we get into the analysis, we need to have a quick review of linear regression. One of the most common types of linear regression is the Least Squares Regression (LSR). The goal with LSR is to find the line that minimizes the residuals between the points in the data and the LSR line. Remember that the residuals are the difference between our predicted value vs the actual. In the plot below the residuals are shown as the red line from the actual data point to the predicted value on the line.



The output of Least Squares Regression is a formula. In single variable regression the formula should look very similar to a typical slope-intercept form of an equation.

$$y = \alpha + \beta \cdot x$$

where α is the y-intercept of our LSR line and β is the coefficient of our independent variable also known as the slope. Remember that intuitively, β is the average increase or decrease for our dependent (y) variable when our independent (x) variable changes by one unit. R will do all of the work for us but it is useful to remember how to calculate both α and β.

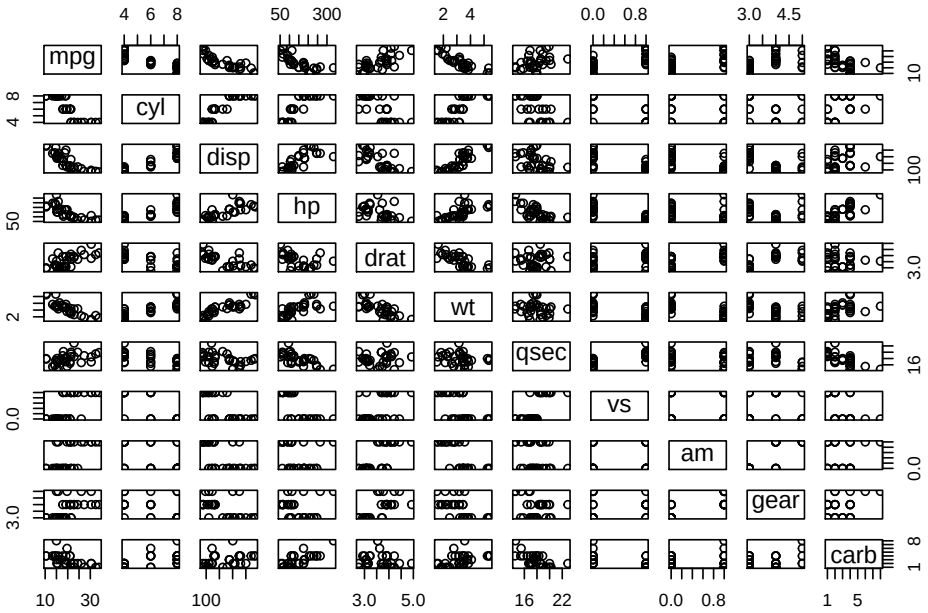
$$\beta = r \cdot \frac{S_y}{S_x}$$

where r is the correlation coefficient between y and x, S_y is the standard deviation of the y variable and S_x is the standard deviation of the x variable.

$$\alpha = \overline{y} - \beta \cdot \overline{x}$$

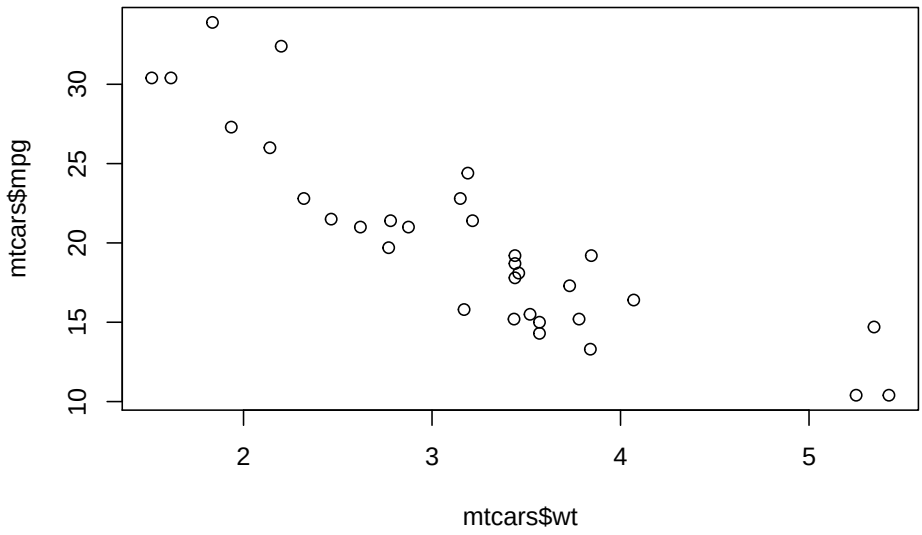
Now we have the necessary equations to allow us to find the LSR line for a set of data. If we had too, we could now compute the equation of the line of best fit using the two formulas above. Before we find the regression formula we need to find a variable that correlates well with our independent variable, mpg. Our first task is to get an overview of how each variable correlates with the mpg variable. A pairplot will give us a visual overview of the correlation between the variables.

```
pairs(mtcars)
```

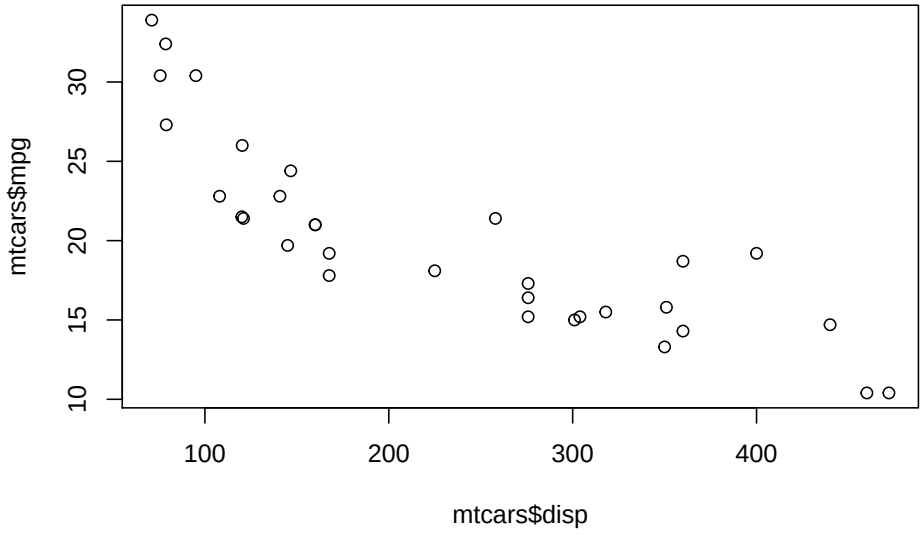


Whoa, that's a lot of information. However, we are interested in predicting mpg from a different dependent variable and we only need to look at the first row. It seems that `wt`, `disp`, `cyl`, `drat`,and `hp` might have some correlation with Temp. Lets plot each relationship separately with the following code.

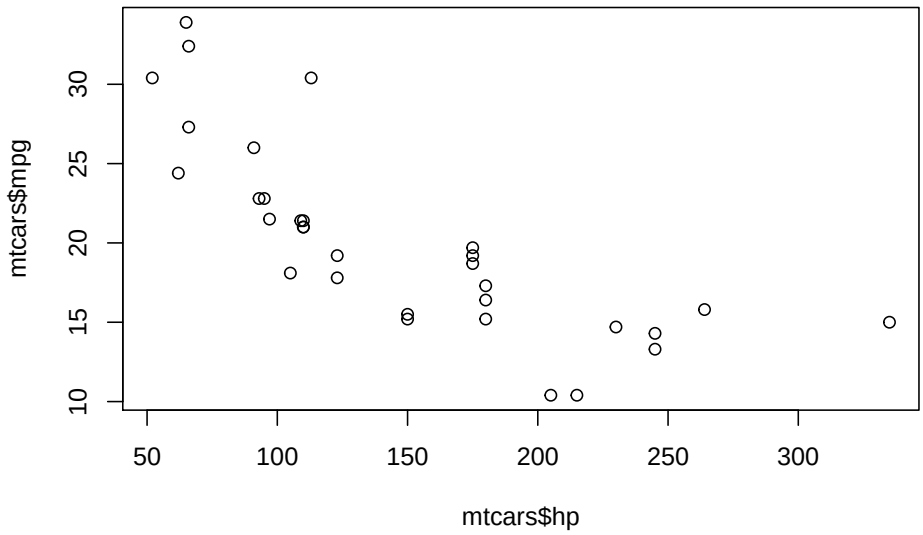
```
plot(mtcars$wt, mtcars$mpg)
```



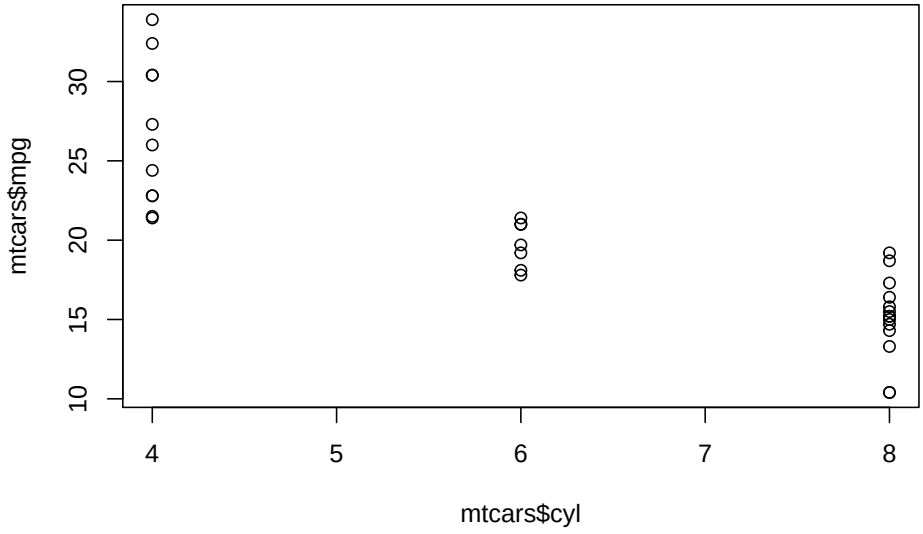

```
plot(mtcars$disp, mtcars$mpg)
```



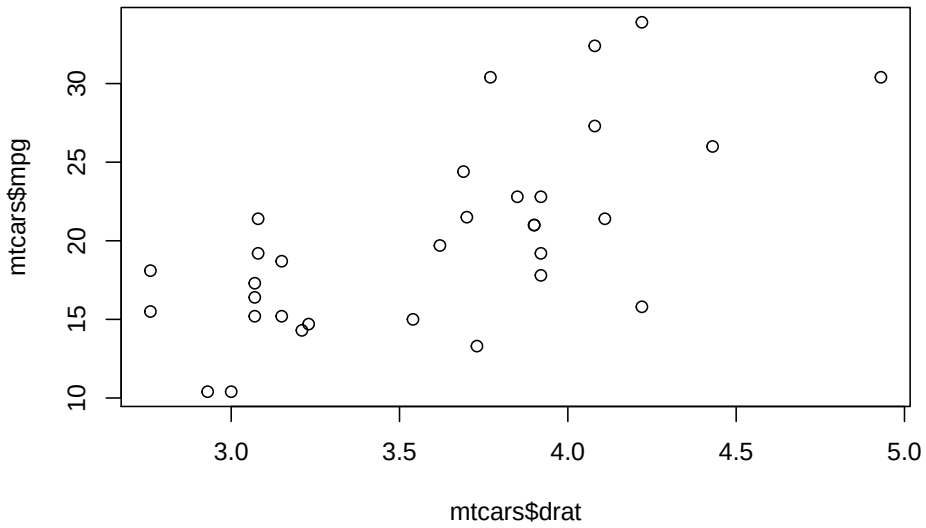

```
plot(mtcars$hp, mtcars$mpg)
```




```
plot(mtcars$cyl, mtcars$mpg)
```




```
plot(mtcars$drat, mtcars$mpg)
```



Which plot seems to show the highest correlation? Possibly `wt` is the highest negative correlation while `drat` seems to have a slightly positive correlation. Lets use the correlation coefficient to see for sure. The `cor()` function will calculate the correlation coefficient for us. One side note, we need to include the `use = 'complete.obs'` option to omit rows with blank values.

```
# cor without blanks and to make it easier to read only include the first row with [1,]
cor(mtcars, use = 'complete.obs')[1,]
```

```
##        mpg        cyl       disp         hp       drat         wt       qsec
##  1.0000000 -0.8521620 -0.8475514 -0.7761684  0.6811719 -0.8676594  0.4186840  0.664
```

Lets use some variables to help us calculate α and β.

```
# r can be calculated using the cor() function
r <- cor(x = mtcars$wt, y = mtcars$mpg)
# sd() calculates the standard deviation
sx <- sd(mtcars$wt)
sy <- sd(mtcars$mpg)
beta <- r*(sy/sx)
beta
```

```
## [1] -5.344472
```

```
ybar <- mean(mtcars$mpg)
xbar <- mean(mtcars$wt)
alpha <- ybar - beta*xbar
alpha
```

```
## [1] 37.28513
```

Now because I'm evil I decided to wait until the end to show a shortcut to calculate the equation of the LSR line. The lm() function allows us to do all of the above steps, and more, in one step. Look at the help documentation for the `lm()` function.

```
?lm
```

To use the `lm()` function, it is best to save the outcome in a variable. Also, the function parameter is a little weird and the syntax for the function parameter is `(dependent_variable ~ independent_variable)`.

```
model <- lm(mtcars$mpg ~ mtcars$wt)
```

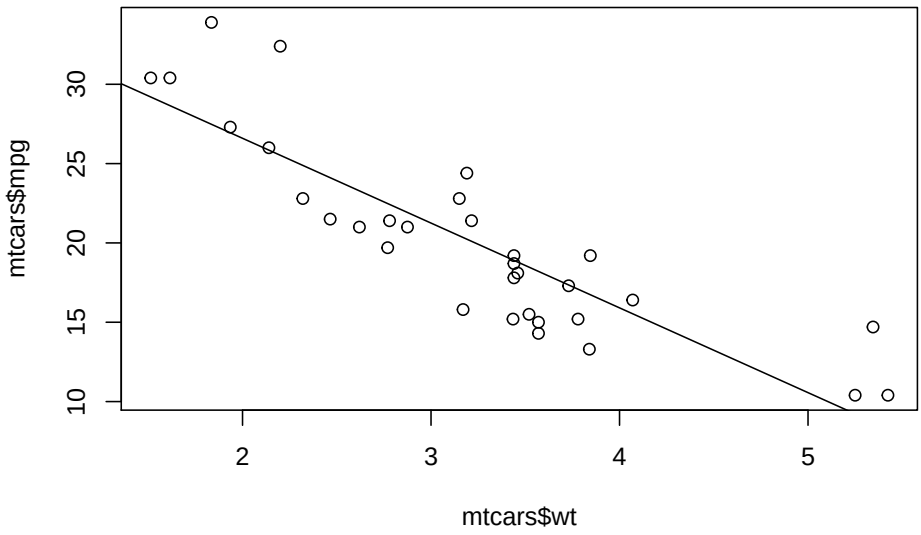
To see the outcome of our linear model we can use the `summary()` function.

```
summary(model)
```

```
##
## Call:
## lm(formula = mtcars$mpg ~ mtcars$wt)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.5432 -2.3647 -0.1252  1.4096  6.8727
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  37.2851     1.8776  19.858  < 2e-16 ***
## mtcars$wt    -5.3445     0.5591  -9.559 1.29e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.046 on 30 degrees of freedom
## Multiple R-squared:  0.7528, Adjusted R-squared:  0.7446
## F-statistic: 91.38 on 1 and 30 DF,  p-value: 1.294e-10
```

We can plot the line on top of our scatter plot to see our handywork!

```
plot(mtcars$wt, mtcars$mpg)
abline(model)
```


mtcars$mpg
10
15
20
25
30
2
3
4
5
mtcars$wt

Chapter 3

Minilab

3.1 R Markdown Intro

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

The way we will use it is to prove your R skills and apply what we have learned so far. The great thing about R Markdown is that you can type regular text as well as run R commands. R markdown files have `chunks` that let you run R in the same document. Try it below by checking the available datasets by calling the `data()` function. Type in `data()` and press the run (little green triangle) button on the chunk.

```
data()
```

To complete this minilab you will need to type your code and answers in the code chunks. You have already run some code but you can answer questions by typing your answer between the single quotes and run the chunk.

3.1.1 Q1: What is the description for the `airquality` dataset?

```
description <- 'New York Air Quality Measurements'
print(description)
```

```
## [1] "New York Air Quality Measurements"
```

The rest of the minilab will require you to type either an answer or some code in every code chunk.

Let's continue by exploring the `airquality` dataset. Type in the code to call for help on the dataset.

```
# type your code on the line below
```

3.1.2 Q2: How many observations and how many variables are included in the `airquality` dataset?

```
# Type your answers between the '' for each answer.
observations <- ''
variables <- ''
# Do not type below this line
paste('There are ' , observations , ' observations and ' , variables, ' variables.')
```

```
## [1] "There are    observations and    variables."
```

Take a look at the first five rows of the `airquality` dataset.

Lets look at the correlation between each of the variables. Remember to only include complete observations.

3.2 Q3: What variable has the highest correlation with the Temp variable? What is the correlation coefficient?

```
variable_name <- ''
cor_coeff <- ''

# Do not type below this line
paste( variable_name , 'is the most highly correlated with Temp and has a correlation
```

```
## [1] " is the most highly correlated with Temp and has a correlation coefficient of
```

Lets look at the correlation in a more visual manner. Plot the a scatterplot of each variable in the dataset, also known as a pair plot.

3.2.1 Q4: Which row and column seems to show the best correlation with Temp with Temp as the independent variable?

```
row_num <- ''
col_num <- ''

# Do not type below this line
paste('Row X Column:',row_num,"X",col_num, 'which is', colnames(airquality)[as.numeric(row_num)],
```

```
## [1] "Row X Column:  X  which is NA and NA"
```

Pair plots are can be hard to read. Plot a scatter plot for Ozone vs Temp, Solar.R vs Temp, and Wind vs Temp.

Finally, we need to fit a regression line to our data. Choosing the independent variable that is most correlated, fit a model and print out the summary of the model.

3.2.2 Q5: What are the α and β values for your regression?

```
alpha <- ''
beta <- ''

paste('Your equation is y = ', alpha, ' + ' , beta, 'X', sep = "")
```

```
## [1] "Your equation is y =  + X"
```

3.2.3 Q6: What is the R^2 value? What does this tell us?

```
r2 <- ''
meaning <- ''

paste('R^2 =' , r2,'and', meaning)
```

```
## [1] "R^2 =  and "
```

Chapter 4

Swirl Minilab

4.1 R Packages

We need to install a package before so that we can complete a minilab. We can think of a package like an app on our phone. When you first turn on a new phone you have the basic functionality that all other phones have. If you need more functionality, you install new apps. Packages are to R as apps are to your phone's operating system. The swirl package is a great way to learn how to use the R programming language. Install the package, run it, and learn R in the R console! Before we start we need to download a swirl course that we can use to test our understanding so far. Download this file so we can use it later.

Type the following code in the console. Notice once you start typing RStudio provides hints. You can press `tab` to select the function you want. We need to first download and install the package with `install.packages()` function.

```
install.packages('swirl')
```

The code above will start a process that will download and install the necessary parts so that we can use the swirl package.

Once the package is installed we need to tell R to use it with the `library()` function.

```
library(swirl)
```

Next, we need to install the course that we downloaded from above. The `install_course()` function will allow us to select a file to install as a course. Type the following command into the console and then find the `AS_Intro_to_R.swc` file. The file should be in your Downloads folder.

```
install_course()
```

Now we can start the swirl app with the `swirl()` function. The swirl program will ask you questions, allow you to type your answer, and provide feedback, all in the R console.

```
swirl()
```

Answer the initial setup questions and be sure to choose the "AS Intro to R" course. The course covers the topics we discussed in the walkthrough. Good Luck!