C02T01_Getting_Started_with_R

Code

Note to the reader

> markdown comments noted by the student/author (John Leonard) are highlighted in red. The final section of the document (section 7) contains the informal written report <>

Task 1: Get Started with R

Introduction Email

FROM.: Danielle Sherman
Subject: Get Started with R Hello,

Although we're generally happy with using RapidMiner as an analytics tool, many people in the industry seem to be moving to R, an open-source statistical programming language and analytics environment that is supported by a huge community of developers. Based on the excellent analysis work you have done thus far I have decided to ask you to explore introducing R into our current processes. To help you get started, I have obtained a walkthrough tutorial in R for you and a script of code to practice with.

Here are the things I would like you to do:

- Learn what R and RStudio are.
- Install R and RStudio.
- Learn how to work within RStudio.
- Upload a data set into RStudio.
- Install packages into RStudio.
- Call a package in RStudio.
- Perform basic exploratory data analysis.
- Preprocess data.
- Create test and training sets using your data.
- Develop a linear regression model
- Evaluate your model.
- Use your model to make to make predictions.

I have attached the data sets that you'll be using for this task. I'll be expecting a report on your experience in a few days.

Thanks, Danielle

Danielle Sherman Chief Technology Officer Blackwell Electronics www.blackwellelectronics.com

Link:R Tutorial Data (https://s3.amazonaws.com/gbstool/emails/2784/R%20Tutorial%20Data.zip?AWSAccessKeyId=AKIAJBIZLMJQ2O6DKIAA&Expires=1547110800&Signature=e1C8TaEwLLba30nj4oSxmemevaU%3D)

Plan of Attack

Introduction

Danielle has asked you to install R and R Studio on your machine, then to work through a tutorial to learn the basics of analytics and visualization using R. **Within your tutorial, you'll be working through a regression analysis using a linear regression model.** In this simple analysis, you'll be **predicting distances through the speed of certain cars.** The dataset that you'll be using through this analysis is the **cars.csv file**, which is located within Danielle's email.

Blackwell believes that the best way to learn is through trial and error, so **after your tutorial, you'll be asked to test your knowledge through running a script of code. The dataset that you'll be using through this analysis is the iris.csv file,** which is also in the zip file attached to Danielle's email. Don't be too surprised if you encounter errors or warning messages. Take a few deep breaths and embrace the errors!

Once you've completed both tasks, **you'll submit an informal report to Danielle about your experience switching to R, the errors that you encountered in the task, the reasons behind them, and how you overcame them. You should also include discussion of the outcomes your model predicted.**

1. Install R and R Studio

What is R?

R is a programming language for statistical use and data visualization. R is an interpreted language, which means you can run a line of code and receive an instant output.

How will I be learning R?

In this course, you will be using and learning R within the RStudio environment. RStudio is an Integrated Development Environment (a graphical user interface for R development.) Some Data Analyst believes that RStudio makes life easier for working with R, but you can also run R through a Command Line Interface.

R and RStudio are both open-source software, which means each piece of software is free and has active user and development communities. Because of these communities, a lot of resources have been created by fellow R programmers. Some examples are Stack-Exchange, Stack Overflow, etc.

Once you've installed R and RStudio you will be ready to move on to the next task.

How do I download R and RStudio?

Here is the website where you download R. https://cran.r-project.org/ (https://cran.r-project.org/)

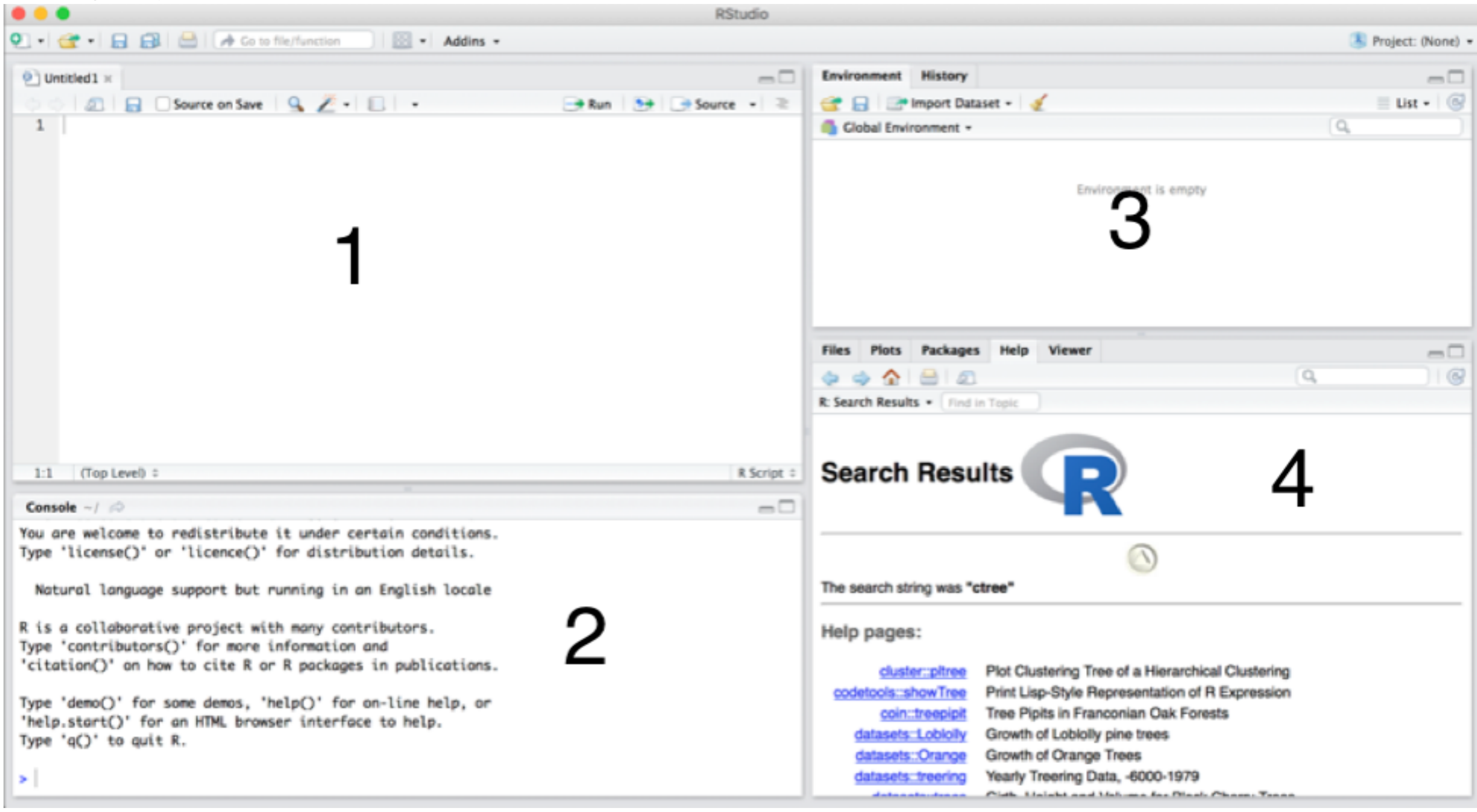
How do I install RStudio? Here is where you can download RStudio. https://www.rstudio.com/products/rstudio/download/ (https://www.rstudio.com/products/rstudio/download/)

2. Get to Know R Studio

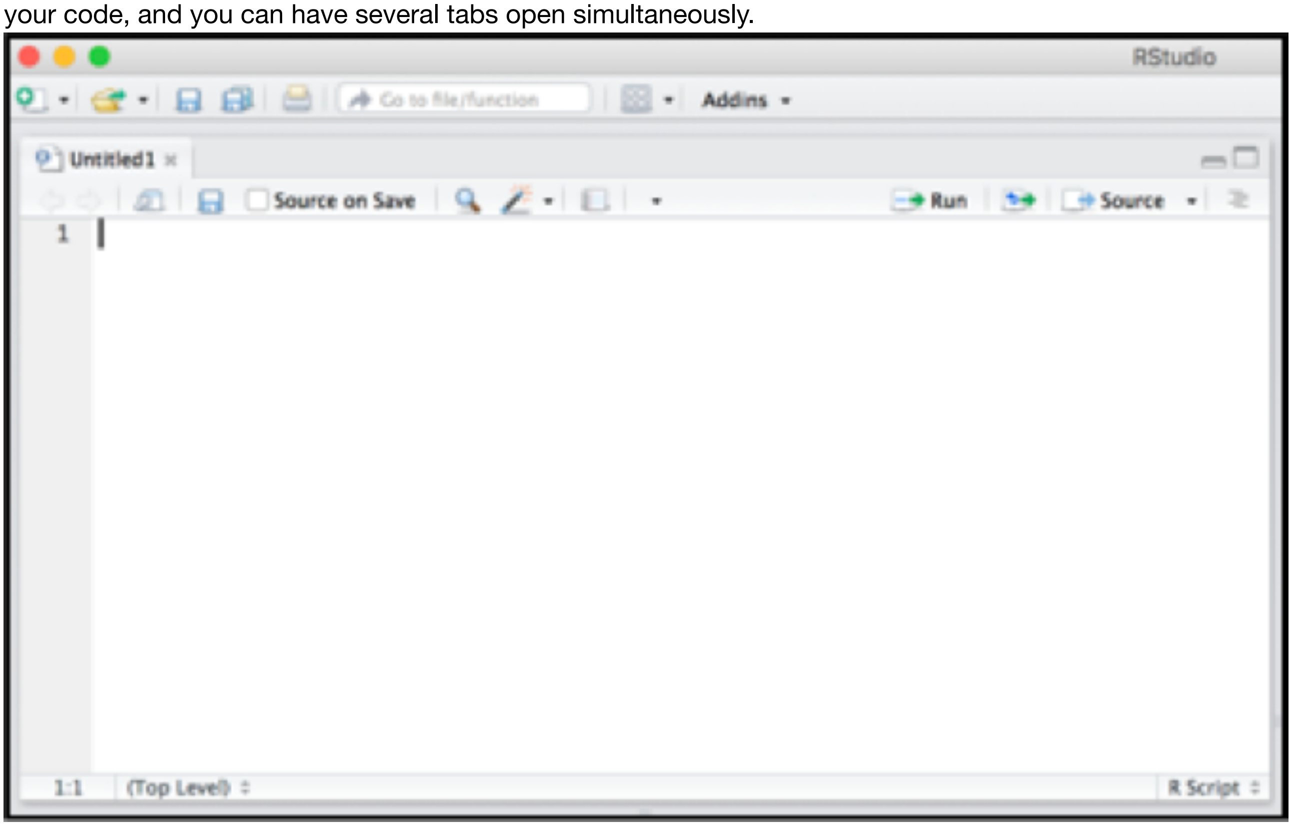
Within this section of the tutorial, you will learn RStudio's general layout and the functionality that each portion of the interface provides.

Becoming Familiar with RStudio

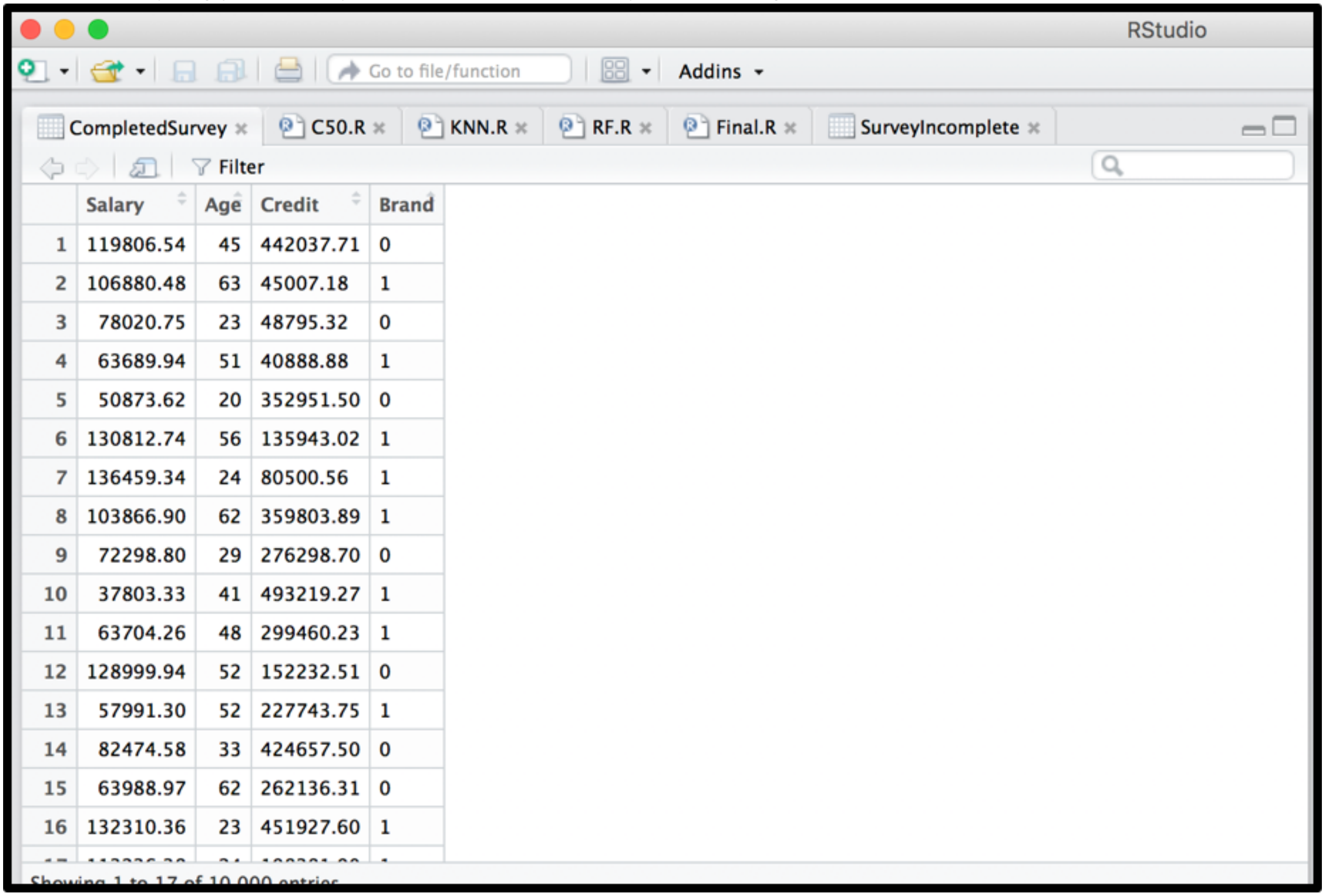
Before you begin learning R, let's open RStudio within your computer to take a look at how to work RStudio. After opening the program you will see four sections within the interface.



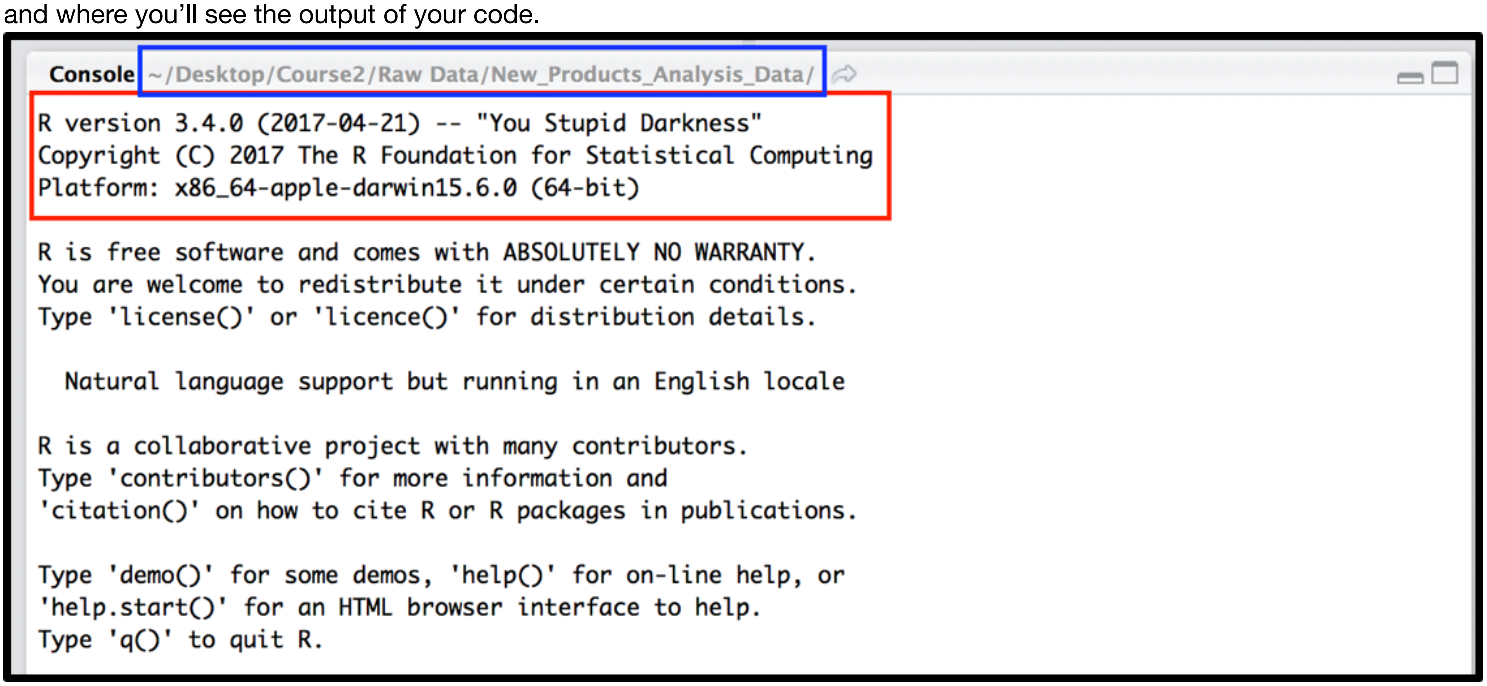
The 1st Section (Upper left box) is the R Script box. The R Script box is 1 of the 2 places that you can type in your code. The advantages of the R Script box are that you can edit your code, save your code (by clicking the blue floppy disk icon), run your code, and you can have several tabs open simultaneously.



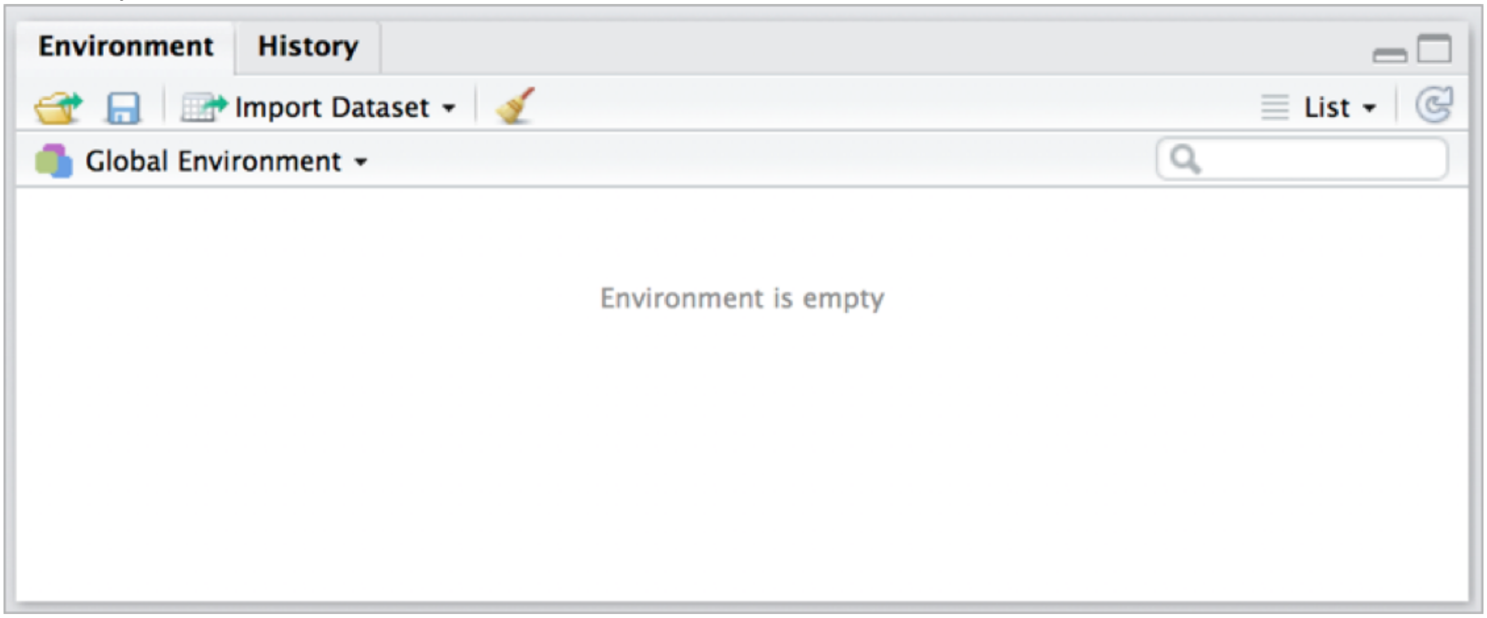
In addition to typing your code, you can also view the data you're working with. For example:



The 2nd section (lower left) is called the Console. The Console is the 2nd place you can type and run your code. It is also where you'll see which R version you are running (in the red box), which folder you're working in on your computer (in blue), and where you'll see the output of your code.



The 3rd section (upper right box) contains the Environment tab and the History tab. The History tab stores all code that you've created during your session. (It does not store outputs.) Within your Environment tab you can upload your dataset and save your environment.



In the 4th section (bottom right box), there are five tabs. The first tab, Files, displays the files and folders within your computer/RStudio. The second tab, Plots, is where you'll be able to see, save, and export the plots your produce. The third tab, Packages, shows the packages that you've downloaded or that are available to download. The fourth tab, Help, is where you can go to receive additional documentation concerning certain topics and packages. The fifth tab, Viewer, is used to for viewing web content. (We won't be using much of this tab during this tutorial.)

You're now ready to move on to the next task to begin learning R. If you'd like additional information on the RStudio IDE, here is a helpful cheat sheet. https://www.rstudio.com/wp-content/uploads/2016/01/rstudio-IDE-cheatsheet.pdf (https://www.rstudio.com/wp-content/uploads/2016/01/rstudio-IDE-cheatsheet.pdf)

1 LAYOUT	Windows/Linux	Mac
Move focus to Source Editor	Ctrl+1	Ctrl+1
Move focus to Console	Ctrl+2	Ctrl+2
Move focus to Help	Ctrl+3	Ctrl+3
Show History	Ctrl+4	Ctrl+4
Show Files	Ctrl+5	Ctrl+5
Show Plots	Ctrl+6	Ctrl+6
Show Packages	Ctrl+7	Ctrl+7
Show Environment	Ctrl+8	Ctrl+8
Show Git/SVN	Ctrl+9	Ctrl+9
Show Build	Ctrl+0	Ctrl+0

2 RUN CODE	Windows/Linux	Mac
Search command history	Ctrl+↑	Cmd+↑
Navigate command history	↑/↓	↑/↓
Move cursor to start of line	Home	Cmd+←
Move cursor to end of line	End	Cmd+ →
Change working directory	Ctrl+Shift+H	Ctrl+Shift+H
Interrupt current command	Esc	Esc
Clear console	Ctrl+L	Ctrl+L
Quit Session (desktop only)	Ctrl+Q	Cmd+Q
Restart R Session	Ctrl+Shift+F10	Cmd+Shift+F10
Run current line/selection	Ctrl+Enter	Cmd+Enter
Run current (retain cursor)	Alt+Enter	Option+Enter
Run from current to end	Ctrl+Alt+E	Cmd+Option+E
Run the current function	Ctrl+Alt+F	Cmd+Option+F
Source a file	Ctrl+Shift+O	Cmd+Shift+O
Source the current file	Ctrl+Shift+S	Cmd+Shift+S
Source with echo	Ctrl+Shift+Enter	Cmd+Shift+Enter

3 NAVIGATE CODE	Windows /Linux	Mac
Goto File/Function	Ctrl+.	Ctrl+.
Fold Selected	Alt+L	Cmd+Option+L
Unfold Selected	Shift+Alt+L	Cmd+Shift+Option+L
Fold All	Alt+O	Cmd+Option+O
Unfold All	Shift+Alt+O	Cmd+Shift+Option+O
Go to line	Shift+Alt+G	Cmd+Shift+Option+G
Jump to	Shift+Alt+J	Cmd+Shift+Option+J
Switch to tab	Ctrl+Shift+.	Ctrl+Shift+.
Previous tab	Ctrl+F11	Ctrl+F11
Next tab	Ctrl+F12	Ctrl+F12
First tab	Ctrl+Shift+F11	Ctrl+Shift+F11
Last tab	Ctrl+Shift+F12	Ctrl+Shift+F12
Navigate back	Ctrl+F9	Cmd+F9
Navigate forward	Ctrl+F10	Cmd+F10
Jump to Brace	Ctrl+P	Ctrl+P
Select within Braces	Ctrl+Shift+Alt+E	Ctrl+Shift+Alt+E
Use Selection for Find	Ctrl+F3	Cmd+E
Find in Files	Ctrl+Shift+F	Cmd+Shift+F
Find Next	Win: F3, Linux: Ctrl+G	Cmd+G
Find Previous	W: Shift+F3, L: Ctrl+Shift	Cmd+Shift+G
Jump to Word	Ctrl+ ←/→	Option+ ←/→
Jump to Start/End	Ctrl+↑/↓	Cmd+↑/↓

4 WRITE CODE	Windows /Linux	Mac
Attempt completion	Tab or Ctrl+Space	Tab or Cmd+Space
Navigate candidates	↑/↓	↑/↓
Accept candidate	Enter, Tab, or →	Enter, Tab, or →
Dismiss candidates	Esc	Esc
Undo	Ctrl+Z	Cmd+Z
Redo	Ctrl+Shift+Z	Cmd+Shift+Z
Cut	Ctrl+X	Cmd+X
Copy	Ctrl+C	Cmd+C
Paste	Ctrl+V	Cmd+V
Select All	Ctrl+A	Cmd+A
Delete Line	Ctrl+D	Cmd+D
Select	Shift+[Arrow]	Shift+[Arrow]
Select Word	Ctrl+Shift+ ←/→	Option+Shift+ ←/→
Select to Line Start	Alt+Shift+←	Cmd+Shift+←
Select to Line End	Alt+Shift+→	Cmd+Shift+→
Select Page Up/Down	Shift+PageUp/Down	Shift+PageUp/Down
Select to Start/End	Shift+Alt+↑/↓	Cmd+Shift+↑/↓
Delete Word Left	Ctrl+Backspace	Ctrl+Opt+Backspace
Delete Word Right		Option+Delete
Delete to Line End		Ctrl+K
Delete to Line Start		Option+Backspace
Indent	Tab (at start of line)	Tab (at start of line)
Outdent	Shift+Tab	Shift+Tab
Yank line up to cursor	Ctrl+U	Ctrl+U
Yank line after cursor	Ctrl+K	Ctrl+K
Insert yanked text	Ctrl+Y	Ctrl+Y
Insert <-	Alt+-	Option+-
Insert %>%	Ctrl+Shift+M	Cmd+Shift+M
Show help for function	F1	F1
Show source code	F2	F2
New document	Ctrl+Shift+N	Cmd+Shift+N
New document (Chrome)	Ctrl+Alt+Shift+N	Cmd+Shift+Alt+N
Open document	Ctrl+O	Cmd+O
Save document	Ctrl+S	Cmd+S
Close document	Ctrl+W	Cmd+W
Close document (Chrome)	Ctrl+Alt+W	Cmd+Option+W
Close all documents	Ctrl+Shift+W	Cmd+Shift+W
Extract function	Ctrl+Alt+X	Cmd+Option+X
Extract variable	Ctrl+Alt+V	Cmd+Option+V
Reindent lines	Ctrl+I	Cmd+I
(Un)Comment lines	Ctrl+Shift+C	Cmd+Shift+C
Reflow Comment	Ctrl+Shift+/	Cmd+Shift+/
Reformat Selection	Ctrl+Shift+A	Cmd+Shift+A
Select within braces	Ctrl+Shift+E	Ctrl+Shift+E
Show Diagnostics	Ctrl+Shift+Alt+P	Cmd+Shift+Alt+P
Transpose Letters		Ctrl+T
Move Lines Up/Down	Alt+↑/↓	Option+↑/↓
Copy Lines Up/Down	Shift+Alt+↑/↓	Cmd+Option+↑/↓
Add New Cursor Above	Ctrl+Alt+Up	Ctrl+Alt+Up
Add New Cursor Below	Ctrl+Alt+Down	Ctrl+Alt+Down
Move Active Cursor Up	Ctrl+Alt+Shift+Up	Ctrl+Alt+Shift+Up
Move Active Cursor Down	Ctrl+Alt+Shift+Down	Ctrl+Alt+Shift+Down
Find and Replace	Ctrl+F	Cmd+F
Use Selection for Find	Ctrl+F3	Cmd+E
Replace and Find	Ctrl+Shift+J	Cmd+Shift+J

5 DEBUG CODE	Windows/Linux	Mac
Toggle Breakpoint	Shift+F9	Shift+F9
Execute Next Line	F10	F10
Step Into Function	Shift+F4	Shift+F4
Finish Function/Loop	Shift+F6	Shift+F6
Continue	Shift+F5	Shift+F5
Stop Debugging	Shift+F8	Shift+F8

6 VERSION CONTROL	Windows/Linux	Mac
Show diff	Ctrl+Alt+D	Ctrl+Option+D
Commit changes	Ctrl+Alt+M	Ctrl+Option+M
Scroll diff view	Ctrl+↑/↓	Ctrl+↑/↓
Stage/Unstage (Git)	Spacebar	Spacebar
Stage/Unstage and move to next	Enter	Enter

7 MAKE PACKAGES	Windows/Linux	Mac
Build and Reload	Ctrl+Shift+B	Cmd+Shift+B
Load All (devtools)	Ctrl+Shift+L	Cmd+Shift+L
Test Package (Desktop)	Ctrl+Shift+T	Cmd+Shift+T
Test Package (Web)	Ctrl+Alt+F7	Cmd+Alt+F7
Check Package	Ctrl+Shift+E	Cmd+Shift+E
Document Package	Ctrl+Shift+D	Cmd+Shift+D

8 DOCUMENTS AND APPS	Windows/Linux	Mac
Preview HTML (Markdown, etc.)	Ctrl+Shift+K	Cmd+Shift+K
Knit Document (knitr)	Ctrl+Shift+K	Cmd+Shift+K
Compile Notebook	Ctrl+Shift+K	Cmd+Shift+K
Compile PDF (TeX and Sweave)	Ctrl+Shift+K	Cmd+Shift+K
Insert chunk (Sweave and Knitr)	Ctrl+Alt+I	Cmd+Option+I
Insert code section	Ctrl+Shift+R	Cmd+Shift+R
Re-run previous region	Ctrl+Shift+P	Cmd+Shift+P
Run current document	Ctrl+Alt+R	Cmd+Option+R
Run from start to current line	Ctrl+Alt+B	Cmd+Option+B
Run the current code section	Ctrl+Alt+T	Cmd+Option+T
Run previous Sweave/Rmd code	Ctrl+Alt+P	Cmd+Option+P
Run the current chunk	Ctrl+Alt+C	Cmd+Option+C
Run the next chunk	Ctrl+Alt+N	Cmd+Option+N
Sync Editor & PDF Preview	Ctrl+F8	Cmd+F8

Previous plot	Ctrl+Alt+F11	Cmd+Option+F11
Next plot	Ctrl+Alt+F12	Cmd+Option+F12
Show Keyboard Shortcuts	Alt+Shift+K	Option+Shift+K

Why RStudio Server Pro?

Do everything you would do with the open source server with a commercial license, support, and more.

- edit the same project at the same time as others
- switch easily from one version of R to a different version
- open and run multiple R sessions simultaneously
- see what you and others are doing on your server
- tune your resources to improve performance
- integrate with your authentication, authorization, and audit practices

Download a free 45 day evaluation at
www.rstudio.com/products/rstudio-server-pro/

3. Walk Through the R Tutorial - Part 1

Within this section of the tutorial, you will learn how to start a new R Studio project, install packages, call on a package (library) and upload a dataset into RStudio.

Reading the Working with Projects in RStudio section of the Resources will help with the next steps.

Starting a new project

R Studio has a system for organizing your projects and keeping all of your work within its own working directory, workspace, history, and source documents. In order to work effectively and efficiently in R Studio it is important that you always start a new R Studio project whenever you start a new task. This will keep your work separate and make the work portable and easy to share with your classmates and your mentor. Here are the steps you should always follow when starting a new project in R Studio:

RStudio projects are associated with R working directories. You can create an RStudio project:

- In a brand new directory
- In an existing directory where you already have R code and data

- By cloning a version control (Git or Subversion) repository To create a new project use the New Project command (available on the RStudio File menu and on the global toolbar):

When a new project is created RStudio the following takes place:

1. Creates a project file (with an .Rproj extension) within the project directory. This file contains various project options (discussed below) and can also be used as a shortcut for opening the project directly from the filesystem.
2. Creates a hidden directory (named .Rproj.user) where project-specific temporary files (e.g. auto-saved source documents, window-state, etc.) are stored. This directory is also automatically added to .Rbuildignore, .gitignore, etc. if required.
3. Loads the project into RStudio and display its name in the Projects toolbar (which is located on the far right side of the main toolbar)

Installing Packages

How to install packages in RStudio: The package that you will need for this part of the tutorial is the readr package. The readr package is designed to read files, like csv files. (Although we will only be going over one alternative, there are multiple ways to install a package.)

You will be using the install.packages() function to install the readr package.

Hide

```
install.packages('readr')
```

```
Error in install.packages : Updating loaded packages
```

TIP: If you are typing in the R Script pane, press COMMAND + ENTER to run your code.

Calling on a Package

After the installation of the package, you need to call on it to have access to its functions. There are multiple ways to call a library. To see one of them, type this code line into your R Script OR Console section.

Hide

```
library(readr)
```

Uploading Your Data

Now it's time to upload the dataset that you'll be using during this tutorial (the .CSV file) into RStudio. When you perform this action, you are creating a data frame, which means that all of the data that is stored within the data frame is separated by columns. Like the previous steps, there are multiple ways to upload a dataset. Here is one option.

TIP: Your dataset is cars.csv, which is located within the zip file attached to Danielle's email, but you'll want to create a name for your dataset. You will also need to make sure that you read how to set up your working environment or this line of code will not work.

Hide

```
df_cars<- read_csv("R Tutorial Data Sets/cars.csv");
```

```
Parsed with column specification:
cols(
  `name of car` = [31mcol_character()[39m,
  `speed of car` = [32mcol_double()[39m,
  `distance of car` = [32mcol_double()[39m
)
```

Hide

```
View(df_cars)
print(df_cars)
```

name of car <chr>	speed of car <dbl>	distance of car <dbl>
Ford	4	2
Jeep	4	4
Honda	7	10
KIA	7	10
Toyota	8	14
BMW	9	16
Mercedes	10	17
GM	10	18
Hyundai	10	20
Infiniti	11	20

1-10 of 50 rows Previous **1** 2 3 4 5 Next

Hide

```
df_iris<- read_csv("R Tutorial Data Sets/iris.csv")
```

```
Missing column names filled in: 'X1' [1]Parsed with column specification:
cols(
  X1 = [32mcol_double()[39m,
  Sepal.Length = [32mcol_double()[39m,
  Sepal.Width = [32mcol_double()[39m,
  Petal.Length = [32mcol_double()[39m,
  Petal.Width = [32mcol_double()[39m,
  Species = [31mcol_character()[39m
)
```

Hide

```
df_iris <- subset(df_iris, select = -c(X1))
View(df_iris)
print(df_iris)
```

Sepal.Length <dbl>	Sepal.Width <dbl>	Petal.Length <dbl>	Petal.Width <dbl>	Species <chr>
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa
4.6	3.4	1.4	0.3	setosa
5.0	3.4	1.5	0.2	setosa
4.4	2.9	1.4	0.2	setosa
4.9	3.1	1.5	0.1	setosa

1-10 of 150 rows Previous **1** 2 3 4 5 6 ... 15 Next

4. Walk Through the R Tutorial - Part 2

Within this section of the tutorial, you will learn how to get to know your data, preprocess your data, and create training and test sets within RStudio.

Getting to Know Your Data

Once you've uploaded your dataset, it's always a good idea to get to know your data. Here are some helpful functions that will help you get acquainted with your data:

Hide

```
Get_to_know_data <- function(DatasetName){
  print(paste("GET TO KNOW: ",deparse(substitute(DatasetName))))

  print('fetching attributes...')
  print(attributes(DatasetName) )#List your attributes within your data set.

  print('fetching summary...')
  print(summary(DatasetName) )#Prints the min, max, mean, median, and quartiles of each attribute
.

  print('fetching data structure...')
  print(str(DatasetName) )#Displays the structure of your data set.

  print('fetching data attribute names...')
  print(names(DatasetName) )#Names your attributes within your data set.
  #DatasetName$ColumnName #Will print out the instances within that particular column in your dat
a set.
}
Get_to_know_data(df_cars)
```

```
[1] "GET TO KNOW:  df_cars"
[1] "fetching attributes..."
$names
[1] "name of car"     "speed of car"    "distance of car"

$class
[1] "spec_tbl_df" "tbl_df"      "tbl"         "data.frame"

$row.names
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29
[30] 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50

$spec
cols(
  `name of car` = [31mcol_character()[39m,
  `speed of car` = [32mcol_double()[39m,
  `distance of car` = [32mcol_double()[39m
)

[1] "fetching summary..."
 name of car         speed of car  distance of car
 Length:50          Min.   : 4.0   Min.   :  2.00
 Class :character   1st Qu.:12.0   1st Qu.: 26.00
 Mode  :character   Median :15.0   Median : 36.00
                    Mean   :15.4   Mean   : 42.98
                    3rd Qu.:19.0   3rd Qu.: 56.00
                    Max.   :25.0   Max.   :120.00
[1] "fetching data structure..."
Classes ‘spec_tbl_df’, ‘tbl_df’, ‘tbl’ and 'data.frame':    50 obs. of  3 variables:
 $ name of car    : chr  "Ford" "Jeep" "Honda" "KIA" ...
 $ speed of car   : num  4 4 7 7 8 9 10 10 10 11 ...
 $ distance of car: num  2 4 10 10 14 16 17 18 20 20 ...
 - attr(*, "spec")=
  .. cols(
  ..   `name of car` = [31mcol_character()[39m,
  ..   `speed of car` = [32mcol_double()[39m,
  ..   `distance of car` = [32mcol_double()[39m
  .. )
NULL
[1] "fetching data attribute names..."
[1] "name of car"     "speed of car"    "distance of car"
```

Hide

```
Get_to_know_data(df_iris)
```

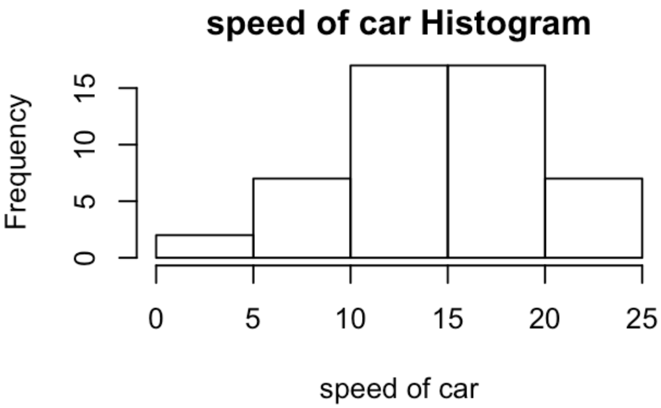
```
[1] "GET TO KNOW:  df_iris"
[1] "fetching attributes..."
$names
[1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Species"

$class
[1] "tbl_df"     "tbl"        "data.frame"

$row.names
  [1]   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21
 [22]  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42
 [43]  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63
 [64]  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84
 [85]  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100 101 102 103 104 105
[106] 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126
[127] 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147
[148] 148 149 150

[1] "fetching summary..."
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width      Species
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   Length:150
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300   Class :character
 Median :5.800   Median :3.000   Median :4.350   Median :1.300   Mode  :character
 Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
 3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
 Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
[1] "fetching data structure..."
Classes ‘tbl_df’, ‘tbl’ and 'data.frame':   150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : chr  "setosa" "setosa" "setosa" "setosa" ...
NULL
[1] "fetching data attribute names..."
[1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Species"
```

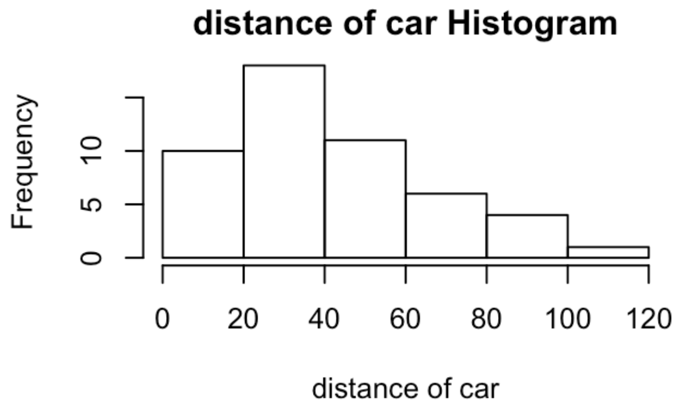
Plotting is also a helpful way to view your dataset. Here are some helpful functions that will help you get acquainted with your data visually:

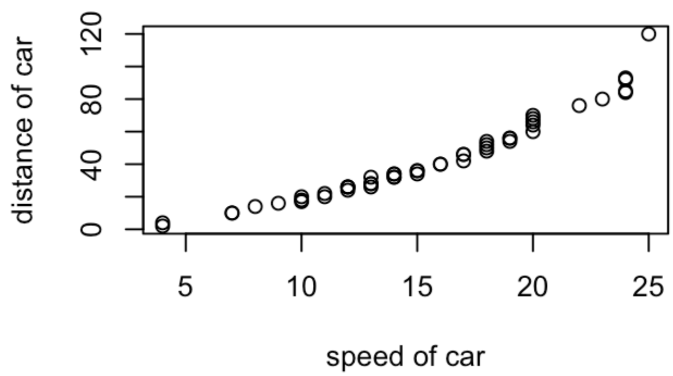
TIP: Your columns must be in numeric form to perform these plots.

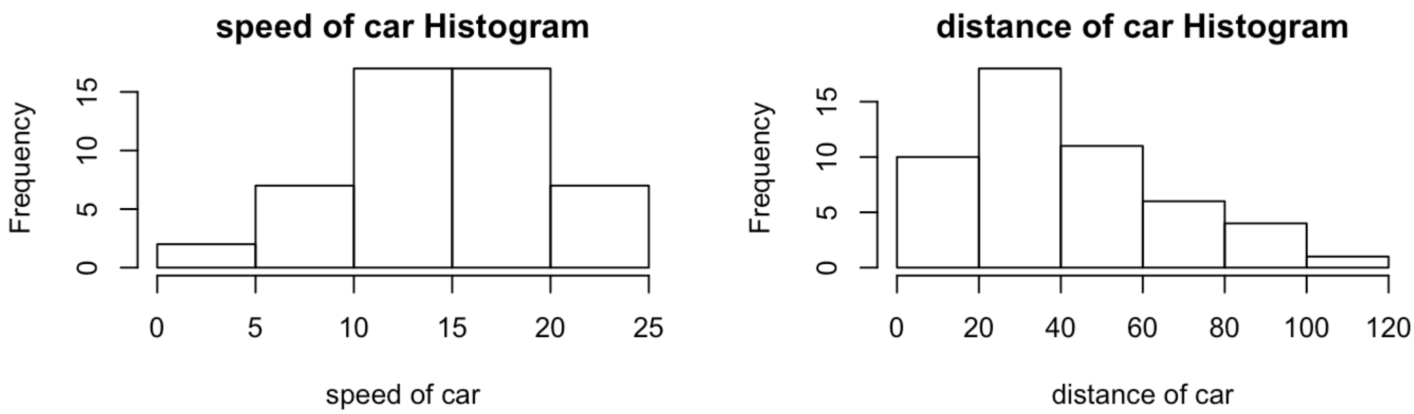
Hide

```
plot_summary_of_data<-function(DatasetName,x_index=1){

  column_names = names(DatasetName)

  subplot_cols = 2
  subplot_rows = 2
  par(mfrow=c(subplot_rows,subplot_cols))

  x <- unlist(DatasetName[,x_index])
  x_header = column_names[x_index]

  for(i in 1:length(column_names)){
    y <- unlist(DatasetName[,i])
    y_header = column_names[i]
    try(hist(y, main = paste(y_header, "Histogram"), xlab = y_header ),silent=TRUE)#Histogram Plo
t
  }

  for(i in 1:length(column_names)){

    if(i != x_index) {
    y <- unlist(DatasetName[,i])
    y_header = column_names[i]

    try(plot(x,y, xlab = x_header, ylab = y_header),silent=TRUE)  #Scatter (Box) Plot
    }
  }

  #Normal Quantile Plot- is a way to see if your data is normally distributed.
  for(i in 1:length(column_names)){

    y <- unlist(DatasetName[,i])
    y_header = column_names[i]

 try(qqnorm(y,main = paste(y_header, " Normal Q-Q Plot")),silent=TRUE) ##Normal Quantile Plot
  }
}
plot_summary_of_data(df_cars,x_index=2)
```


speed of car Histogram
Frequency
0
5
10
15
0
5
10
15
20
25
speed of car

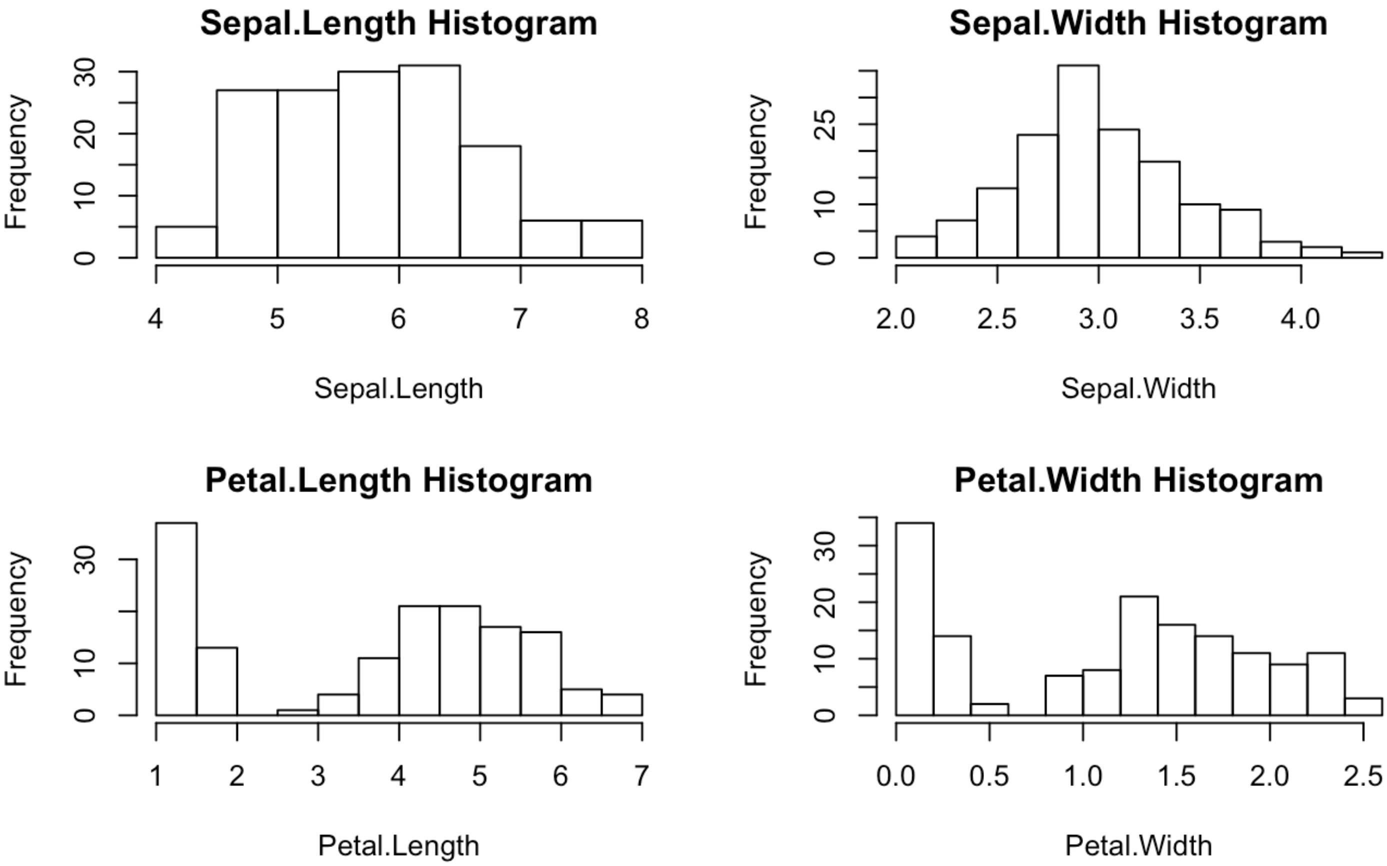

distance of car Histogram
Frequency
0
5
10
0
20
40
60
80
100
120
distance of car

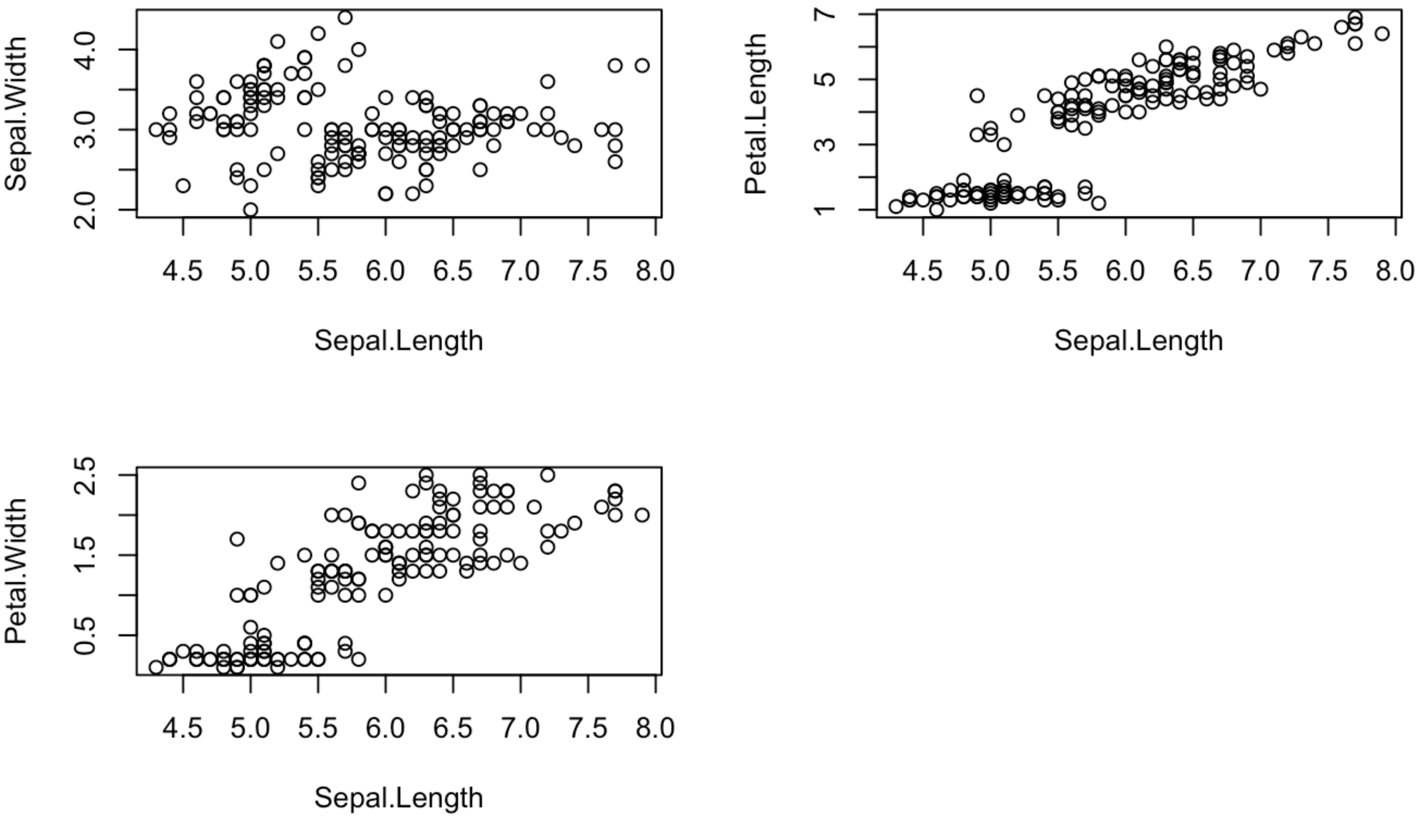

distance of car
0
40
80
120
5
10
15
20
25
speed of car

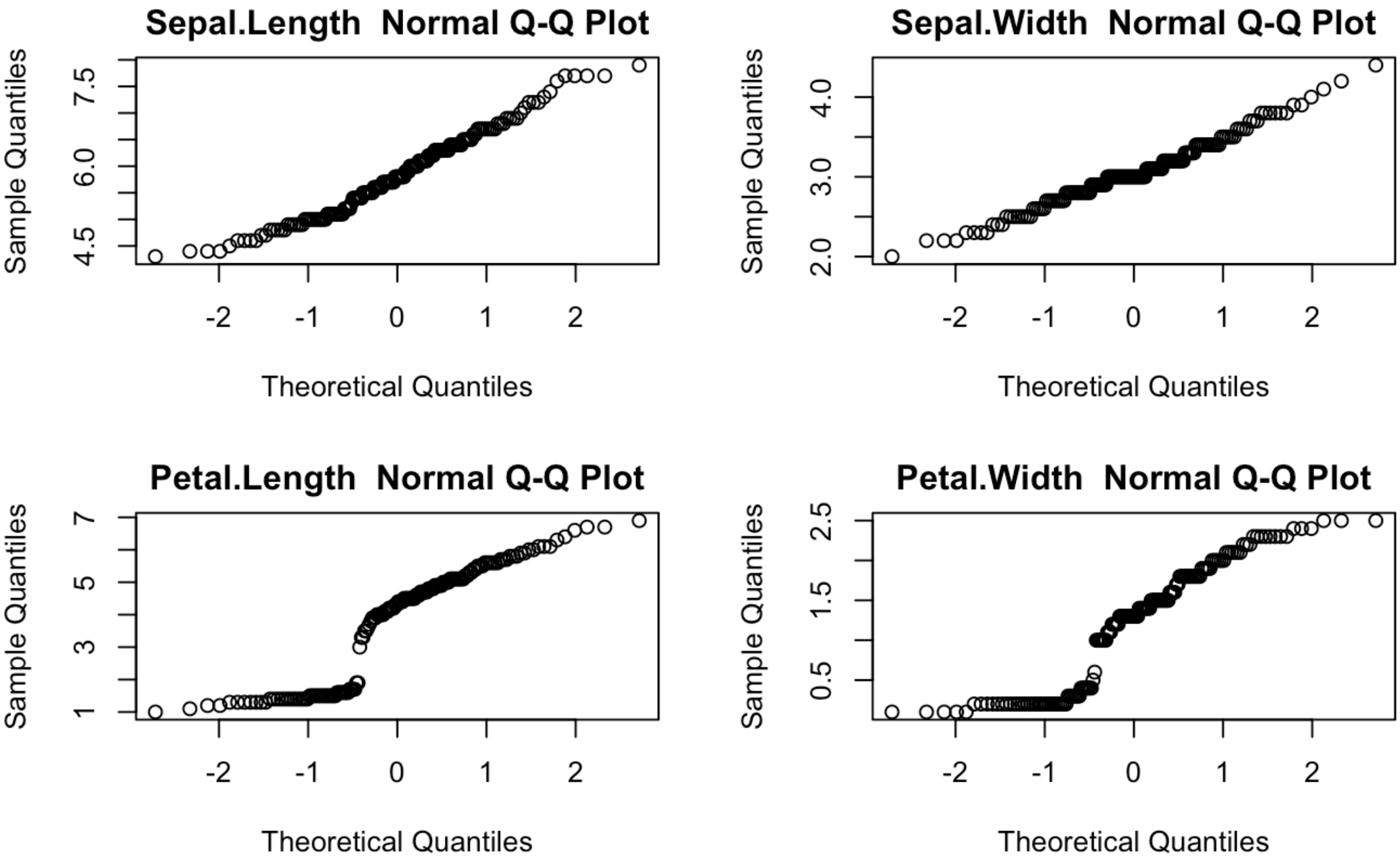


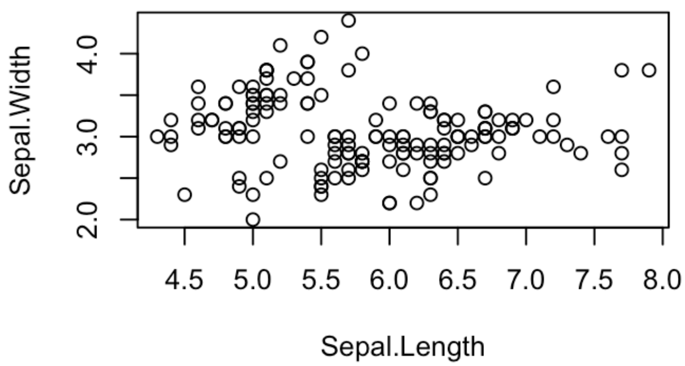
Hide

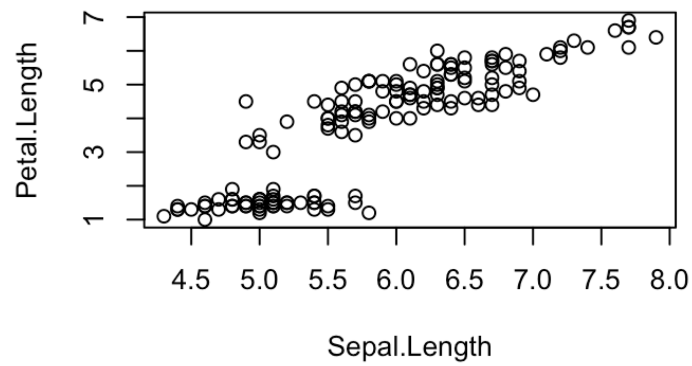
```
plot_summary_of_data(df_iris,x_index=1)
```

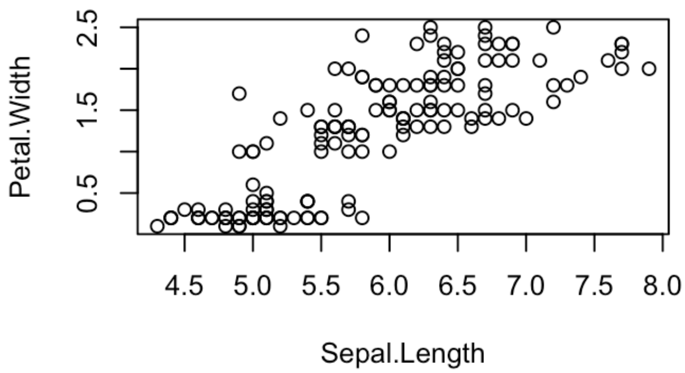

Sepal.Length Histogram
Frequency
Sepal.Length
Sepal.Width Histogram
Frequency
Sepal.Width
Petal.Length Histogram
Frequency
Petal.Length
Petal.Width Histogram
Frequency
Petal.Width


Sepal.Width
Sepal.Length
Petal.Length
Sepal.Length
Petal.Width
Sepal.Length


Sepal.Length Normal Q-Q Plot
Sample Quantiles
4.5
6.0
7.5
-2
-1
0
1
2
Theoretical Quantiles
Sepal.Width Normal Q-Q Plot
Sample Quantiles
2.0
3.0
4.0
-2
-1
0
1
2
Theoretical Quantiles
Petal.Length Normal Q-Q Plot
Sample Quantiles
1
3
5
7
-2
-1
0
1
2
Theoretical Quantiles
Petal.Width Normal Q-Q Plot
Sample Quantiles
0.5
1.5
2.5
-2
-1
0
1
2
Theoretical Quantiles







Preprocessing your Data

Preprocessing, also known as data cleaning, is a vital step in your analysis process. Some reasons to prepare your data is so it can be analyzed, it might be noisy data (missing values/outliers), it could have attributes that aren't helpful, etc. There are many steps that one may take when preparing your data, we will only be discussing a handful at a high-level view.

When working with data, you will need to be aware of what a vector is. A vector is a sequence of the same data type.

Here are the data types that you will encounter when working in R:

- Numeric- Numbers with decimals. (Ex: 1.0, 10.5, 4.5, etc.)
- Integer Data- Whole numbers (Ex: 11, 45, 78, etc.)
- Factor Data- Categorical data (Ex: Red, Blue, Green, Purple, etc.)
- Ordinal Data- Ordered data (Ex: educational levels, temperature, etc.)
- Character Data- String values, which are characters (words) with quotes around them. (Ex: "Red", "Blue", "Green", "Purple", etc.)
- Logical- TRUE or TRUE (Always capitalize TRUE or FALSE)

Do you see any data types that need changing within your data set? If so, how do you convert data types? Converting data types is a helpful skill to learn for this tutorial and future analyses. Here is an example of how one would change a column's data type within a data set:

Hide

```
#DatasetName$ColumnName<-as.typeofdata(DatasetName$ColumnName)
```

Do the columns/attributes within your dataset need renaming?

> the numeric data seems fine (all 'dbl'), however the character data should be changed to 'Factor' data as it is all categorical

Hide

```
df_cars$"name of car"<-as.factor(df_cars$"name of car")
df_iris$"Species"<-as.factor(df_iris$"Species")
```

To rename the attributes/columns in your dataset, you'll want to use the c() function, specifying a name for each column.

Hide

```
reformat_column_headers <- function(DatasetName){
  column_names = names(DatasetName)

  column_names_reformated = gsub(" ","_",fixed=TRUE,column_names)

  names(DatasetName)<-c(column_names_reformated)

  return(DatasetName)
}
df_cars = reformat_column_headers(df_cars)
print(names(df_cars))
```

```
[1] "name_of_car"     "speed_of_car"    "distance_of_car"
```

Hide

```
df_iris = reformat_column_headers(df_iris)
print(names(df_iris))
```

```
[1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Species"
```

Do any of your variables have missing values? How do you know if your dataset has any missing values? If you do not address missing values certain functions will not work properly, so it's smart to start the practice checking for missing values. R labels missing as NA (Not Available). Here are two ways to know if you have any missing values:

Hide

```
summary(df_cars) #Will count how many NA's you have.
```

```
  name_of_car  speed_of_car  distance_of_car
 Dodge  : 3   Min.   : 4.0   Min.   :  2.00  
 Honda  : 3   1st Qu.:12.0   1st Qu.: 26.00  
 Jeep   : 3   Median :15.0   Median : 36.00  
 KIA    : 3   Mean   :15.4   Mean   : 42.98  
 Acura  : 2   3rd Qu.:19.0   3rd Qu.: 56.00  
 Audi   : 2   Max.   :25.0   Max.   :120.00  
 (Other):34                                  
```

Hide

```
summary(df_iris)
```

```
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width          Species  
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   setosa    :50  
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300   versicolor:50  
 Median :5.800   Median :3.000   Median :4.350   Median :1.300   virginica :50  
 Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199                  
 3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800                  
 Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500                  
```

Hide

```
is.na(df_cars) #Will show your NA’s through logical data. (TRUE if it’s missing, FALSE if it’s no
t.)
```

```
      name_of_car speed_of_car distance_of_car
 [1,]       FALSE        FALSE           FALSE
 [2,]       FALSE        FALSE           FALSE
 [3,]       FALSE        FALSE           FALSE
 [4,]       FALSE        FALSE           FALSE
 [5,]       FALSE        FALSE           FALSE
 [6,]       FALSE        FALSE           FALSE
 [7,]       FALSE        FALSE           FALSE
 [8,]       FALSE        FALSE           FALSE
 [9,]       FALSE        FALSE           FALSE
[10,]       FALSE        FALSE           FALSE
[11,]       FALSE        FALSE           FALSE
[12,]       FALSE        FALSE           FALSE
[13,]       FALSE        FALSE           FALSE
[14,]       FALSE        FALSE           FALSE
[15,]       FALSE        FALSE           FALSE
[16,]       FALSE        FALSE           FALSE
[17,]       FALSE        FALSE           FALSE
[18,]       FALSE        FALSE           FALSE
[19,]       FALSE        FALSE           FALSE
[20,]       FALSE        FALSE           FALSE
[21,]       FALSE        FALSE           FALSE
[22,]       FALSE        FALSE           FALSE
[23,]       FALSE        FALSE           FALSE
[24,]       FALSE        FALSE           FALSE
[25,]       FALSE        FALSE           FALSE
[26,]       FALSE        FALSE           FALSE
[27,]       FALSE        FALSE           FALSE
[28,]       FALSE        FALSE           FALSE
[29,]       FALSE        FALSE           FALSE
[30,]       FALSE        FALSE           FALSE
[31,]       FALSE        FALSE           FALSE
[32,]       FALSE        FALSE           FALSE
[33,]       FALSE        FALSE           FALSE
[34,]       FALSE        FALSE           FALSE
[35,]       FALSE        FALSE           FALSE
[36,]       FALSE        FALSE           FALSE
[37,]       FALSE        FALSE           FALSE
[38,]       FALSE        FALSE           FALSE
[39,]       FALSE        FALSE           FALSE
[40,]       FALSE        FALSE           FALSE
[41,]       FALSE        FALSE           FALSE
[42,]       FALSE        FALSE           FALSE
[43,]       FALSE        FALSE           FALSE
[44,]       FALSE        FALSE           FALSE
[45,]       FALSE        FALSE           FALSE
[46,]       FALSE        FALSE           FALSE
[47,]       FALSE        FALSE           FALSE
[48,]       FALSE        FALSE           FALSE
[49,]       FALSE        FALSE           FALSE
[50,]       FALSE        FALSE           FALSE
```

How to address missing values? There are multiple ways to confront missing values in your dataset – all depend on how much they will affect your dataset. Here are a few options:

- Remove any observations containing missing data. (If the missing data is less than 10% of the total data and only after comparing the min/max of all the features both with and without the missing data.)

Hide

```
na.omit(df_cars)#Drops any rows with missing values and omits them forever.
```

name_of_car <fctr>	speed_of_car <dbl>	distance_of_car <dbl>
Ford	4	2
Jeep	4	4
Honda	7	10
KIA	7	10
Toyota	8	14
BMW	9	16
Mercedes	10	17
GM	10	18
Hyundai	10	20
Infiniti	11	20

1-10 of 50 rows Previous **1** 2 3 4 5 Next

Hide

```
na.omit(df_iris)
```

Sepal.Length <dbl>	Sepal.Width <dbl>	Petal.Length <dbl>	Petal.Width <dbl>	Species <fctr>
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa
4.6	3.4	1.4	0.3	setosa
5.0	3.4	1.5	0.2	setosa

4.4	2.9	1.4	0.2	setosa
4.9	3.1	1.5	0.1	setosa

1-10 of 150 rows Previous **1** 2 3 4 5 6 ... 15 Next

Hide

```
#na.exclude(DatasetName$ColumnName)#Drops any rows with missing values, but keeps track of where
they were.
```

Replace the missing values with the mean, which is common technique, but something to use with care with as it can skew the data.

Hide

```
#DatasetName$ColumnName[is.na(DatasetName$ColumnName)]<-mean(DatasetName$ColumnName,na.rm = TRUE)
```

Creating Testing and Training Sets

Once you've preprocessed your dataset, it's now time to create training and testing sets for the linear regression model you will be creating.

How to begin? In order to create your training and testing sets, you need to use the set.seed() function. The seed is a number that you choose for a starting point used to create a sequence of random numbers. It is also helpful for others who want to recreate your same results. Here is the function:

TIP: A common set.seed number is 123. To use the same set of random numbers, you'll want to use the same seed number throughout your modeling process.

Hide

```
set.seed(1)
```

How do you split the data into training and test sets? You'll now want to split your data into two sets for modeling. One is the training set and the other one being the test set. A common split is 70/30, which means that 70% of the data will be the training set's size and 30% of the data will be the test set's size. You will be using the 70/30 split, but another common split is 80/20.

Setting the training set's size and the testing set's size can be done by performing these two lines of code. These two lines calculate the sizes of each set but do not create the sets:

Hide

```
train_test_size<-function(DatasetName,training_size_ratio=0.7){
  trainSize<-round(nrow(DatasetName)*training_size_ratio)
  testSize<-nrow(DatasetName)-trainSize
  df_out = data.frame('trainSize'=c(trainSize),'testSize'=c(testSize))
  return(df_out)
}
train_test_size(df_cars,0.7)
```

trainSize	testSize
<dbl>	<dbl>

35	15

1 row

Hide

```
train_test_size(df_iris,0.7)
```

trainSize <dbl>	testSize <dbl>
105	45

1 row

How do you create the training and test sets? It's now time for you to create the training and test sets. We also want these sets to be in a randomized order, which will create the most optimal model.

To perform this, you need to run these lines of code. Type in this code into R Script or Console:

Hide

```
train_test_split<-function(DatasetName,training_size_ratio=0.7){

  df_train_test_size <- train_test_size(df_cars,training_size_ratio)  #get train test sizes
  trainSize<-df_train_test_size[1,1]

  training_indices<-sample(seq_len(nrow(DatasetName)),size = trainSize)

  trainSet<-DatasetName[training_indices,]

  testSet<-DatasetName[-training_indices,]

  return(list(trainSet,testSet))
}
list_df_train_test_sets = train_test_split(df_cars,0.7)
df_trainSet = list_df_train_test_sets[[1]]
print(df_trainSet)
```

name_of_car <fctr>	speed_of_car <dbl>	distance_of_car <dbl>
Mitsubishi	12	26
Dodge	13	32
Toyota	16	40
Acura	20	70
Infiniti	11	20
Chrysler	20	66
Dodge	20	68

BMW	17	42
KIA	16	40
Honda	7	10

1-10 of 35 rows Previous **1** 2 3 4 Next

Hide

```
df_testSet = list_df_train_test_sets[[2]]
print(df_testSet)
```

name_of_car <fctr>	**speed_of_car** <dbl>	**distance_of_car** <dbl>
Jeep	4	4
Toyota	8	14
BMW	9	16
Land Rover	11	22
GMC	13	26
Fiat	13	28
Audi	14	32
Chevrolet	14	34
Ford	15	34
Honda	15	36

1-10 of 15 rows Previous **1** 2 Next

5. Walk Through the R Tutorial - Part 3

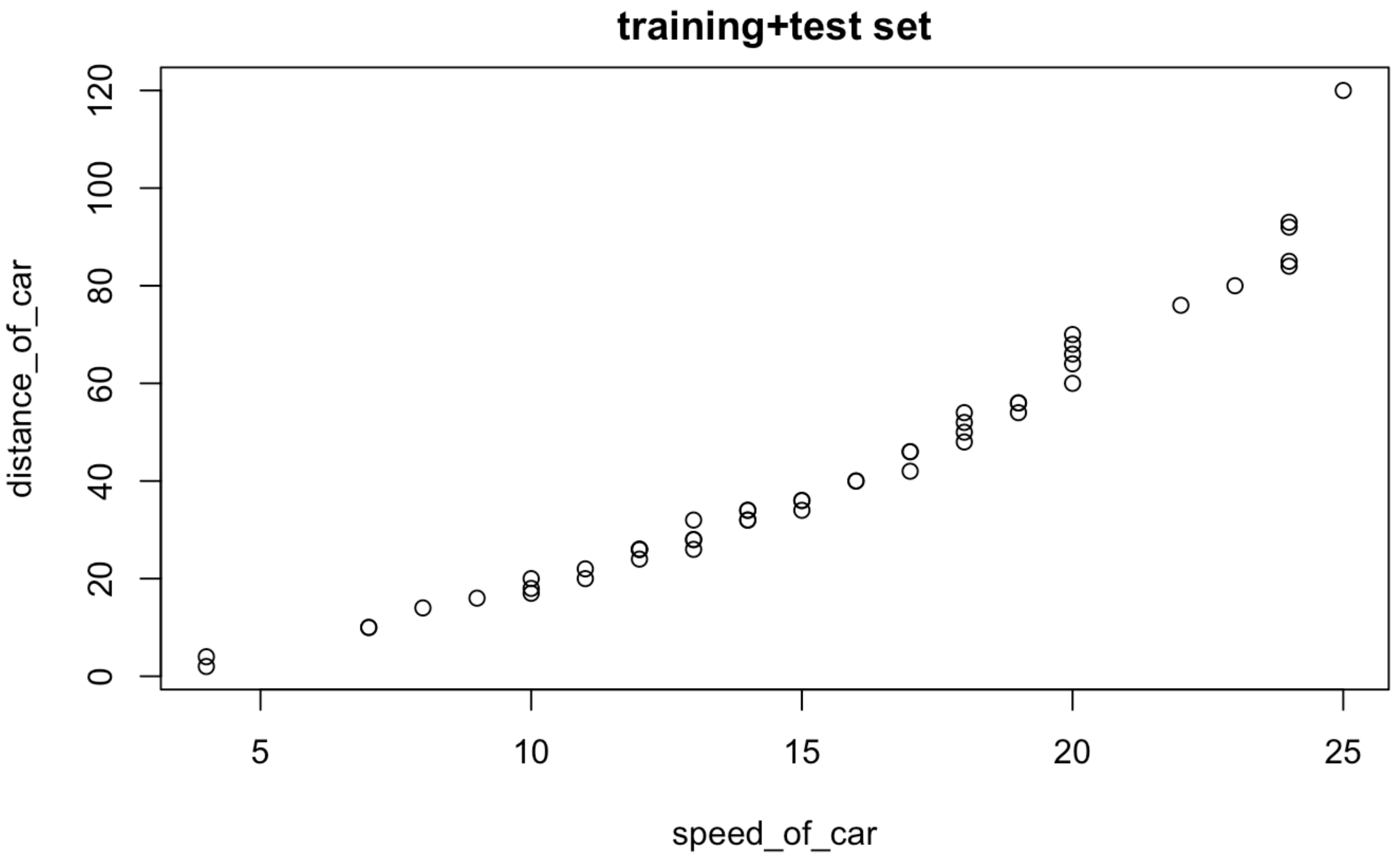
Within this section of the tutorial, you will learn how to create a linear regression model, understand its output and use the prediction function.

Linear Regression Model

You're now ready to run your data through your modeling algorithm. The model that we will be using is the Linear Regression Model, which is helpful when trying to discover the relationship between two variables. These two variables represent the X and Y within the linear equation. The X variable is the predictor variable, also known as the independent variable because it doesn't depend on other attributes while making predictions. Y is the response variable, also known as the dependent variable because its value depends on the other variables. (We will be keeping this at a high level. If you'd like to discover more about this equation, please feel free to do your own research.) In our case, these two variables will be Speed and Distance. We are trying to predict Distance, so it is our dependent/response/Y variable. Speed is our independent/predictor/X variable.

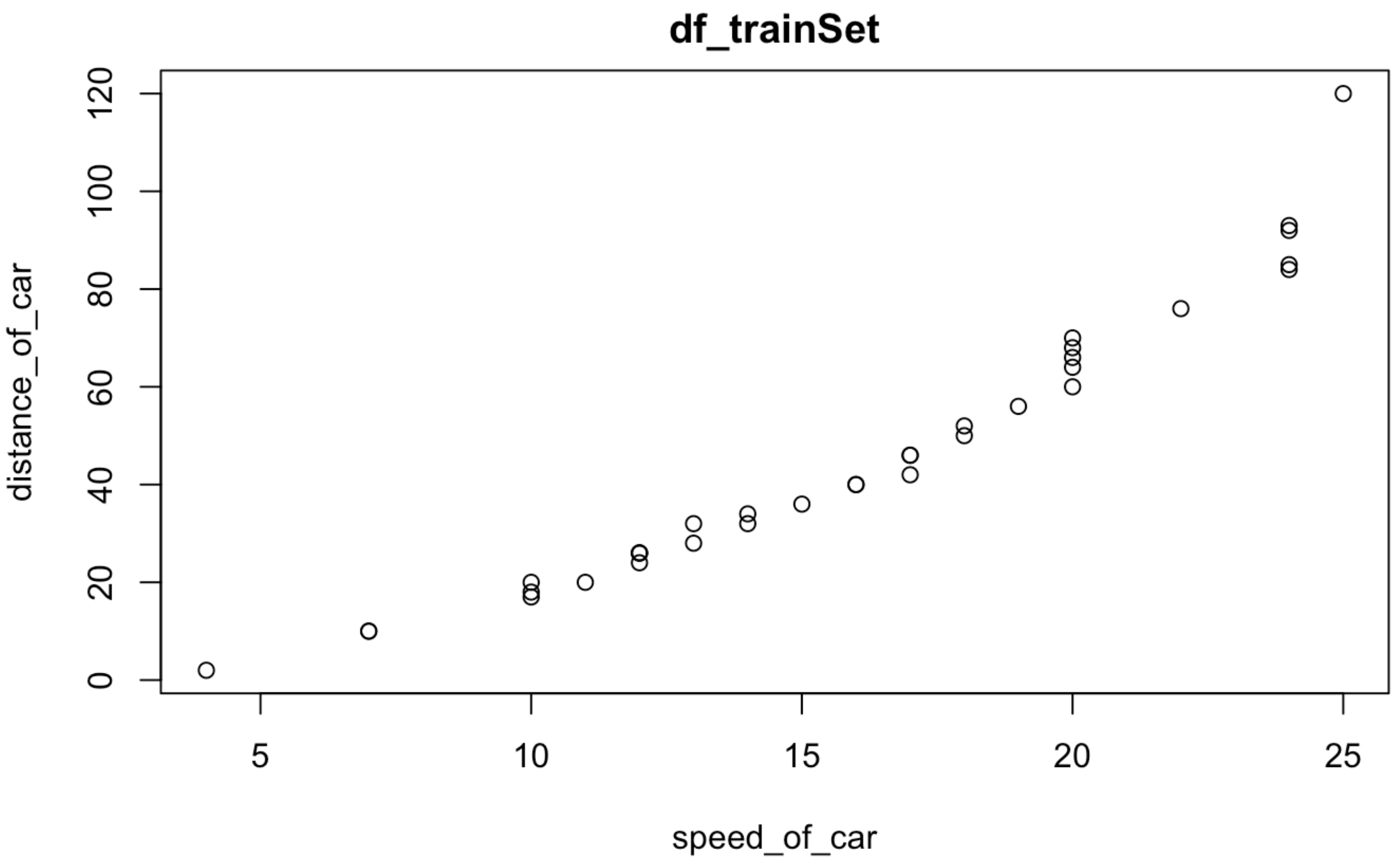
Hide

```
plot(df_cars$'speed_of_car',df_cars$'distance_of_car',xlab = 'speed_of_car', ylab = 'distance_of_
car',main='training+test set')
```



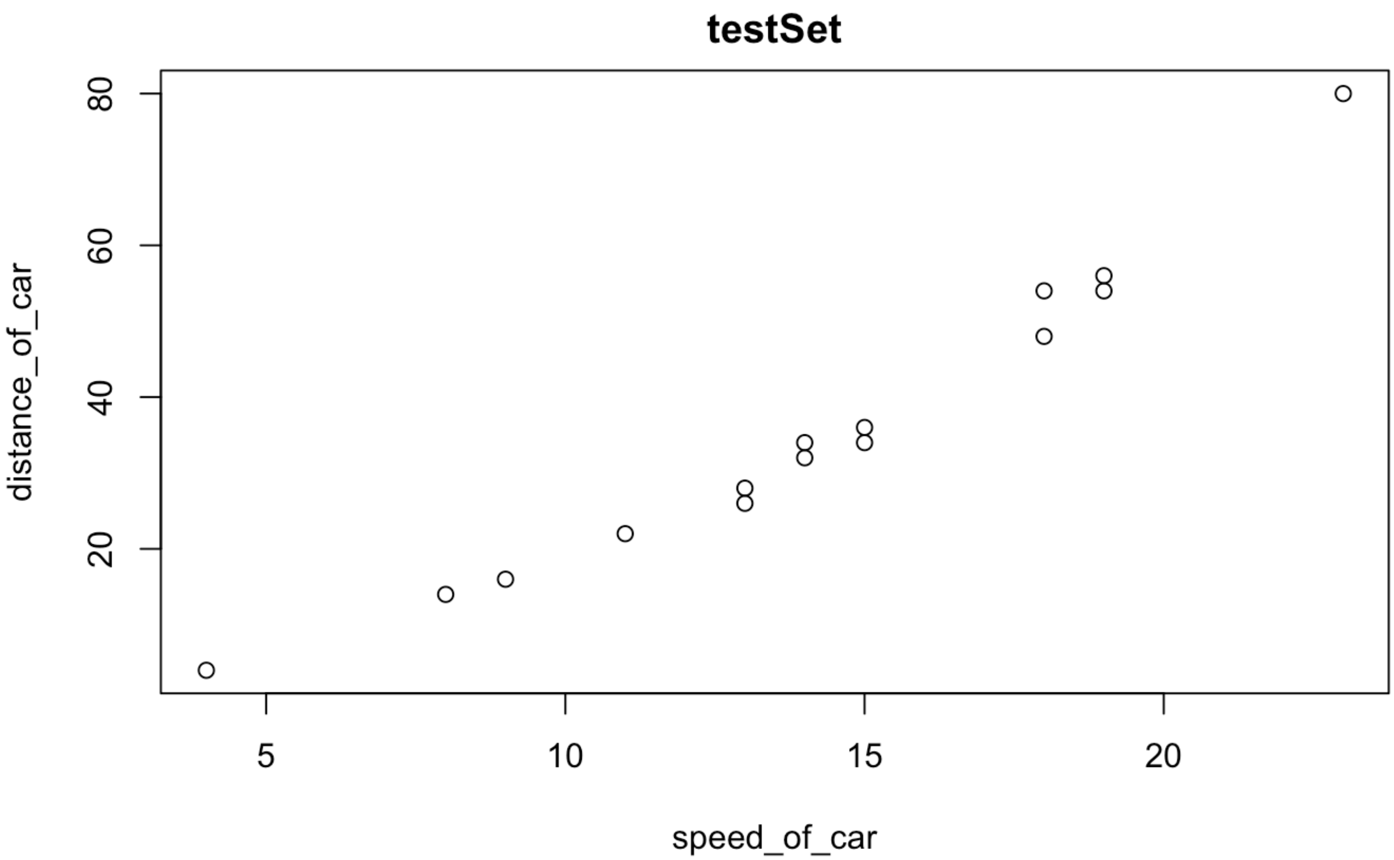
Hide

```
plot(df_trainSet$'speed_of_car',df_trainSet$'distance_of_car',xlab = 'speed_of_car', ylab = 'dist
ance_of_car',main='df_trainSet')
```



Hide

```
plot(df_testSet$'speed_of_car',df_testSet$'distance_of_car',xlab = 'speed_of_car', ylab = 'distan
ce_of_car',main='testSet')
```



To create this model, we will be using the linear model function – lm(). Here is the basic line of code for the linear model function.

Hide

```
lm_cars_speed_vs_distance<-lm(distance_of_car ~ speed_of_car, df_trainSet)
```

Did you create an optimal model? To see key metrics of your model, type in this code into R Script or Console

Hide

```
summary(lm_cars_speed_vs_distance)
```

```
Call:
lm(formula = distance_of_car ~ speed_of_car, data = df_trainSet)

Residuals:
    Min      1Q  Median      3Q     Max 
-9.3709 -4.9106 -0.8151  1.4703 29.2644 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept)  -32.2791     3.8682  -8.345 1.22e-09 ***
speed_of_car   4.9206     0.2304  21.358  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 7.295 on 33 degrees of freedom
Multiple R-squared:  0.9325,	Adjusted R-squared:  0.9305 
F-statistic: 456.2 on 1 and 33 DF,  p-value: < 2.2e-16
```

The two metrics that we will be discussing are:

- Multiple R-squared- How well the regression line fits the data (1 means it’s a perfect fit).
- p-value - Tells you how much the Independent Variable/Predictor affects the Dependent Variable/Response/. A p-value of more than 0.05 means the Independent Variable has no effect on the Dependent Variable; less than 0.05 means the relationship is statistically significant.
- If you’d like to learn more about Adjusted R-squared and the F-statistic, see the Resources tab. For now, you will be leaving this model as is. You will learn how to adjust your model’s parameters to create the most optimal model in later courses.

Predictions

The next step is to predict the cars distances through the speed of the cars. To do this, we’ll be using the prediction function – predict()

Hide

```
predict_lm_cars_vs_distance <- predict(lm_cars_speed_vs_distance,df_testSet)
print("predictions:")
```

```
[1] "predictions:"
```

Hide

```
predict_lm_cars_vs_distance
```

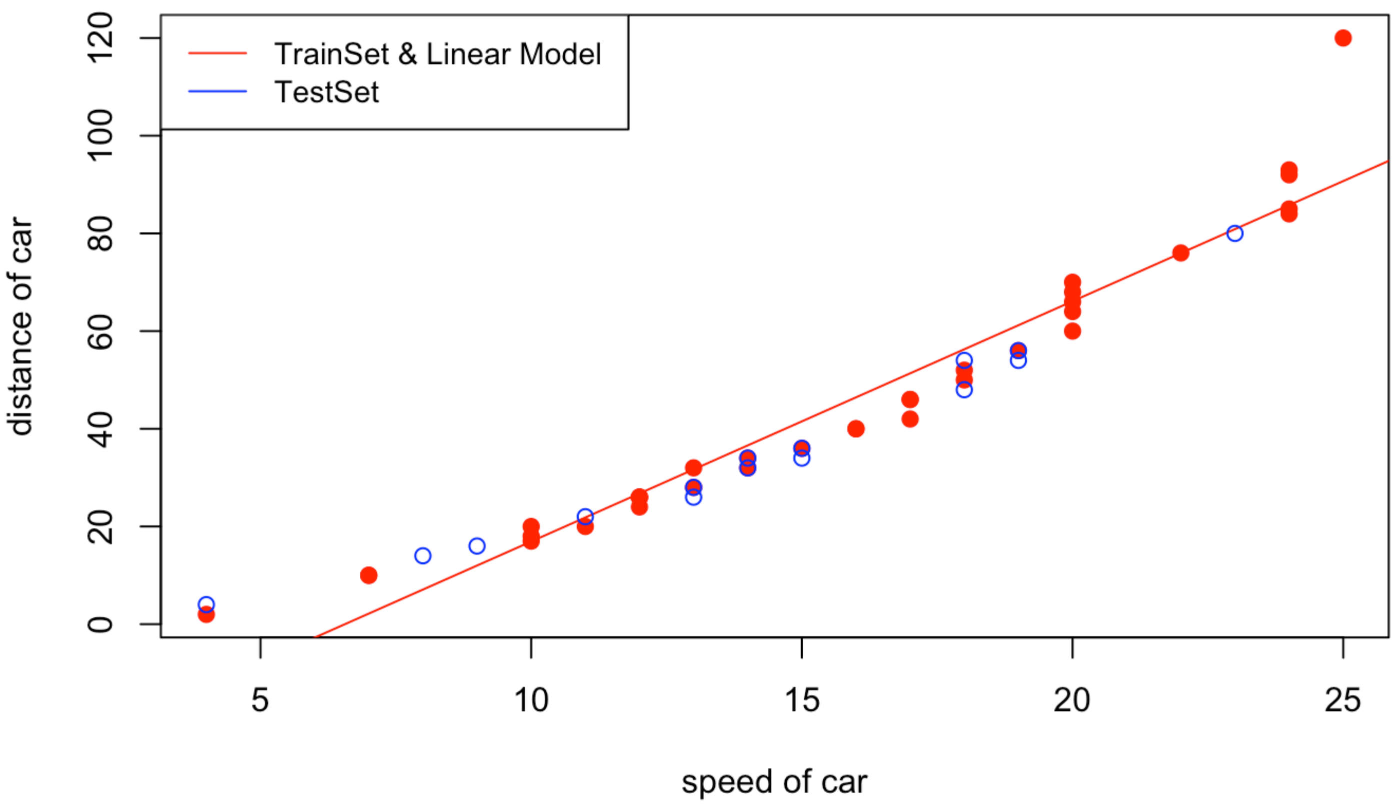
```
         1          2          3          4          5          6          7          8 
-12.596735   7.085622  12.006212  21.847390  31.688568  31.688568  36.609158  36.609158 
         9         10         11         12         13         14         15 
 41.529747  41.529747  56.291515  56.291515  61.212104  61.212104  80.894461 
```

Hide

```
plot(df_trainSet$speed_of_car, df_trainSet$distance_of_car, xlab = 'speed of car', ylab = 'distan
ce of car', col='red',pch = 19)
abline(lm(distance_of_car ~ speed_of_car, df_trainSet),col='red')
```

Hide

```
points(df_testSet$speed_of_car, df_testSet$distance_of_car,col='blue')
legend("topleft", legend=c("TrainSet & Linear Model","TestSet"),
       col=c("red","blue"),lty=1, cex=.9)
```



TIP: Make a note of your predictions. (You will need to include them in your informal report to Blackwell.)

Find the Errors in an R Scripts

Congrats, you've now completed a Regression Analysis in R!

Blackwell believes that reading and deciphering code is just as important as coding itself. Therefore, it's now time to test your new-found knowledge of R by running a script of code. Beware that you will be encountering errors and warnings, but you have the power to overcome these!

What does an error look like? An error message will be in red and will prevent you from going further due to a mistake in the previous line or lines of code.

What does a warning message look like? A warning message will also be in red. You will still be able to continue, but it might or might not affect your analysis in the long run.

You've encountered an error/warning message, what now?

Review your tutorial. Check your spelling/spacing. Research the error/warning message — Has anyone else encountered this error? TIP: REMEMBER:

Don't forget your analysis goal! If you are typing in the Console, press ENTER to see run your code. If you are typing in the R Script, press COMMAND + ENTER to see run your code. The dataset that is discussed in the code concerns the Iris flower, which is within the zip file attached to Danielle's email. The analysis goal is to predict a petal's length using the petal's width.

Here is the script that you will run (and fix any errors it contains):

Hide

```
install.packages(readr) # Original Code
```

```
Error in install.packages : object 'readr' not found
```

Hide

```
install.packages('readr') # New Code | Note: make sure you use quotes when calling a package
```

Hide

```
#library(“readr”) # Original Code
#Error: unexpected input in "library(�"
library(readr) # New Code | Note: reference by unquoted name when reading library package.
```

Hide

```
#IrisDataset <- read.csv(iris.csv) # Original Code
#Error in read.table(file = file, header = header, sep = sep, quote = quote, : object 'iris.csv'
not found
IrisDataset <- read.csv("R Tutorial Data Sets/iris.csv") # New Code | Note: point to directory lo
cation using quotes
```

Hide

```
attributes(IrisDataset) #Original code
```

```
$names
[1] "X"            "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"
[6] "Species"

$class
[1] "data.frame"

$row.names
  [1]   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21
 [22]  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42
 [43]  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63
 [64]  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84
 [85]  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100 101 102 103 104 105
[106] 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126
[127] 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147
[148] 148 149 150
```

Hide

```
#summary(risDataset) #Original code
#Error in summary(risDataset) : object 'risDataset' not found
summary(IrisDataset) # new code | note: missing "I" at start of "iris..."
```

```
       X           Sepal.Length    Sepal.Width     Petal.Length    Petal.Width   
 Min.   :  1.00   Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100  
 1st Qu.: 38.25   1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300  
 Median : 75.50   Median :5.800   Median :3.000   Median :4.350   Median :1.300  
 Mean   : 75.50   Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199  
 3rd Qu.:112.75   3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800  
 Max.   :150.00   Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500  
       Species  
 setosa    :50  
 versicolor:50  
 virginica :50  
```

Hide

```
# str(IrisDatasets) # Original code
# Error in str(IrisDatasets) : object 'IrisDatasets' not found
str(IrisDataset) # new code | typo
```

```
'data.frame':	150 obs. of  6 variables:
 $ X           : int  1 2 3 4 5 6 7 8 9 10 ...
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

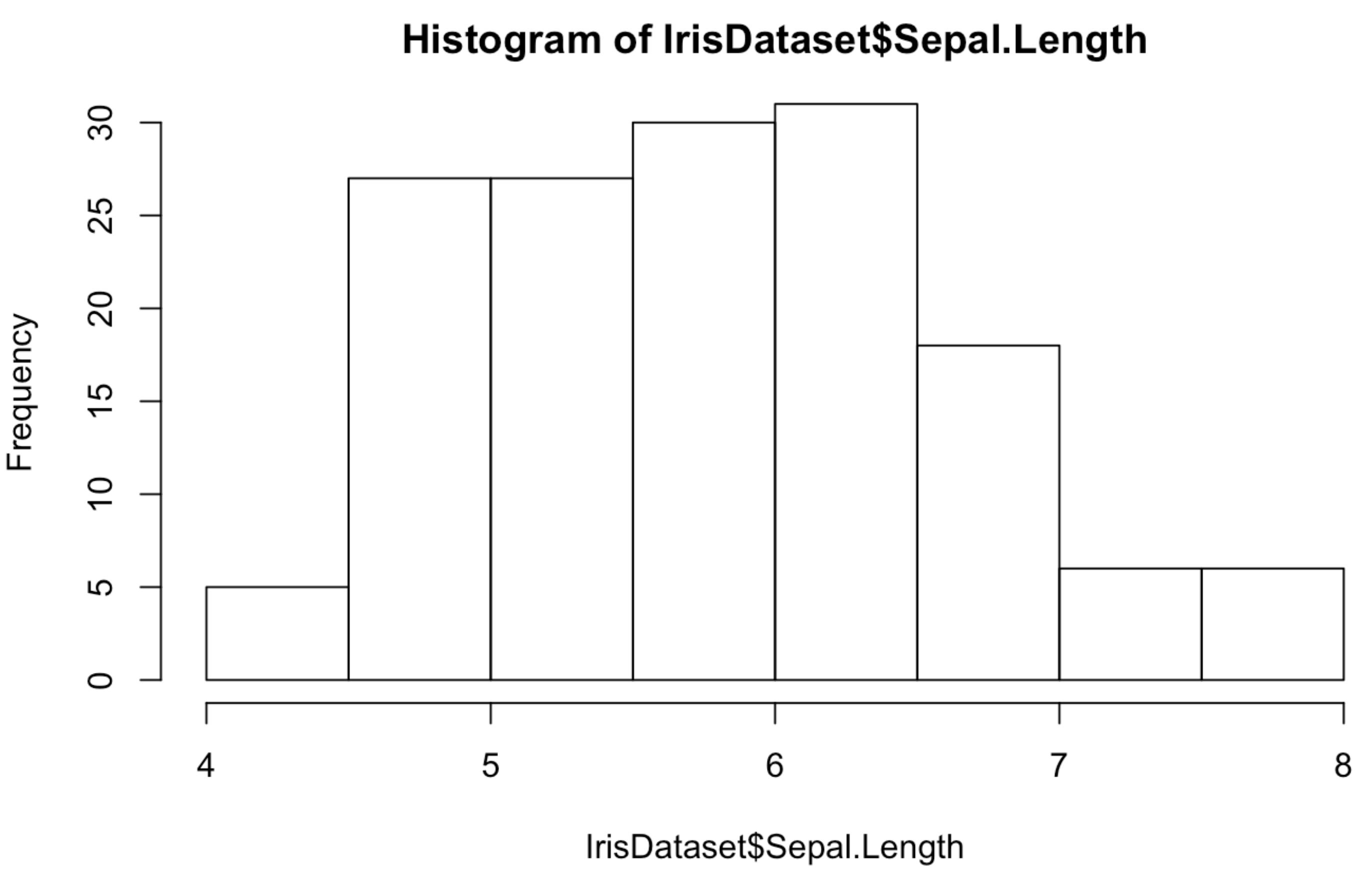
Hide

```
names(IrisDataset) #original code
```

```
[1] "X"            "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"
[6] "Species"
```

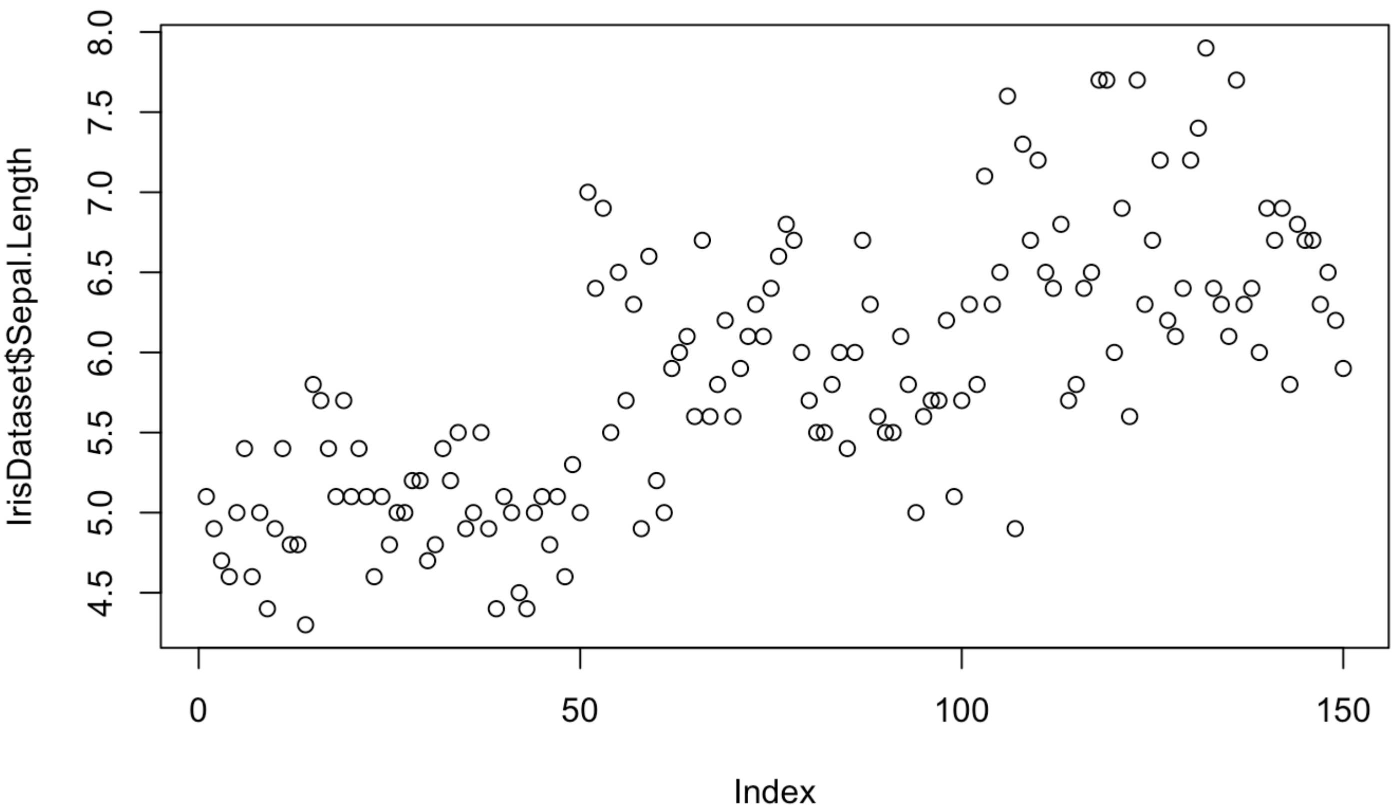
Hide

```
# hist(IrisDataset$Species) #original code
# Error in hist.default(IrisDataset$Species) : 'x' must be numeric
hist(IrisDataset$Sepal.Length) #new code | histogram can only accept numeric columns
```



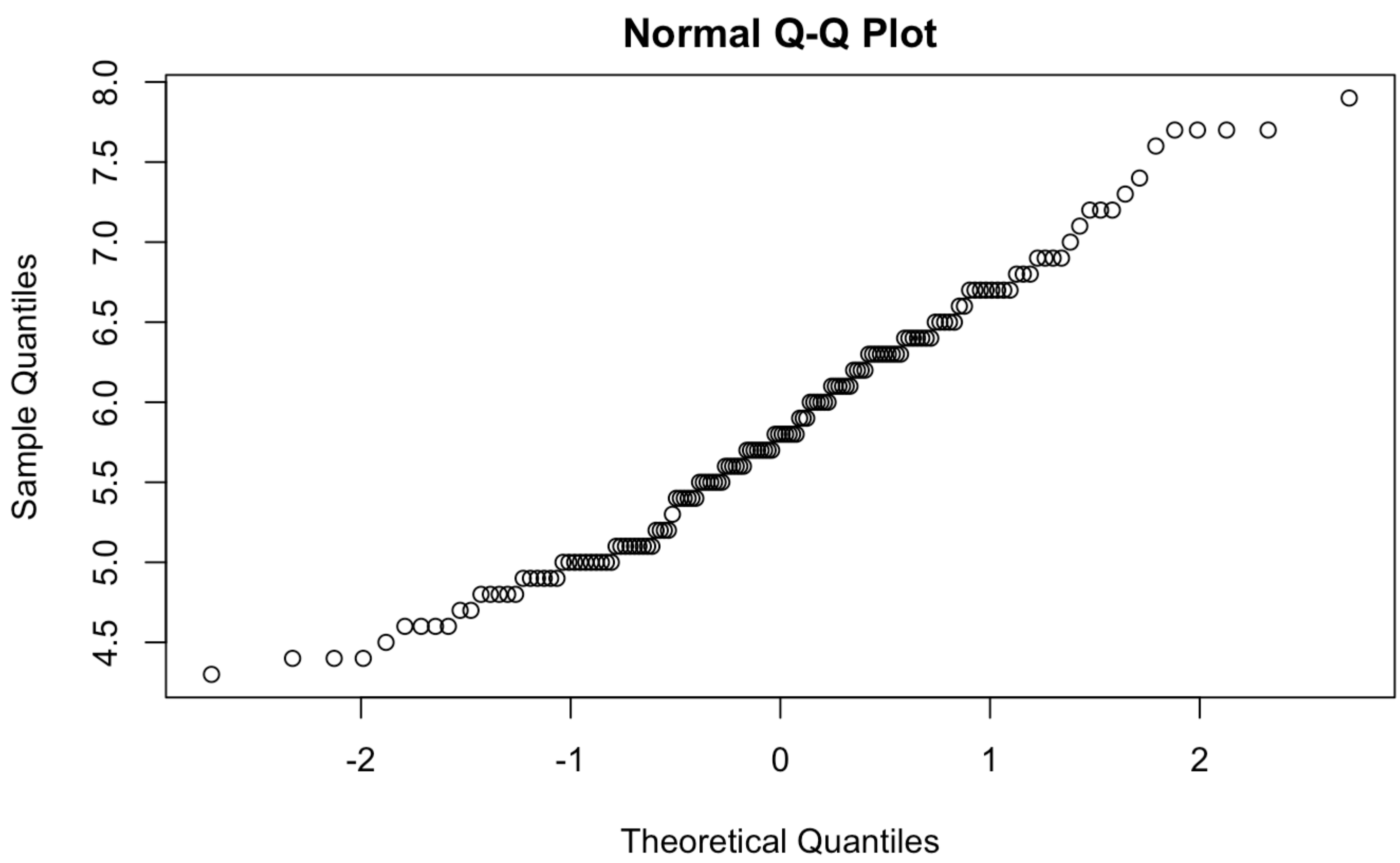
Hide

```
# plot(IrisDataset$Sepal.Length #original code
# Error: Incomplete expression: plot(IrisDataset$Sepal.Length #original code
plot(IrisDataset$Sepal.Length) #new code | missing end bracket
```



Hide

```
#qqnorm(IrisDataset) #original code
#Error in FUN(X[[i]], ...) : only defined on a data frame with all numeric variables
qqnorm(IrisDataset$Sepal.Length) #original code | must select a specific numeric column for Q-Q p
lot
```



Hide

```
IrisDataset$Species<- as.numeric(IrisDataset$Species)  # original code
```

Hide

```
set.seed(123) # original code
```

Hide

```
trainSize <- round(nrow(IrisDataset) * 0.2)# original code
trainSize
```

```
[1] 30
```

Hide

```
testSize <- nrow(IrisDataset) - trainSet # original code
testSize
```

	X <int>	Sepal.Length <dbl>	Sepal.Width <dbl>	Petal.Length <dbl>	Petal.Width <dbl>	Species <dbl>
128	22	143.9	147.0	145.1	148.2	147

83	67	144.2	147.3	146.1	148.8	148
58	92	145.1	147.6	146.7	149.0	148
10	140	145.1	146.9	148.5	149.9	149
53	97	143.1	146.9	145.1	148.5	148
30	120	145.3	146.8	148.4	149.8	149
11	139	144.6	146.3	148.5	149.8	149
65	85	144.4	147.1	146.4	148.7	148
148	2	143.5	147.0	144.8	148.0	147
43	107	145.6	146.8	148.7	149.8	149

1-10 of 30 rows Previous **1** 2 3 Next

Hide

```
testSize <- round(nrow(IrisDataset)) - trainSize # new code | replace trainSet with trainSize
testSize
```

```
[1] 120
```

Hide

```
# trainSizes # orgiinal code
# Error: object 'trainSizes' not found
trainSize # new code | drop "s" from original code
```

```
[1] 30
```

Hide

```
testSize #original code
```

```
[1] 120
```

Hide

```
trainSet <- IrisDataset[training_indices, ] #original code
```

Hide

```
#new code
training_indices<-sample(seq_len(nrow(IrisDataset)),size = trainSize) #get indices
trainSet<-IrisDataset[training_indices,]
print(trainSet)
```

	X	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
	<int>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
44	44	5.0	3.5	1.6	0.6	1
118	118	7.7	3.8	6.7	2.2	3
61	61	5.0	2.0	3.5	1.0	2
130	130	7.2	3.0	5.8	1.6	3
138	138	6.4	3.1	5.5	1.8	3
7	7	4.6	3.4	1.4	0.3	1
77	77	6.8	2.8	4.8	1.4	2
128	128	6.1	3.0	4.9	1.8	3
79	79	6.0	2.9	4.5	1.5	2
65	65	5.6	2.9	3.6	1.3	2

1-10 of 30 rows Previous **1** 2 3 Next

Hide

```
testSet <- IrisDataset[-training_indices, ]#original code
```

Hide

```
set.seed(405)#original code
```

Hide

```
trainSet <- IrisDataset[training_indices, ] #original code (repreated previously)
testSet <- IrisDataset[-training_indices, ] #original code (repreated previously)
```

Hide

```
# LinearModel<- lm(trainSet$Petal.Width ~ testingSet$Petal.Length) #original code
# Error in eval(predvars, data, env) : object 'testingSet' not found
LinearModel<- lm(Petal.Width ~ Petal.Length, trainSet) #new code | both data sets must come from
same data frame
```

Hide

```
summary(LinearModel) # original code
```

```
Call:
lm(formula = Petal.Width ~ Petal.Length, data = trainSet)

Residuals:
     Min       1Q   Median       3Q      Max 
-0.36434 -0.15638 -0.01071  0.08594  0.60354 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept)   -0.2552     0.1165  -2.191   0.0369 *  
Petal.Length   0.3827     0.0275  13.917 4.18e-14 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.2368 on 28 degrees of freedom
Multiple R-squared:  0.8737,	Adjusted R-squared:  0.8692 
F-statistic: 193.7 on 1 and 28 DF,  p-value: 4.185e-14
```

Hide

```
# prediction<-predict(LinearModeltestSet) #orignal code
# Error in predict(LinearModeltestSet) : object 'LinearModeltestSet' not found
prediction<-predict(LinearModel,testSet) #new code
```

Hide

```
# predictions #original code
# Error: object 'predictions' not found
prediction #new code
```

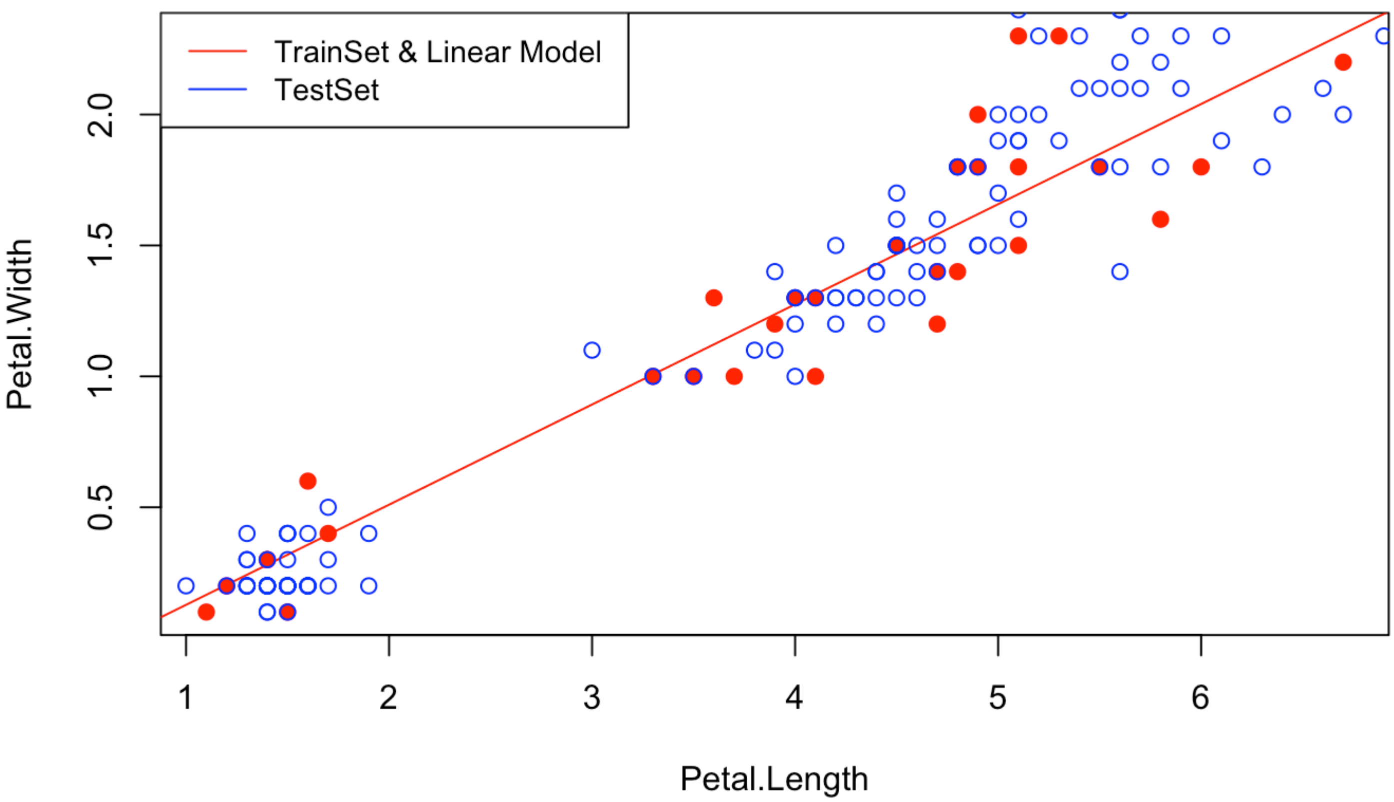
```
        1         2         3         4         5         8         9        10        11 
0.2805356 0.2805356 0.2422673 0.3188039 0.2805356 0.3188039 0.2805356 0.3188039 0.3188039 
       12        13        15        16        17        19        20        21        22 
0.3570723 0.2805356 0.2039989 0.3188039 0.2422673 0.3953406 0.3188039 0.3953406 0.3188039 
       23        24        25        26        27        28        29        30        31 
0.1274623 0.3953406 0.4718772 0.3570723 0.3570723 0.3188039 0.2805356 0.3570723 0.3570723 
       32        34        35        37        38        39        40        41        42 
0.3188039 0.2805356 0.3188039 0.2422673 0.2805356 0.2422673 0.3188039 0.2422673 0.2422673 
       43        45        46        47        48        49        50        51        52 
0.2422673 0.4718772 0.2805356 0.3570723 0.2805356 0.3188039 0.2805356 1.5433905 1.4668539 
       53        54        55        56        57        58        59        60        62 
1.6199272 1.2755122 1.5051222 1.4668539 1.5433905 1.0076339 1.5051222 1.2372439 1.3520489 
       63        66        67        69        70        71        72        73        75 
1.2755122 1.4285855 1.4668539 1.4668539 1.2372439 1.5816589 1.2755122 1.6199272 1.3903172 
       76        78        80        81        84        85        86        87        88 
1.4285855 1.6581955 1.0841706 1.1989756 1.6964639 1.4668539 1.4668539 1.5433905 1.4285855 
       91        92        93        95        96        97        98        99       100 
1.4285855 1.5051222 1.2755122 1.3520489 1.3520489 1.3520489 1.3903172 0.8928289 1.3137806 
      101       102       103       104       105       106       107       108       109 
2.0408789 1.6964639 2.0026105 1.8878055 1.9643422 2.2704889 1.4668539 2.1556839 1.9643422 
      110       111       112       113       114       115       117       119       120 
2.0791472 1.6964639 1.7730005 1.8495372 1.6581955 1.6964639 1.8495372 2.3852939 1.6581955 
      121       123       124       125       129       131       132       133       135 
1.9260739 2.3087572 1.6199272 1.9260739 1.8878055 2.0791472 2.1939522 1.8878055 1.8878055 
      136       137       139       140       141       143       144       145       146 
2.0791472 1.8878055 1.5816589 1.8112689 1.8878055 1.6964639 2.0026105 1.9260739 1.7347322 
      147       148       149 
1.6581955 1.7347322 1.8112689 
```

Hide

```r
plot(trainSet$Petal.Length, trainSet$Petal.Width, xlab = 'Petal.Length', ylab = 'Petal.Width', co
l='red',pch = 19)
abline(lm(Petal.Width ~ Petal.Length, trainSet),col='red')
```

Hide

```r
points(testSet$Petal.Length, testSet$Petal.Width,col='blue')
legend("topleft", legend=c("TrainSet & Linear Model","TestSet"),
       col=c("red","blue"),lty=1, cex=0.9)
```



7. Write and Informal Report

Now that you've completed the R tutorial and the Find The Errors tasks, write an informal report in Word to Danielle from Blackwell.

Your report should include the following:

Your predictions concerning how far a certain car can travel based on speed. (From the R tutorial.) Your predictions concerning the petal length through using the petal's width. (From the Find the Errors task.) The errors/warning messages that you encountered and how you overcame them. In addition, think about adding the answers to these questions within your report:

Was it straightforward to install R and RStudio? Was the tutorial useful? Would you recommend it to others? What are the main lessons you've learned from this experience? What recommendations would you give to other employees who need to get started using R and doing predictive analytics in R instead of Rapidminer?

Getting Started with R: Informal Report

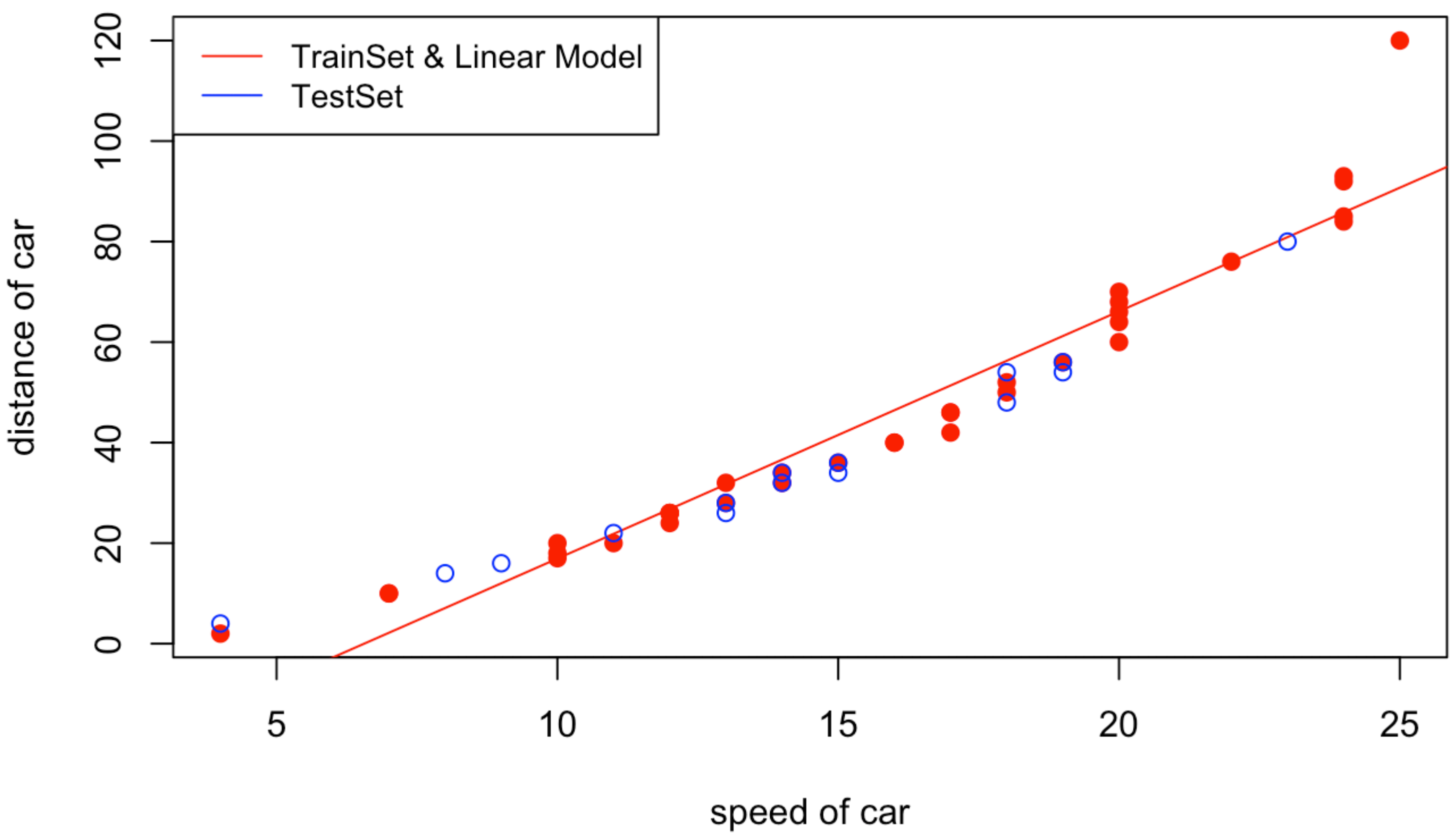
Summary

In this task we explored using R as a tool for data science/analytics. Compared to Rapid Miner, R is significantly more powerful and flexible, though it may seem more complex to new users. In particular, utilizing R via RStudio gives the user a kind of integraded development environment (IDE) for data science. The task covered the basics of a data science pipeline: loading key modules/libraries, loading data, cleaning & preprocessing data, apply a train-test split, training a machine learning model (ML), and making a prediction using the ML model. Overall, the ability to implement these steps in code and via a computational notebook style, makes data reproducability and much more efficient. This is because the notebook style of data analysis allows you to integrate written text with code and output, allowing you to create a kind of user manual for reproducing your results. Furthermore, utilizing code to perform data analysis gives you access to useful tricks, such as for loops, if-then statements, custom functions, and more.

Results

Two data sets were analyzed: (1) “cars.csv” & (2) “iris.csv”. In both data sets, double (numeric), and categorical data columns were present, however in this task we focused only on the numeric data. Specifically, after some preliminary data exploration via scatterplots, histograms, and Q-Q plots, we performed a 70-30% train-test split and fit a linear regression model to the data sets.

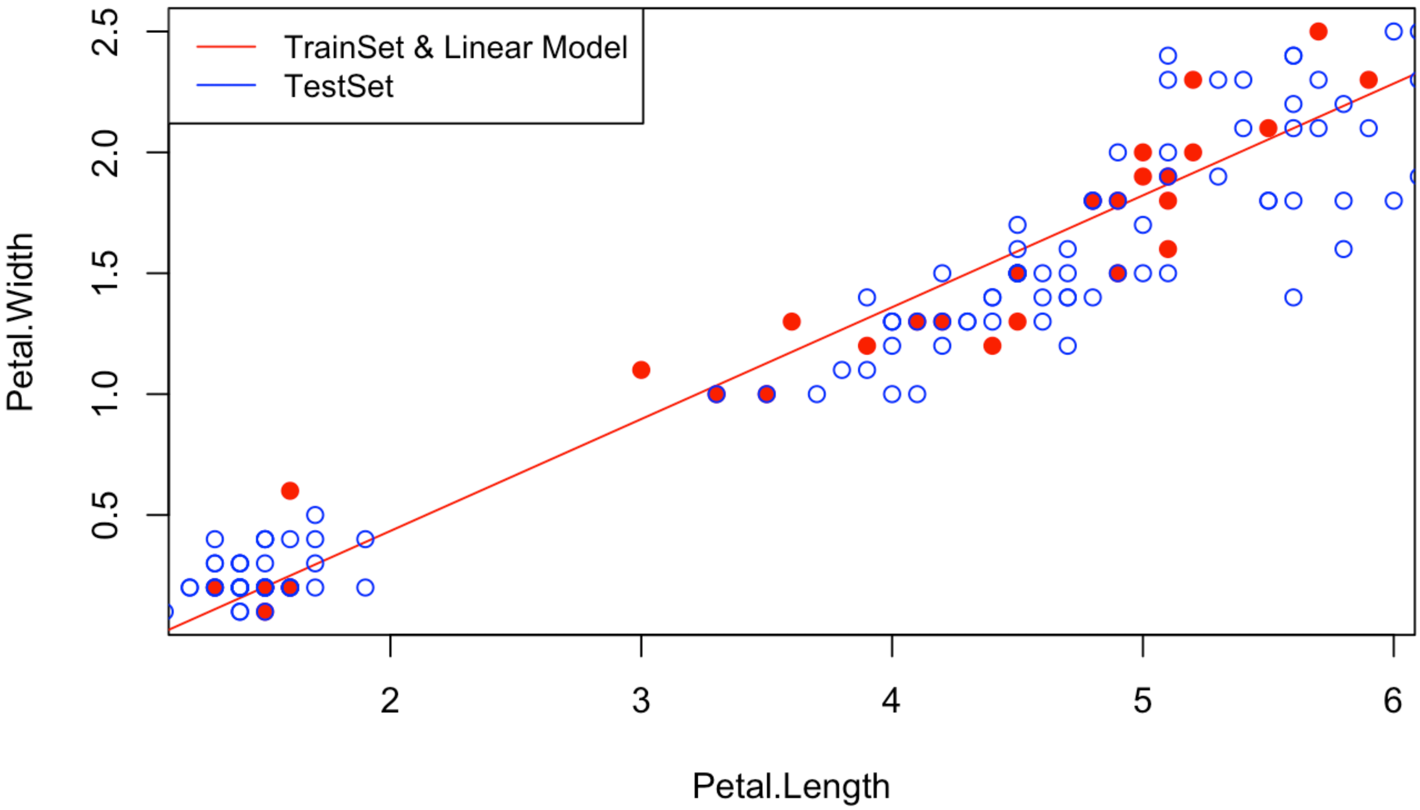
In the case of “cars.csv”, we applied linear regression to predict the distance of a card, given it’s speed. The result of this linear model can be seen in the figure below.



Here, we have highlighted the data points from the training set in red. The linear model was trained on this data set, thus the prediciton line is also colored red. The test set can be in blue. Note that the test set covers a similar distribution/range as the training set, though there are more data points. This linear model achieve an R^2 score of 0.93 on the testing set, implying the a good model fit, as can be seen from the plot. It should be noted that one could likely transform the x-axis, possibly using a semi-log scale of some degree, to force the relationship between the distance of the car and the speed of the car to become event more linear. This would then give bette prediction results in the extremes of the data set and probably lead to better extrapolation

Following the “cars.csv” data analysis, we performed an analysis on the “iris.csv” data set. Part of the analysis for this data set involved debugging code provided by the instructor. This code, along with the corrected code, can be found in the “Find the Errors in an R Scripts” section of this notebook. The original code is denoted by the comment “# original code”, the resulting errors have been pasted below the original code as a comment as well, followed by the new code (#new code). Most of the errors were simply syntactic in nature. The types of errors are most easily avoided using the autofill (tab-out) functionality common to most scripting languages.

After debugging the provided code, we performed another linear regression fit, but this time to the “iris.csv” petal length vs. petal width. The results of the model are shown below, with color coding identical to the previously analyzed figure.



Here we see a linear model appears to represent the true nature of the trend in the data better than it did in the car case, however there is also more spread in the data for a given petal length.

Concluding Comments

Overall, I found it pretty straightforward to download and install R and RStudio. Most of the activity was pretty straightforward, however I also have a lot of experience with notebooks generally and a few different coding languages, all of which have their own unique ‘features’. One thing I enjoyed the most was actually exploring beyond what the assignment explictly asked. Specifically, I found it useful looking into how to build custom functions in R and implement for loops. This was my first time using R and overall I would say it is useful for data scientists to be familiar with R, but I honestly would not recommend new data scientists spend too much time in R. This is primarily because R seems a bit clunky and less scalable/flexible compared to python. In particular, more advanced machine learning models, such as neural networks, are not commonly developed in R. Additionally, python is used to develop professional software interfaces, interact with analytical hardware machines/tools, and much more, this it is more likely that a data scientist could integrate a code into some software engineers data acquisition/analysis pipeline if it is written in R. Another complaint about R is that some of the syntax is very non-intuiative and clunky. For example, why do I need to do ‘<-’ to make expressions all the time. It would be some much nicer if I could just press “=”. Finally, the error handling in R seems much less transparent the in python. In R it isn’t immediately obvious which line of code threw an error.

All that being said, I still think knowing R is useful because it is still a popular language in data science and being able to "speak" many different languages is always useful.

As for recommendations to others in learning R, I would say with coding in general the key is to just practice & practice. Things feel slow and uncomfortable at first, but there is a large community that can help you solve almost any problem and the only thing stopping you from building whatever code or ML model you want is just time and perseverance.