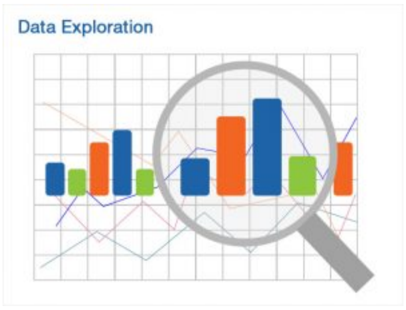
Diving Deeper into Data Exploration

Background

This post is dedicated to diving deeper into data analysis, and learning to use some of the tools offered by R libraries, most notably dplyr and ggplot2. While it may seem daunting to work with large tables of hundreds, or sometimes even thousands of rows of data, there are many simple commands and tricks you can use to not only organize the data effectively, but also visualize it in colorful and easily-interpretable graphs so that you can make sense out of this mass of text! Perhaps the greatest talent of someone who performs data analysis is the ability to make sense out of confusing data and help anyone understand what the data means. In this post, I will go over all steps of basic data analysis with R packages from start to finish to hopefully aid you in your own data exploration!



Getting Started

Before we start, we are going to need a data set we are interested in to analyze! However, we must load in the packages we want to use for the analysis before we actually start to do any analysis. For this project, we will be using the readr, dplyr, and ggplot2 packages. You can load them into your Rmd file with the following commands once you have downloaded them:

```
library(readr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
```

The readr package is to help aid in downloading files and doing text manipulations with them if necessary. To read more about its documentation, you can visit https://cran.r-project.org/web/packages/readr/readr.pdf. The data I have chosen today some weather data from Charlotte International Airport This data was found at this link in the fivethirtyeight github, a popular statistical analysis website; https://github.com/fivethirtyeight/data/blob/master/us-weather-history/KCLT.csv. Once you download the data from an online source and save it onto your computer, to load it into your current R session, use the following code onto the file path:

```
weather_data = read_csv("~/Documents/GitHub/stat133-hws-fall17/Post01/Data/KCLT.csv")
```

```
## Parsed with column specification:
## cols(
##   date = col_character(),
##   actual_mean_temp = col_integer(),
##   actual_min_temp = col_integer(),
##   actual_max_temp = col_integer(),
##   average_min_temp = col_integer(),
##   average_max_temp = col_integer(),
##   record_min_temp = col_integer(),
##   record_max_temp = col_integer(),
##   record_min_temp_year = col_integer(),
##   record_max_temp_year = col_integer(),
##   actual_precipitation = col_double(),
##   average_precipitation = col_double(),
##   record_precipitation = col_double()
## )
```

Exploring the Data

Great! Now we have loaded in the data. In many cases, the data we have loaded in will be unclean, and we will have to look through the data to see whether errors or missing/unknown values exist. Thankfully, our dataset has been cleaned as my focus will be on data exploration and analysis, and not cleaning. Now we want to know more about our data, so we'd like to peek into the first few rows. This is something which should be done with every new dataset so that you know what the data is telling us and supposed to convey. We can do this with the command:

```
weather_data = weather_data[,c("date","actual_min_temp","actual_max_temp")]
head(weather_data,5)
```

```
## # A tibble: 5 x 3
##       date actual_min_temp actual_max_temp
##      <chr>           <int>           <int>
## 1 2014-7-1              70              91
## 2 2014-7-2              74              95
## 3 2014-7-3              71              93
## 4 2014-7-4              64              86
## 5 2014-7-5              60              84
```

Now we can see the format of the data inside weather_data! We see that the first weather record is from July 1st, 2014 along with all the statistics computed. How many days of data is in the file? We can compute this with the code

```
nrow(weather_data)
```

```
## [1] 365
```

Since we see that we have 365 rows, and there is 1 record for each day, it means we have weather data for exactly 1 year. Now lets say we want to see what the hottest temperature observed during this interval was. We can find the maximum value in the actual_max_temp column of the table with the following code

```
max(weather_data$actual_max_temp)
```

```
## [1] 100
```

So in Charlotte, North Carolina, the hottest day had a temperature of 100 degrees. But lets say we want to know which day this was. Then we can find the date with the command

```
weather_data[which.max(weather_data$actual_max_temp),]
```

```
## # A tibble: 1 x 3
##        date actual_min_temp actual_max_temp
##       <chr>           <int>           <int>
## 1 2015-6-18              74             100
```

The which command gives us the index where the max temperature occurs, and then by calling weather_data(i,) where i is the index, it returns the ith row. So here, we see the 100 degree day ocurred on June 18th, 2016.

Data Manipulation and dplyr

Now rather than just finding values, we are going to actually add values to the table and manipulate values. The dplyr package will really help with this. Lets say we want to create a column which tells us the difference between the high and low temperature of the day. Then we can modify the weather_data table by

```
new_weather_data = mutate(weather_data, temp_difference = actual_max_temp-actual_min_temp)
head(new_weather_data,5)
```

```
## # A tibble: 5 x 4
##       date actual_min_temp actual_max_temp temp_difference
##      <chr>           <int>           <int>           <int>
## 1 2014-7-1              70              91              21
## 2 2014-7-2              74              95              21
## 3 2014-7-3              71              93              22
## 4 2014-7-4              64              86              22
## 5 2014-7-5              60              84              24
```

Now if we look at our code, we see that a new column called temp_difference has been added with the mutate command! We have saved this in a new table. This is good practice, because you should always have 1 copy of the original data without any changes. We can also use dplyr to do more sophisticated filtering functions. Lets say we want to see all the days which had a temperature difference of over 40 degrees. Then we can use the command

```
big_change = filter(new_weather_data, temp_difference>40)
big_change
```

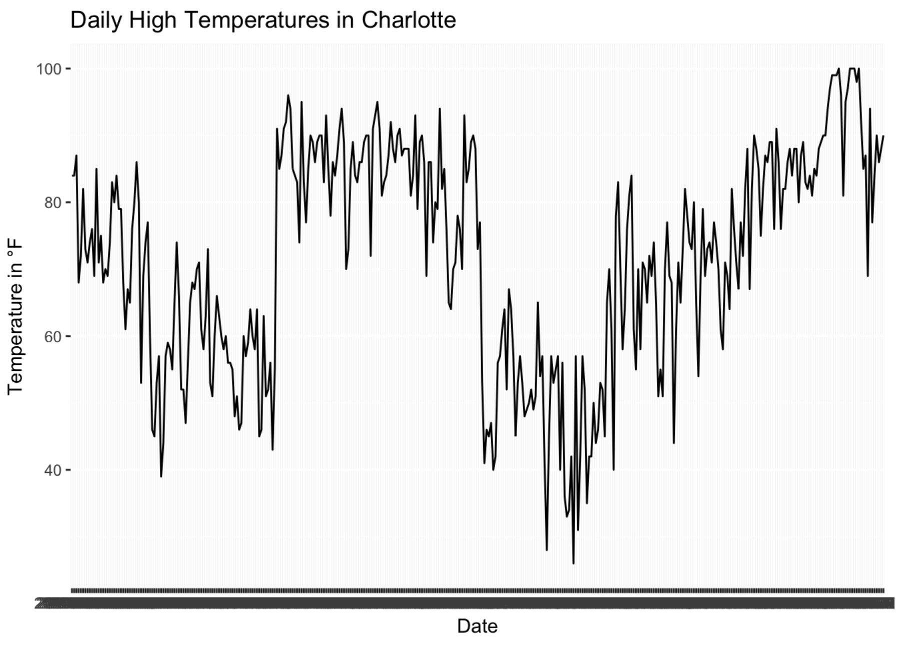
```
## # A tibble: 4 x 4
##         date actual_min_temp actual_max_temp temp_difference
##        <chr>           <int>           <int>           <int>
## 1  2014-11-3              24              65              41
## 2 2014-12-13              25              66              41
## 3  2015-3-17              41              84              43
## 4  2015-3-31              35              77              42
```

So there were 4 days with a temperature difference of over 40 degrees. The day with the greatest temperature difference, March 31, 2015 had a high of 77 degrees and low of 35 degrees. Talk about crazy temperature swings! These are just some of the simple data manipulations you can do with dplyr. To see more simple operations you can do with the dplyr package, check out https://www.rstudio.com/wp-content/uploads/2015/02/data-wrangling-cheatsheet.pdf. For a more detailed walkthrough, I found this video by Data School very informative and helpful: https://www.youtube.com/watch?v=jWjqLW-u3hc.

Data Visualization With ggplot2

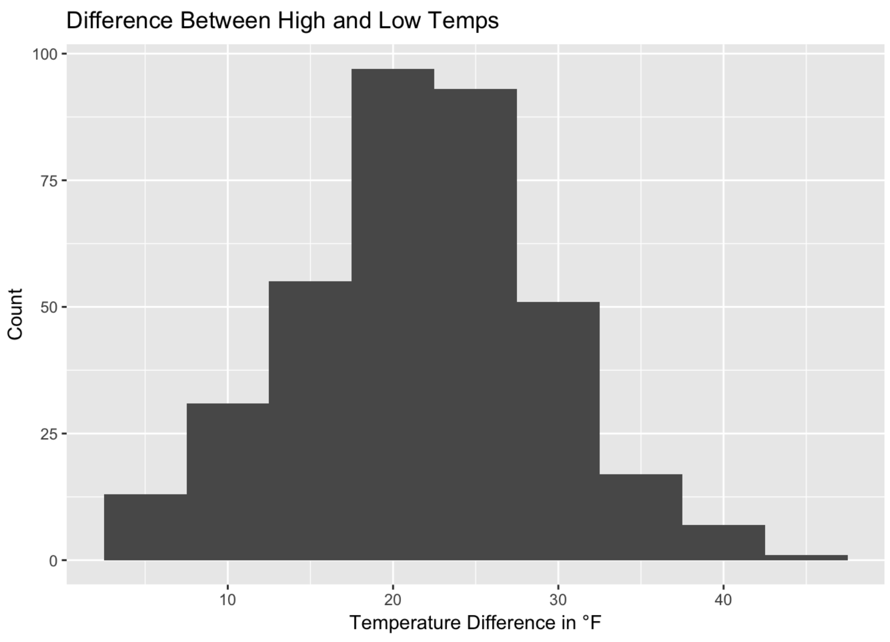
Now comes the fun part, making the graphs! Up until now, we have only been manipulating and changing the data within our dataframe. However, when it comes to showing our findings, a data table is definitely not the most user-friendly way of showing findings or trends. While R does have built-in plot functions, there generally aren't very good, and most people who need to make graphs in R use ggplot2, so we will get familiar with this tool as well. We have already loaded this package in earlier, so lets get down to making graphs! Lets say that we want to make a simple line graph visualization of the temperatures for the whole year in Charlotte. Then using the following commands

```
ggplot(data = new_weather_data, aes(x = date, y = actual_max_temp, group = 1))+geom_line() + ggtitle("Daily High T
emperatures in Charlotte") + xlab("Date") + ylab("Temperature in °F")
```



Cool! We are not just limited to line graphs though. Ggplot2 is very flexible, and we can make any kind of graph imagineable. Here is a hisogram of the temperature differences for each day:

```
ggplot(data = new_weather_data, aes(temp_difference))+geom_histogram(binwidth = 5) + ggtitle("Difference Between H
igh and Low Temps") + xlab("Temperature Difference in °F") + ylab("Count")
```



Here we can see that on average, the difference between the high and low temperature was between 20-25 degrees, which makes sense because we expect there to be at least some difference, and the nights to be colder than the day. To learn how to make some more basic ggplot graphs, check out this guide at https://www.rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf. To get a more detailed video guide, you can watch https://www.youtube.com/watch?v=HeqHMM4ziXA.

Putting it All Together

Congratulations, you have finished your first data exploration project! While most projects in the real world deal with a lot more data, sometimes on the order of millions of rows, the general data exploration and analysis process is almost always the same. Here are the general steps.

1. Initially, you look at the data and try to make sense of what you have.
2. Once you are able to understand the kind of data you have, try and clean it before performing any analysis.
3. Once it is clean and ready, try and compute the statistics you are looking for or filter out for results you want.
4. Make a graph and visualize your findings to find trends and show other people what you found.

This seems simple, and that is the point - Data exploration is fun! While the nitty-gritty of calculating statistics or complex data manipulation can get technical, by breaking down larger problems into smaller ones and following a general organized guide, it becomes much easier to get things done. Thanks for reading, and I hope you all learned something! To get some of your own simple data sets on which you can perform your own exploratory analysis, you can download sets from https://github.com/fivethirtyeight/data. This data is good for starters because it is already cleaned for you, and clearly structured and organized. For more advanced datasets with more interesting data which might also be uncleaned and unstructured, you can visit https://www.kaggle.com/datasets.