intRo 3

Data and Graphing

Billy Fryer with inspiration from Brendan Kent, Graham Pash, and Jason Thompson

Reading in CSVs

There are many ways to read in data sets. We are going to focus on CSVs since that is one of the most common data types found. CSV stands for Comma Separated Values. This that the data is *delimited* or seperated with commas. These will eventually make up our columns.

Tidyverse Method (read_csv())

Using the read_csv() function is probably the easiest way to read in a csv. See code below

```
library(tidyverse)

# setwd() means to set your working directory.
# A working Directory is where your computer searches for files
# when loading them and where new files will be outputted to
setwd("C:/Users/billy/OneDrive/Desktop/Sports Analytics/intRo Tutorials/intRo 3 Data and Graphing")

kenpom1 <- read_csv("kenpom_2019-02-11.csv")
```

There are other options to specify if data is missing and other unusual things. To learn more, look at the R Documentation.

Base R Method (read.table())

The read.table() method is much more flexible for reading in data. It comes in Base R which means you do not have to import any packages to read the data (though you'll probably have tidyverse in anyways).

One of the ways read.table() allows for flexibility is that is allows you to specify separators, using the sep = option. The header = option is for when the first row of your data set is the names of your columns.Example Below:

```
kenpom2 <- read.table(file = "kenpom_2019-02-11.csv",
                      header = TRUE,
                      sep = ",")
```

Same as before, there are other options to specify if data is missing and other unusual things. To learn more, look at the R Documentation.

Basics of Web Scraping

Web Scraping is a topic that seems big and scary, but actually isn't as bad as one would think. Web Scraping is pulling data from the Internet so it can be used in a functional form. It is an individualized process however, meaning that you have to change the way you do it for every individual website. The example here is for the Kenpom website front page, one of the best sources for men's college basketball information

```
library(rvest)


url <- "https://kenpom.com/"
url %>% read_html() %>% html_table() -> df
df <- df[[1]] # only want the first (main) table

colnames(df) <- df[1,] # Assigning the first row as the headers
df <- df[-1,] # Drops 1st row from data frame
df <- df[!is.na(as.numeric(df$Rk)),] # drop column name rows
row.names(df) <- 1:nrow(df) # makes rownames the numbers 1:number of rows

# handle duplicate column names by renaming everything
colnames(df) <- c('Rk','Team','Conf','W-L','AdjEM','AdjO','AdjO_Rk','AdjD','AdjD_Rk', 'AdjT','AdjT_Rk',

# subset only useful variables
df <- df %>% select(c('Rk','Team','Conf','W-L','AdjEM','AdjO','AdjD','AdjT','Luck', 'OppAdjEM','OppO','

# Change types to numeric
# This line of code applies the as.numeric() function to column
# 1 and columns 5 to 13 which makes them usable later. Ideally,
# We would use more tidyverse functions to make this better, but
# for now, it will do.
df[,c(1,5:13)] <- sapply(df[,c(1,5:13)], as.numeric)
```

We can go much more in depth about web scraping, but that would get very time consuming. It would be a better use of our time to look at data that is easier to access.

Packages with Good Sports Data

I want to give credit to Brendan Kent's Website https://brendankent.com for the majority of these R packages mentioned. For a link to this specific article, check out https://brendankent.com/2021/03/09/free-sports-data-sources/.

One of the unique things about R is that anyone can create a package and share it with others for public use on Github. Github is a code sharing platform that we will discuss more about in a future meeting. For now, we will learn how to get packages off of Github by using the devtools package.

```
# Install and load devtools package
#install.packages("devtools")
library(devtools)

# Use the devtools function install_github to install packages from github
#install_github("lbenz730/ncaahoopR")
```

```r
library(ncaahoopR) # Then load it just as you would any other package
```

If this is not working, look and see if your packages need updating. To do this, click on the packages tab on the right side of the screen and then click Update.

We will now discuss packages by sports. If any of these sound interesting to you, just look them up to see documentation and how to download them!

Just a side note: There are a lot of puns using R in the names of packages. You have been warned.

American Football

- nflscrapR
- nflfastR
- NFLSimulatoR
- cfbscrapR

Basketball

- nbastatR
- ncaahoopR
- hoopR
- wehoop
- airball

Baseball

- baseballr
- Lahman (Note: this is not on GitHub, it is built directly into R)

Soccer

- socceR
- footballR
- ggsoccer
- worldfootballR
- tyrone_mings

For other sports, I again point you to Brendan Kent's article https://brendankent.com/2021/03/09/free-sports-data-sources/. More sports are included there as well as websites you could possible scrape from.

Graphing

One of the most useful things that sports analysts do is create graphics. Graphics are simply easy ways do digest information. These include charts, small tables, and graphs. A great way of doing this is with the `ggplot2` package! Let's do some examples based on the Kenpom data we scraped earlier.

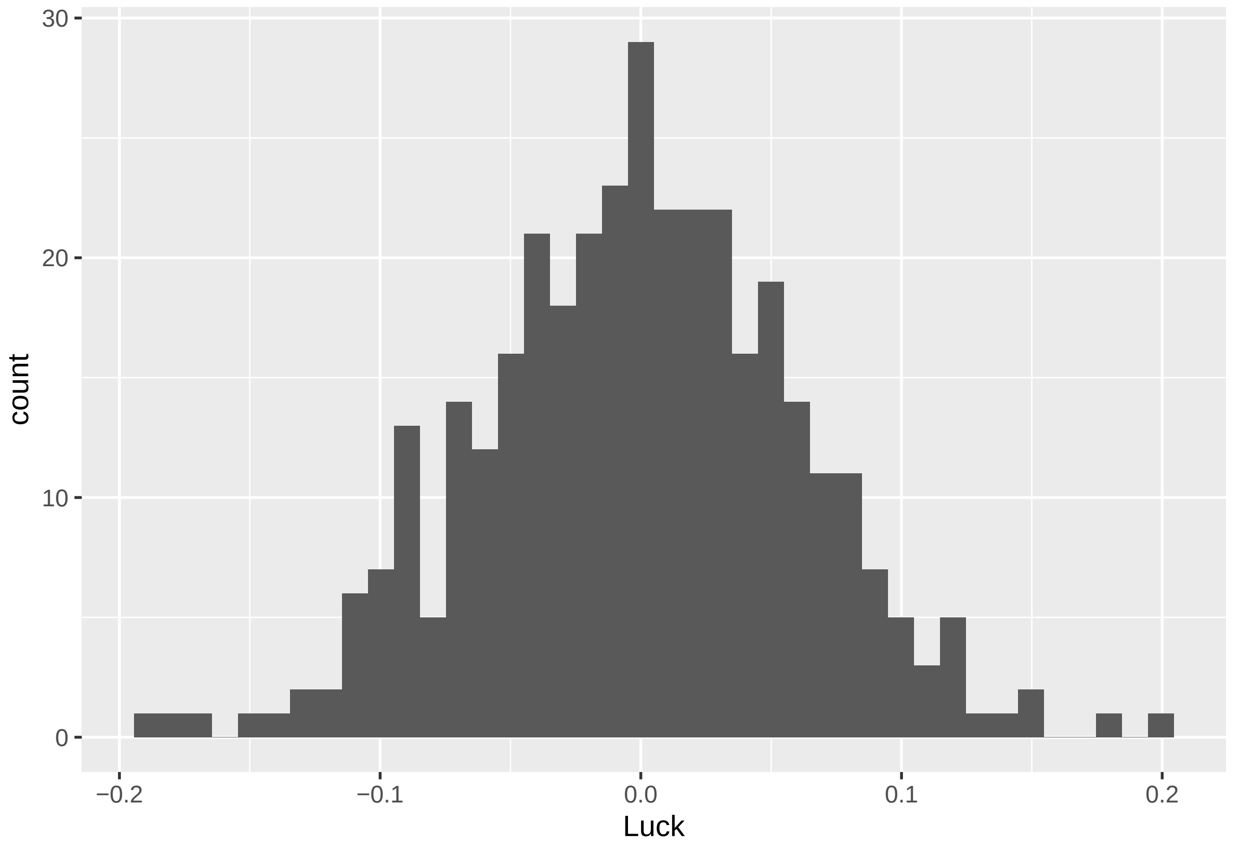
Histogram of Team Luck

To Start, we will make use the `ggplot` function to give the data parameters. The first argument will be the name of the data frame we want to use.

The second argument, mapping, is normally the trickiest to wrap your head around. If you are mapping something *based on a value inside your data*, you must wrap it inside `aes()`. If you want to set something a *constant value*, it does not go inside `aes()`. For example, since we want to make a histogram based on the Luck variable, that needs to go inside an `aes()` statement. This construct applies later on as well inside of the `geom_***()` functions as well.

Finally, we are ready to make the actual graph. Depending on what we want, we would want to use the corresponding `geom_***()` function. Since we want a histogram, we use the `geom_histogram()` function. We can specify the number of bins we want with the `bins =` option. Since we want to specify this as a set number, we do not wrap it in a `aes()` function.

```r
ggplot(df, aes(x = Luck)) +
  geom_histogram(bins = 40)
```



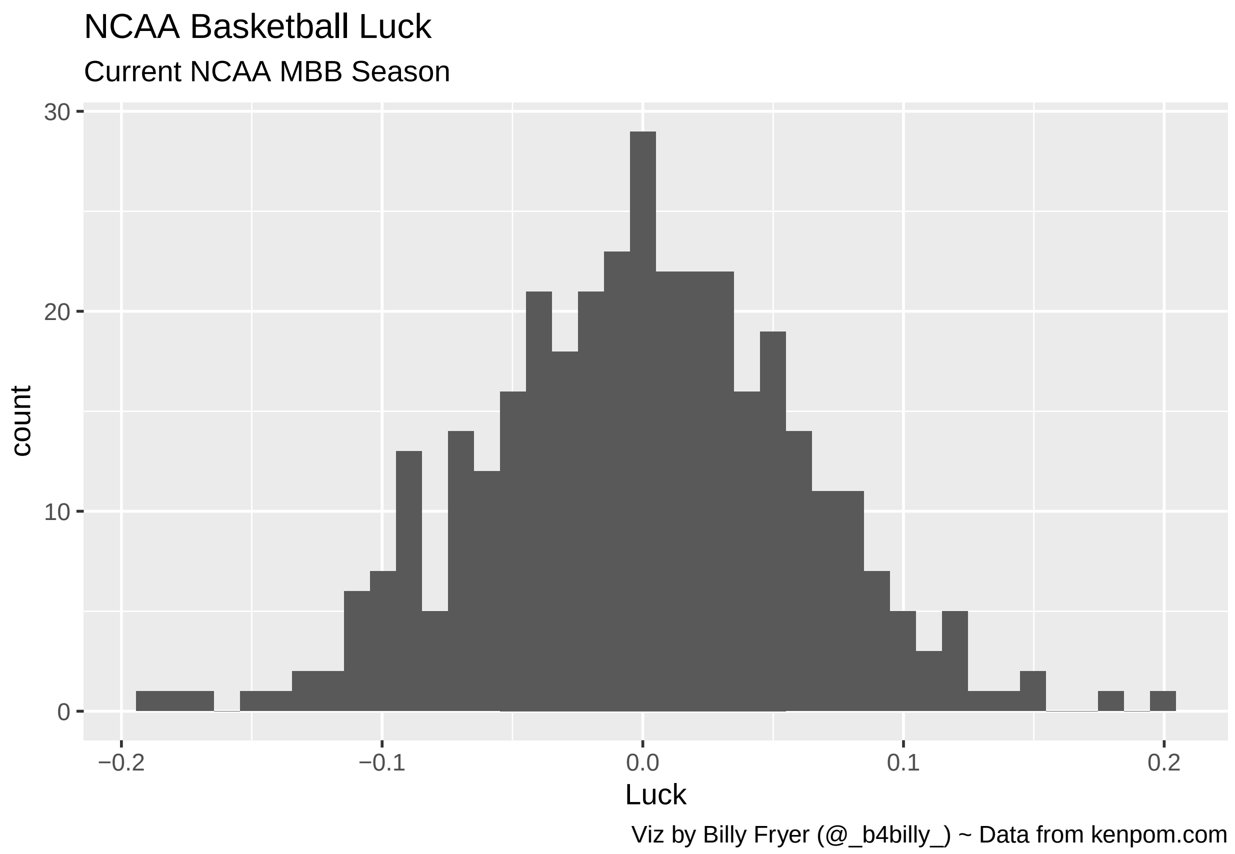
Upgrading our Plot

If we want to add on to a plot, we can do that to! To add layers on to a plot, we put a + between layers. Plots can also be saved to an object which allows us to add on at different times. To see a plot after saving it, wrap it in a `print()` statement.

```r
luck_plot <- ggplot(df, aes(x = Luck)) +
  geom_histogram(bins = 40)
```

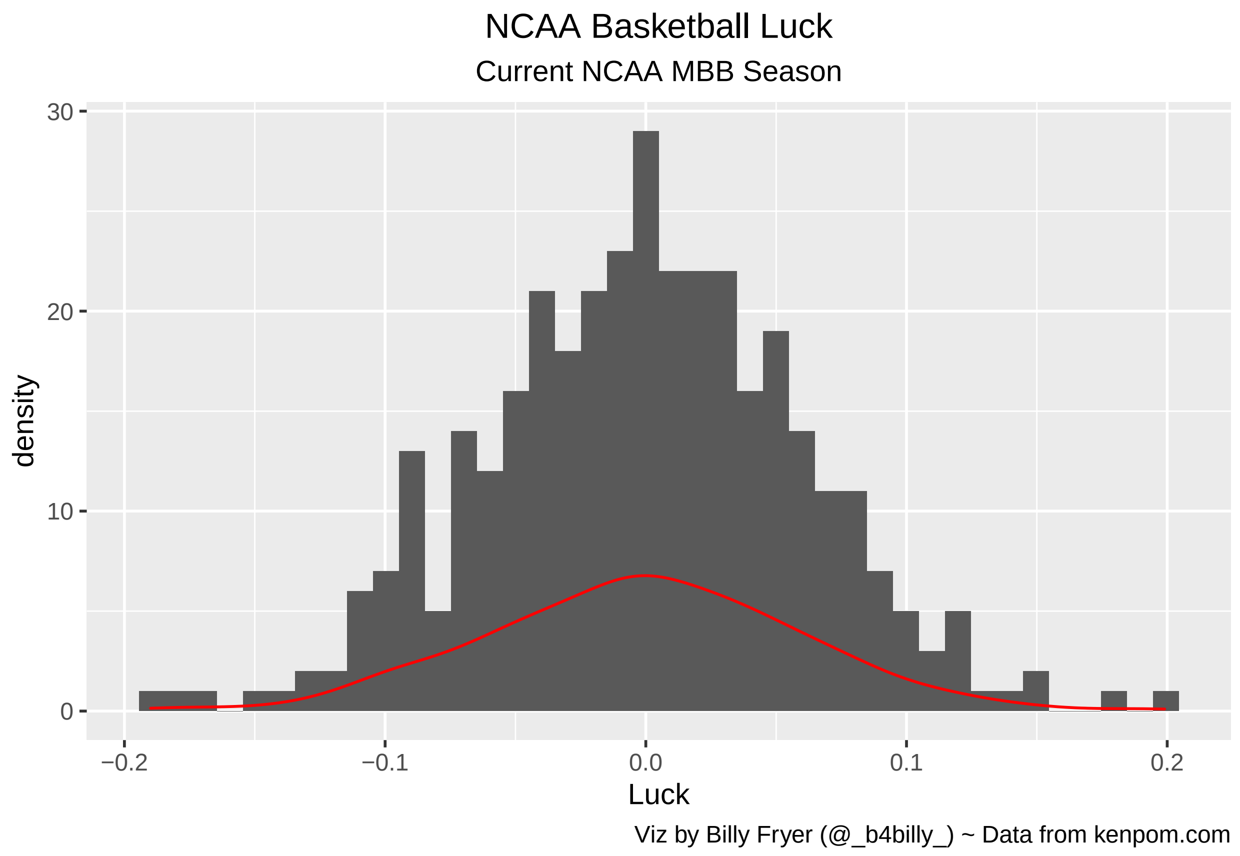
To add a title, subtitle, and caption use the `labs()` function.

```r
luck_plot <- luck_plot +
  labs(title = "NCAA Basketball Luck",
       subtitle = "Current NCAA MBB Season",
       caption = "Viz by Billy Fryer (@_b4billy_) ~ Data from kenpom.com")

print(luck_plot)
```



If we want a density curve around the histogram, that's easy too. We can even change the color to make it stand out more. Finally, to output the file to your computer, we use the `ggsave()` function

```r
luck_plot <- luck_plot +
  # Add a Red Density curve
  geom_density(color = "red") +
  # Center Title and subtitle
  theme(plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5))

print(luck_plot)
```



```r
#ggsave("Luckiest Teams.png", luck_plot)
```

The ggplot2 package is one of the best packages in R and there are tons of resources that one can learn more about it. For more information, search "ggplot2 cheat sheet" which gives more information about available plots in the package or "Tidy Tuesday" which is a R Community initiative to teach others about manipulating data with the `Tidyverse` package and plotting with `ggplot2`.

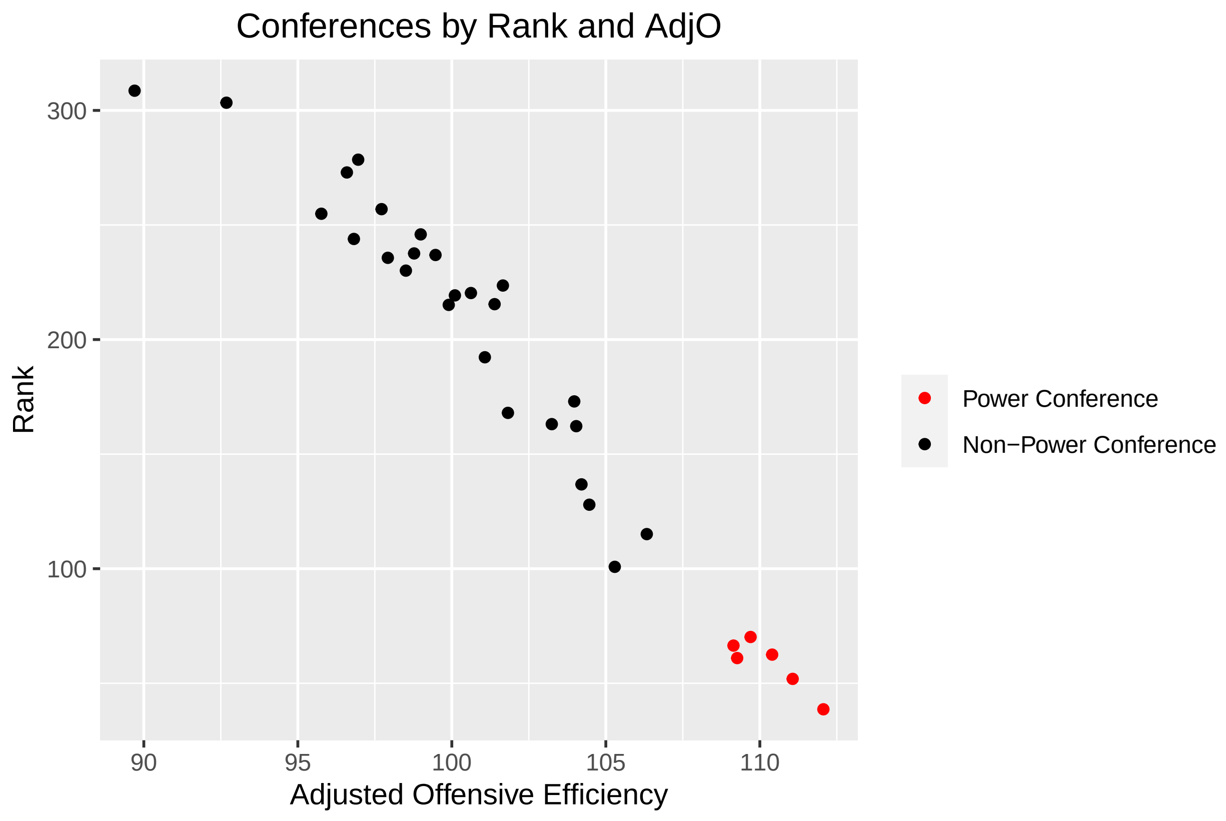
Graphing in Groups

There may be times where you want to group objects together and perform analysis on that. This is very useful when doing analysis where players may be represented in different rows, such as a pitch by pitch analysis in baseball. We will use the same data frame to group by Conference.

```r
# Make a copy of df and call it df_by_conference
df_by_conference <- df

df_by_conference <- df_by_conference %>%
  # Group By Conference
  group_by(Conf) %>%
  # Summarize Statistics
  summarize( # Find Mean Rank by Conference
            Rk = mean(Rk, na.rm = TRUE),
            # Find Mean AdjO by Conference
            AdjO = mean(AdjO, na.rm = TRUE)) %>%
  # Can't forget to Ungroup!
  ungroup() %>%
  # Add a Power 6 Column
  # ACC, SEC BIG 10, BIG 12, BIG EAST
  mutate(Power6 = case_when(Conf %in% c("ACC", "SEC", "B10", "B12", "BE", "P12") ~ "Power Conference",
                            TRUE ~ "Non-Power Conference"))

# Make Power6 a factor variable
df_by_conference$Power6 <- factor(df_by_conference$Power6)
```

Now that we have a new data set, it's time to graph it!

```r
# Give the df name then the x and y coordinates
ggplot(df_by_conference, aes(x = AdjO,
                             y = Rk)) +
  # We want to color by whether or not they
  # are a Power Conference, so we use the color = option
  # geom_point() plots data points
  geom_point(aes(color = Power6)) +
  # Labels
  labs(title = "Conferences by Rank and AdjO",
       # I didn't want to include a Legend Title
       color = "",
       y = "Rank",
       x = "Adjusted Offensive Efficiency",
       # Give Credit to Yourself and the Data Source
       caption = "Viz by Billy Fryer (@_b4billy_) | Data from kenpom.com") +
  # Centering the title
  theme(plot.title = element_text(hjust = 0.5)) +
  # Choosing colors. The # then the letters and numbers
  # Represents Hex Colors!
  scale_color_manual(values =  c(
                    "Power Conference" = "#FF0000",
                     "Non-Power Conference" = "#000000")
                    )
```



Viz by Billy Fryer (@_b4billy_) | Data from kenpom.com

Conclusion

As usual, this is only an introduction to sports data in R. There are also many more websites and packages that could be listed here. Many thanks again to Brendan Kent (Twitter: @brendankent, @MeasurablesPod), Graham Pash (Twitter: @gtpash), and Jason Thompson (Twitter: @jason24thompson) for allowing me to use some of their stuff.