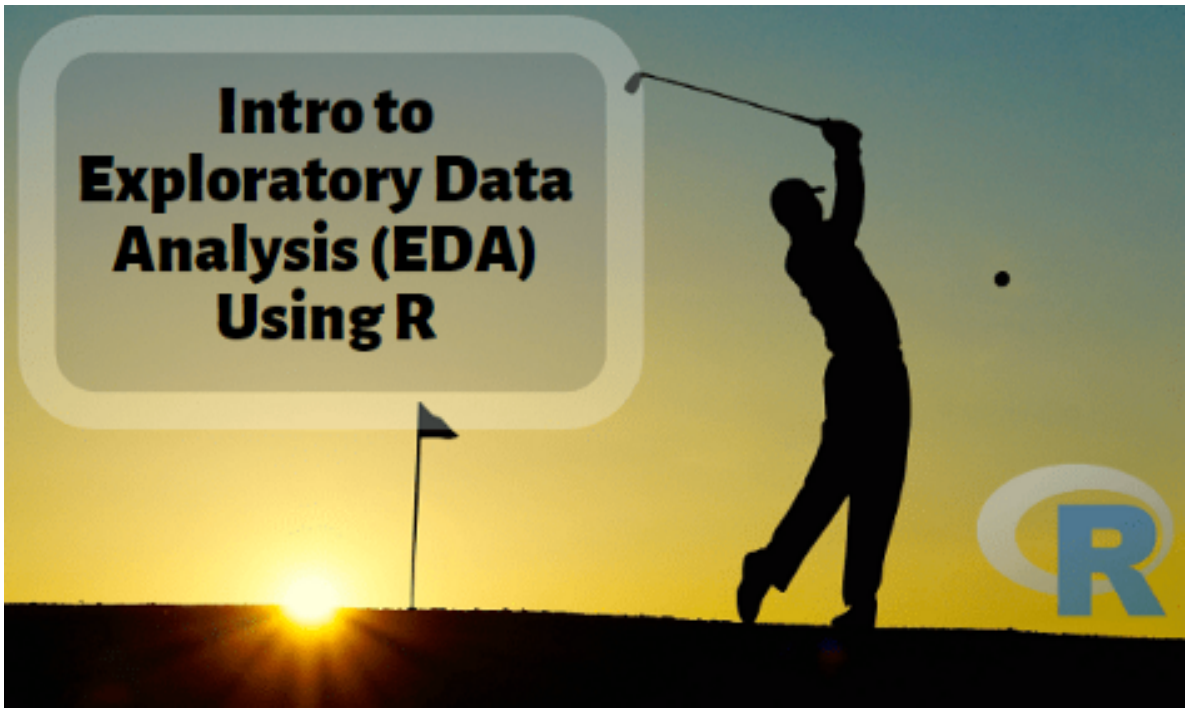
# Using R for Exploratory Data Analysis (EDA) — Analyzing Golf Stats

Jeff Griesemer Follow
Jul 4, 2019 · 6 min read ★




Whether you're a Data Architect, Data Engineer, Data Analyst, or Data Scientist, we have to work with unfamiliar datasets when starting a new data project. It's a little like having a blind date with a new dataset. You'll want to get to know more about it before feeling comfortable.

So, how do we get there? The answer is **Exploratory Data Analysis (EDA)**.

Exploratory Data Analysis is a term for initial analysis and findings done with data sets, usually early on in an analytical process.

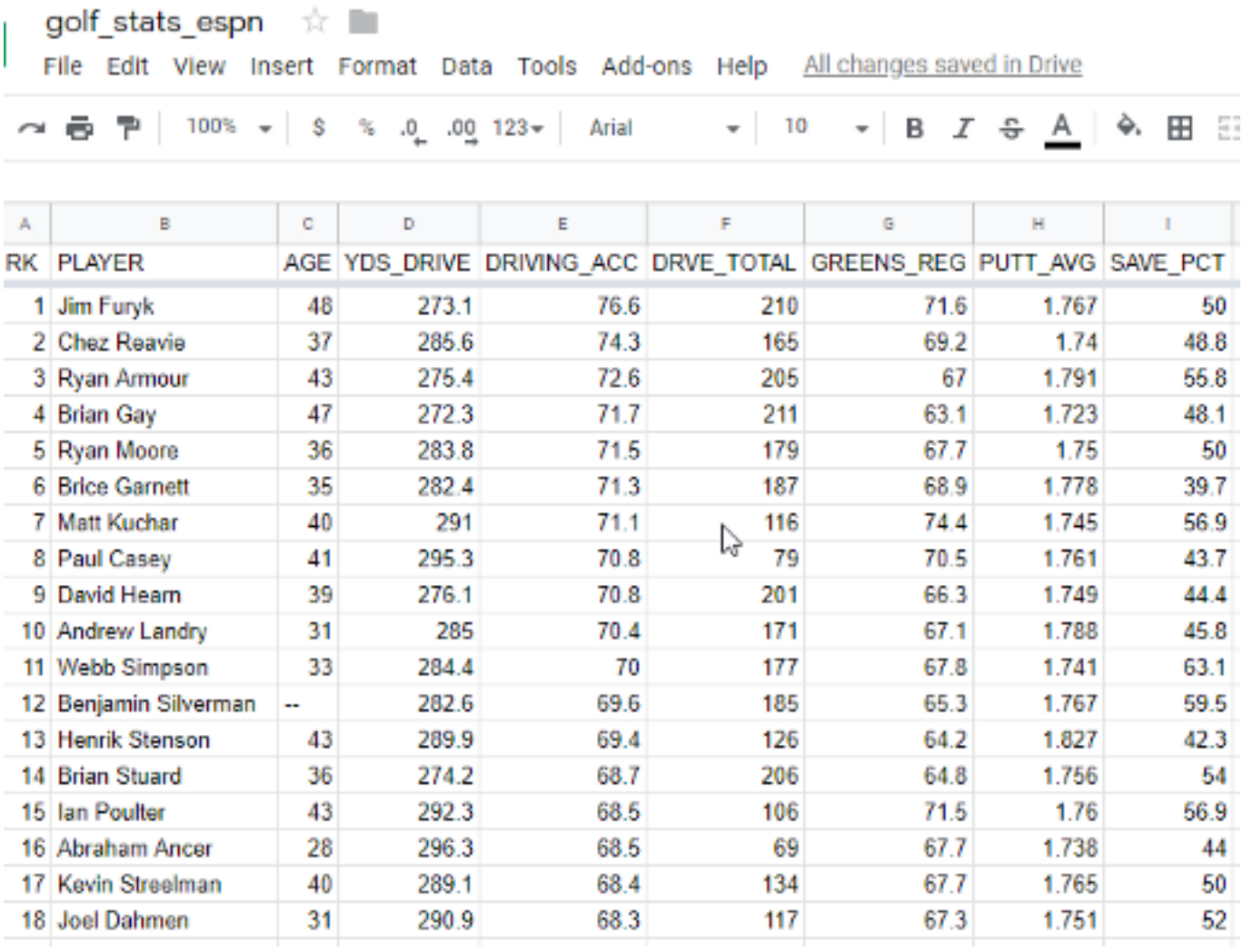
As a data professional, we'll sleep much better having gone through this process. Much time is wasted in future steps if this step is ignored.. re-work is needed to resolve data issues well after architecture foundations and data processing pipelines have been built.

. . .

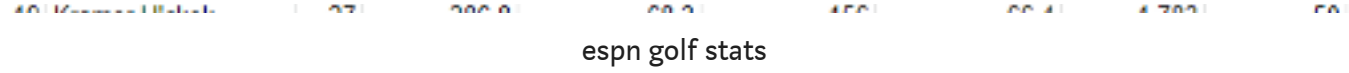
### Sample Dataset

With all the 4 majors done, and Tiger coming off his historic Master's victory, why not look at some golf statistics. I found high-level stats for 2019 on the espn.com website http://www.espn.com/golf/statistics. I put the contents into a Googlesheet for easy access. As you will soon see, this is a very basic dataset but will allow us to focus on the EDA process.

Below are sample rows of our dataset.




| A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|
| RK | PLAYER | AGE | YDS_DRIVE | DRIVING_ACC | DRVE_TOTAL | GREENS_REG | PUTT_AVG | SAVE_PCT |
| 1 | Jim Furyk | 48 | 273.1 | 76.6 | 210 | 71.6 | 1.767 | 50 |
| 2 | Chez Reavie | 37 | 285.6 | 74.3 | 165 | 69.2 | 1.74 | 48.8 |
| 3 | Ryan Armour | 43 | 275.4 | 72.6 | 205 | 67 | 1.791 | 55.8 |
| 4 | Brian Gay | 47 | 272.3 | 71.7 | 211 | 63.1 | 1.723 | 48.1 |
| 5 | Ryan Moore | 36 | 283.8 | 71.5 | 179 | 67.7 | 1.75 | 50 |
| 6 | Brice Garnett | 35 | 282.4 | 71.3 | 187 | 68.9 | 1.778 | 39.7 |
| 7 | Matt Kuchar | 40 | 291 | 71.1 | 116 | 74.4 | 1.745 | 56.9 |
| 8 | Paul Casey | 41 | 295.3 | 70.8 | 79 | 70.5 | 1.761 | 43.7 |
| 9 | David Hearn | 39 | 276.1 | 70.8 | 201 | 66.3 | 1.749 | 44.4 |
| 10 | Andrew Landry | 31 | 285 | 70.4 | 171 | 67.1 | 1.788 | 45.8 |
| 11 | Webb Simpson | 33 | 284.4 | 70 | 177 | 67.8 | 1.741 | 63.1 |
| 12 | Benjamin Silverman | -- | 282.6 | 69.6 | 185 | 65.3 | 1.767 | 59.5 |
| 13 | Henrik Stenson | 43 | 289.9 | 69.4 | 126 | 64.2 | 1.827 | 42.3 |
| 14 | Brian Stuard | 36 | 274.2 | 68.7 | 206 | 64.8 | 1.756 | 54 |
| 15 | Ian Poulter | 43 | 292.3 | 68.5 | 106 | 71.5 | 1.76 | 56.9 |
| 16 | Abraham Ancer | 28 | 296.3 | 68.5 | 69 | 67.7 | 1.738 | 44 |
| 17 | Kevin Streelman | 40 | 289.1 | 68.4 | 134 | 67.7 | 1.765 | 50 |
| 18 | Joel Dahmen | 31 | 290.9 | 68.3 | 117 | 67.3 | 1.751 | 52 |



espn golf stats

We'll load the dataset into R using the "googlesheets" library. (Googlesheet filename is "golf_stats_espn" and the sheetname is "2019_stats").

```
library(googlesheets)

googlesheet <- gs_title("golf_stats_espn")
df_2019 <- googlesheet %>% gs_read(ws = "2019_stats")
```

. . .

### EDA Step 1 : Data Validation and Data Quality

The *str()* function will do a sanity check on the structure and show sample data for each variable.

```
str(df_2019)
```

```
> str(df_2019)
Classes 'spec_tbl_df', 'tbl_df', 'tbl' and 'data.frame':        120 obs. of  8 variables:
 $ RK         : num  1 2 3 4 5 6 7 8 9 10 ...
 $ PLAYER     : chr  "Jim Furyk" "Chez Reavie" "Ryan Armour" "Brian Gay" ...
 $ AGE        : chr  "48" "37" "43" "47" ...
 $ YDS_DRIVE  : num  273 286 275 272 284 ...
 $ DRIVING_ACC: num  76.6 74.3 72.6 71.7 71.5 71.3 71.1 70.8 70.8 70.4 ...
 $ GREENS_REG : num  71.6 69.2 67 63.1 67.7 68.9 74.4 70.5 66.3 67.1 ...
 $ PUTT_AVG   : num  1.77 1.74 1.79 1.72 1.75 ...
 $ SAVE_PCT   : num  50 48.8 55.8 48.1 50 39.7 56.9 43.7 44.4 45.8 ...
```

str() function

If you have experience with R, you are probably familiar with the *summary()* function. It works well, but a more complete function is the *skim()* function from the "skimr" package. It breaks down the variables by type with relevant summary information, PLUS a small histogram for each numeric variable.

```
library(skimr)
skim(df_2019)
```

```
> library(skimr)
> skim(df_2019)
Skim summary statistics
 n obs: 120
 n variables: 8

-- Variable type:character ---------------------------------------------------
 variable missing complete   n min max empty n_unique
      AGE       0      120 120   2   2     0       27
   PLAYER       0      120 120   8  20     0      120

-- Variable type:numeric ------------------------------------------------------
    variable missing complete   n   mean     sd    p0    p25    p50    p75   p100     hist
 DRIVING_ACC       0      120 120  65.16   3.33   60.3  62.1   64.8   67.4   76.6 ▃▇▃▁▁▁▁▁
  GREENS_REG       0      120 120  67.01   3.21   59    64.9   67     69     75.6 ▁▃▇▃▂▁▁▁
    PUTT_AVG       0      120 120   1.76  0.029   1.7   1.74   1.76   1.78   1.84 ▂▃▇▃▃▂▂▁
          RK       0      120 120  60.5  34.79    1    30.75  60.5   90.25 120    ▇▇▇▇▇▇▇▇
    SAVE_PCT       0      120 120  50.52  7.33   36.1  45.75  50     54.85  71.9 ▁▃▇▃▂▁▁▁
   YDS_DRIVE       0      120 120 289.29  8.1   269.3 284.4  289.1  294.9  309.9 ▁▃▇▃▂▁▁▁
```

skim() function

Looks good **except** for the "AGE" variable. With R and many other analytics tools, data-types are assigned automatically as contents are read in, referred to as "schema-on-read". For "AGE", a character-type was assigned to our variable rather than a numeric-type. However, I'd like to treat AGE as a numeric-type to potentially apply numeric functions downstream. Why did R create the AGE variable as a character-type?

Let's do more EDA and run a *table()* function on the AGE variable.

```
with(df_2019, table(AGE))
```

```
AGE
-- 21 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 47 48 49 55
 1  1  2  4  4  7  9  7  4 10  4  9  5  8  6  2  3  6  8  2  2  9  1  1  2  2  1
```

table() function

The table() function shows distinct values for the variable on the top line, and the number of occurrences in the line below it. For the AGE variable, we can see 1 occurrence of "--". R had no choice but to define the variable as a "character" type.

We can fix that by using the “DPLYR” package. DPLYR specializes in “data wrangling”. You can efficiently accomplish a lot with this package — dataframe manipulation, transformations, filtering, aggregations, etc.

The R command below creates a new dataframe after performing the actions described by the bullets.

```
df_2019_filtered <- df_2019 %>%          # create new dataframe
 mutate(AGE_numeric = !(is.na(as.numeric(AGE)))) %>%
 filter(AGE_numeric == TRUE) %>%
 mutate(AGE = as.numeric(AGE))
```

- mutate → creates a new boolean variable identifying whether it’s value is numeric
- filter → uses the new boolean variable created in the mutate line above it to filter off non-numeric
- mutate → replaces the “AGE” variable; now defined as a numeric variable

Below is our new dataframe.

```
> str(df_2019_filtered)
Classes 'spec_tbl_df', 'tbl_df', 'tbl' and 'data.frame':       119 obs. of  9 variables:
 $ RANK_DRV_ACC: num  1 2 3 4 5 6 7 8 9 10 ...
 $ PLAYER      : chr  "Jim Furyk" "Chez Reavie" "Ryan Armour" "Brian Gay" ...
 $ AGE         : num  48 37 43 47 36 35 40 41 39 31 ...
 $ YDS_DRIVE   : num  273 286 275 272 284 ...
 $ DRIVING_ACC : num  76.6 74.3 72.6 71.7 71.5 71.3 71.1 70.8 70.8 70.4 ...
 $ GREENS_REG  : num  71.6 69.2 67 63.1 67.7 68.9 74.4 70.5 66.3 67.1 ...
 $ PUTT_AVG    : num  1.77 1.74 1.79 1.72 1.75 ...
 $ SAVE_PCT    : num  50 48.8 55.8 48.1 50 39.7 56.9 43.7 44.4 45.8 ...
 $ AGE_numeric : logi  TRUE TRUE TRUE TRUE TRUE TRUE ...
```

str() function

One more adjustment, I’m going to rename the column “RK” to a better name.

```
df_2019_filtered <- rename(df_2019_filtered, “RANK_DRV_ACC” = “RK”)
```

*Let’s pause here for a minute.*

### *Handling Dirty Data*

- *For this article, the missing AGE rows were filtered out. In a real analytics project, we'll have to look at the best course of action to take (filter the row, replace the character data, replace with NULL, etc).*

. . .

## Exploratory Data Analysis (EDA) — Part 2

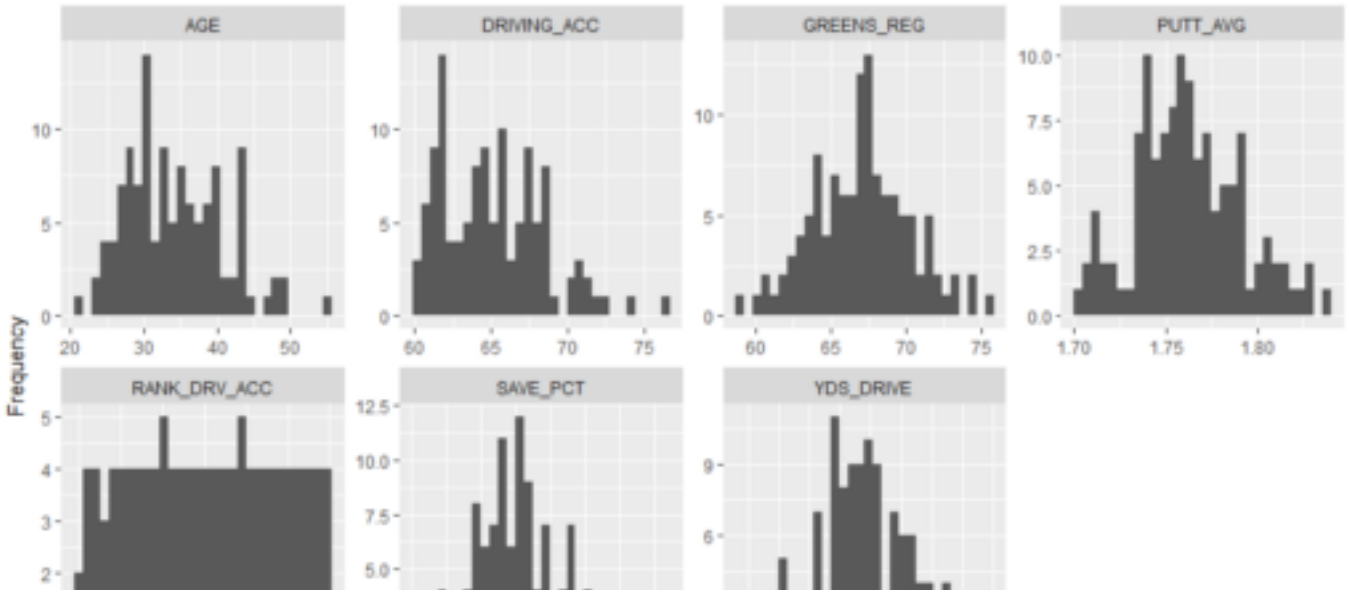
With our dataset examined and cleaned…

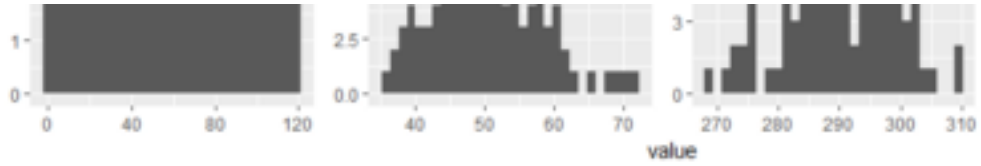
Part 2 leans more toward Data Analysts and Data Scientists. You may be surprised at the insights that can be derived during this phase, even on this very basic dataset.

### *plot_histogram()*

We'll use the "DataExplorer" library to learn more about our dataset. The *plot_histogram()* function will return a separate bar chart for each of our numeric variables. It shows the frequency (number of occurrences) for each value in the variable.

```
library(DataExplorer)
plot_histogram(df_2019_filtered)
```






plot_histogram() function

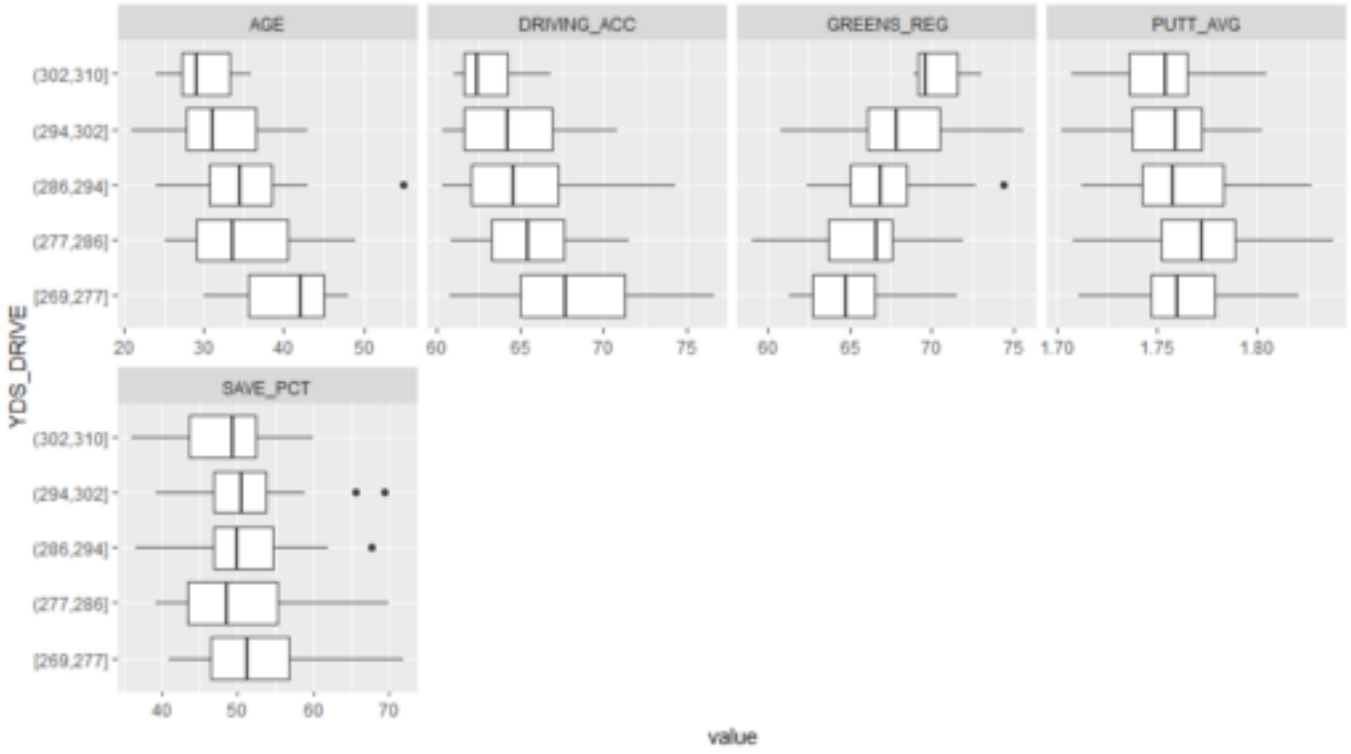
For example, the “GREENS_REG” variable contains values roughly between 55 and 75. Based on the bar chart, we see most golfers are hitting Greens about 65–70% of the time.

### *plot_boxplot()*

The boxplot (box and whisker diagram) displays the distribution of data for a variable. The box shows us a “five number summary” — minimum, first quartile, median, third quartile, and maximum.

The *plot_boxplot()* function below created 5 bins/partitions. We’ll focus first on the Yards-per-Drive (YDS_DRIVE) variable.

```
plot_boxplot(df_2019_filtered, by = “YDS_DRIVE”)
```



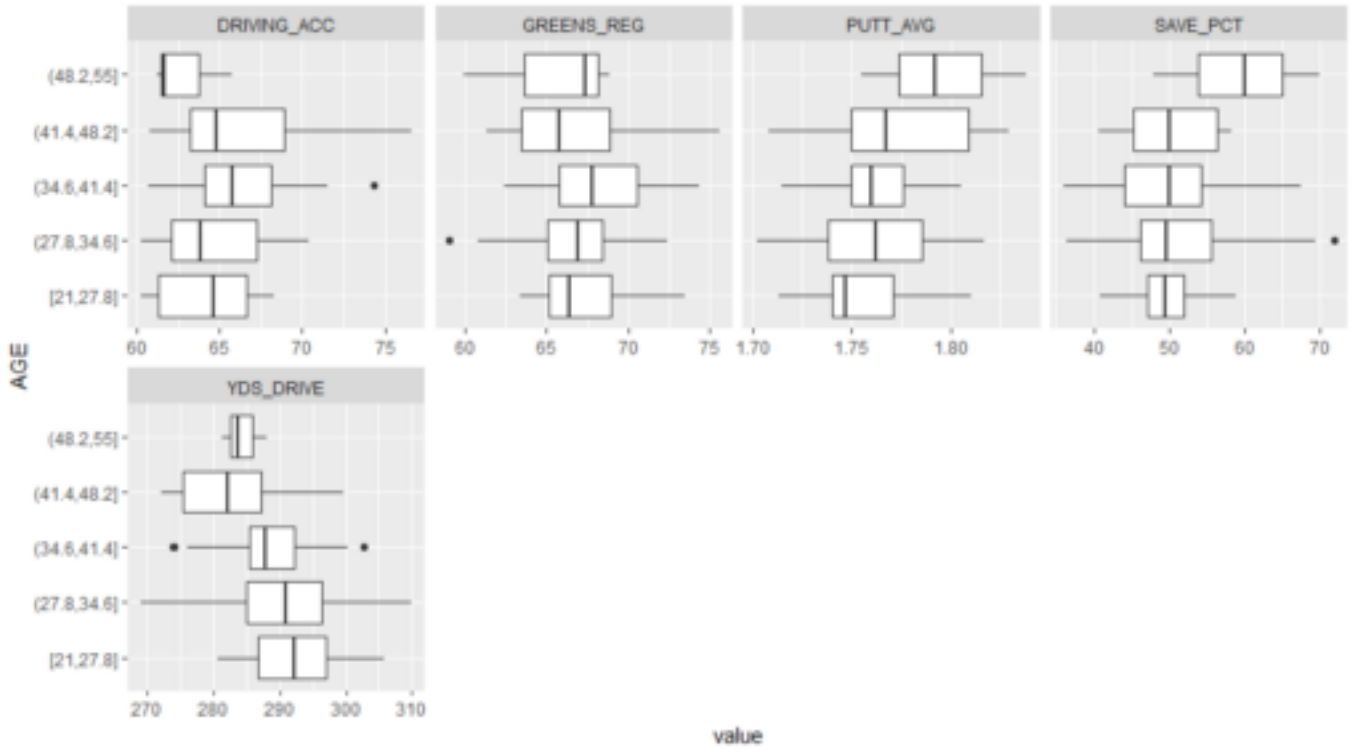
plot_boxplot() function

We can see some very interesting correlations look at the “AGE” variable compared with the “YDS per DRIVE”.

- The older guys don’t hit as far.
- There is one outlier. Someone in their mid-50’s is still hitting it quite far compared to the others in that age group.

Next, let’s do another boxplot from the “AGE” perspective.

```
plot_boxplot(df_2019_filtered, by = "AGE")
```




plot_boxplot() function

- The oldest group (48–55) in the upper left corner has a very low “Driving Accuracy” (DRIVING_ACC). I would expect the older players to hit the ball shorter, but more accurate… that is not true, the data does not lie.

- The older group (48–55) also struggles with putting. They have the highest average putts per hole (PUTT_AVG).

. . .

**Exploratory Data Analysis (EDA) — Part 3**

Let's take this analysis to another level.

***ggcorrplot()***

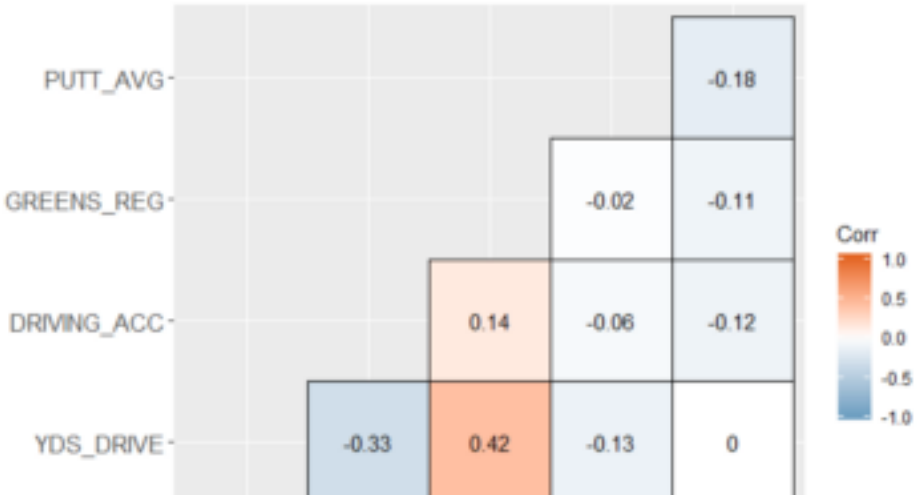
The functionality in this library gets closer to where a Data Scientist spends time.

The "ggcorrplot" provides us with a "heatmap" showing the significance (or lack of significance) between the relationships. A human cannot possibly stare at a spreadsheet and determine patterns/relationships between columns and rows of data.

Let's put the "ggcorrplot" library to work. Sometimes it's all about working smarter, not harder!

```
library(ggcorrplot)

ggcorrplot(corr, type = "lower", outline.col = "black",
 lab=TRUE,
 ggtheme = ggplot2::theme_gray,
 colors = c("#6D9EC1", "white", "#E46726"))
```




ggcorrplot() function

**1 = highly related ; 0 = no relationship ; -1 = inverse relationship**

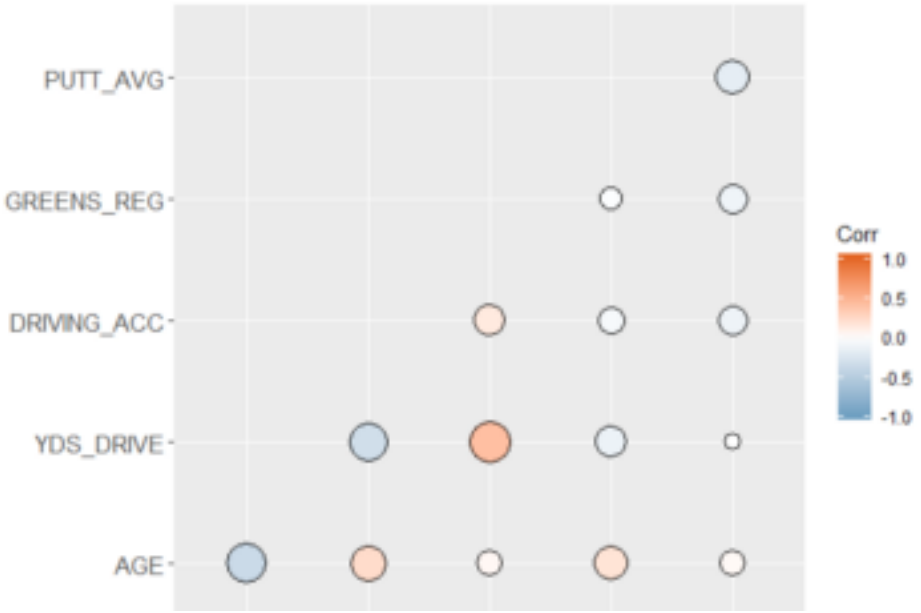
Running that function was much easier than staring at a spreadsheet, attempting to see relationships between rows and columns!

1 . The most significant relationship between our variables is “Yards per Drive” and “Greens Hit in Regulation”.

2 . On the opposite side, the most significant inverse relationship exists between “Age” and “Yard per Drive”.

If you like circles better than squares, below is the same data using the circle method.

```
ggcorrplot(corr, type = “lower”, outline.col = “black”,
 method=”circle”,
 ggtheme = ggplot2::theme_gray,
 colors = c(“#6D9EC1”, “white”, “#E46726”))
```




. . .

## Conclusion

Although the EDA process is critically important, it is only the beginning of a typical data analytics project lifecycle. Most likely an organization's valuable data will not be coming from a googlesheet, but will more likely be buried in disparate databases, or coming from a 3rd party vendor, or possibly even an IoT data stream.

Leveraging R and the EDA process will help pave the way toward a successful analytics project.

Jeff Griesemer is an Analytics Developer at Froedtert Health.

Data Science Eda R Analytics Data