Project 2 – Exploratory Data Analysis (EDA) of Two Data Sets
ALY 6000

Project Instructions

In this two-part project, you will explore core functions within the set of libraries known as the tidyverse.

Note: Utilize the file **project2_tests.R** with the code below to run a series of tests (not comprehensive) on your code. Any failed test signals that something is wrong with the results or that you have not utilized the specified variable names.

```
p_load(testthat)
#testthat::test_file("project2_tests.R")
```

Setting Up Your Project

Complete the following steps to create and organize your initial R project.

1. Create a new R Project called **Lastname_Project2**.
2. Create a new R Script and save it into the R folder of your project as **Project2_Script.R**.
3. Download the data set **2015.csv** from Canvas and save it into the project folder.
4. Download the data set **baseball.csv** from Canvas and save it into the project folder.
5. Download cheat sheets for the tidyr and dplyr packages for quick reference. You can access them from the help menu in RStudio.
6. Include the following boilerplate code at the top of your file to clear the environment each time you run your complete script.
```
cat("\014")  # clears console
rm(list = ls())  # clears global environment
try(dev.off(dev.list()["RStudioGD"]), silent = TRUE) # clears plots
try(p_unload(p_loaded(), character.only = TRUE), silent = TRUE) #
clears packages
options(scipen = 100) # disables scientific notion for entire R session
```
7. Include the following code at the top of your script (but below the boilerplate code) to load the pacman loader library. Then load the entire tidyverse.

```r
library(pacman)
p_load(tidyverse)
```

Assignment Part 1

Data can measure many things. Countries, for example, can be assessed against a variety of metrics. In addition to the gross domestic product (GDP) of a given country, researchers consider other data points in assessing the quality of life across the globe. To understand how data can be wrangled to measure freedom, trust, and other measures of human life, complete the following steps. The assignment displays the expected outcome after each step.

1. Read the data set **2015.csv** and store it in a variable called **data_2015**. You can test that you loaded it correctly with the code utilizing the head function below.

```r
head(data_2015)

# A tibble: 6 × 12
  Country  Region Happi…¹ Happi…² Stand…³ Econo…⁴ Family Healt…⁵
Freedom Trust…⁶
  <chr>    <chr>    <dbl>   <dbl>   <dbl>   <dbl>  <dbl>   <dbl>
<dbl>   <dbl>
1 Switzer… Weste…       1    7.59  0.0341    1.40   1.35   0.941
0.666   0.420
2 Iceland  Weste…       2    7.56  0.0488    1.30   1.40   0.948
0.629   0.141
3 Denmark  Weste…       3    7.53  0.0333    1.33   1.36   0.875
0.649   0.484
4 Norway   Weste…       4    7.52  0.0388    1.46   1.33   0.885
0.670   0.365
5 Canada   North…       5    7.43  0.0355    1.33   1.32   0.906
0.633   0.330
6 Finland  Weste…       6    7.41  0.0314    1.29   1.32   0.889
0.642   0.414
# … with 2 more variables: Generosity <dbl>, `Dystopia Residual` <dbl>,
and
#   abbreviated variable names ¹`Happiness Rank`, ²`Happiness Score`,
#   ³`Standard Error`, ⁴`Economy (GDP per Capita)`,
#   ⁵`Health (Life Expectancy)`, ⁶`Trust (Government Corruption)`
```

2. Use the function **names** to produce the column names for your data set.

```r
names(data_2015)

 [1] "Country"                       "Region"
 [3] "Happiness Rank"                "Happiness Score"
 [5] "Standard Error"                "Economy (GDP per Capita)"
 [7] "Family"                        "Health (Life Expectancy)"
```

```
 [9] "Freedom"                          "Trust (Government Corruption)"
[11] "Generosity"                       "Dystopia Residual"
```

3. Use the **view** function to view the data set in a separate tab.

4. Use the **glimpse** function to view your data set in another configuration.

```
glimpse(data_2015)
```

5. Use **p_load** to install the **janitor** package. Janitor has a function called **clean_names** that can be given a data frame to make the names more R friendly. Be sure to store the resulting converted data frame in a variable.

```
p_load(janitor)
data_2015 <- clean_names(data_2015)
data_2015
```

6. Select from the data set the **country**, **region**, **happiness_score**, and **freedom columns**. Store this new table as **happy_df**.

```
# A tibble: 158 × 4
   country     region                    happiness_score freedom
   <chr>       <chr>                               <dbl>   <dbl>
 1 Switzerland Western Europe                       7.59   0.666
 2 Iceland     Western Europe                       7.56   0.629
 3 Denmark     Western Europe                       7.53   0.649
 4 Norway      Western Europe                       7.52   0.670
 5 Canada      North America                        7.43   0.633
 6 Finland     Western Europe                       7.41   0.642
 7 Netherlands Western Europe                       7.38   0.616
 8 Sweden      Western Europe                       7.36   0.660
 9 New Zealand Australia and New Zealand            7.29   0.639
10 Australia   Australia and New Zealand            7.28   0.651
# … with 148 more rows
```

7. Slice the first 10 rows from **happy_df** and store it as **top_ten_df**.

```
# A tibble: 10 × 4
   country     region                    happiness_score freedom
   <chr>       <chr>                               <dbl>   <dbl>
 1 Switzerland Western Europe                       7.59   0.666
 2 Iceland     Western Europe                       7.56   0.629
 3 Denmark     Western Europe                       7.53   0.649
 4 Norway      Western Europe                       7.52   0.670
 5 Canada      North America                        7.43   0.633
 6 Finland     Western Europe                       7.41   0.642
 7 Netherlands Western Europe                       7.38   0.616
 8 Sweden      Western Europe                       7.36   0.660
 9 New Zealand Australia and New Zealand            7.29   0.639
10 Australia   Australia and New Zealand            7.28   0.651
```

8. From **happy_df** filter the table for freedom values under 0.20. Store this new table as **no_freedom_df.**

```
# A tibble: 12 × 4
   country                region
happiness_sc…¹ freedom
   <chr>                  <chr>
<dbl>   <dbl>
 1 Pakistan               Southern Asia
5.19  0.121
 2 Montenegro             Central and Eastern Europe
5.19  0.183
 3 Bosnia and Herzegovina Central and Eastern Europe
4.95  0.0924
 4 Greece                 Western Europe
4.86  0.0770
 5 Iraq                   Middle East and Northern Africa
4.68  0
 6 Sudan                  Sub-Saharan Africa
4.55  0.101
 7 Armenia                Central and Eastern Europe
4.35  0.198
 8 Egypt                  Middle East and Northern Africa
4.19  0.173
 9 Angola                 Sub-Saharan Africa
4.03  0.104
10 Madagascar             Sub-Saharan Africa
3.68  0.192
11 Syria                  Middle East and Northern Africa
3.01  0.157
12 Burundi                Sub-Saharan Africa
2.90  0.118
# … with abbreviated variable name ¹happiness_score
```

9. Arrange the values in **happy_df** in descending order by their freedom values. Store this new table as **best_freedom_df**.

```
# A tibble: 158 × 4
   country              region                          happiness_score
freedom
   <chr>                <chr>                                     <dbl>
<dbl>
 1 Norway               Western Europe                             7.52
0.670
 2 Switzerland          Western Europe                             7.59
0.666
 3 Cambodia             Southeastern Asia                          3.82
0.662
 4 Sweden               Western Europe                             7.36
0.660
```

```
 5 Uzbekistan           Central and Eastern Europe                          6.00
0.658
 6 Australia            Australia and New Zealand                           7.28
0.651
 7 Denmark              Western Europe                                      7.53
0.649
 8 Finland              Western Europe                                      7.41
0.642
 9 United Arab Emirates Middle East and Northern Africa                     6.90
0.642
10 Qatar                Middle East and Northern Africa                     6.61
0.640
# … with 148 more rows
```

10. Create a new column with **mutate** in **data_2015** called **gff_stat.** For each row, the **gff_stat** is the sum of the family, freedom, and generosity values. Store the resulting table right in the **data_2015** variable.

```
# A tibble: 158 × 13
   country region happi…¹ happi…² stand…³ econo…⁴ family healt…⁵
freedom trust…⁶
   <chr>   <chr>    <dbl>   <dbl>   <dbl>   <dbl>  <dbl>   <dbl>
<dbl>   <dbl>
 1 Switze… Weste…       1    7.59  0.0341    1.40   1.35   0.941
0.666   0.420
 2 Iceland Weste…       2    7.56  0.0488    1.30   1.40   0.948
0.629   0.141
 3 Denmark Weste…       3    7.53  0.0333    1.33   1.36   0.875
0.649   0.484
 4 Norway  Weste…       4    7.52  0.0388    1.46   1.33   0.885
0.670   0.365
 5 Canada  North…       5    7.43  0.0355    1.33   1.32   0.906
0.633   0.330
 6 Finland Weste…       6    7.41  0.0314    1.29   1.32   0.889
0.642   0.414
 7 Nether… Weste…       7    7.38  0.0280    1.33   1.28   0.893
0.616   0.318
 8 Sweden  Weste…       8    7.36  0.0316    1.33   1.29   0.911
0.660   0.438
 9 New Ze… Austr…       9    7.29  0.0337    1.25   1.32   0.908
0.639   0.429
10 Austra… Austr…      10    7.28  0.0408    1.33   1.31   0.932
0.651   0.356
# … with 148 more rows, 3 more variables: generosity <dbl>,
#   dystopia_residual <dbl>, gff_stat <dbl>, and abbreviated variable
names
#   ¹happiness_rank, ²happiness_score, ³standard_error,
#   ⁴economy_gdp_per_capita, ⁵health_life_expectancy,
#   ⁶trust_government_corruption
```

11. Summarize the **happy_df** data set. Your summary should contain the **mean** happiness_score in a column called **mean_happiness,** the **max** happiness_score in a column called **max_happiness**, the **mean** freedom in a column called **mean_freedom**, and the **max** freedom in a column called **max_freedom.** Store the resulting table as **happy_summary**.

```
# A tibble: 1 × 4
  mean_happiness max_happiness mean_freedom max_freedom
           <dbl>         <dbl>        <dbl>       <dbl>
1           5.38          7.59        0.429       0.670
```

12. Group the **happy_df** data set by region. Run a summary that provides the number of countries in each region in a column called **country_count**, the **mean** happiness for each region in a column called **mean_happiness**, and the **mean** freedom of each region in a column called **mean_freedom.** Store your resulting table in a variable called **regional_stats_df.**

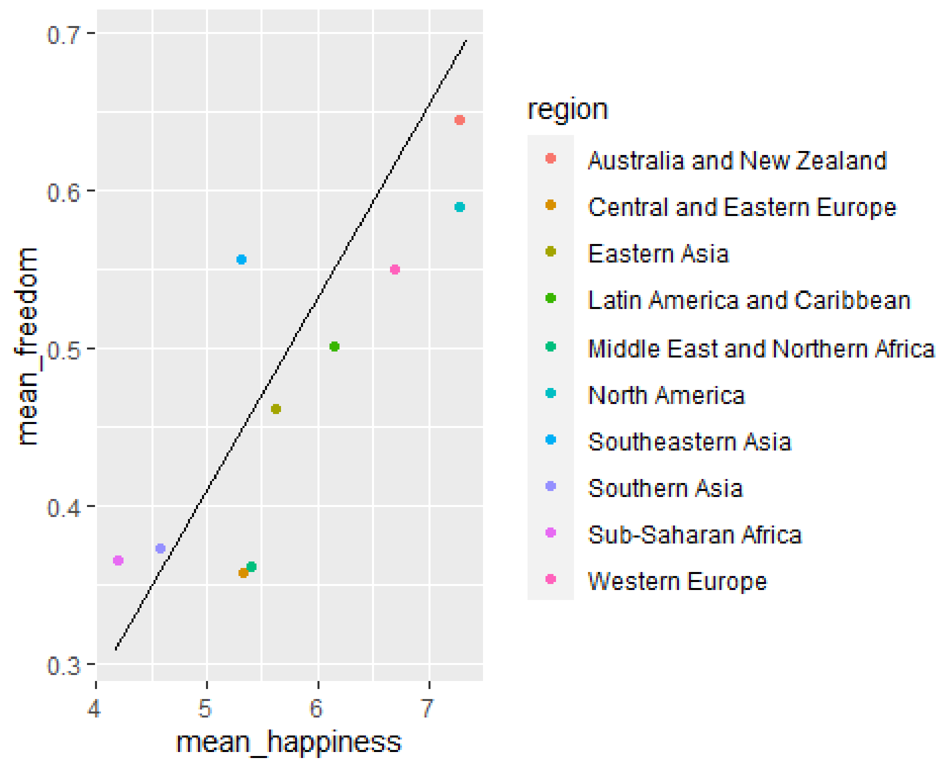
```
# A tinble: 10 × 4
   region                          country_count mean_happiness
mean_freedom
   <chr>                                   <int>          <dbl>
<dbl>
 1 Australia and New Zealand                   2           7.28
0.645
 2 Central and Eastern Europe                 29           5.33
0.358
 3 Eastern Asia                                6           5.63
0.462
 4 Latin America and Caribbean                22           6.14
0.502
 5 Middle East and Northern Africa            20           5.41
0.362
 6 North America                               2           7.27
0.590
 7 Southeastern Asia                           9           5.32
0.557
 8 Southern Asia                               7           4.58
0.373
 9 Sub-Saharan Africa                         40           4.20
0.366
10 Western Europe                             21           6.69
0.550
```

13. Compare the average gdp per capita of the ten *least* happy Western European countries with the ten *happiest* Sub-Saharan African countries. For testing, you can store the resulting data.frame or table as **gdp_df**.

```
# A tibble: 1 × 2
  europe_gdp africa_gdp
```

```
        <dbl>      <dbl>
1        1.23      0.523
```

14. From your **regional_stats_df,** create a scatterplot of mean_happiness vs. mean_freedom. Draw a line segment from the smallest of these values to the largest.



Assignment Part 2

In Part Two of this R Project, you will analyze a data set of batting statistics from the 1986 Major League Baseball season. You will then draft a brief executive summary that corresponds to the data analysis. Details for both the data analysis and executive summary follow below.

1. Download the **baseball.csv** data set. data set that represents batting statistics from the 1986 Major League Baseball season. Read this data set in a **variable** called **baseball.**
2. Spend time with the data using various exploration functions to get a general feel for what you are working with. For more information on this data set and its various columns, see Baseball Reference's [1986 Major League Standard Batting](#).
3. Use the ***class*** function to discover the type of class represented in the **baseball** data set.

```
[1] "spec_tbl_df" "tbl_df"      "tbl"         "data.frame"
```

4. For each age, compute the following: the number of people at that age, the average number of home runs (HRs), the average number of hits, and the average number of runs scored. Store these computations in a variable called **age_stats_df.**

```
# A tibble: 24 × 5
     Age Count    HR     H     R
   <dbl> <int> <dbl> <dbl> <dbl>
 1    20     5  3.4   24    11.8
 2    21    18  3.28  22.4  14.1
 3    22    38  2.32  28.5  14.3
 4    23    38  3.74  36.7  20.0
 5    24    65  4.37  42.6  22.1
 6    25    94  4.5   42.8  21.0
 7    26    86  5.70  49.8  24.9
 8    27    63  4.62  52.0  27.1
 9    28    64  3.94  49.3  25.8
10    29    53  5.26  52.6  26.4
# … with 14 more rows
```

5. Remove (**filter)** from **baseball** any player with 0 at bats (AB). Store the result in **baseball.**

```
# A tibble: 726 × 16
   Last  First   Age     G    PA    AB     R     H  `2B`  `3B`    HR
RBI    SB
   <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
<dbl> <dbl>
 1 Acker Jim      27    21    28    28     1     3     1     0     0
0     0
 2 Addu… Jim      26     3    13    11     2     1     1     0     0
0     0
 3 Agua… Luis     27    62   146   133    17    28     6     1     4
13     1
 4 Agui… Rick     24    32    57    51     4     8     0     0     2
6     0
 5 Aldr… Mike     25    84   256   216    27    54    18     3     2
25     1
 6 Alex… Doyle    35    18    45    38     2     8     1     0     0
5     0
 7 Alla… Andy     24   101   324   293    30    66     7     3     1
29    10
 8 Almon Bill     33   102   230   196    29    43     7     2     7
27    11
 9 Amel… Ed       27     8    11    11     0     1     0     0     0
0     0
10 Ande… Larry    33    48     7     6     0     0     0     0     0
0     0
# … with 716 more rows, and 3 more variables: CS <dbl>, BB <dbl>, SO
<dbl>
```

6. Add a new column batting average called **BA.** Batting average is computed by the number of hits (H) divided by the number of at bats (AB). Store the result in **baseball.**

```
# A tibble: 726 × 17
   Last  First   Age     G    PA    AB     R     H  `2B`  `3B`    HR
RBI    SB
   <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
<dbl> <dbl>
 1 Acker Jim      27    21    28    28     1     3     1     0     0
0     0
 2 Addu… Jim      26     3    13    11     2     1     1     0     0
0     0
 3 Agua… Luis     27    62   146   133    17    28     6     1     4
13     1
 4 Agui… Rick     24    32    57    51     4     8     0     0     2
6     0
 5 Aldr… Mike     25    84   256   216    27    54    18     3     2
25     1
 6 Alex… Doyle    35    18    45    38     2     8     1     0     0
5     0
 7 Alla… Andy     24   101   324   293    30    66     7     3     1
29    10
 8 Almon Bill     33   102   230   196    29    43     7     2     7
27    11
 9 Amel… Ed       27     8    11    11     0     1     0     0     0
0     0
10 Ande… Larry    33    48     7     6     0     0     0     0     0
0     0
# … with 716 more rows, and 4 more variables: CS <dbl>, BB <dbl>, SO
<dbl>,
#   BA <dbl>
```

7. Modify your new BA column so that the value is **rounded** to three (3) decimal places.

```
# A tibble: 726 × 17
   Last  First   Age     G    PA    AB     R     H  `2B`  `3B`    HR
RBI    SB
   <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
<dbl> <dbl>
 1 Acker Jim      27    21    28    28     1     3     1     0     0
0     0
 2 Addu… Jim      26     3    13    11     2     1     1     0     0
0     0
 3 Agua… Luis     27    62   146   133    17    28     6     1     4
13     1
 4 Agui… Rick     24    32    57    51     4     8     0     0     2
6     0
 5 Aldr… Mike     25    84   256   216    27    54    18     3     2
25     1
```

```
 6 Alex… Doyle    35    18    45    38     2     8     1     0     0
5     0
 7 Alla… Andy     24   101   324   293    30    66     7     3     1
29    10
 8 Almon Bill     33   102   230   196    29    43     7     2     7
27    11
 9 Amel… Ed       27     8    11    11     0     1     0     0     0
0     0
10 Ande… Larry    33    48     7     6     0     0     0     0     0
0     0
# … with 716 more rows, and 4 more variables: CS <dbl>, BB <dbl>, SO
<dbl>,
#   BA <dbl>
```

8. On-base percentage (OBP) is arguably a better statistic than batting average. Create a column called **OBP** that computes this stat as (H + BB) / (AB + BB). Store the result in **baseball.**

```
# A tibble: 726 × 18
   Last  First   Age     G    PA    AB     R     H  `2B`  `3B`    HR
RBI    SB
   <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
<dbl> <dbl>
 1 Acker Jim      27    21    28    28     1     3     1     0     0
0     0
 2 Addu… Jim      26     3    13    11     2     1     1     0     0
0     0
 3 Agua… Luis     27    62   146   133    17    28     6     1     4
13     1
 4 Agui… Rick     24    32    57    51     4     8     0     0     2
6     0
 5 Aldr… Mike     25    84   256   216    27    54    18     3     2
25     1
 6 Alex… Doyle    35    18    45    38     2     8     1     0     0
5     0
 7 Alla… Andy     24   101   324   293    30    66     7     3     1
29    10
 8 Almon Bill     33   102   230   196    29    43     7     2     7
27    11
 9 Amel… Ed       27     8    11    11     0     1     0     0     0
0     0
10 Ande… Larry    33    48     7     6     0     0     0     0     0
0     0
# … with 716 more rows, and 5 more variables: CS <dbl>, BB <dbl>, SO
<dbl>,
#   BA <dbl>, OBP <dbl>
```

9. Modify your new OBP column so that the value is **rounded** to three (3) decimal places.

```
# A tibble: 726 × 18
   Last  First   Age     G    PA    AB     R     H  `2B`  `3B`    HR
```

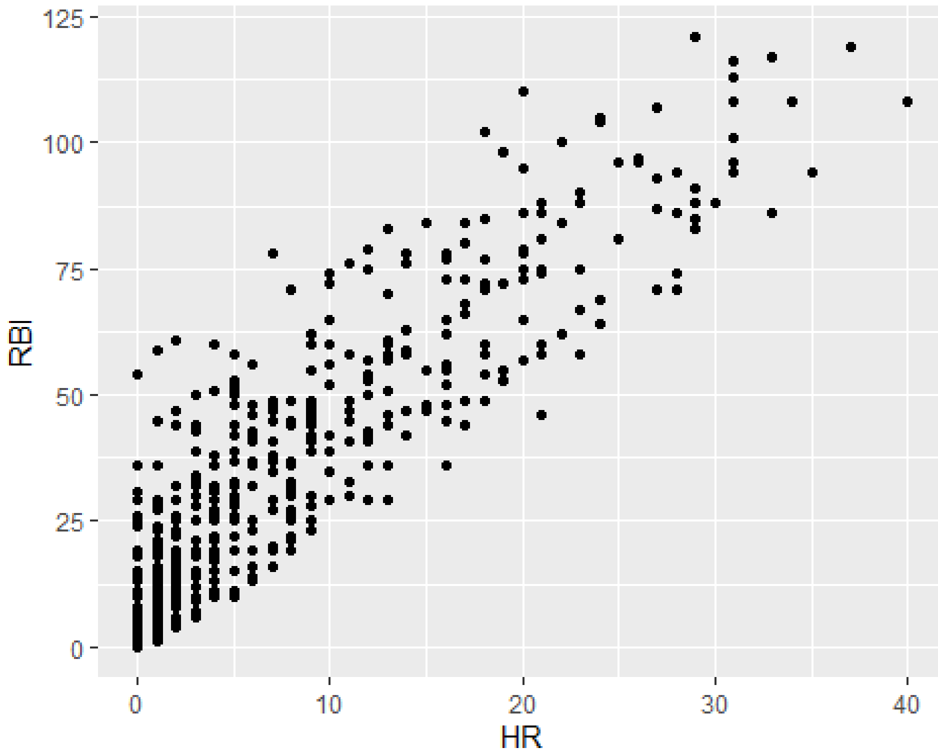
```
RBI    SB
   <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
<dbl> <dbl>
 1 Acker Jim      27    21    28    28     1     3     1     0     0
0     0
 2 Addu… Jim      26     3    13    11     2     1     1     0     0
0     0
 3 Agua… Luis     27    62   146   133    17    28     6     1     4
13     1
 4 Agui… Rick     24    32    57    51     4     8     0     0     2
6     0
 5 Aldr… Mike     25    84   256   216    27    54    18     3     2
25     1
 6 Alex… Doyle    35    18    45    38     2     8     1     0     0
5     0
 7 Alla… Andy     24   101   324   293    30    66     7     3     1
29    10
 8 Almon Bill     33   102   230   196    29    43     7     2     7
27    11
 9 Amel… Ed       27     8    11    11     0     1     0     0     0
0     0
10 Ande… Larry    33    48     7     6     0     0     0     0     0
0     0
# … with 716 more rows, and 5 more variables: CS <dbl>, BB <dbl>, SO
<dbl>,
#   BA <dbl>, OBP <dbl>
```

10. Determine the 10 players who struck out the most this season. Store these results as **strikeout_artist.**

```
# A tibble: 10 × 18
   Last  First   Age     G    PA    AB     R     H  `2B`  `3B`    HR
RBI    SB
   <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
<dbl> <dbl>
 1 Inca… Pete     22   153   606   540    82   135    21     2    30
88     3
 2 Deer  Rob      25   134   546   466    75   108    17     3    33
86     5
 3 Cans… Jose     21   157   682   600    85   144    29     1    33
117    15
 4 Pres… Jim      24   155   660   616    83   163    33     4    27
107     0
 5 Tart… Danny    23   137   578   511    76   138    25     6    25
96     4
 6 Balb… Steve    29   138   562   512    54   117    25     1    29
88     0
 7 Barf… Jesse    26   158   671   589   107   170    35     2    40
108     8
 8 Samu… Juan     25   145   633   591    90   157    36    12    16
78    42
```

```
 9 Murp… Dale      30   160   692   614    89   163    29     7    29
83     7
10 Stra… Darr…     24   136   562   475    76   123    27     5    27
93    28
# … with 5 more variables: CS <dbl>, BB <dbl>, SO <dbl>, BA <dbl>, OBP
<dbl>
```

11. Using a scatterplot (**geom_point**), plot the number of home runs (HRs) (the x-axis), versus the number of RBIs (the y-axis) per player.

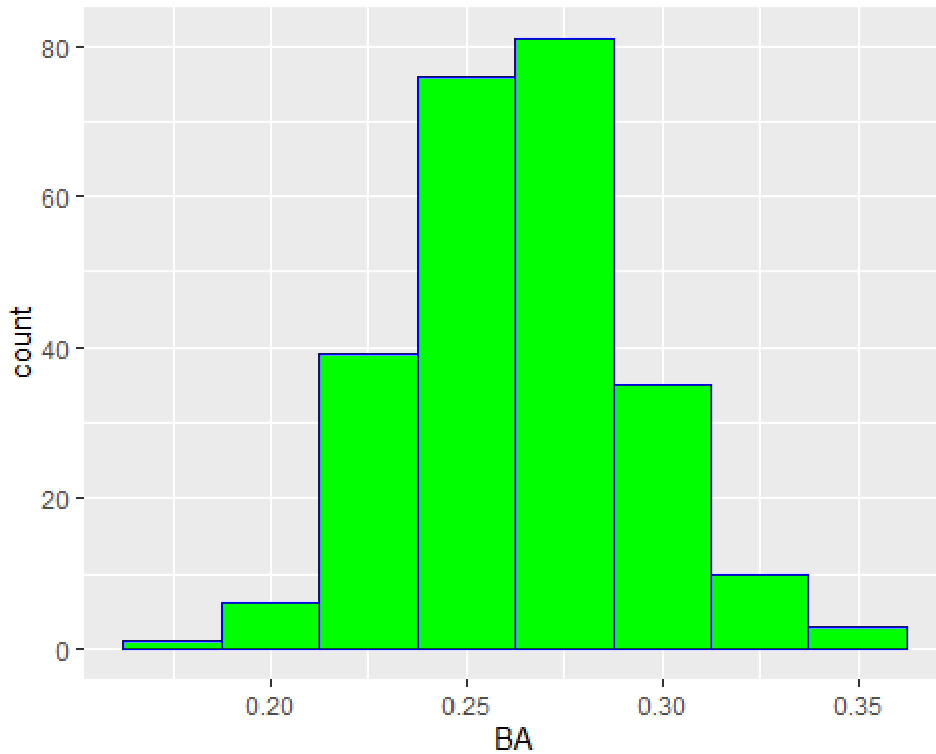


12. To be eligible for end-of-season awards, a player must have either at least 300 at bats or appear in at least 100 games. Keep only the players who are eligible to be considered and store them in a variable called **eligible_df.**

```
# A tibble: 251 × 18
   Last  First  Age     G    PA    AB     R     H  `2B`  `3B`    HR
RBI    SB
   <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
<dbl> <dbl>
 1 Alla… Andy      24   101   324   293    30    66     7     3     1
29    10
 2 Almon Bill      33   102   230   196    29    43     7     2     7
27    11
 3 Armas Tony      32   121   453   425    40   112    21     4    11
58     0
 4 Ashby Alan      34   120   361   315    24    81    15     0     7
38     1
 5 Back… Wally     26   124   440   387    67   124    18     2     1
```

```
27     13
  6 Bain… Haro…     27    145    618    570     72    169     29      2     21
88      2
  7 Balb… Steve     29    138    562    512     54    117     25      1     29
88      0
  8 Barf… Jesse     26    158    671    589    107    170     35      2     40
108      8
  9 Barr… Marty     28    158    713    625     94    179     39      4      4
60     15
10 Bass   Kevin     27    157    640    591     83    184     33      5     20
79     22
# … with 241 more rows, and 5 more variables: CS <dbl>, BB <dbl>, SO
<dbl>,
#   BA <dbl>, OBP <dbl>
```

13. For eligible players, create a histogram of batting average. Use a binwidth of .025 in your graph. The graph should be drawn in blue and filled in green.



14. Use the following code to create a ranking column of **eligible** players with regard to home runs (HRs). Store the result in **eligible_df.**

```
eligible_df <- eligible_df |>
            mutate(RankHR =rank(-1 * HR, ties.method = "min"))
eligible_df

# A table: 251 × 19
   Last  First   Age     G    PA    AB     R     H  `2B`  `3B`    HR
RBI    SB
```

```
   <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
<dbl> <dbl>
 1 Alla… Andy     24   101   324   293    30    66     7     3     1
29    10
 2 Almon Bill     33   102   230   196    29    43     7     2     7
27    11
 3 Armas Tony     32   121   453   425    40   112    21     4    11
58     0
 4 Ashby Alan     34   120   361   315    24    81    15     0     7
38     1
 5 Back… Wally    26   124   440   387    67   124    18     2     1
27    13
 6 Bain… Haro…    27   145   618   570    72   169    29     2    21
88     2
 7 Balb… Steve    29   138   562   512    54   117    25     1    29
88     0
 8 Barf… Jesse    26   158   671   589   107   170    35     2    40
108     8
 9 Barr… Marty    28   158   713   625    94   179    39     4     4
60    15
10 Bass  Kevin    27   157   640   591    83   184    33     5    20
79    22
# … with 241 more rows, and 6 more variables: CS <dbl>, BB <dbl>, SO
<dbl>,
#   BA <dbl>, OBP <dbl>, RankHR <int>
```

15. Repeat the prior step to create rankings for both runs batted in (RBI) and on-base percentage (OBP). Store the result in **eligible_df.**

```
# A tibble: 251 × 21
   Last  First   Age     G    PA    AB     R     H  `2B`  `3B`    HR
RBI    SB
   <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
<dbl> <dbl>
 1 Alla… Andy     24   101   324   293    30    66     7     3     1
29    10
 2 Almon Bill     33   102   230   196    29    43     7     2     7
27    11
 3 Armas Tony     32   121   453   425    40   112    21     4    11
58     0
 4 Ashby Alan     34   120   361   315    24    81    15     0     7
38     1
 5 Back… Wally    26   124   440   387    67   124    18     2     1
27    13
 6 Bain… Haro…    27   145   618   570    72   169    29     2    21
88     2
 7 Balb… Steve    29   138   562   512    54   117    25     1    29
88     0
 8 Barf… Jesse    26   158   671   589   107   170    35     2    40
108     8
```

```
 9 Barr… Marty    28   158   713   625    94   179    39     4     4
60    15
10 Bass  Kevin    27   157   640   591    83   184    33     5    20
79    22
# … with 241 more rows, and 8 more variables: CS <dbl>, BB <dbl>, SO
<dbl>,
#   BA <dbl>, OBP <dbl>, RankHR <int>, RankRBI <int>, RankOBP <int>
```

16. Create a TotalRank column that is the sum of the prior three (3) ranks. If a player was ranked first in HR, RBI, and OBP, then their total rank would be 3. Store the result in **eligible_df**.

```
# A tibble: 251 × 22
   Last  First   Age     G    PA    AB     R     H  `2B`  `3B`    HR
RBI    SB
   <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
<dbl> <dbl>
 1 Alla… Andy     24   101   324   293    30    66     7     3     1
29    10
 2 Almon Bill     33   102   230   196    29    43     7     2     7
27    11
 3 Armas Tony     32   121   453   425    40   112    21     4    11
58     0
 4 Ashby Alan     34   120   361   315    24    81    15     0     7
38     1
 5 Back… Wally    26   124   440   387    67   124    18     2     1
27    13
 6 Bain… Haro…    27   145   618   570    72   169    29     2    21
88     2
 7 Balb… Steve    29   138   562   512    54   117    25     1    29
88     0
 8 Barf… Jesse    26   158   671   589   107   170    35     2    40
108     8
 9 Barr… Marty    28   158   713   625    94   179    39     4     4
60    15
10 Bass  Kevin    27   157   640   591    83   184    33     5    20
79    22
# … with 241 more rows, and 9 more variables: CS <dbl>, BB <dbl>, SO
<dbl>,
#   BA <dbl>, OBP <dbl>, RankHR <int>, RankRBI <int>, RankOBP <int>,
#   TotalRank <int>
```

17. Arrange the data in ascending order by TotalRank and store the twenty (20) lowest TotalRank scores in a variable called **mvp_candidates.**

```
# A tibble: 20 × 22
   Last  First   Age     G    PA    AB     R     H  `2B`  `3B`    HR
RBI    SB
   <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
<dbl> <dbl>
```

```
 1 Matt… Don       25   162   742   677   117   238    53     2    31
113     0
 2 Schm… Mike      36   160   657   552    97   160    29     1    37
119     1
 3 Barf… Jesse     26   158   671   589   107   170    35     2    40
108     8
 4 Evans Dwig…     34   152   640   529    86   137    33     2    26
97     3
 5 Puck… Kirby     26   161   723   680   119   223    37     6    31
96    20
 6 Rice  Jim       33   157   693   618    98   200    39     2    20
110     0
 7 O'Br… Pete      28   156   641   551    86   160    23     3    23
90     4
 8 Bell  Geor…     26   159   690   641   101   198    38     6    31
108     7
 9 McRe… Kevin     26   158   641   560    89   161    31     6    26
96     8
10 Gibs… Kirk      29   119   521   441    84   118    11     2    28
86    34
11 Gaet… Gary      27   157   661   596    91   171    34     1    34
108    14
12 Hayes Von       27   158   690   610   107   186    46     2    19
98    24
13 Down… Brian     35   152   631   513    90   137    27     4    20
95     4
14 Stra… Darr…     24   136   562   475    76   123    27     5    27
93    28
15 Evans Darr…     39   151   601   507    78   122    15     0    29
85     3
16 Hrbek Kent      26   149   634   550    85   147    27     1    29
91     2
17 Davis Eric      24   132   487   415    97   115    15     3    27
71    80
18 Winf… Dave      34   154   652   565    90   148    31     5    24
104     6
19 Parr… Larry     32   129   524   464    67   128    22     1    28
94     3
20 Murr… Eddie     30   137   578   495    61   151    25     1    17
84     3
# … with 9 more variables: CS <dbl>, BB <dbl>, SO <dbl>, BA <dbl>, OBP
<dbl>,
#   RankHR <int>, RankRBI <int>, RankOBP <int>, TotalRank <int>
```

18. Create a variable called **mvp_candidates_abbreviated** with the First, Last, RankHR, RankRBI, and RankOBP selected from **mvp_candidates**.

```
# A tibble: 20 × 6
   First   Last       RankHR RankRBI RankOBP TotalRank
   <chr>   <chr>       <int>   <int>   <int>     <int>
```

```
 1 Don     Mattingly       7        5        8         20
 2 Mike    Schmidt         2        2       16         20
 3 Jesse   Barfield        1        7       45         53
 4 Dwight  Evans          27       17       30         74
 5 Kirby   Puckett         7       18       50         75
 6 Jim     Rice           52        6       18         76
 7 Pete    O'Brien        36       28       17         81
 8 George  Bell            7        7       74         88
 9 Kevin   McReynolds     27       18       45         90
10 Kirk    Gibson         19       34       41         94
11 Gary    Gaetti          4        7       86         97
12 Von     Hayes          61       16       21         98
13 Brian   Downing        52       22       28        102
14 Darryl  Strawberry     23       26       57        106
15 Darrell Evans          14       38       57        109
16 Kent    Hrbek          14       27       71        112
17 Eric    Davis          23       71       22        116
18 Dave    Winfield       32       12       74        118
19 Larry   Parrish        19       23       77        119
20 Eddie   Murray         74       40        6        120
```

19. Make a recommendation for the league most valuable player (MVP). Keep in mind that the dataset completely ignores pitchers. You can decide whether a pitcher should be eligible for the MVP. Base your decision on the data you have analyzed. You may choose to do additional analysis at your discretion. You should produce a concise, written executive summary that, in addition to the title page and citations, contains an introduction, presentation of written key findings supported by visualizations, and a conclusion that contains your recommendations as supported by the data. Your executive summary should adhere to basic APA guidelines.

Submitting to Canvas

When you are satisfied with your solution, take the following steps:

1. **Remove** any lines in your code with “include.packages” or “install.packages.”
2. **Remove** any lines in your code that use the **view** function.
3. Submit two (2) files under the appropriate assignment in Canvas:
    1. Your R script named **Project2_Script.R**.
    2. A PDF file of your report titled **Lastname_Project2_Report.pdf**.

   In addition to the problem descriptions and results your report should contain the following information formatted as specified below:

Title Page
Include your name, assignment title, and submission date

Introduction and Key Findings
Include an overview of the assignment and any findings

Conclusion/Recommendations
Include evidence-based recommendations and visualizations or direct presentation of tabular data

Works Cited
Include all sources, including YouTube videos, instruction materials, Google search results, and texts that informed your study of statistics and R

Your report should be as concise as possible while maintaining fluency. Your key findings will be strongest if supported by visualizations or direct presentation of tabular data.

Your summary must adhere to APA guidelines, including page numbers on each page (including the title page) in the upper right corner. See the following examples for title pages, citations, and general APA formatting.

Congratulations on completing your second project!