Project – 2

Introductions to Analytics

Prepared by: Niharika Yarlagadda

Date: January 17th, 2024

Introduction:

Two datasets are examined and analysed in this project. The first dataset, that provides information on characteristics including happiness scores, freedom, economy, and health, focusses on happiness rankings from nations worldwide in 2015. Using data manipulation techniques, we obtain various statistics and summaries, such as freedom levels and regional happiness.

The second dataset examines batting statistics from the 1986 Major League Baseball season. Here, we clean and explore the data to calculate key metrics like batting average and on-base percentage. The last assignment entails choosing the most valuable player (MVP) based on performance statistics and justifying this pick with data-driven research.

Assignment 1

1. In the first task, analysis of the 2015 World Happiness dataset, we first loaded the data from the 2015.csv file into a variable named data_2015**.**

   To confirm that the data was loaded correctly, we used the head () function, which displays the first few rows of the dataset.

```
> data_2015 <-
  read.csv("C:/users/nihar/OneDrive/Desktop/Niharika_Project2/2015.csv"
> head(data_2015)

      Country          Region Happiness.Rank Happiness.Score Standard.Error
1 Switzerland Western Europe               1           7.587        0.03411
2     Iceland Western Europe               2           7.561        0.04884
3     Denmark Western Europe               3           7.527        0.03328
4      Norway Western Europe               4           7.522        0.03880
5      Canada  North America               5           7.427        0.03553
6     Finland Western Europe               6           7.406        0.03140
  Economy..GDP.per.Capita.  Family Health..Life.Expectancy. Freedom
1                  1.39651 1.34951                   0.94143 0.66557
2                  1.30232 1.40223                   0.94784 0.62877
3                  1.32548 1.36058                   0.87464 0.64938
4                  1.45900 1.33095                   0.88521 0.66973
5                  1.32629 1.32261                   0.90563 0.63297
6                  1.29025 1.31826                   0.88911 0.64169
  Trust..Government.Corruption. Generosity Dystopia.Residual
1                       0.41978    0.29678           2.51738
2                       0.14145    0.43630           2.70201
3                       0.48357    0.34139           2.49204
4                       0.36503    0.34699           2.46531
5                       0.32957    0.45811           2.45176
6                       0.41372    0.23351           2.61955
```

2. I used the **names ()** function to list all the column names in the dataset. This helps us see what each column represents, like country names, happiness scores, and factors such as economy and health.

```
> 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. In this task, **View()** function was used to open the dataset in a separate tab, allowing us to see the data in a spreadsheet-like format.

   >library(tidyverse)
   > View(data_2015)

4. Here, **glimpse()** function was used to view the dataset in a compact format. It provides an overview of the data, including the column names, data types, and a preview of the first few values in each column. This helps us quickly understand the data

>library(dplyr)
> glimpse(data_2015)

```
Rows: 158
Columns: 12
$ Country                      <chr> "Switzerland", "Iceland", "Denmark", "Norw…
$ Region                       <chr> "Western Europe", "Western Europe", "Weste…
$ Happiness.Rank               <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,…
$ Happiness.Score              <dbl> 7.587, 7.561, 7.527, 7.522, 7.427, 7.406, …
$ Standard.Error               <dbl> 0.03411, 0.04884, 0.03328, 0.03880, 0.0355…
$ Economy..GDP.per.Capita.     <dbl> 1.39651, 1.30232, 1.32548, 1.45900, 1.3262…
$ Family                       <dbl> 1.34951, 1.40223, 1.36058, 1.33095, 1.3226…
$ Health..Life.Expectancy.     <dbl> 0.94143, 0.94784, 0.87464, 0.88521, 0.9056…
$ Freedom                      <dbl> 0.66557, 0.62877, 0.64938, 0.66973, 0.6329…
$ Trust..Government.Corruption. <dbl> 0.41978, 0.14145, 0.48357, 0.36503, 0.3295…
$ Generosity                   <dbl> 0.29678, 0.43630, 0.34139, 0.34699, 0.4581…
$ Dystopia.Residual            <dbl> 2.51738, 2.70201, 2.49204, 2.46531, 2.4517…
```

5. First, I Installed **janitor** package and loaded. The **clean_names()** function was then applied to the dataset to make the column names more R-friendly by converting them to a consistent format. The cleaned data frame was stored in a new variable for further analysis.

>library(janitor)

```
> data_2015 <- clean_names(data_2015)
> data_2015
               country                          region happiness_rank
1          Switzerland                  Western Europe              1
2              Iceland                  Western Europe              2
3              Denmark                  Western Europe              3
4               Norway                  Western Europe              4
5               Canada                   North America              5
6              Finland                  Western Europe              6
7          Netherlands                  Western Europe              7
8               Sweden                  Western Europe              8
9          New Zealand       Australia and New Zealand              9
```

6. I created a new dataset called **happy_df** by selecting the most relevant columns: **country**, **region**, **happiness_score**, and **freedom**. This focused approach helps to analyze the connections between a country's region, its happiness score, and the level of freedom enjoyed by its citizens.

   Here I used %>% operator, or pipe operator, is a powerful tool in R that simplifies code readability by allowing you to connect multiple functions together. It takes the output of one function and feeds it directly into the next, making it easier to perform complex data manipulations.

```
> library(dplyr)
> happy_df <- data_2015 %>%
    select(country,region, happiness_score,freedom)
>
> head(happy_df)
      country          region happiness_score freedom
1 Switzerland Western Europe           7.587 0.66557
2     Iceland Western Europe           7.561 0.62877
3     Denmark Western Europe           7.527 0.64938
4      Norway Western Europe           7.522 0.66973
5      Canada  North America           7.427 0.63297
6     Finland Western Europe           7.406 0.64169
```

7. In this task, I extracted the first ten rows from the **happy_df** dataset and stored this selection in a new variable called **top_ten_df**. This allows to focus on a smaller subset of the data, making it easier to review and analyze the top ten countries based on the selected criteria of happiness score and freedom.
   The slice() function is useful when we want to extract specific rows based on their position rather than conditions or values.

```
> top_ten_df <- happy_df %>%
    slice(1:10)
> print(top_ten_df)
       country                     region happiness_score freedom
1  Switzerland            Western Europe           7.587 0.66557
2      Iceland            Western Europe           7.561 0.62877
3      Denmark            Western Europe           7.527 0.64938
4       Norway            Western Europe           7.522 0.66973
5       Canada             North America           7.427 0.63297
6      Finland            Western Europe           7.406 0.64169
7  Netherlands            Western Europe           7.378 0.61576
8       Sweden            Western Europe           7.364 0.65980
9  New Zealand Australia and New Zealand           7.286 0.63938
10   Australia Australia and New Zealand           7.284 0.65124
```

8. Here, I filtered the **happy_df** dataset to focus on countries with freedom values under 0.20, creating a new dataset and stored the data called **no_freedom_df.**
   I used filter () function This to subset a data frame, returning only the rows that meet certain criteria.

```
> no_freedom_df <- happy_df%>%
    filter(freedom < 0.20)
> head(no_freedom_df)
                country                          region happiness_score freedom
1              Pakistan                      Southern Asia           5.194 0.12102
2            Montenegro       Central and Eastern Europe           5.192 0.18260
3 Bosnia and Herzegovina       Central and Eastern Europe           4.949 0.09245
4                Greece                  Western Europe           4.857 0.07699
5                  Iraq Middle East and Northern Africa           4.677 0.00000
6                 Sudan               Sub-Saharan Africa           4.550 0.10081
```

9. In this task, I sorted the happy_df dataset by freedom values in descending order, creating a new dataset called best_freedom_df. This makes it very simple to see which countries have the highest levels of freedom, helping us compare these freedom levels with their happiness scores.

   I used arrang() function comes from the dplyr package and is used to reorder rows of a data frame based on the values of one or more columns.
   desc () function indicates that the sorting is done in descending order for the **freedom** column

```
> best_freedom_df <- happy_df%>%
    arrange(desc(freedom))
> head(best_freedom_df)
      country                        region happiness_score freedom
1      Norway                Western Europe           7.522 0.66973
2 Switzerland                Western Europe           7.587 0.66557
3    Cambodia             Southeastern Asia           3.819 0.66246
4      Sweden                Western Europe           7.364 0.65980
5  Uzbekistan Central and Eastern Europe           6.003 0.65821
6   Australia  Australia and New Zealand           7.284 0.65124
```

10. In this task, I added a new column named gff_stat to the data_2015 dataset. This column calculates the total of the family**,** freedom, and generosity values for each entry.
    Here, modified dataset, which now includes this new column, is stored back into data_2015. I used mutate() function to add new variables or modify existing ones in the dataset.

```
> data_2015 <- data_2015 %>%
    mutate(gff_stat = family + freedom + generosity)
> head(data_2015)

      country         region happiness_rank happiness_score standard_error
1 Switzerland Western Europe              1           7.587        0.03411
2     Iceland Western Europe              2           7.561        0.04884
3     Denmark Western Europe              3           7.527        0.03328
4      Norway Western Europe              4           7.522        0.03880
5      Canada  North America              5           7.427        0.03553
6     Finland Western Europe              6           7.406        0.03140
  economy_gdp_per_capita  family health_life_expectancy freedom
1                1.39651 1.34951                0.94143 0.66557
2                1.30232 1.40223                0.94784 0.62877
3                1.32548 1.36058                0.87464 0.64938
```

```
4                1.45900 1.33095                0.88521 0.66973
5                1.32629 1.32261                0.90563 0.63297
6                1.29025 1.31826                0.88911 0.64169
  trust_government_corruption generosity dystopia_residual gff_stat
1                     0.41978    0.29678           2.51738  2.31186
2                     0.14145    0.43630           2.70201  2.46730
3                     0.48357    0.34139           2.49204  2.35135
4                     0.36503    0.34699           2.46531  2.34767
5                     0.32957    0.45811           2.45176  2.41369
6                     0.41372    0.23351           2.61955  2.19346
```

11. I grouped the **happy_df** dataset by region and generated a summary that includes three key statistics:

    1. The number of countries in each region (labeled as **country_count**),
    2. The average happiness score for each region (labeled as **mean_happiness**)
    3. The average freedom score for each region (labeled as **mean_freedom**).

    The summarized data is stored in a new variable called **regional_stats_df**.

    I used group_by(region) function for groups the dataset by the region column. This means that summarizing will be performed separately for each region.

    I used summarise() function to creates a new summary table based on the grouped data. Inside the summarise function, you define new columns based on calculations.

```
> regional_stats_df <- happy_df %>%
    group_by(region) %>%
    summarise(
      country_count = n(),
      mean_happiness = mean(happiness_score),
      mean_freedom = mean(freedom)
    )
> # View the resulting table
> print(regional_stats_df)

# A tibble: 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
```

Assignment 2

12. In this task, analysis of the batting statistics from the 1986 Major League Baseball season, we first loaded the data from the baseball.csv file into a variable named baseball**.**

    To confirm that the data was loaded correctly, we used the head () function, which displays the first few rows of the dataset.

```
> baseball <-
  read.csv("C:/Users/nihar/OneDrive/Desktop/Niharika_Project2/base
  ball.csv")
> head(baseball)
       Last  First Age  G  PA  AB  R  H X2B X3B HR RBI SB CS BB SO
1     Acker    Jim  27 21  28  28  1  3   1   0  0   0  0  0  0 21
2    Adduci    Jim  26  3  13  11  2  1   1   0  0   0  0  0  1  2
3    Aguayo   Luis  27 62 146 133 17 28   6   1  4  13  1  1  8 26
4  Aguilera   Rick  24 32  57  51  4  8   0   0  2   6  0  0  3 12
5 Akerfelds Darrel  24  1   0   0  0  0   0   0  0   0  0  0  0  0
6   Aldrete   Mike  25 84 256 216 27 54  18   3  2  25  1  3 33 34
```

13. Here, I took time to explore the dataset to understand its structure and details better. This included looking at the data types, checking for any missing values, and summarizing important statistics. By doing this, I can spot patterns and trends, which will help guide our analysis and findings.

    >glimpse(baseball)

Rows: 771
Columns: 16
$ Last *<chr>* "Acker", "Adduci", "Aguayo", "Aguilera", "Akerfelds", "Aldrete…
$ First *<chr>* "Jim", "Jim", "Luis", "Rick", "Darrel", "Mike", "Doyle", "Andy…
$ Age *<int>* 27, 26, 27, 24, 24, 25, 35, 24, 33, 27, 33, 25, 29, 22, 32, 23…
$ G *<int>* 21, 3, 62, 32, 1, 84, 18, 101, 102, 8, 48, 92, 15, 1, 121, 15,…
$ PA *<int>* 28, 13, 146, 57, 0, 256, 45, 324, 230, 11, 7, 241, 12, 0, 453,…
$ AB *<int>* 28, 11, 133, 51, 0, 216, 38, 293, 196, 11, 6, 216, 11, 0, 425,…
$ R *<int>* 1, 2, 17, 4, 0, 27, 2, 30, 29, 0, 0, 31, 1, 0, 40, 9, 24, 0, 0…
$ H *<int>* 3, 1, 28, 8, 0, 54, 8, 66, 43, 1, 0, 53, 1, 0, 112, 20, 81, 0,…
$ X2B *<int>* 1, 1, 6, 0, 0, 18, 1, 7, 7, 0, 0, 9, 0, 0, 21, 5, 15, 0, 0, 18…
$ X3B *<int>* 0, 0, 1, 0, 0, 3, 0, 3, 2, 0, 0, 0, 0, 0, 4, 0, 0, 0, 0, 2, 0,…
$ HR *<int>* 0, 0, 4, 2, 0, 2, 0, 1, 7, 0, 0, 1, 0, 0, 11, 0, 7, 0, 0, 1, 0…
$ RBI *<int>* 0, 0, 13, 6, 0, 25, 5, 29, 27, 0, 0, 15, 0, 0, 58, 7, 38, 0, 0…
$ SB *<int>* 0, 0, 1, 0, 0, 1, 0, 10, 11, 0, 0, 5, 0, 0, 0, 1, 1, 0, 0, 13,…
$ CS *<int>* 0, 0, 1, 0, 0, 3, 0, 1, 4, 0, 0, 1, 0, 0, 3, 2, 0, 0, 0, 7, 0,…
$ BB *<int>* 0, 1, 8, 3, 0, 33, 0, 14, 30, 0, 0, 22, 0, 0, 24, 3, 39, 2, 0,…
$ SO *<int>* 21, 2, 26, 12, 0, 34, 8, 36, 38, 4, 3, 39, 4, 0, 77, 13, 56, 3…

```
> summary(baseball)
   Last          First             Age              G
 Length:771       Length:771       Min.   :20.00   Min.   :  1.0
 Class :character   Class :character   1st Qu.:25.00   1st Qu.: 17.0
 Mode  :character   Mode  :character   Median :27.00   Median : 56.0
                                       Mean   :27.98   Mean   : 66.2
                                       3rd Qu.:31.00   3rd Qu.:111.0
                                       Max.   :45.00   Max.   :163.0
    PA              AB              R               H
 Min.   :  0.0   Min.   :  0.0   Min.   :  0.00   Min.   :  0.00
 1st Qu.: 16.0   1st Qu.: 14.0   1st Qu.:  1.00   1st Qu.:  2.00
 Median :101.0   Median : 90.0   Median :  9.00   Median : 19.00
 Mean   :208.6   Mean   :185.6   Mean   : 24.05   Mean   : 47.83
 3rd Qu.:366.5   3rd Qu.:328.5   3rd Qu.: 42.00   3rd Qu.: 84.00
 Max.   :742.0   Max.   :687.0   Max.   :130.00   Max.   :238.00
    X2B             X3B             HR              RBI
 Min.   : 0.000   Min.   : 0.000   Min.   : 0.000   Min.   :  0.00
 1st Qu.: 0.000   1st Qu.: 0.000   1st Qu.: 0.000   1st Qu.:  0.00
 Median : 3.000   Median : 0.000   Median : 1.000   Median :  8.00
 Mean   : 8.445   Mean   : 1.109   Mean   : 4.946   Mean   : 22.56
 3rd Qu.:14.000   3rd Qu.: 2.000   3rd Qu.: 7.000   3rd Qu.: 38.00
 Max.   :53.000   Max.   :14.000   Max.   :40.000   Max.   :121.00
    SB              CS              BB              SO
 Min.   :  0.000   Min.   : 0.000   Min.   :  0.00   Min.   :  0.00
 1st Qu.:  0.000   1st Qu.: 0.000   1st Qu.:  1.00   1st Qu.:  4.00
 Median :  0.000   Median : 0.000   Median :  7.00   Median : 20.00
 Mean   :  4.296   Mean   : 2.101   Mean   : 18.45   Mean   : 32.04
 3rd Qu.:  3.000   3rd Qu.: 3.000   3rd Qu.: 30.00   3rd Qu.: 51.50
 Max.   :107.000   Max.   :19.000   Max.   :105.00   Max.   :185.00
```

14. In this task, I filtered the baseball dataset to remove any players with zero at-bats (AB).

    The cleaned dataset is now stored back in the variable baseball. This step says that our analysis focuses only on players who have participated in games, providing more relevant insights.
    I used filter () function This to subset a data frame, returning only the rows that meet certain criteria. AB (at-bats) column has values greater than 0. This removes any players who did not have any at-bats.

```
> baseball <- baseball %>%
    filter(AB > 0)
> head(baseball)
       Last First Age  G  PA  AB  R  H X2B X3B HR RBI SB CS BB SO
1     Acker   Jim  27 21  28  28  1  3   1   0  0   0  0  0  0 21
2    Adduci   Jim  26  3  13  11  2  1   1   0  0   0  0  0  1  2
3    Aguayo  Luis  27 62 146 133 17 28   6   1  4  13  1  1  8 26
4  Aguilera  Rick  24 32  57  51  4  8   0   0  2   6  0  0  3 12
5   Aldrete  Mike  25 84 256 216 27 54  18   3  2  25  1  3 33 34
6 Alexander Doyle  35 18  45  38  2  8   1   0  0   5  0  0  0  8
```

15. Here, I added a new column to the baseball dataset called BA(batting average).
    This average is calculated by dividing the number of hits (H) by the number of (AB) for each player.
    The updated dataset is now stored in the variable baseball, allowing us to analyze players' performance more effectively.
    I used mutate () function creates a new column.

```
> baseball <- baseball %>%
  mutate(BA = H/AB)
> head(baseball)

Last First Age G PA AB R H X2B X3B HR RBI SB CS BB SO       BA
1     Acker  Jim 27 21 28 28 1 3 1      0 0    0 0 0 0 21 0.10714286
2     Adduci Jim 26 3 13 11 2 1  1      0 0    0 0 0 1 2 0.09090909
3     Aguayo Luis 27 62 146 133 17 28  6      1 4 13 1 1 8 26 0.21052632
4 Aguilera Rick 24 32 57 51 4 8  0      0 2    6 0 0 3 12 0.15686275
5     Aldrete Mike 25 84 256 216 27 54 18     3 2 25 1 3 33 34 0.25000000
6 Alexander Doyle 35 18 45 38 2 8       1      0 0    5 0 0 0 8 0.21052632
```

16. I added a new column called OBP (On-base Percentage) to the baseball dataset.
    In this I calculated using the formula ((H + BB) / (AB + BB)),
        Where H stands for hits, BB is walks, and AB is at-bats.
    This new column helps to better understand how often players reach base.
    Also, I used mutate () function creates a new column.

```
> baseball <- baseball %>%
    mutate(OBP =  (H + BB) / (AB + BB))
> head(baseball)

       Last First Age  G  PA  AB  R  H X2B X3B HR RBI SB CS BB SO          BA        OBP
1     Acker   Jim  27 21  28  28  1  3   1   0  0   0  0  0  0 21 0.10714286 0.1071429
2    Adduci   Jim  26  3  13  11  2  1   1   0  0   0  0  0  1  2 0.09090909 0.1666667
3    Aguayo  Luis  27 62 146 133 17 28   6   1  4  13  1  1  8 26 0.21052632 0.2553191
4  Aguilera  Rick  24 32  57  51  4  8   0   0  2   6  0  0  3 12 0.15686275 0.2037037
5   Aldrete  Mike  25 84 256 216 27 54  18   3  2  25  1  3 33 34 0.25000000 0.3493976
6 Alexander Doyle  35 18  45  38  2  8   1   0  0   5  0  0  0  8 0.21052632 0.2105263
```

17. Here, I identified the top 10 players who strike out the most this season and stoed this information in a new variable called trikeout_artist.
    This helps to see which players might be having trouble hitting and could use some changes in their batting technique.
    The slice() function is useful when you want to extract specific rows based on their position rather Than conditions or values.

```
> strikeout_artist <- baseball%>%

    arrange(desc(SO))%>%
    slice(1:10)
> print(strikeout_artist)
```

```
          Last  First Age   G  PA  AB   R   H X2B X3B HR RBI SB CS BB  SO         BA
1  Incaviglia   Pete  22 153 606 540  82 135  21   2 30  88  3  2 55 185 0.2500000
2        Deer    Rob  25 134 546 466  75 108  17   3 33  86  5  2 72 179 0.2317597
3     Canseco   Jose  21 157 682 600  85 144  29   1 33 117 15  7 65 175 0.2400000
4     Presley    Jim  24 155 660 616  83 163  33   4 27 107  0  4 32 172 0.2646104
5   Tartabull  Danny  23 137 578 511  76 138  25   6 25  96  4  8 61 157 0.2700587
6     Balboni  Steve  29 138 562 512  54 117  25   1 29  88  0  0 43 146 0.2285156
7    Barfield  Jesse  26 158 671 589 107 170  35   2 40 108  8  8 69 146 0.2886248
8      Samuel   Juan  25 145 633 591  90 157  36  12 16  78 42 14 26 142 0.2656514
9      Murphy   Dale  30 160 692 614  89 163  29   7 29  83  7  7 75 141 0.2654723
10 Strawberry Darryl  24 136 562 475  76 123  27   5 27  93 28 12 72 141 0.2589474
```

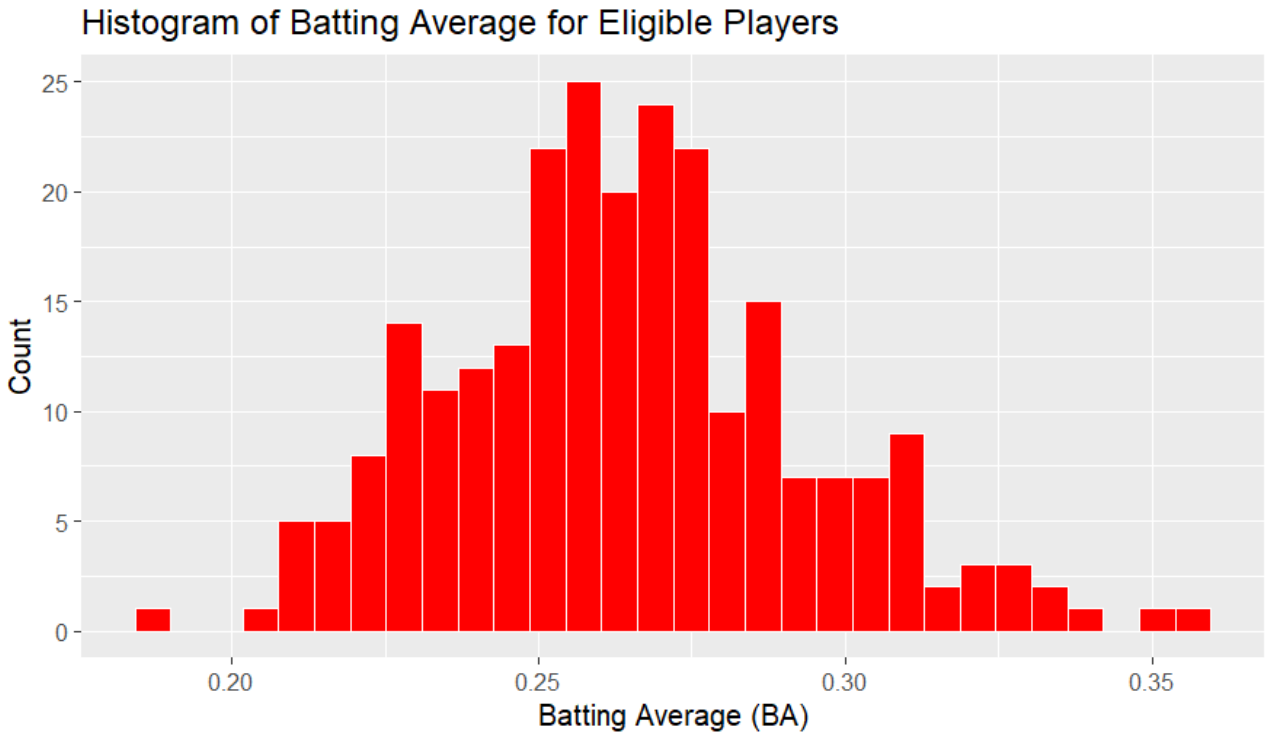
18. In this task, I filtered the dataset to include only players who eligible for end-of-season awards. To qualify, they must have either 300 or more at-bats or have played in at least 100 games. The extracted data is stored in eligible_df, so easily can focus on players who meet the criteria for awards.

```
> eligible_df <- baseball %>%
  filter(AB >= 300 | G >= 100)
>  head(eligible_df)
```

```
Last  First Age   G  PA  AB  R   H X2B X3B HR RBI SB CS BB SO        BA
1 Allanson   Andy  24 101 324 293 30  66   7   3  1  29 10  1 14 36 0.2252560
2    Almon   Bill  33 102 230 196 29  43   7   2  7  27 11  4 30 38 0.2193878
3    Armas   Tony  32 121 453 425 40 112  21   4 11  58  0  3 24 77 0.2635294
4    Ashby   Alan  34 120 361 315 24  81  15   0  7  38  1  0 39 56 0.2571429
5  Backman  Wally  26 124 440 387 67 124  18   2  1  27 13  7 36 32 0.3204134
6   Baines Harold  27 145 618 570 72 169  29   2 21  88  2  1 38 89 0.2964912
OBP
1 0.2605863
2 0.3230088
3 0.3028953
4 0.3389831
5 0.3782506
6 0.3404605
```

19. I made a histogram showing the batting averages of eligible players. It helps to understand how batting performance is distributed among players who qualify for end-of-season awards.

```
> ggplot(eligible_df, aes(x = BA)) +
  geom_histogram( fill = "red", color = "white") +
  labs(title = "Histogram of Batting Average for Eligible Players",
       x = "Batting Average (BA)",
       y = "Count")
> theme_minimal()
```



20. In this task, I evaluated players who eligible for the MVP award based on key statistics:
    On-Base Percentage (OBP), Home Runs (HR), and Runs Batted In (RBI).
    The extracted data is stored in eligible_df, so easily can focus on players who meet the criteria for awards.

```
> mvp_players <- baseball %>%
      select(First,Last, OBP, HR, RBI) %>%
      arrange(desc(OBP), desc(HR), desc(RBI))
>
>   head(mvp_players)

  First     Last       OBP HR RBI
1   Tim   Lollar 1.0000000  0   0
2   Don Robinson 0.6666667  0   1
3  Greg   Minton 0.5714286  0   0
4 Garry   Maddox 0.5555556  0   1
5  John  Gibbons 0.5454545  1   1
6   Pat   Dodson 0.5333333  1   3


library(pacman)
```

Summary

In this exercise, I learned how to use several important data manipulation functions in R, which are crucial for working with large datasets. Functions like head () and view () let me quickly look at the data, while names() and glimpse() helped me understand the structure of the dataset.

Using filter () and slice (), I focused on specific parts of the data, like rows where freedom was less than 0.20 or baseball players with at least one at-bat.

I also used arrange () to sort the data and summaries () to group and summarize it, for example by region or player performance. The mutate () function was useful for creating new columns, like batting average and on-base percentage. Overall, these functions helped me analyze the data more effectively and make decisions based on key statistics like OBP, HR, and RBI.