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

by

Sri Ram Prabu

ALY6000 : Introduction to Analytics

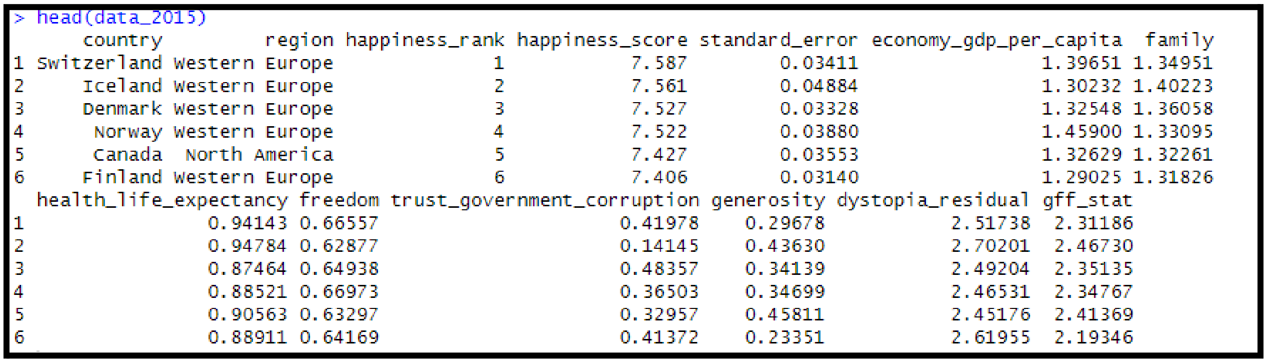
January , 24, 2025

Assignment Part 1

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.

```
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)`
```

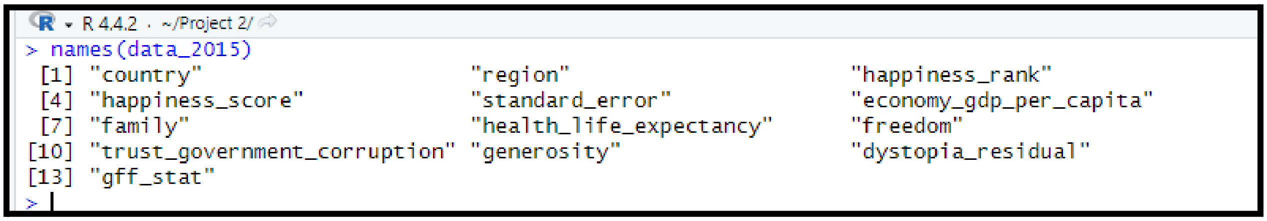


```
> head(data_2015)
      country         region happiness_rank happiness_score standard_error economy_gdp_per_capita  family
1 Switzerland Western Europe              1           7.587        0.03411                1.39651 1.34951
2     Iceland Western Europe              2           7.561        0.04884                1.30232 1.40223
3     Denmark Western Europe              3           7.527        0.03328                1.32548 1.36058
4      Norway Western Europe              4           7.522        0.03880                1.45900 1.33095
5      Canada  North America              5           7.427        0.03553                1.32629 1.32261
6     Finland Western Europe              6           7.406        0.03140                1.29025 1.31826
  health_life_expectancy freedom trust_government_corruption generosity dystopia_residual gff_stat
1                0.94143 0.66557                     0.41978    0.29678           2.51738  2.31186
2                0.94784 0.62877                     0.14145    0.43630           2.70201  2.46730
3                0.87464 0.64938                     0.48357    0.34139           2.49204  2.35135
4                0.88521 0.66973                     0.36503    0.34699           2.46531  2.34767
5                0.90563 0.63297                     0.32957    0.45811           2.45176  2.41369
6                0.88911 0.64169                     0.41372    0.23351           2.61955  2.19346
```

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

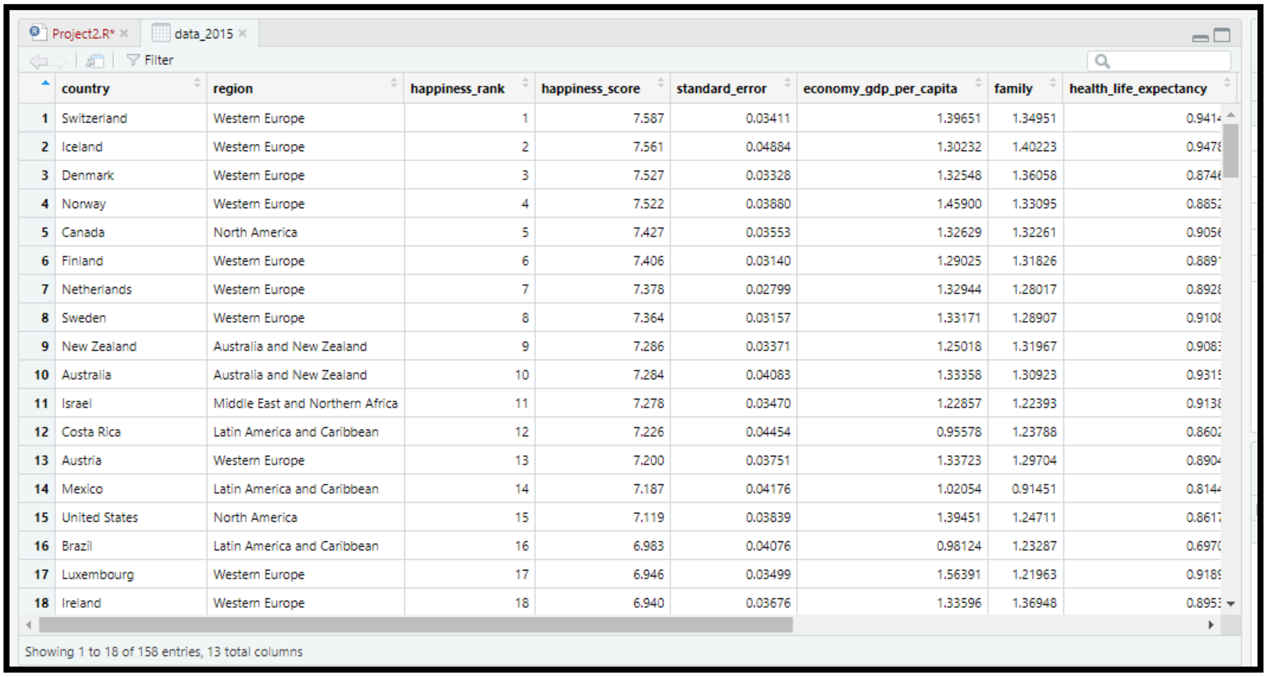
```
names(data_2015)

 [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"
```



```
> names(data_2015)
 [1] "country"                       "region"                        "happiness_rank"
 [4] "happiness_score"               "standard_error"                "economy_gdp_per_capita"
 [7] "family"                        "health_life_expectancy"        "freedom"
[10] "trust_government_corruption"   "generosity"                    "dystopia_residual"
[13] "gff_stat"
> |
```

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

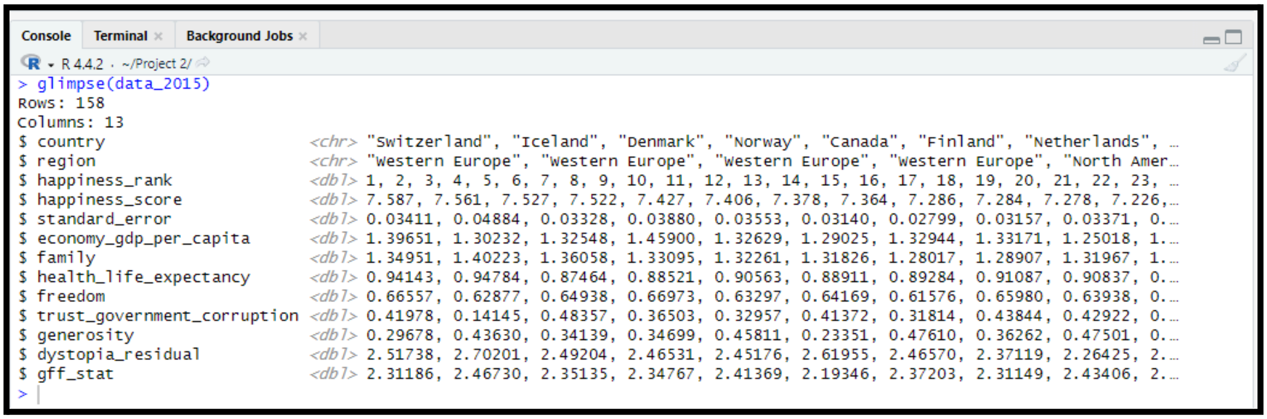


	country	region	happiness_rank	happiness_score	standard_error	economy_gdp_per_capita	family	health_life_expectancy
1	Switzerland	Western Europe	1	7.587	0.03411	1.39651	1.34951	0.9414
2	Iceland	Western Europe	2	7.561	0.04884	1.30232	1.40223	0.9478
3	Denmark	Western Europe	3	7.527	0.03328	1.32548	1.36058	0.8746
4	Norway	Western Europe	4	7.522	0.03880	1.45900	1.33095	0.8852
5	Canada	North America	5	7.427	0.03553	1.32629	1.32261	0.9056
6	Finland	Western Europe	6	7.406	0.03140	1.29025	1.31826	0.8891
7	Netherlands	Western Europe	7	7.378	0.02799	1.32944	1.28017	0.8928
8	Sweden	Western Europe	8	7.364	0.03157	1.33171	1.28907	0.9108
9	New Zealand	Australia and New Zealand	9	7.286	0.03371	1.25018	1.31967	0.9083
10	Australia	Australia and New Zealand	10	7.284	0.04083	1.33358	1.30923	0.9315
11	Israel	Middle East and Northern Africa	11	7.278	0.03470	1.22857	1.22393	0.9138
12	Costa Rica	Latin America and Caribbean	12	7.226	0.04454	0.95578	1.23788	0.8602
13	Austria	Western Europe	13	7.200	0.03751	1.33723	1.29704	0.8904
14	Mexico	Latin America and Caribbean	14	7.187	0.04176	1.02054	0.91451	0.8144
15	United States	North America	15	7.119	0.03839	1.39451	1.24711	0.8617
16	Brazil	Latin America and Caribbean	16	6.983	0.04076	0.98124	1.23287	0.6970
17	Luxembourg	Western Europe	17	6.946	0.03499	1.56391	1.21963	0.9189
18	Ireland	Western Europe	18	6.940	0.03676	1.33596	1.36948	0.8953

Showing 1 to 18 of 158 entries, 13 total columns

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

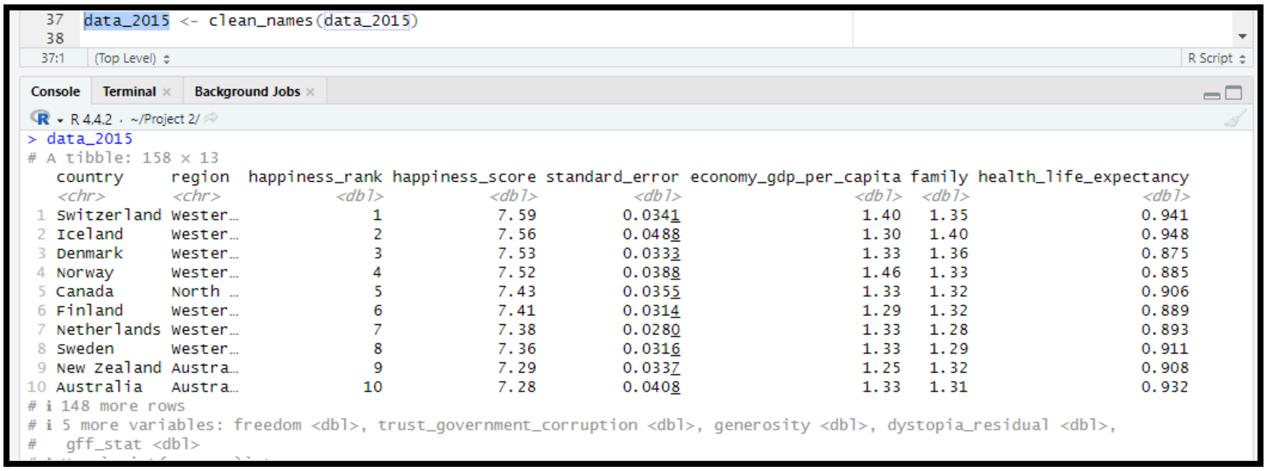
```
glimpse(data_2015)
```



```
> glimpse(data_2015)
Rows: 158
Columns: 13
$ country                    <chr> "Switzerland", "Iceland", "Denmark", "Norway", "Canada", "Finland", "Netherlands", …
$ region                     <chr> "Western Europe", "Western Europe", "Western Europe", "Western Europe", "North Amer…
$ happiness_rank             <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, …
$ happiness_score            <dbl> 7.587, 7.561, 7.527, 7.522, 7.427, 7.406, 7.378, 7.364, 7.286, 7.284, 7.278, 7.226,…
$ standard_error             <dbl> 0.03411, 0.04884, 0.03328, 0.03880, 0.03553, 0.03140, 0.02799, 0.03157, 0.03371, 0.…
$ economy_gdp_per_capita     <dbl> 1.39651, 1.30232, 1.32548, 1.45900, 1.32629, 1.29025, 1.32944, 1.33171, 1.25018, 1.…
$ family                     <dbl> 1.34951, 1.40223, 1.36058, 1.33095, 1.32261, 1.31826, 1.28017, 1.28907, 1.31967, 1.…
$ health_life_expectancy     <dbl> 0.94143, 0.94784, 0.87464, 0.88521, 0.90563, 0.88911, 0.89284, 0.91087, 0.90837, 0.…
$ freedom                    <dbl> 0.66557, 0.62877, 0.64938, 0.66973, 0.63297, 0.64169, 0.61576, 0.65980, 0.63938, 0.…
$ trust_government_corruption <dbl> 0.41978, 0.14145, 0.48357, 0.36503, 0.32957, 0.41372, 0.31814, 0.43844, 0.42922, 0.…
$ generosity                 <dbl> 0.29678, 0.43630, 0.34139, 0.34699, 0.45811, 0.23351, 0.47610, 0.36262, 0.47501, 0.…
$ dystopia_residual          <dbl> 2.51738, 2.70201, 2.49204, 2.46531, 2.45176, 2.61955, 2.46570, 2.37119, 2.26425, 2.…
$ gff_stat                   <dbl> 2.31186, 2.46730, 2.35135, 2.34767, 2.41369, 2.19346, 2.37203, 2.31149, 2.43406, 2.…
> 
```

5. Install and load 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.

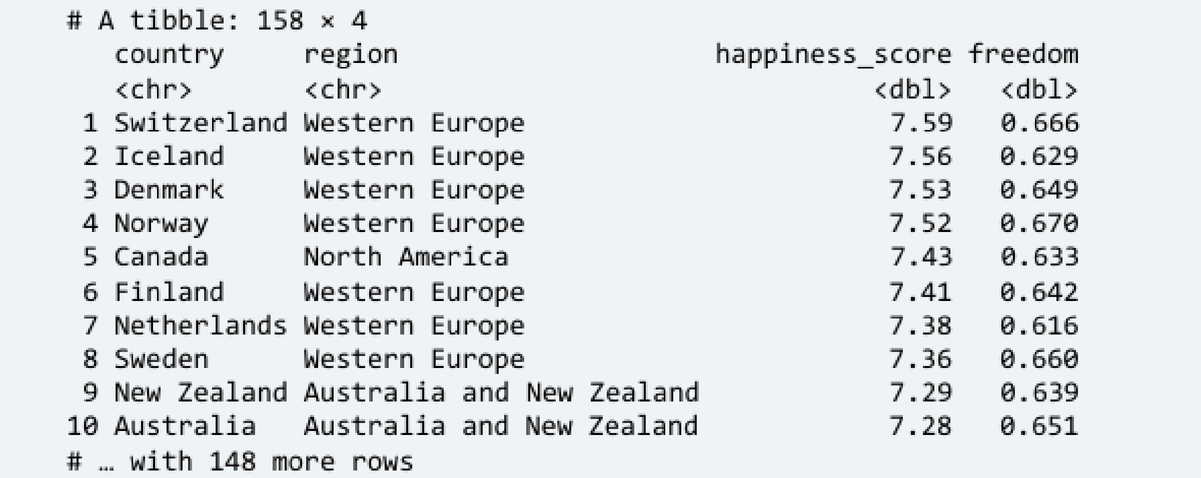
```
library(janitor)
data_2015 <- clean_names(data_2015)
data_2015
```



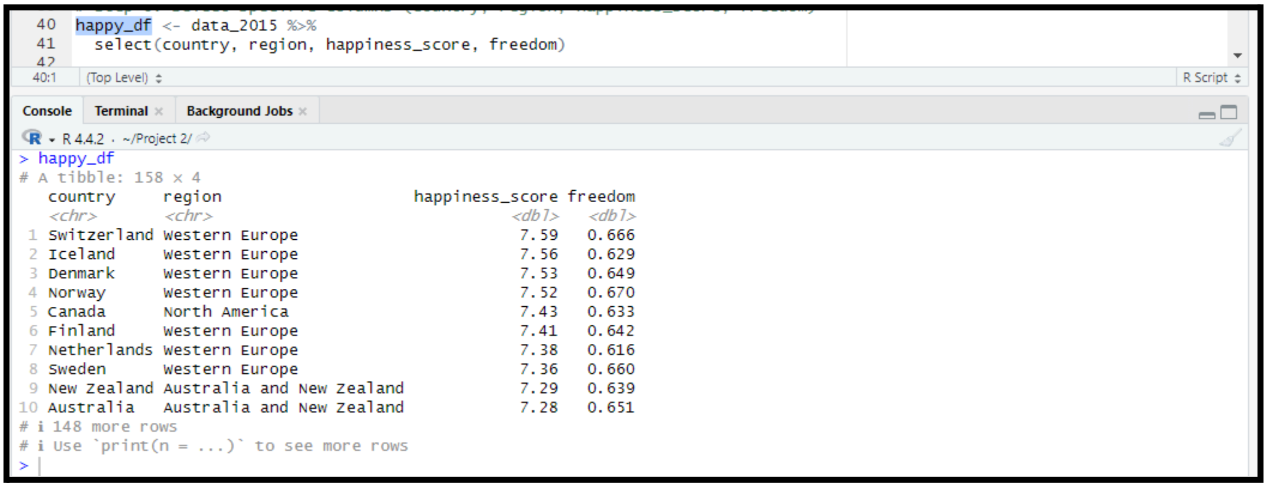
```
data_2015 <- clean_names(data_2015)
```

```
> data_2015
# A tibble: 158 × 13
   country     region  happiness_rank happiness_score standard_error economy_gdp_per_capita family health_life_expectancy
   <chr>       <chr>            <dbl>           <dbl>          <dbl>                  <dbl>  <dbl>                  <dbl>
 1 Switzerland Wester…              1            7.59         0.0341                   1.40   1.35                  0.941
 2 Iceland     Wester…              2            7.56         0.0488                   1.30   1.40                  0.948
 3 Denmark     Wester…              3            7.53         0.0333                   1.33   1.36                  0.875
 4 Norway      Wester…              4            7.52         0.0388                   1.46   1.33                  0.885
 5 Canada      North …              5            7.43         0.0355                   1.33   1.32                  0.906
 6 Finland     Wester…              6            7.41         0.0314                   1.29   1.32                  0.889
 7 Netherlands Wester…              7            7.38         0.0280                   1.33   1.28                  0.893
 8 Sweden      Wester…              8            7.36         0.0316                   1.33   1.29                  0.911
 9 New Zealand Austra…              9            7.29         0.0337                   1.25   1.32                  0.908
10 Australia   Austra…             10            7.28         0.0408                   1.33   1.31                  0.932
# ℹ 148 more rows
# ℹ 5 more variables: freedom <dbl>, trust_government_corruption <dbl>, generosity <dbl>, dystopia_residual <dbl>,
#   gff_stat <dbl>
```

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
```

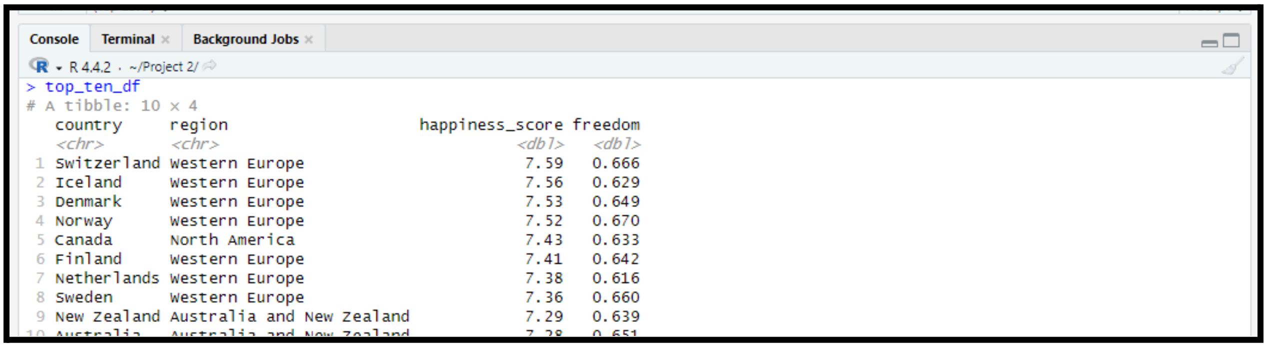


```
happy_df <- data_2015 %>%
  select(country, region, happiness_score, freedom)
```

```
> 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
# ℹ 148 more rows
# ℹ Use `print(n = ...)` to see more rows
>
```

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

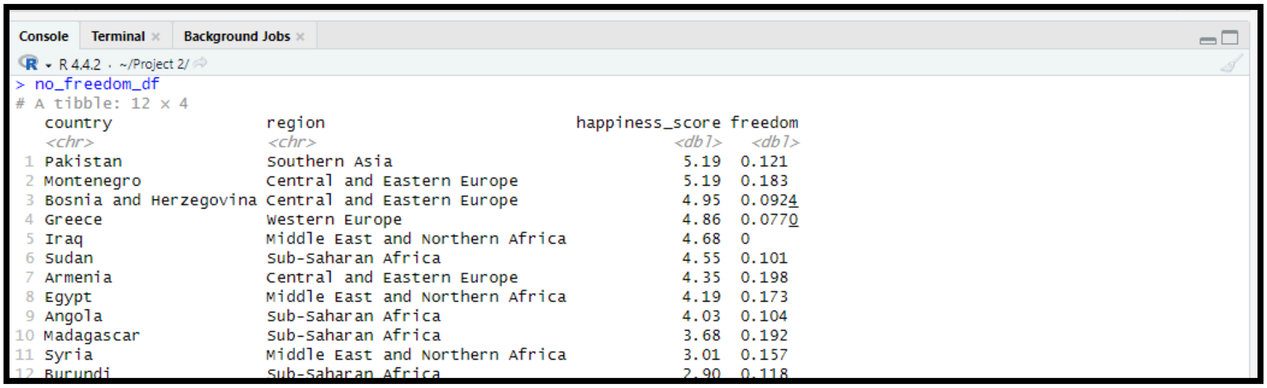
```
# A tibble: 10 × 4
   country     region                    happiness_score freedom
   <chr>       <chr>                               <dbl>   <dbl>
 1 Switzerland Western Europe                       7.59   0.666
 2 Iceland     Western Europe                       7.56   0.629
 3 Denmark     Western 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
```



```
> 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.**

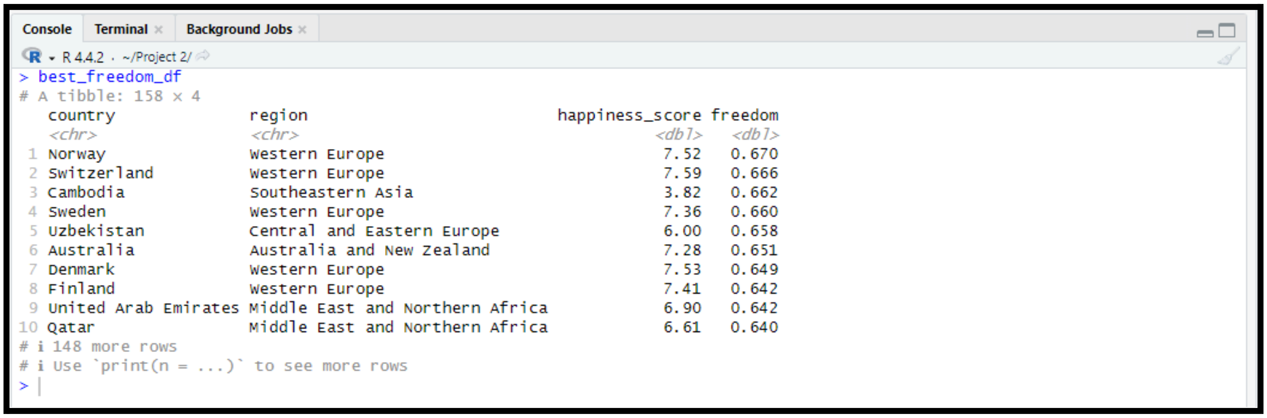
```
# A tibble: 12 × 4
   country                      region                   
happiness_sc…¹ freedom
   <chr>                        <chr>                    
<dbl>   <dbl>
```



```
> no_freedom_df
# A tibble: 12 × 4
   country                region                          happiness_score 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
```

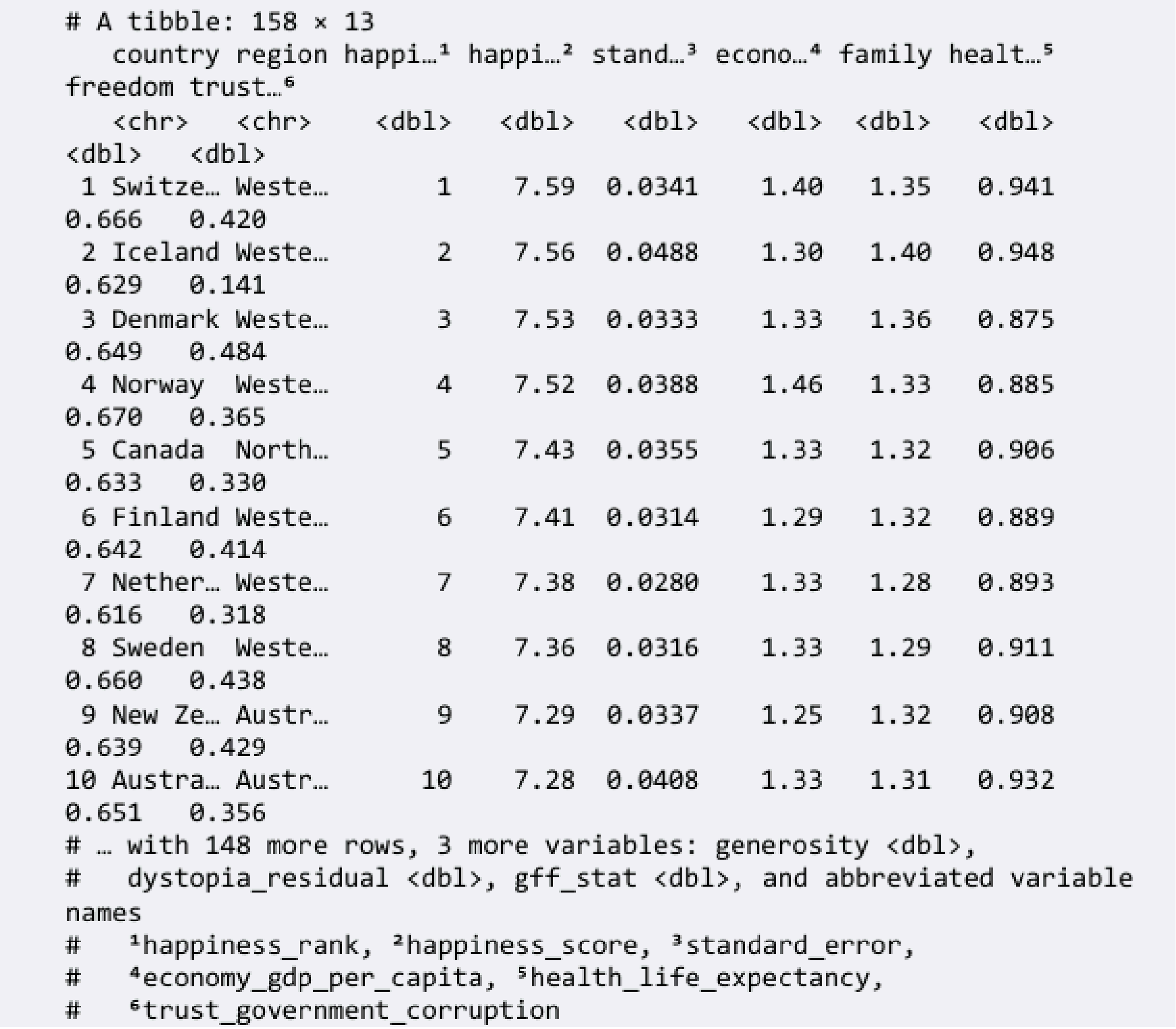
9. Arrange the values in **happy_df** in descending order by their freedom values. Store this new table as **best_freedom_df**.

```
# 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
```

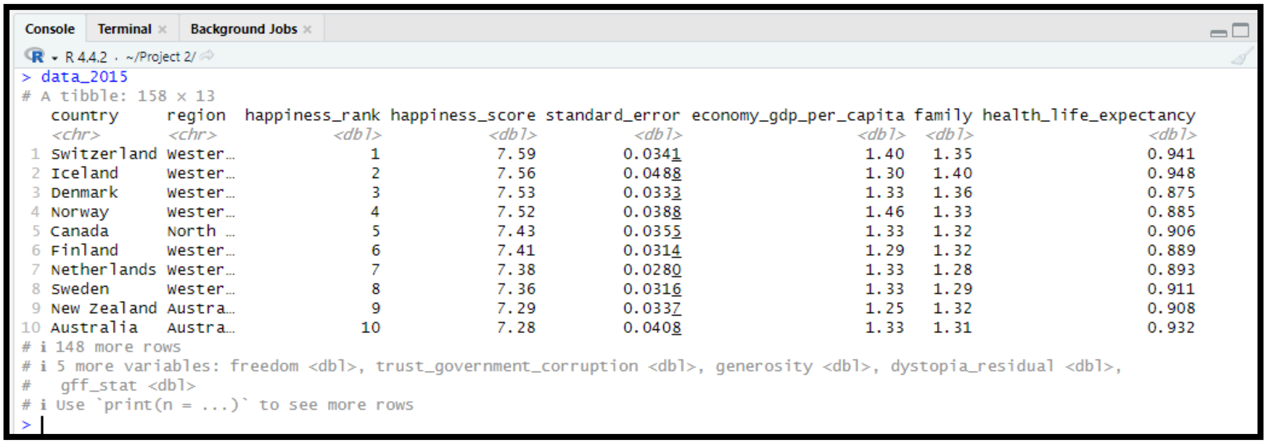


```
> best_freedom_df
# A tibble: 158 × 4
   country              region                          happiness_score freedom
   <chr>                <chr>                                     <dbl>   <dbl>
 1 Norway               Western Europe                             7.52   0.670
 2 Switzerland          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
# ℹ 148 more rows
# ℹ Use `print(n = ...)` to see more rows
> 
```

10. Create a new column **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 back into the **data_2015** variable.



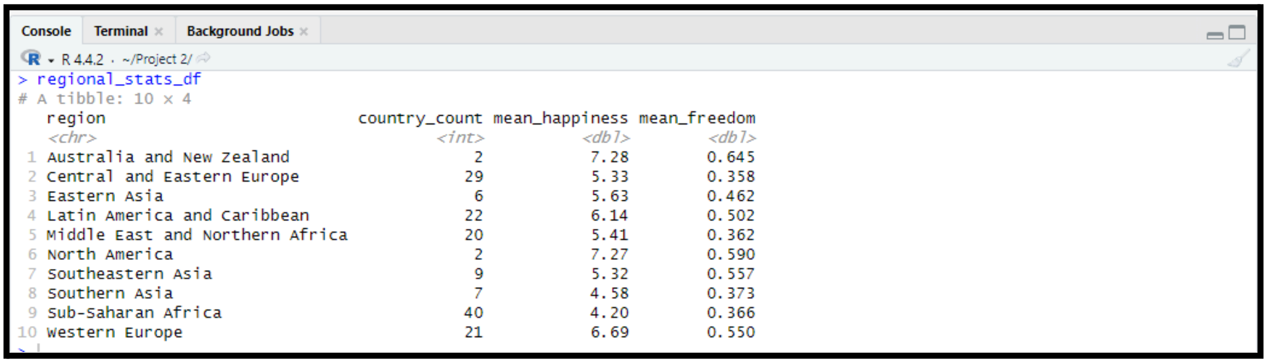
```
# A tibble: 158 × 13
   country region happi…¹ happi…² stand…³ econo…⁴ family healt…⁵
freedom trust…⁶
   <chr>   <chr>    <dbl>   <dbl>   <dbl>   <dbl>  <dbl>   <dbl>
<dbl>   <dbl>
 1 Switze… Weste…       1    7.59  0.0341    1.40   1.35   0.941
0.666   0.420
 2 Iceland Weste…       2    7.56  0.0488    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
```



```
> data_2015
# A tibble: 158 × 13
   country     region  happiness_rank happiness_score standard_error economy_gdp_per_capita family health_life_expectancy
   <chr>       <chr>            <dbl>           <dbl>          <dbl>                  <dbl>  <dbl>                  <dbl>
 1 Switzerland Wester…              1            7.59         0.0341                   1.40   1.35                  0.941
 2 Iceland     Wester…              2            7.56         0.0488                   1.30   1.40                  0.948
 3 Denmark     Wester…              3            7.53         0.0333                   1.33   1.36                  0.875
 4 Norway      Wester…              4            7.52         0.0388                   1.46   1.33                  0.885
 5 Canada      North …              5            7.43         0.0355                   1.33   1.32                  0.906
 6 Finland     Wester…              6            7.41         0.0314                   1.29   1.32                  0.889
 7 Netherlands Wester…              7            7.38         0.0280                   1.33   1.28                  0.893
 8 Sweden      Wester…              8            7.36         0.0316                   1.33   1.29                  0.911
 9 New Zealand Austra…              9            7.29         0.0337                   1.25   1.32                  0.908
10 Australia   Austra…             10            7.28         0.0408                   1.33   1.31                  0.932
# ℹ 148 more rows
# ℹ 5 more variables: freedom <dbl>, trust_government_corruption <dbl>, generosity <dbl>, dystopia_residual <dbl>,
#   gff_stat <dbl>
# ℹ Use `print(n = ...)` to see more rows
> 
```

11. 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.**

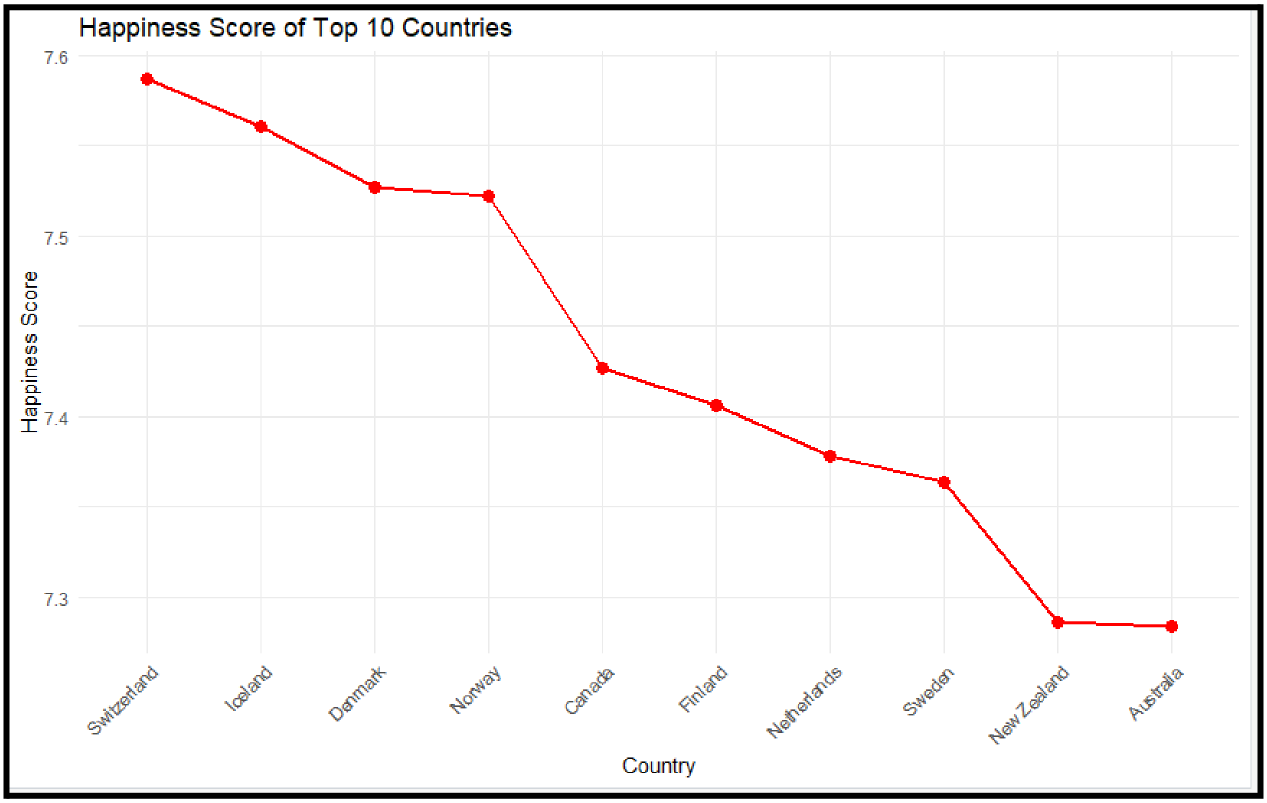
```
# A tibble: 10 × 4
  region                          country_count mean_happiness
```



```
> 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
```

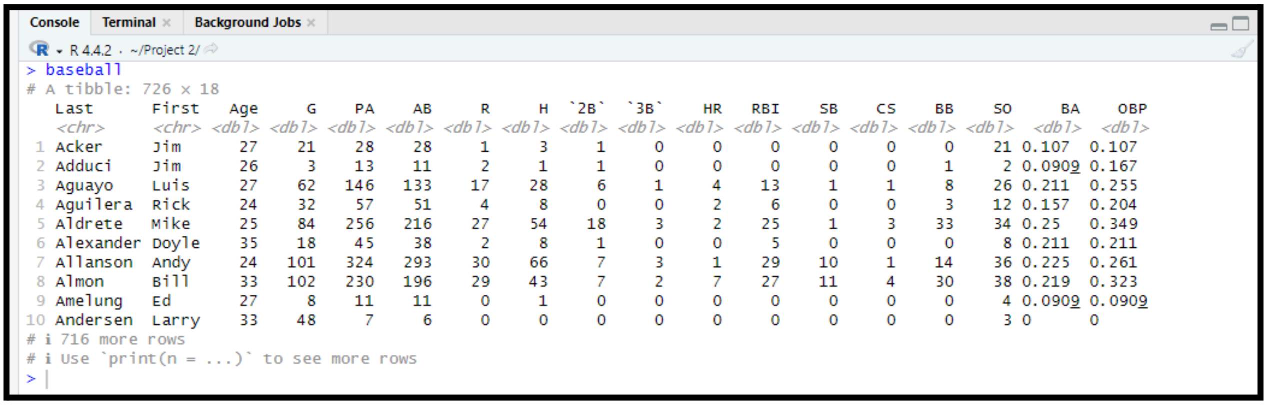
The analysis of the 2015 happiness data revealed significant disparities across regions and countries in terms of happiness, freedom, and related metrics. Countries with higher freedom, family support, and generosity tend to have higher happiness scores, reinforcing the interconnectedness of these variables.

This data can inform policymakers, researchers, and organizations aiming to improve global happiness and quality of life through targeted interventions.



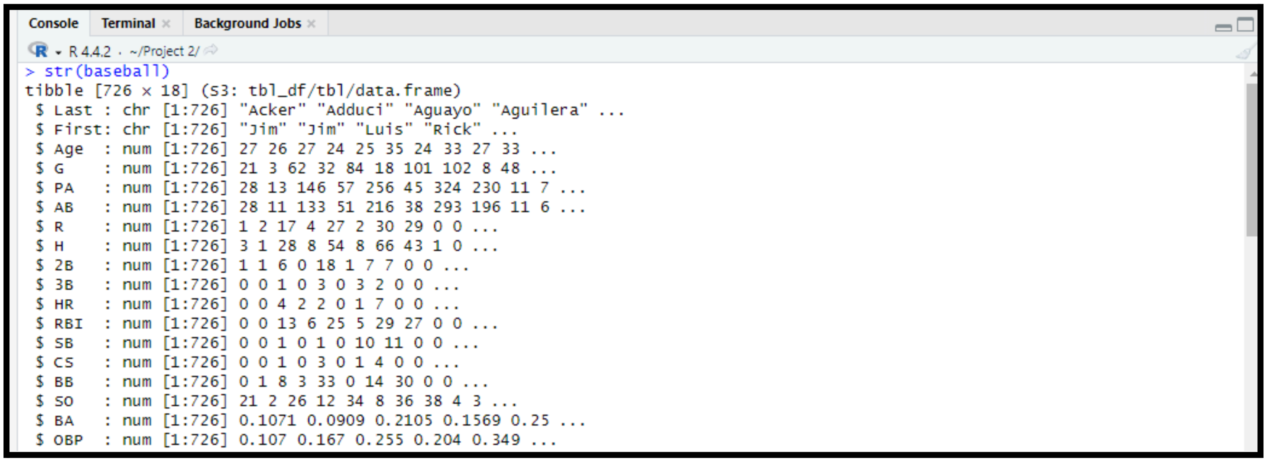
Assignment Part 2

12. Download the **baseball.csv** data set that represents batting statistics from the 1986 Major League Baseball season. Read this data set in a **variable** called **baseball.**

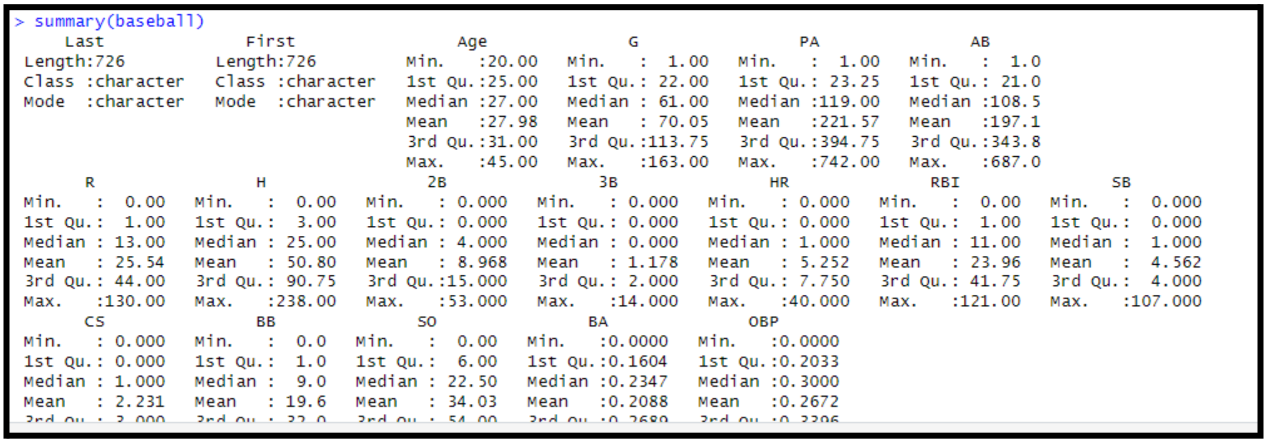


```
> baseball
# A tibble: 726 × 18
   Last      First   Age     G    PA    AB     R     H  `2B`  `3B`    HR   RBI    SB    CS    BB    SO BA     OBP   
   <chr>     <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  <dbl>  <dbl>
 1 Acker     Jim      27    21    28    28     1     3     1     0     0     0     0     0     0    21 0.107  0.107 
 2 Adduci    Jim      26     3    13    11     2     1     1     0     0     0     0     0     1     2 0.0909 0.167 
 3 Aguayo    Luis     27    62   146   133    17    28     6     1     4    13     1     1     8    26 0.211  0.255 
 4 Aguilera  Rick     24    32    57    51     4     8     0     0     2     6     0     0     3    12 0.157  0.204 
 5 Aldrete   Mike     25    84   256   216    27    54    18     3     2    25     1     3    33    34 0.25   0.349 
 6 Alexander Doyle    35    18    45    38     2     8     1     0     0     5     0     0     0     8 0.211  0.211 
 7 Allanson  Andy     24   101   324   293    30    66     7     3     1    29    10     1    14    36 0.225  0.261 
 8 Almon     Bill     33   102   230   196    29    43     7     2     7    27    11     4    30    38 0.219  0.323 
 9 Amelung   Ed       27     8    11    11     0     1     0     0     0     0     0     0     0     4 0.0909 0.0909
10 Andersen  Larry    33    48     7     6     0     0     0     0     0     0     0     0     0     3 0      0     
# ℹ 716 more rows
# ℹ Use `print(n = ...)` to see more rows
> 
```

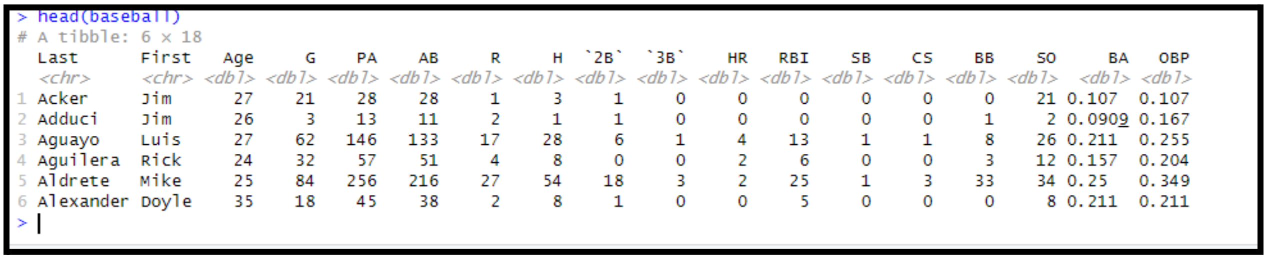
13. Spend time with the data using various exploration functions to get a general feel for what you are working with. For more information on this data set and its various columns, see Baseball Reference's 1986 Major League Standard Batting.



```
> str(baseball)
tibble [726 × 18] (S3: tbl_df/tbl/data.frame)
 $ Last : chr [1:726] "Acker" "Adduci" "Aguayo" "Aguilera" ...
 $ First: chr [1:726] "Jim" "Jim" "Luis" "Rick" ...
 $ Age  : num [1:726] 27 26 27 24 25 35 24 33 27 33 ...
 $ G    : num [1:726] 21 3 62 32 84 18 101 102 8 48 ...
 $ PA   : num [1:726] 28 13 146 57 256 45 324 230 11 7 ...
 $ AB   : num [1:726] 28 11 133 51 216 38 293 196 11 6 ...
 $ R    : num [1:726] 1 2 17 4 27 2 30 29 0 0 ...
 $ H    : num [1:726] 3 1 28 8 54 8 66 43 1 0 ...
 $ 2B   : num [1:726] 1 1 6 0 18 1 7 7 0 0 ...
 $ 3B   : num [1:726] 0 0 1 0 3 0 3 2 0 0 ...
 $ HR   : num [1:726] 0 0 4 2 2 0 1 7 0 0 ...
 $ RBI  : num [1:726] 0 0 13 6 25 5 29 27 0 0 ...
 $ SB   : num [1:726] 0 0 1 0 1 0 10 11 0 0 ...
 $ CS   : num [1:726] 0 0 1 0 3 0 1 4 0 0 ...
 $ BB   : num [1:726] 0 1 8 3 33 0 14 30 0 0 ...
 $ SO   : num [1:726] 21 2 26 12 34 8 36 38 4 3 ...
 $ BA   : num [1:726] 0.1071 0.0909 0.2105 0.1569 0.25 ...
 $ OBP  : num [1:726] 0.107 0.167 0.255 0.204 0.349 ...
```



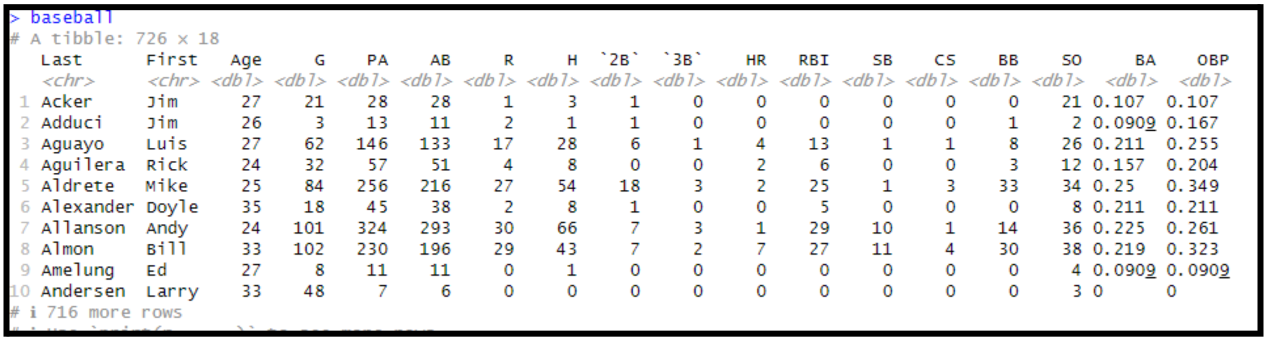
```
> summary(baseball)
     Last              First                Age              G                PA               AB       
 Length:726         Length:726         Min.   :20.00   Min.   :  1.00   Min.   :  1.00   Min.   :  1.0  
 Class :character   Class :character   1st Qu.:25.00   1st Qu.: 22.00   1st Qu.: 23.25   1st Qu.: 21.0  
 Mode  :character   Mode  :character   Median :27.00   Median : 61.00   Median :119.00   Median :108.5  
                                       Mean   :27.98   Mean   : 70.05   Mean   :221.57   Mean   :197.1  
                                       3rd Qu.:31.00   3rd Qu.:113.75   3rd Qu.:394.75   3rd Qu.:343.8  
                                       Max.   :45.00   Max.   :163.00   Max.   :742.00   Max.   :687.0  
       R                H                2B               3B               HR              RBI               SB         
 Min.   :  0.00   Min.   :  0.00   Min.   : 0.000   Min.   : 0.000   Min.   : 0.000   Min.   :  0.00   Min.   :  0.000  
 1st Qu.:  1.00   1st Qu.:  3.00   1st Qu.: 0.000   1st Qu.: 0.000   1st Qu.: 0.000   1st Qu.:  1.00   1st Qu.:  0.000  
 Median : 13.00   Median : 25.00   Median : 4.000   Median : 0.000   Median : 1.000   Median : 11.00   Median :  1.000  
 Mean   : 25.54   Mean   : 50.80   Mean   : 8.968   Mean   : 1.178   Mean   : 5.252   Mean   : 23.96   Mean   :  4.562  
 3rd Qu.: 44.00   3rd Qu.: 90.75   3rd Qu.:15.000   3rd Qu.: 2.000   3rd Qu.: 7.750   3rd Qu.: 41.75   3rd Qu.:  4.000  
 Max.   :130.00   Max.   :238.00   Max.   :53.000   Max.   :14.000   Max.   :40.000   Max.   :121.00   Max.   :107.000  
       CS               BB               SO               BA              OBP        
 Min.   : 0.000   Min.   :  0.0   Min.   :  0.00   Min.   :0.0000   Min.   :0.0000  
 1st Qu.: 0.000   1st Qu.:  1.0   1st Qu.:  6.00   1st Qu.:0.1604   1st Qu.:0.2033  
 Median : 1.000   Median :  9.0   Median : 22.50   Median :0.2347   Median :0.3000  
 Mean   : 2.231   Mean   : 19.6   Mean   : 34.03   Mean   :0.2088   Mean   :0.2672  
 3rd Qu.: 3.000   3rd Qu.: 32.0   3rd Qu.: 54.00   3rd Qu.:0.2689   3rd Qu.:0.3296  
```



```
> head(baseball)
# A tibble: 6 × 18
  Last      First   Age     G    PA    AB     R     H  `2B`  `3B`    HR   RBI    SB    CS    BB    SO     BA   OBP
  <chr>     <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <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     0     0    21 0.107  0.107
2 Adduci    Jim      26     3    13    11     2     1     1     0     0     0     0     0     1     2 0.0909 0.167
3 Aguayo    Luis     27    62   146   133    17    28     6     1     4    13     1     1     8    26 0.211  0.255
4 Aguilera  Rick     24    32    57    51     4     8     0     0     2     6     0     0     3    12 0.157  0.204
5 Aldrete   Mike     25    84   256   216    27    54    18     3     2    25     1     3    33    34 0.25   0.349
6 Alexander Doyle    35    18    45    38     2     8     1     0     0     5     0     0     0     8 0.211  0.211
> 
```

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

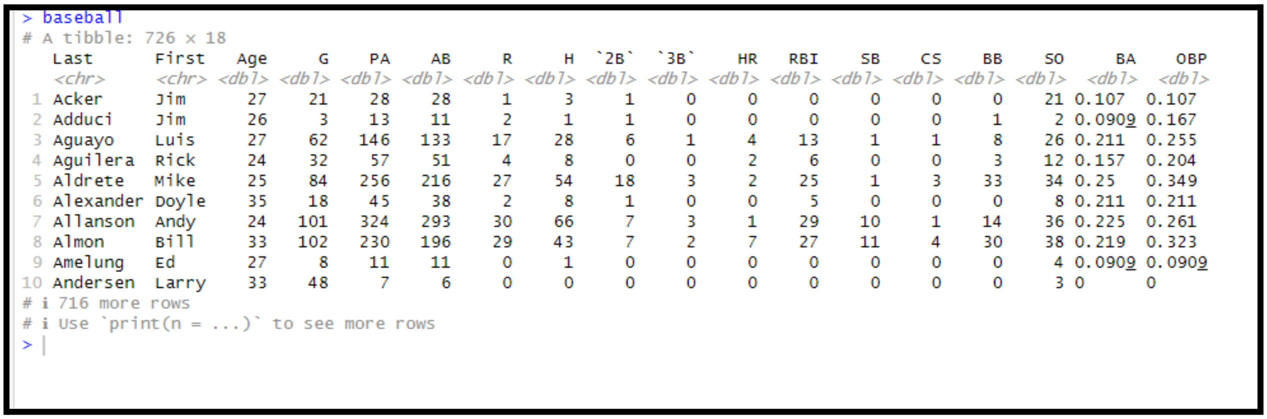
```
# A tibble: 726 × 16
   Last  First   Age     G    PA    AB     R     H `2B`  `3B`    HR
RBI    SB
   <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
<dbl> <dbl>
 1 Acker Jim      27    21    28    28     1     3     1     0     0
0     0
```



```
> baseball
# A tibble: 726 × 18
   Last      First   Age     G    PA    AB     R     H  `2B`  `3B`    HR   RBI    SB    CS    BB    SO     BA    OBP
   <chr>     <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <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     0     0    21 0.107  0.107 
 2 Adduci    Jim      26     3    13    11     2     1     1     0     0     0     0     0     1     2 0.0909 0.167 
 3 Aguayo    Luis     27    62   146   133    17    28     6     1     4    13     1     1     8    26 0.211  0.255 
 4 Aguilera  Rick     24    32    57    51     4     8     0     0     2     6     0     0     3    12 0.157  0.204 
 5 Aldrete   Mike     25    84   256   216    27    54    18     3     2    25     1     3    33    34 0.25   0.349 
 6 Alexander Doyle    35    18    45    38     2     8     1     0     0     5     0     0     0     8 0.211  0.211 
 7 Allanson  Andy     24   101   324   293    30    66     7     3     1    29    10     1    14    36 0.225  0.261 
 8 Almon     Bill     33   102   230   196    29    43     7     2     7    27    11     4    30    38 0.219  0.323 
 9 Amelung   Ed       27     8    11    11     0     1     0     0     0     0     0     0     0     4 0.0909 0.0909
10 Andersen  Larry    33    48     7     6     0     0     0     0     0     0     0     0     0     3 0      0     
# ℹ 716 more rows
# ℹ Use `print(n = ...)` to see more rows
```

15. 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.**

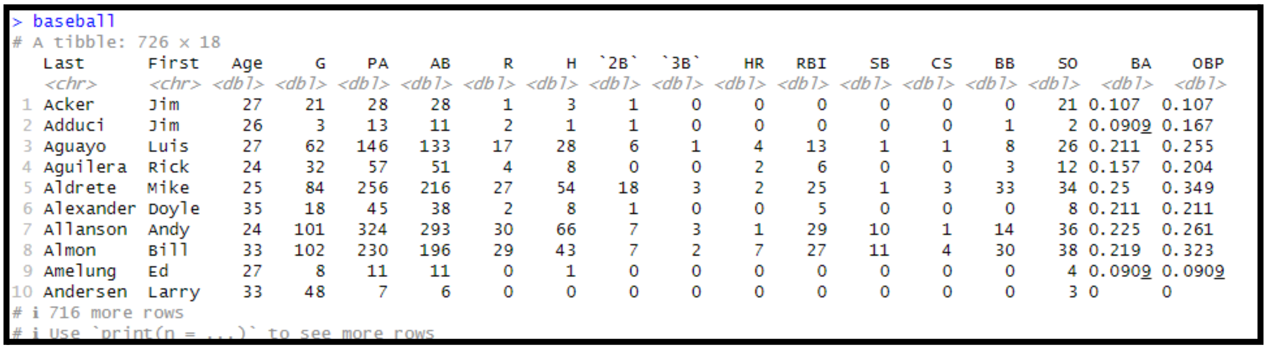
```
# A tibble: 726 × 17
   Last  First   Age     G    PA    AB     R     H `2B` `3B`    HR
RBI    SB
   <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
<dbl> <dbl>
 1 Acker Jim      27    21    28    28     1     3     1     0     0
0     0
 2 Addu… Jim      26     3    13    11     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
```



```
> baseball
# A tibble: 726 × 18
   Last      First   Age     G    PA    AB     R     H `2B` `3B`    HR   RBI    SB    CS    BB    SO     BA    OBP
   <chr>     <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <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     0     0    21 0.107  0.107
 2 Adduci    Jim      26     3    13    11     2     1     1     0     0     0     0     0     1     2 0.0909 0.167
 3 Aguayo    Luis     27    62   146   133    17    28     6     1     4    13     1     1     8    26 0.211  0.255
 4 Aguilera  Rick     24    32    57    51     4     8     0     0     2     6     0     0     3    12 0.157  0.204
 5 Aldrete   Mike     25    84   256   216    27    54    18     3     2    25     1     3    33    34 0.25   0.349
 6 Alexander Doyle    35    18    45    38     2     8     1     0     0     5     0     0     0     8 0.211  0.211
 7 Allanson  Andy     24   101   324   293    30    66     7     3     1    29    10     1    14    36 0.225  0.261
 8 Almon     Bill     33   102   230   196    29    43     7     2     7    27    11     4    30    38 0.219  0.323
 9 Amelung   Ed       27     8    11    11     0     1     0     0     0     0     0     0     0     4 0.0909 0.0909
10 Andersen  Larry    33    48     7     6     0     0     0     0     0     0     0     0     0     3 0      0
# ℹ 716 more rows
# ℹ Use `print(n = ...)` to see more rows
> |
```

16. 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.**

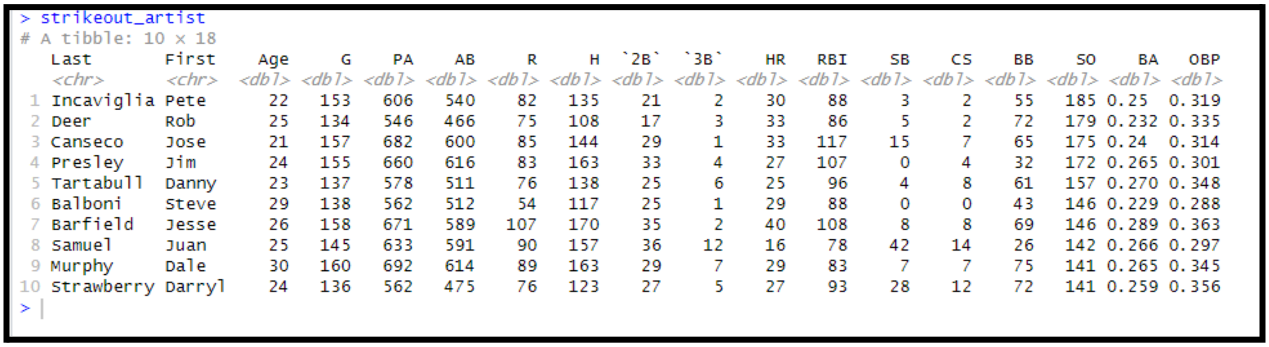
```
# A tibble: 726 × 18
   Last  First   Age     G    PA    AB     R     H `2B`  `3B`    HR
RBI    SB
   <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
<dbl> <dbl>
 1 Acker Jim      27    21    28    28     1     3     1     0     0
0     0
 2 Addu… Jim      26     3    13    11     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>
```



```
> baseball
# A tibble: 726 × 18
   Last      First   Age     G    PA    AB     R     H  `2B`  `3B`    HR   RBI    SB    CS    BB    SO     BA    OBP
   <chr>     <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <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     0     0    21 0.107  0.107 
 2 Adduci    Jim      26     3    13    11     2     1     1     0     0     0     0     0     1     2 0.0909 0.167 
 3 Aguayo    Luis     27    62   146   133    17    28     6     1     4    13     1     1     8    26 0.211  0.255 
 4 Aguilera  Rick     24    32    57    51     4     8     0     0     2     6     0     0     3    12 0.157  0.204 
 5 Aldrete   Mike     25    84   256   216    27    54    18     3     2    25     1     3    33    34 0.25   0.349 
 6 Alexander Doyle    35    18    45    38     2     8     1     0     0     5     0     0     0     8 0.211  0.211 
 7 Allanson  Andy     24   101   324   293    30    66     7     3     1    29    10     1    14    36 0.225  0.261 
 8 Almon     Bill     33   102   230   196    29    43     7     2     7    27    11     4    30    38 0.219  0.323 
 9 Amelung   Ed       27     8    11    11     0     1     0     0     0     0     0     0     0     4 0.0909 0.0909
10 Andersen  Larry    33    48     7     6     0     0     0     0     0     0     0     0     0     3 0      0     
# ℹ 716 more rows
# ℹ Use `print(n = ...)` to see more rows
```

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

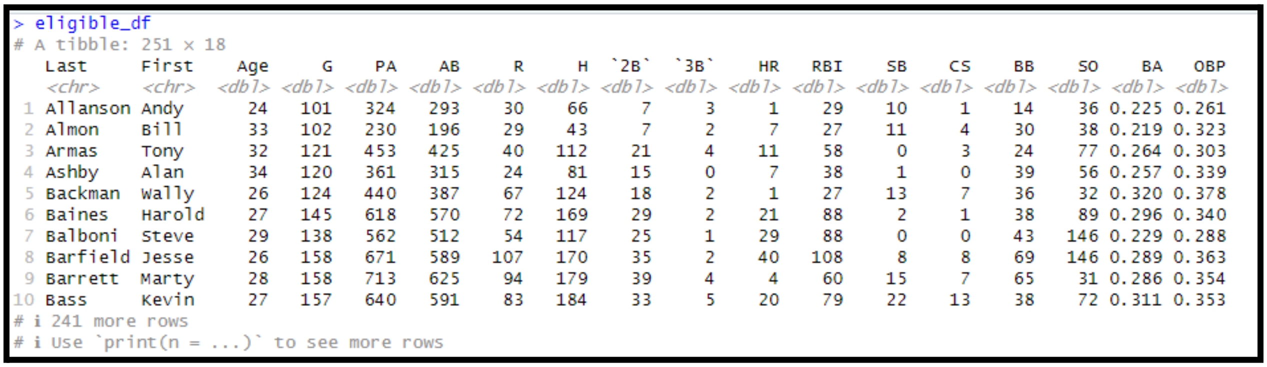
```
# A tibble: 10 × 18
   Last  First   Age     G    PA    AB     R     H `2B`  `3B`    HR
RBI     SB
   <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
<dbl> <dbl>
 1 Inca… Pete     22   153   606   540    82   135    21     2    30
88      3
 2 Deer  Rob      25   134   546   466    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
```



```
> strikeout_artist
# A tibble: 10 × 18
   Last       First    Age     G    PA    AB     R     H  `2B`  `3B`    HR   RBI    SB    CS    BB    SO    BA   OBP
   <chr>      <chr>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1 Incaviglia Pete      22   153   606   540    82   135    21     2    30    88     3     2    55   185 0.25  0.319
 2 Deer       Rob       25   134   546   466    75   108    17     3    33    86     5     2    72   179 0.232 0.335
 3 Canseco    Jose      21   157   682   600    85   144    29     1    33   117    15     7    65   175 0.24  0.314
 4 Presley    Jim       24   155   660   616    83   163    33     4    27   107     0     4    32   172 0.265 0.301
 5 Tartabull  Danny     23   137   578   511    76   138    25     6    25    96     4     8    61   157 0.270 0.348
 6 Balboni    Steve     29   138   562   512    54   117    25     1    29    88     0     0    43   146 0.229 0.288
 7 Barfield   Jesse     26   158   671   589   107   170    35     2    40   108     8     8    69   146 0.289 0.363
 8 Samuel     Juan      25   145   633   591    90   157    36    12    16    78    42    14    26   142 0.266 0.297
 9 Murphy     Dale      30   160   692   614    89   163    29     7    29    83     7     7    75   141 0.265 0.345
10 Strawberry Darryl    24   136   562   475    76   123    27     5    27    93    28    12    72   141 0.259 0.356
> |
```

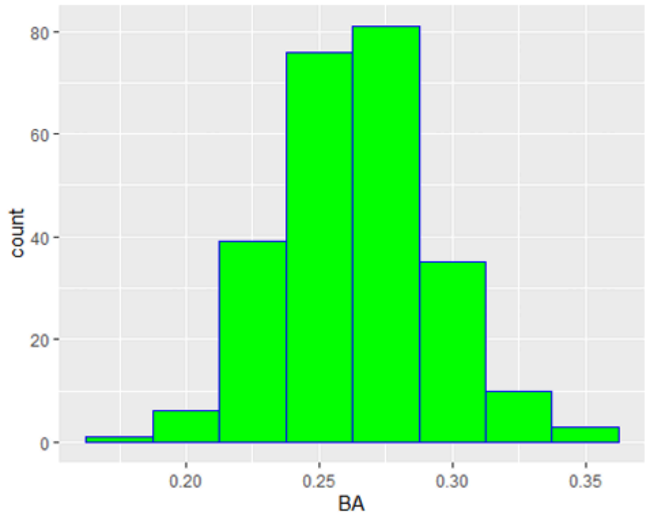
18. To be eligible for end-of-season awards, a player must have either at least 300 at bats or appear in at least 100 games. Keep only the players who are eligible to be considered and store them in a variable called **eligible_df.**

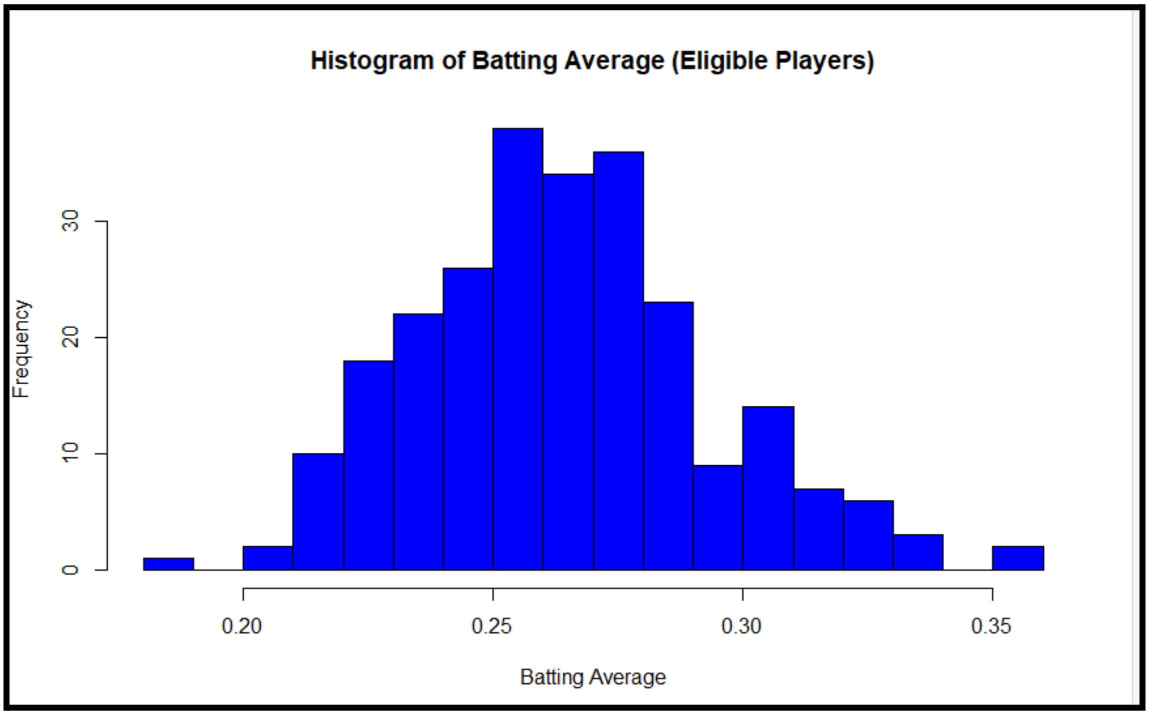
```
# A tibble: 251 × 18
   Last  First   Age     G    PA    AB     R     H  `2B`  `3B`    HR
RBI    SB
   <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
<dbl> <dbl>
 1 Alla… Andy     24   101   324   293    30    66     7     3     1
29    10
 2 Almon Bill     33   102   230   196    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>
```



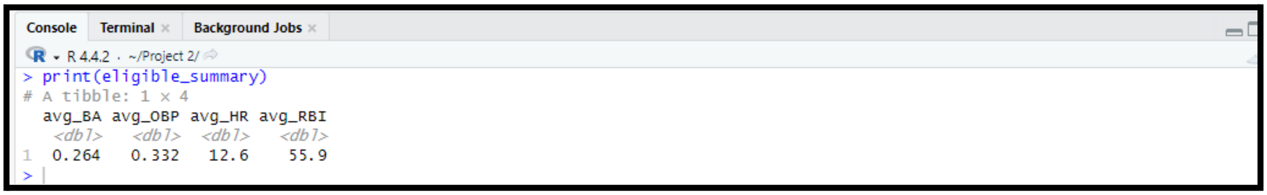
```
> eligible_df
# A tibble: 251 × 18
   Last     First    Age     G    PA    AB     R     H  `2B`  `3B`    HR   RBI    SB    CS    BB    SO    BA   OBP
   <chr>    <chr>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1 Allanson Andy      24   101   324   293    30    66     7     3     1    29    10     1    14    36 0.225 0.261
 2 Almon    Bill      33   102   230   196    29    43     7     2     7    27    11     4    30    38 0.219 0.323
 3 Armas    Tony      32   121   453   425    40   112    21     4    11    58     0     3    24    77 0.264 0.303
 4 Ashby    Alan      34   120   361   315    24    81    15     0     7    38     1     0    39    56 0.257 0.339
 5 Backman  Wally     26   124   440   387    67   124    18     2     1    27    13     7    36    32 0.320 0.378
 6 Baines   Harold    27   145   618   570    72   169    29     2    21    88     2     1    38    89 0.296 0.340
 7 Balboni  Steve     29   138   562   512    54   117    25     1    29    88     0     0    43   146 0.229 0.288
 8 Barfield Jesse     26   158   671   589   107   170    35     2    40   108     8     8    69   146 0.289 0.363
 9 Barrett  Marty     28   158   713   625    94   179    39     4     4    60    15     7    65    31 0.286 0.354
10 Bass     Kevin     27   157   640   591    83   184    33     5    20    79    22    13    38    72 0.311 0.353
# ℹ 241 more rows
# ℹ Use `print(n = ...)` to see more rows
```

19. For eligible players, create a histogram of batting average.

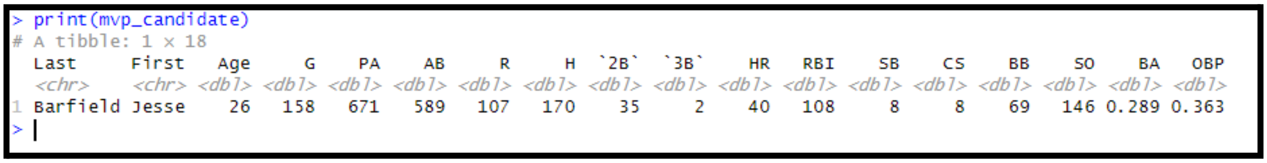




20. Important statistics for baseball players include the on-base percentage (OBP), the number of home runs (HR), the number of runs batted-in (RBI) among others. Analyze the eligible players and select a player that in your opinion is deserving of the Most Valuable Player (MVP) award. This choice must be supported by your data. In your report, you should present your data analysis supported by relevant data points and statistics that supports your recommendation. Produce a concise, written executive summary that focuses on the baseball data analysis. In addition to the title page and citations, it contains an introduction, presentation of written key findings, and a conclusion that contains your recommendations as supported by the data. Your executive summary should adhere to basic APA guidelines.



```
> print(eligible_summary)
# A tibble: 1 × 4
  avg_BA avg_OBP avg_HR avg_RBI
   <dbl>   <dbl>  <dbl>   <dbl>
1  0.264   0.332   12.6    55.9
> 
```



```
> print(mvp_candidate)
# A tibble: 1 × 18
  Last     First    Age     G    PA    AB     R     H  `2B`  `3B`    HR   RBI    SB    CS    BB    SO    BA   OBP
  <chr>    <chr>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 Barfield Jesse     26   158   671   589   107   170    35     2    40   108     8     8    69   146 0.289 0.363
> 
```

The analysis identified notable players in the dataset, such as strikeout leaders and top performers based on key metrics. Eligible players were evaluated, and **Jesse Barfield** emerged as a strong MVP candidate based on his offensive statistics, which surpassed the league averages for eligible players.

This analysis can guide team management and stakeholders in making informed decisions about player awards and team composition.

References

World Happiness Report. (2015). *World Happiness Report 2015 Dataset*. Retrieved from https://worldhappiness.report/ed/2015/

Wickham, H., François, R., Henry, L., & Müller, K. (2023). *tidyverse: Data Science Tools for R*. Retrieved from https://www.tidyverse.org

Firke, S. (2021). *janitor: Simple Tools for Examining and Cleaning Dirty Data*. R package version 2.1.0. Retrieved from https://cran.r-project.org/package=janitor

Baseball Reference. (n.d.). *1986 Major League Standard Batting*. Retrieved January 25, 2025, from https://www.baseball-reference.com/leagues/MLB/1986-standard-batting.shtml

Baseball Dataset. (n.d.). *Source of player statistics*. Details about dataset origin should be included here if available (e.g., the publisher or author of the dataset).