Step by step basic data computing in R

Fernando Iscar Fernandez de Alarcon

2022-09-05

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

Presentation of the script

You will find a step by step introduction to some basic R formulas using a baseball stats dataframe. You will find below some brief explanations and examples, plus, at the end, you will also learn how to make some diverse graphs from the dataset.

Setting up a working directory

First, I'll set up a new working directory located at my PC desktop assigning a wd to variable curr_wd and later using setwd. I'll also use the getwd() function to check if code worked properly after the set up. For privacy reasons I'll hide the wd of my desktop using include=FALSE in R chunks, so neither code or results of this exercise will be displayed in HTLM doc.

It matches! The function getwd () gave me the current wd in working in, which is the one I have just assigned.

Importing a dataset

Let's move on to importing a dataset.

```
df3 <- read.csv("data/baseball.csv" , stringsAsFactors = FALSE)
```

Let's now check the dataframe we have just retrieved. But before that, I'll assign our dataframe with a new name, more convenient than "df3" and will delete the latter with the rm() function:

```
baseball_df <- (df3)
rm(df3)
str(baseball_df)
```

```
## 'data.frame':    86 obs. of  20 variables:
##  $ playerID    : chr  "anderbo01" "averiea02" "bankser01" "barondi01" ...
##  $ birthYear   : int  1935 1931 1931 1932 1953 1925 1930 1927 1925 1924 ...
##  $ birthMonth  : int  9 9 1 10 6 5 4 8 4 12 ...
```

```
##  $ birthDay    : int  29 9 31 13 6 12 5 7 16 11 ...
##  $ birthCountry: chr  "USA" "USA" "USA" "USA" ...
##  $ birthState  : chr  "IN" "OH" "TX" "CA" ...
##  $ birthCity   : chr  "East Chicago" "Cleveland" "Dallas" "San Jose" ...
##  $ deathYear   : int  2015 2015 2015 2015 2015 2015 2015 2015 2016 2015 ...
##  $ deathMonth  : int  3 5 1 4 2 9 11 1 1 12 ...
##  $ deathDay    : int  12 13 23 23 2 22 7 28 10 17 ...
##  $ nameFirst   : chr  "Bob" "Earl" "Ernie" "Dick" ...
##  $ nameLast    : chr  "Anderson" "Averill" "Banks" "Barone" ...
##  $ nameGiven   : chr  "Robert Carl" "Earl Douglas" "Ernest" "Richard Anthony" ...
##  $ weight      : int  210 185 180 165 185 185 200 170 195 180 ...
##  $ height      : int  76 70 73 69 73 67 75 68 74 74 ...
##  $ bats        : chr  "R" "R" "R" "R" ...
##  $ throws      : chr  "R" "R" "R" "R" ...
##  $ debut       : int  1957 1956 1953 1960 1975 1946 1956 1951 1951 1951 ...
##  $ finalGame   : int  1963 1963 1971 1960 1992 1965 1956 1961 1951 1964 ...
##  $ activeYears : int  6 7 18 0 17 19 0 10 0 13 ...
```

Checking how the first 6 variables look in the dataframe with function head():

```
head(baseball_df)
```

```
##    playerID birthYear birthMonth birthDay birthCountry birthState    birthCity
## 1 anderbo01      1935          9       29          USA         IN East Chicago
## 2 averiea02      1931          9        9          USA         OH    Cleveland
## 3 bankser01      1931          1       31          USA         TX       Dallas
## 4 barondi01      1932         10       13          USA         CA     San Jose
## 5 bergmda01      1953          6        6          USA         IL     Evanston
## 6 berrayo01      1925          5       12          USA         MO    St. Louis
##   deathYear deathMonth deathDay nameFirst nameLast       nameGiven weight
## 1      2015          3       12       Bob Anderson     Robert Carl    210
## 2      2015          5       13      Earl  Averill    Earl Douglas    185
## 3      2015          1       23     Ernie    Banks          Ernest    180
## 4      2015          4       23      Dick   Barone Richard Anthony    165
## 5      2015          2        2      Dave  Bergman     David Bruce    185
## 6      2015          9       22      Yogi    Berra  Lawrence Peter    185
##   height bats throws debut finalGame activeYears
## 1     76    R      R  1957      1963           6
## 2     70    R      R  1956      1963           7
## 3     73    R      R  1953      1971          18
## 4     69    R      R  1960      1960           0
## 5     73    L      L  1975      1992          17
## 6     67    L      R  1946      1965          19
```

How many rows/columns we'll find in the dataframe? Let's discover it using nrow/ncol functions below. Also, we'll create meaningful names for both results.

```
nrow(baseball_df)
```

```
## [1] 86
```

```r
ncol(baseball_df)
```

```
## [1] 20
```

```r
baseball_df_nrows <- nrow(baseball_df)
baseball_df_ncols <- ncol(baseball_df)
```

Vectors

Let's create a vector with the height of the three shortest baseball players in our dataset. First, we'll check what is the minimum value for that variable.

```r
min(baseball_df$height)
```

```
## [1] 66
```

```r
sort(baseball_df$height)[1:3]
```

```
## [1] 66 66 67
```

```r
shortest3 <- c(sort(baseball_df$height)[1:3])
```

Frequencies

Table () will give us the unique values in a variable, along with their total occurrences. After running it, we can guess the top 1 and 2 States with more players born in. So we'll calculate the difference between top 1 and 2 assigning some variables and making the correspondent calculation.

```r
table(baseball_df$birthState)
```

```
##
## AL AR CA CO CT FL GA HI ID IL IN KY LA MA MI MO MS NC NJ NY OH OK OR PA SC TX
##  5  4 12  1  2  3  2  1  1  4  2  1  2  5  3  4  2  3  4  6  4  2  1  3  1  4
## VA WA
##  3  1
```

```r
CA_players <- 12
NY_players <- 6
difference_top1and2_birthState <- CA_players - NY_players
difference_top1and2_birthState
```

```
## [1] 6
```

Average

In order to guess the average of the players height, we will use mean() function. Since result will have lots of decimals, we can round it till 2 digits as shown below:

```r
avg_height <- mean(baseball_df$height)
avg_height <- round(avg_height , digits = 2)
avg_height
```

```
## [1] 72.53
```

Removing/adding variables from Dataset

Suppose we are interested in removing variables from the dataset. In this case, we want to remove variable nameGiven. We will first check in the code the number of columns, then, we will remove the variable and check again the number of columns to confirm the success of our procedure:

```r
ncol(baseball_df)
```

```
## [1] 20
```

```r
baseball_df$nameGiven <- NULL
ncol(baseball_df)
```

```
## [1] 19
```

As we can see, we removed the column nameGiven using "NULL". As a result, we now have 19 columns instead of 20.

We will now add a new variable to replace the deleted one which will be named as ageAtDeath. As its name indicates, we will deduct it from comparing the difference between players' birth and death years.

- First we deduct the ageAtDeath variable resting birthYear to deathYear
- Secondly we check if the length of the new variable equals the nº rows in the dataframe. If so, we'll get a "TRUE" as an answer
- Later we add the new column to the df
- Then we rename it. By deafult, it was called "new_col"
- Finally, we can check our code by using the function head() and looking forthe column we have just created

```r
ageAtDeath <- baseball_df$deathYear - baseball_df$birthYear
length(ageAtDeath) == nrow(baseball_df)
```

```
## [1] TRUE
```

```r
baseball_df$new_col <- ageAtDeath
colnames(baseball_df)[which(names(baseball_df)== "new_col")] <- "ageAtDeath"
head(baseball_df)
```

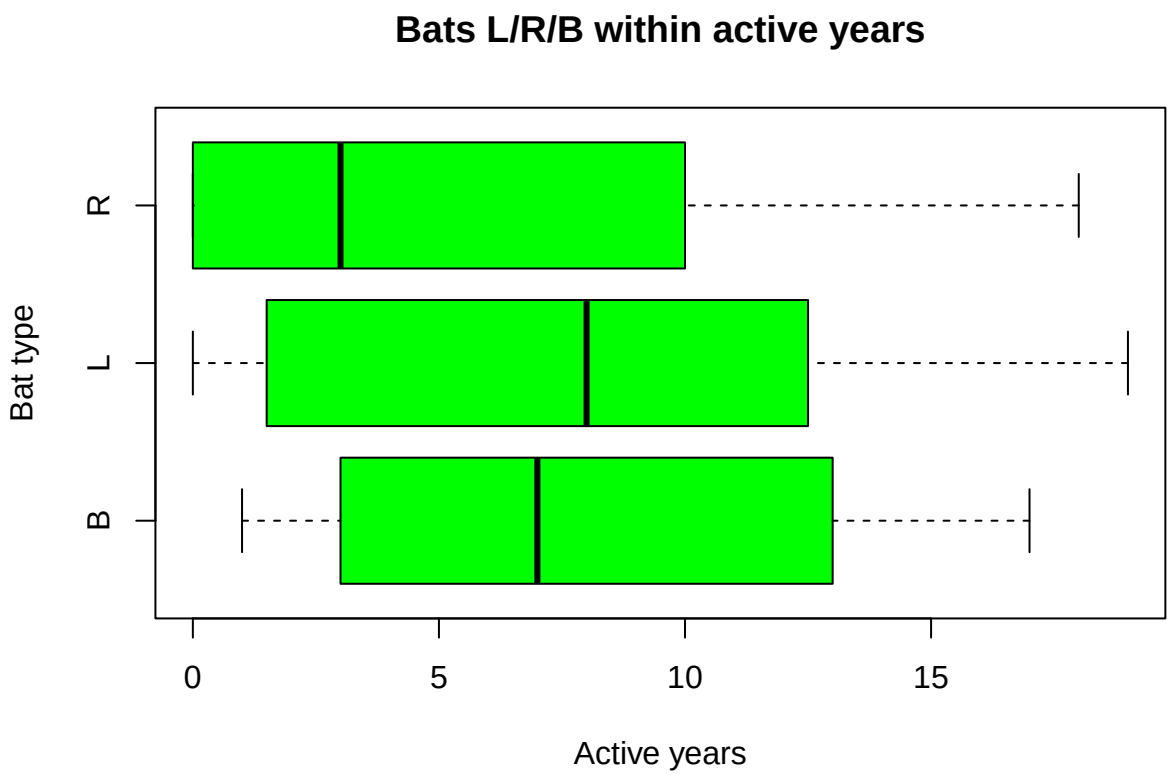
```
##    playerID birthYear birthMonth birthDay birthCountry birthState    birthCity
## 1 anderbo01      1935          9       29          USA         IN East Chicago
## 2 averiea02      1931          9        9          USA         OH    Cleveland
## 3 bankser01      1931          1       31          USA         TX       Dallas
## 4 barondi01      1932         10       13          USA         CA     San Jose
```

```
## 5 bergmda01      1953          6        6          USA         IL     Evanston
## 6 berrayo01      1925          5       12          USA         MO    St. Louis
##   deathYear deathMonth deathDay nameFirst nameLast weight height bats throws
## 1      2015          3       12       Bob Anderson    210     76    R      R
## 2      2015          5       13      Earl  Averill    185     70    R      R
## 3      2015          1       23     Ernie    Banks    180     73    R      R
## 4      2015          4       23      Dick   Barone    165     69    R      R
## 5      2015          2        2      Dave  Bergman    185     73    L      L
## 6      2015          9       22      Yogi    Berra    185     67    L      R
##   debut finalGame activeYears ageAtDeath
## 1  1957      1963           6         80
## 2  1956      1963           7         84
## 3  1953      1971          18         84
## 4  1960      1960           0         83
## 5  1975      1992          17         62
## 6  1946      1965          19         90
```

Boxplot

We will now make a boxplot of variables activeYears and bats, adjust the variables if necessary. We will add title, labels and change the color of the plot.

```
boxplot(activeYears~bats, data = baseball_df, main = "Bats L/R/B within active years", xlab = "Active y
```



Creating a new dataset

We can create a new dataset from the original one. In this case, we will create a dataset using the height and weight of the players:

```
baseball_df2 <- cbind(baseball_df$weight , baseball_df$height)
baseball_df2.0 <- data.frame(baseball_df2)
head(baseball_df2.0)
```

```
##    X1 X2
## 1 210 76
## 2 185 70
## 3 180 73
## 4 165 69
## 5 185 73
## 6 185 67
```

Calculation of BMI using vectors

We'll proceed with calculating the Body Mass Index formula since we already have the height and weight variables of the players.

- The BMI formula = kg/m^2

As per the BMI formula, first we have to do is to pass inches into m and then, elevate m to m^2 (height). Then we can divide it to the weight in kg. However, we have to calculate the kg as well because in the dataframe are presented in pounds.

After all, we will round the result to 1 decimal.

```
height_inch <- (baseball_df$height)
height_m <- height_inch*0.0254
height_msqrd <- height_m^2
weight_pound <- (baseball_df$weight)
weight_kg <- weight_pound*0.453592
playerbmi <- weight_kg/height_msqrd
round(playerbmi, digits = 1)
```

```
##  [1] 25.6 26.5 23.7 24.4 24.4 29.0 25.0 25.8 25.0 23.1 22.9 23.7 26.4 25.0 24.4
## [16] 23.7 21.8 23.0 23.3 24.4 25.1 25.1 25.8 29.0 23.7 23.7 25.4 22.4 27.0 24.4
## [31] 27.8 25.7 26.2 27.8 25.0 25.6 25.8 23.6 25.1 25.4 23.7 27.1 25.8 25.8 23.9
## [46] 24.7 23.7 23.8 28.1 23.1 23.2 23.0 24.4 24.1 26.5 24.4 24.3 25.3 25.8 23.7
## [61] 25.1 23.0 25.7 21.3 25.8 25.1 25.8 25.7 24.4 27.5 26.9 25.7 25.7 24.3 23.1
## [76] 23.0 25.5 23.7 24.8 24.4 23.7 24.4 25.7 26.6 23.8 37.9
```

Now, we are going to add the BMI vector we have just created to the baseball_df2.0 with a similar procedure done before in this script.

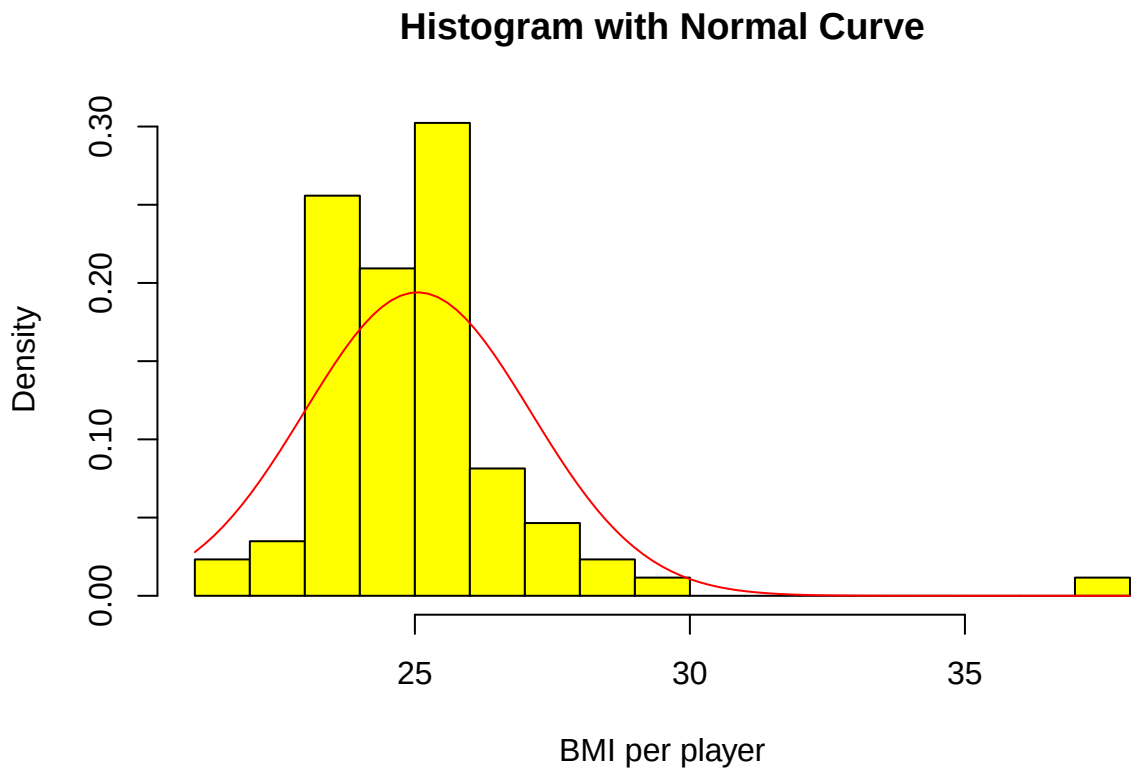
```
baseball_df2.0$new_col <- playerbmi
colnames(baseball_df2.0) <- c("Weight", "Height", "BMI")
head(baseball_df2.0)
```

```
##   Weight Height      BMI
## 1    210     76 25.56172
## 2    185     70 26.54444
## 3    180     73 23.74787
## 4    165     69 24.36597
## 5    185     73 24.40754
## 6    185     67 28.97478
```

Histogram

Histograms are frequently used in data analyses for visualizing the data. Through histograms, we can identify the distribution and frequency of the data. Histograms divide the continues variable into groups (x-axis) and give the frequency (y-axis) in each group. The function that histogram uses is hist().

```
hist(baseball_df2.0$BMI, breaks=20,freq=FALSE, col="yellow",
     xlab="BMI per player",
     main="Histogram with Normal Curve")
curve(dnorm(x, mean=mean(baseball_df2.0$BMI), sd=sd(baseball_df2.0$BMI)),
      add=TRUE, col="red")
```



From the data shown by our histogram, we can deduct the mean is around 25 kg/m^2 for variable "BMI". Also, we can observe there is an outlier close to 38. We'll check this assumptions with the following code:

```
min(baseball_df2.0$BMI)
```

```
## [1] 21.30143
```

```
mean(baseball_df2.0$BMI)
```

```
## [1] 25.04871
```

```
max(baseball_df2.0$BMI)
```

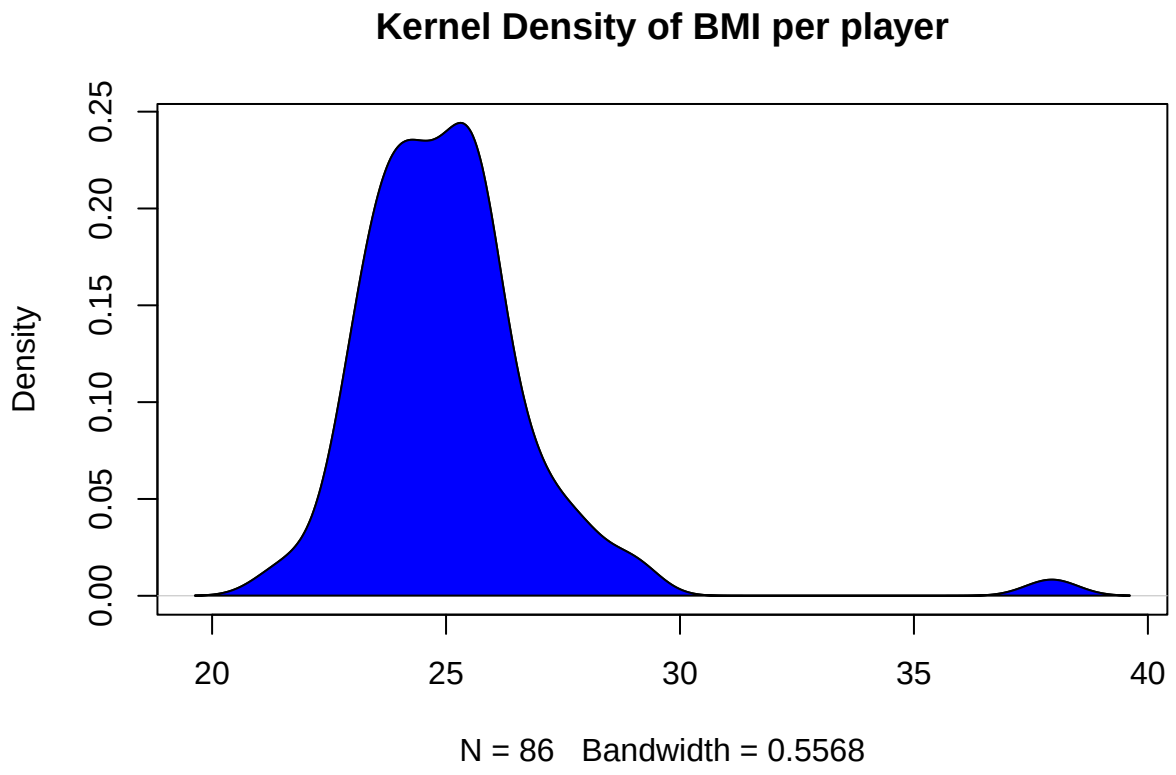
```
## [1] 37.94604
```

Sometimes, in data science, its quite useful to delete this kind of outliers from the observations since its a very unusual observation which can make estimations biased.

Density plot

Using Kernel's density plot, we can easily see the behavior of the density distribution for the Body Mass Index, appreciating better the outlier mentioned before:

```
d <- density(baseball_df2.0$BMI)
plot(d, main="Kernel Density of BMI per player")
polygon(d, col="blue", border="black")
```



Making a ggplot graph

ggplot is a system for creating graphics, based on The Grammar of Graphics. You provide the data, tell ggplot2 how to map variables to aesthetics, what graphical primitives to use, and it takes care of the details. Here is an example of violin plot of throws and height:

```
library(ggplot2)

p <- ggplot(data = baseball_df, aes(x = factor(throws), y = height, fill = throws)) +
geom_violin() +
scale_color_manual(values = c('L' = '#8E518D', 'R' = '#9B7EDE'),
limits = c('L', 'R'), aesthetics = c("colour", "fill")) +
labs(title = 'Violin plot of throws and height', x = 'Throwing hand', y = 'Height')
p
```