Midterm II Key

Fall 2018

##### 1. First write code that loads the workspace “Midterm II.RData”. This should read five different tibbles into your workspace.

The code is:

load("C:/Users/rdn3.BYU/Documents/GitHub/r4ds/Midterm II/Midterm II.RData")

##### 2. Write code that recreates the tibbles “player” and “player\_attributes” by importing the files “player.csv” and “player\_attributes.csv”. Compare the tibbles that you create with those in the Midterm II workspace to make sure that they are the same.

We use the read\_csv function to read the files. We have to make sure that the file structure is correct when we call the read\_csv function. The default is that we have variable names in the first row of the file.

player\_test <- read\_csv("../../Data/player.csv")  
player\_attributes\_test <- read\_csv("../../Data/player\_attributes.csv")

If we compare player with player\_test, we can see that they both have 11,060 observations and 3 variables. Similarly, the comparison of player\_attributes with player\_attributes\_test gives 183,978 observations and 4 variables.

##### 3. Write code that recreates the season factor variable in the player\_attributes tibble. Season is defined as the year from the match\_date variable. Make sure your code converts your variable into a factor.

The seasons are first defined as the integers 2007 through 2016. We add the variable season to the player\_attributes tibble using the mutate command. We also extract the year from the match\_date variable and then declare that season is a factor with the levels specified by the seasons object that was declared in the first step of this section.

seasons <- 2007:2016  
player\_attributes <- player\_attributes %>%   
 mutate(season = year(match\_date) %>% factor(levels = seasons))

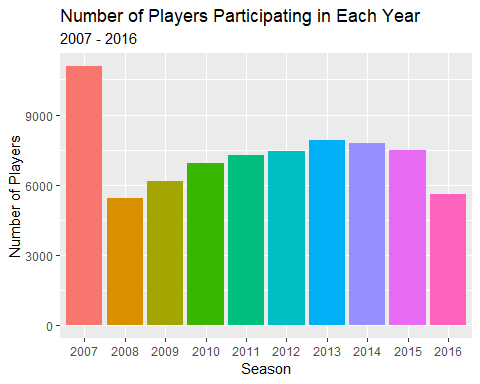
##### 4. We want to determine the total number of players that participate in any given season by using group\_by, summarise, and geom\_bar. Write code that creates the following bar chart:

These are the steps to create the graph:

1. Group the player attributes by player and then season.
2. Use the summarise function to count the number of games for each player during each season.
3. Pipe the resulting tibble into ggplot.
4. Calling geom\_bar triggers another count of players in each season.
5. Label the graph appropriately.

The code to accomplish the above is given by:

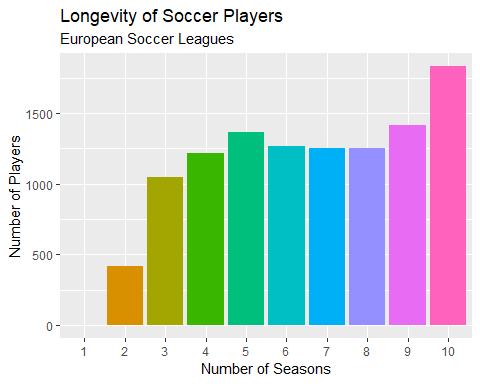
player\_attributes %>%   
 group\_by(player\_id, season) %>%   
 summarise(count = n()) %>%   
 ggplot(aes(x = season, fill = season)) +  
 geom\_bar(show.legend = FALSE) +  
 labs(  
 title = "Number of Players Participating in Each Year",  
 subtitle = "2007 - 2016",  
 x = "Season",  
 y = "Number of Players"  
 )



##### 5. Explain the code in the script file that creates the following graph:

For the following script:

# Number of seasons played  
number\_of\_seasons <- player\_attributes %>%   
 group\_by(player\_id, season) %>%   
 count() %>%   
 group\_by(player\_id) %>%   
 count()  
colnames(number\_of\_seasons) <- c("player\_id", "number\_of\_seasons")  
number\_of\_seasons <- number\_of\_seasons %>%   
 mutate(number\_of\_seasons = factor(number\_of\_seasons, levels = 1:10))  
  
number\_of\_seasons %>%   
 ggplot(aes(x = number\_of\_seasons, fill = number\_of\_seasons)) +  
 geom\_bar(show.legend = FALSE) +  
 labs(  
 title = "Longevity of Soccer Players",  
 subtitle = "European Soccer Leagues",  
 x = "Number of Seasons",  
 y = "Number of Players"  
 )



The steps are:

1. Create a new tibble called number of seasons beginning with the player\_attributes tibble.
2. Group by player\_id and season as done in the previous problem.
3. Counting once again gives the number of games played by each player in each season.
4. Grouping next by player id and then counting again gives the number of seasons in each each player participated.
5. Name the first column of the resulting tibble “player\_id” and the second “number\_of\_seasons”.
6. Redefine the “number\_of\_seasons” variable as a factor between 1 and 10. (There are only ten possible seasons in this data set.)
7. Pipe the number\_of\_seasons tibble into ggplot and count the number of players with one, two, three, etc. seasons using the implicit counting in geom\_bar.
8. Label the graph appropriately.

##### 6. Write a function named age that will create a variable age for each player based on the birthday of the player and the day of the match.

The following code will calculate the age of the player when they participated in each match. The first part of the function calculates the interval between the birthday and match date and then divides by one year to give the age as a decimal or double.

### Function to calculate the age as of the player for each match  
age <- function(birthday, match\_date){  
 (birthday %--% match\_date) / years(1)  
}  
ymd(19510414) %--% ymd(20181206) / years(1)

## [1] 67.64658

##### 7. Interpret the code in the script file that creates the tibble player\_attributes\_complete.

The objective of this code is to create a tibble that adds each players information to the player\_attributes tibble so that we can complete the analysis in parts 8 through 10.

player\_attributes\_complete <- player\_attributes %>%   
 left\_join(player, key = "player\_id") %>%   
 mutate(age = age(birthday, match\_date))

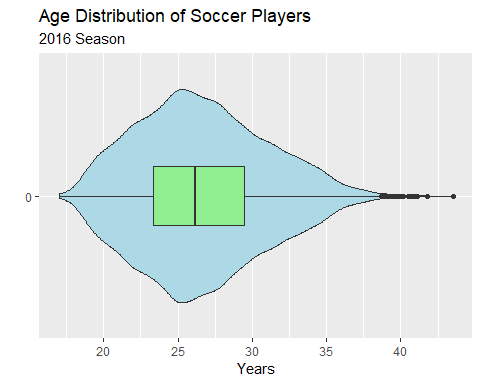
The above steps accomplish this:

1. Use a left join to append the information in players to each row of player attributes.
2. Use mutate to create a new variable age which is the age of the player as a decimal at the date of each match in which the player participates.

##### 8. We are interested in the distribution of ages of the players that participated in the 2016 season. Recreate the player\_attributes\_2016 tibble and then write code that creates a violin plot for the ages of the players who participated in the 2016 season.

One possibltility for creating the desired violin plot is the following:

# Violin plot of the ages of those that participated in 2016  
player\_attributes\_2016 <- player\_attributes\_complete %>%   
 filter(year(match\_date) == 2016)  
  
player\_attributes\_2016 %>%   
 ggplot(aes(x = factor(0), y = age)) +  
 geom\_violin(fill = "lightblue") +  
 geom\_boxplot(width = 0.25, fill = "lightgreen") +  
 coord\_flip() +  
 labs(  
 title = "Age Distribution of Soccer Players",  
 subtitle = "2016 Season",  
 x = "",  
 y = "Years"  
 )



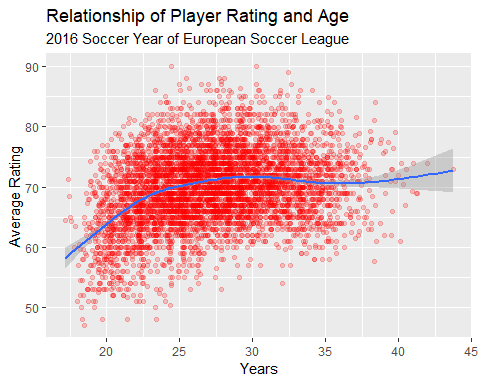
The following steps occur in this code:

1. Create a new tibble called player\_attributes\_2016 from player\_attributes\_complete from the previous problem. We then choose only those observations that correspond to the 2016 season.
2. Pipe this tibble into ggplot. The desired box and violin plots require both x and y arguments. We declare that x is a one level null factor.
3. The two geom’s first create a violin density trace and then superimpose a boxplot on top of that.
4. The coord\_flip switches the x and y axes so that we have a horizontal violin plot.
5. The finally lines label the graph appropriately.

##### 9. Explain the code in the script file that creates the following graph:

The following code generates a smoothed scatterplot:

# Relationship of Age and Average Rating for 2016 players  
player\_attributes\_2016 %>%   
 group\_by(player\_id) %>%   
 summarize(  
 average\_rating = mean(rating)  
 ) %>%   
 left\_join(player, key = "player\_id") %>%  
 mutate(age = age(birthday, ymd(20160701))) %>%   
 ggplot(aes(x = age, y = average\_rating)) +  
 geom\_point(position = "jitter", alpha = 1/5, shape = 19, color = "red") +  
 geom\_smooth() +  
 labs(  
 title = "Relationship of Player Rating and Age",  
 subtitle = "2016 Soccer Year of European Soccer League",  
 x = "Years",  
 y = "Average Rating"  
 )



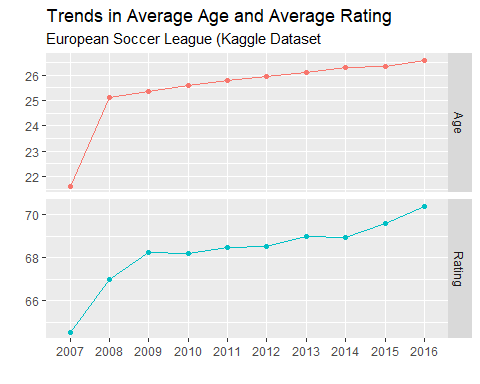
This graph is drawn using the following steps:

1. First group the player\_attributes\_2016 by player id and then calculate their average rating all the games in which they participated. The result is a tibble that has player id and average rating.
2. Because we lost the birthday variable in the calculations in step 1, put the birthday variable back in with a left join.
3. Now calculate age realtive to the date July 1, 2016.
4. Pipe the result into ggplot and create a scatterplot. We need to jitter the points because we have thousands of observations. We also set the transparency to 20% and choose the plotting symbol of a red filled circle.
5. We add a loess smoothed line which is the default (including a confidence interval).
6. Add the appropriate labeling for the graph.

##### 10. Interpret the following code in the script file that compares the time series for average age and average rating.

The following code creates looks for time series trends in age and rating:

# Time series of average age and average rating  
player\_attributes\_complete %>%   
 group\_by(season) %>%   
 summarize(  
 Age = mean(age, na.rm = TRUE),  
 Rating = mean(rating, na.rm = TRUE)  
 ) %>%   
 gather(Age, Rating, key = "measure", value = "amount") %>%   
 ggplot(aes(y = amount, color = measure)) +  
 geom\_point(aes(x = season), show.legend = FALSE) +  
 geom\_path(aes(x = as.numeric(season)), show.legend = FALSE) +  
 facet\_grid(rows = vars(measure), scales = "free\_y") +  
 labs(  
 title = "Trends in Average Age and Average Rating",  
 subtitle = "European Soccer League (Kaggle Dataset",  
 x = "",  
 y = ""  
 )



This graph is the result of the following steps:

1. Group the player\_attributes\_complete tibble by season
2. Calculate the mean age and mean rating for each season. This gives a tibble with three columns: season, Age, and Rating.
3. Create a long tibble using gather. This stacks Age and Rating into one column of values called amount and another column measure which is either Age or Rating. The third column is the season.
4. Pipe the result into ggplot to create the time plots with season on the x axis and measure on the y axis.
5. The geom\_point puts points on the graph.
6. The geom\_path connects the points.
7. The facet creates two graphs, one for Age and one for Rating. The “free\_y” argument allows the scales of the y to correspond to the ranges of the respective Age and Rating variables.
8. The labs creates the appropriate labeling.