Semester assignment in STV2020: Subtasks on FPL

27/05/2021

1.0. Introduction

In this paper, I will wrap up a series of subtasks regarding Fantasy Premier League (https://fantasy.premierleague.com/) (FPL hereafter) through the use of RStudio. There is no denying that FPL is growing more popular. As of today, the game platform has over 7 million users worldwide. There are several ways to play the game. FPL is convoluted in the sense that one can play the game in many ways, applying comprehensive strategies, competing with many people, and it has an extensive bonus- and point system.

The content of this paper is the following: I will present a (i) performance chart that displays the performance of the users within a mini league over a period of 35 weeks and (ii) analyze and visualize one player's journey as an example to how one can retrieve data on a single FPL-user. In addition, I will make use of (iii) global data. All subtasks will, once wrapped up, display the ways to which one can analyze FPL and how useful R can be for managing a great deal of data.

2.0. Fantasy Premier League (FPL)

First, we are in need of further understanding what FPL is. Fantasy Premier League is a virtual game where users all over the world come together and decide on which football players in Premier League they are to include on their teams. The FPL follows the same deadlines and scheduling as Premier League does. The users compete by joining mini-leagues and major leagues.

The principle is fairly straight-forward: One gathers many points week by week by including good footballers on a team. It is possible to apply tactics to the game, although some randomness in terms of player performances may be present. However, given the convoluted point system and bonus system, the available tactics for each user are many. For instance, one can analyze a fixture chart such as this one, where I began by loading thr required packages `fplscrapr`, `dplyr` and `ggplot2`.

```
library(fplscrapR)
library(dplyr)
library(ggplot2)
```

Then, I created some new variables to get both types, namely `home` and `away`:

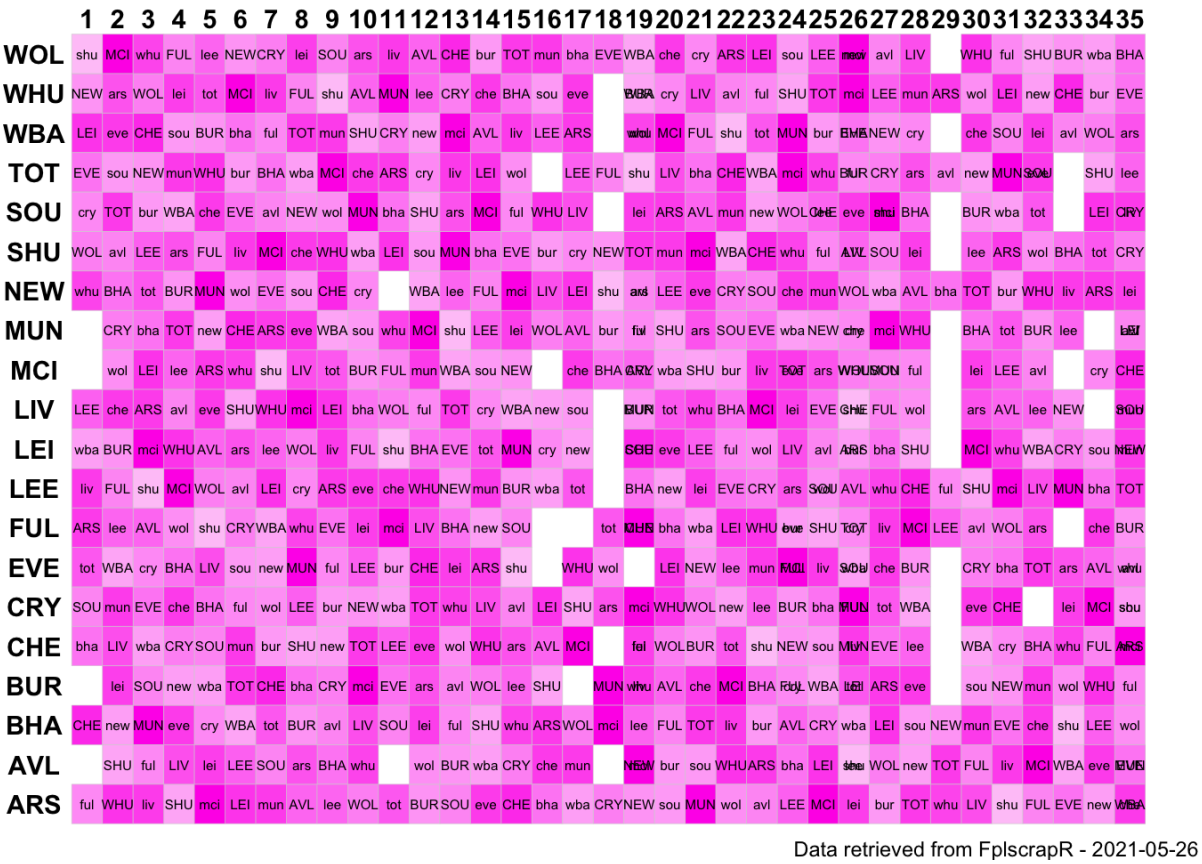
```
fdr <- get_fdr()

gamelist <- get_game_list()


fdrfixtures <- rbind(
  gamelist %>% mutate(team=home,oppo=away,homeaway="home"),
  gamelist %>% mutate(team=away,oppo=tolower(home),homeaway="away"))
```

I created a loop, and wanted to return many fixtures, where I also made it possible to define strength of each team:

```
for (i in 1:nrow(fdrfixtures)){
  ifelse(fdrfixtures$homeaway[i]=="home",
    fdrfixtures$fdr[i] <- fdr$strength_overall_away[which(fdr$short_name==toupper(fdrfixtu
res$oppo[i]))],
    fdrfixtures$fdr[i] <- fdr$strength_overall_home[which(fdr$short_name==toupper(fdrfixtu
res$oppo[i]))])
}
```

Then, I plotted it. The stronger the color of pink, the more difficult the match for the team in play.

```
fdrfixtures %>%
  filter(GW %in% 1:35) %>%
    ggplot() +
      geom_tile(aes(x=GW,y=team,fill=fdr),colour="lightgrey") +
      geom_text(aes(x=GW,y=team,label=oppo),size=2) +
      theme_void() +
      theme(axis.text = element_text(face = "bold")) +
      theme(axis.text.y = element_text(margin=margin(0,-20,0,0))) +
      scale_x_continuous(position="top",breaks=1:35) +
      labs(caption=paste("Data retrieved from FplscrapR - ",Sys.Date(),sep="")) +
      scale_fill_gradient2(guide = F, low="#fce6fa",mid="#ff87f4",high="#ff00e8", midpoint
=median(fdrfixtures$fdr))
```



I also defined the breaks (35 gameweeks), and cited `FplscrapR` as source of the data. As one can see only by watching the fixture, each user of FPL has a massive amount of data, tactics and players to be familiarized with. *This fixture was inspired by wiscostret (https://wiscostret.github.io/fplscrapR/articles/fdrtable.html).*

Some ground rules are present in FPL. One may only have 15 footballers available each “gameweek”, with different formations available, but the requirement is to have 4 players benched. One gets points for a player if goals are scored, assists provided, clean sheets acquired, when a player has played for such and such minutes, bonus points, man of the match (best player), shots saved (for goal keepers), etc. The bonus point system is extensive as well.

Additionally, there are many “chips” that can be used throughout the football season, such as the wildcard (where one can make transfers without additional costs), free-hit (where one may change the existing team for one gameweek), triple-captain (where the choice of captain will triple the points received during the gameweek) and bench-boost (where the points scored by the benched players will be added to the overall score of the gameweek). These options make FPL even more diverse in the various ways of which the users play their game and choose their strategies.

2.1. Mini leagues in FPL

I have chosen to make mini leagues central for this paper. Mini leagues are subgroups of users in FPL competing against each other. Usually, a mini league consists of about 10-15 users, usually they know each other. There is a table available for each mini league to view the current ranking of the users consisting of accumulated points throughout the Premier League season.

3.0. Case: A mini-league in FPL

The case study of mini leagues consists of a selection made of 14 players in a mini league called “FIF-league”. FIF-league stands for the “Frogner Idrettsforening” league. This group are friends competing against each other in the season 2020/21 of Premier League in FPL.

Special thanks to the FIF-league for allowing me to make use of their league data for this paper.

4.0. Data

To track the trends of the FIF-league, I had to retrieve (a) the league ID and (b) the user IDs to each league member. First, I retrieved (a) through a volunteer from the league who gave it to me, then I retrieved (b) by finding each user ID in the URL at FPL-website (https://fantasy.premierleague.com/). In addition, I received consent from the league members to use their decisions for the purpose of writing this paper. Second, the page of the league displays the user IDs attached to every league member. It is public information, available for everyone to find. This makes for transparent and accessible data with the potential for further verification.

The package `fplscrapR` is a package using the FPL JSON API. It consists of functions that help R-users parse, collect and consequently analyze data from the official FPL website. fplscrapR (https://wiscostret.github.io/fplscrapR/) was developed by Rasmus Christensen and Eivind Hammers. The package is of great significance for this paper, as it is used in almost all of the following subtasks.

Additionally, I got much help from using the `fplr` (https://ewenme.github.io/fplr/) package. In addition to these, the `tidyverse` package is used. `tidyverse` is a collection of packages used to manage data in R. The goal is to have manageable data that is tidy, meaning: well organized, easy to work with, easy to document and easy to share. Some of the core packages from `tidyverse` used in this paper are `ggplot` for visualizations and `dplyr` for manipulating data frames.

4.1. Preparations

We begin by loading…

```
library(fplscrapR)
library(fplr)
library(remotes)
library(tidyverse)
library(jsonlite)
```

And some more loading…

```
library(stringr)
library(fplscrapR)
library(fplr)
library(dplyr)
```

Then, I use the `fplscrapR` package to load the information from the Fantasy Premier League website.

First I define a string of the relevant user ids using the `cbind` function. Then I define the variable `myleague_df` in addition to loading the package `dplyr`.

```
#I have gathered the league members' IDs.
ids <- c(4420185,3407983,540808,134972,4487263,3863412,3728201,1174865,4667552,4337609,429
7961,3091921,2872750,3093408,931424)
myleague <- lapply(ids, get_entry_season)
get_entry_season(4667552)
```

Then, I load `dplyr`. Additionally, I will create a data frame called `myleague_df`:

```
library(dplyr)
myleague_df <- bind_rows(myleague)
myleague_df <- arrange(myleague_df,event,overall_rank)
```

In line with tidyverse, I tidy up the code by using pipes (%>%). In addition, I use the `mutate` function to add new variables and preserve the existing ones.

```
myleague_df <- myleague_df %>%
  dplyr::group_by(event) %>%
  dplyr::mutate(week_rank = rank(overall_rank, ties.method = 'first'))
```

Next, I load the `kableExtra` package. I then use `kable()` to construct a complex visualization of the performance of the league members of FIF.

```
library(kableExtra)
kable(head(myleague_df))
```

4.2. Performance chart of a mini-league in FPL

I am not interested in the last names of the league players for privacy concerns. First, it would not look tidy and visually appealing in a chart to have long names featured. Second, I am interested in preserving some anonymity of the users. As a way of tidying up the visualization, I have therefore used the `$first_name` function, allowing for the visualization to only feature the users' first names.

```
myleague_df$first_name <- myleague_df$name
myleague_df$first_name <- sub(" .*$", "", myleague_df$first_name)
```

Now, it is time for the main part of the code consisting of visuals and so on. I use `ggplot` (as part of tidyverse) to visualize the performance of the league members. The events displayed are gameweeks up until week 35 of the Premier League season. In total, there are 37 gameweeks for the season of 2020/21, but due to time limit, I will only include the first 35 weeks.

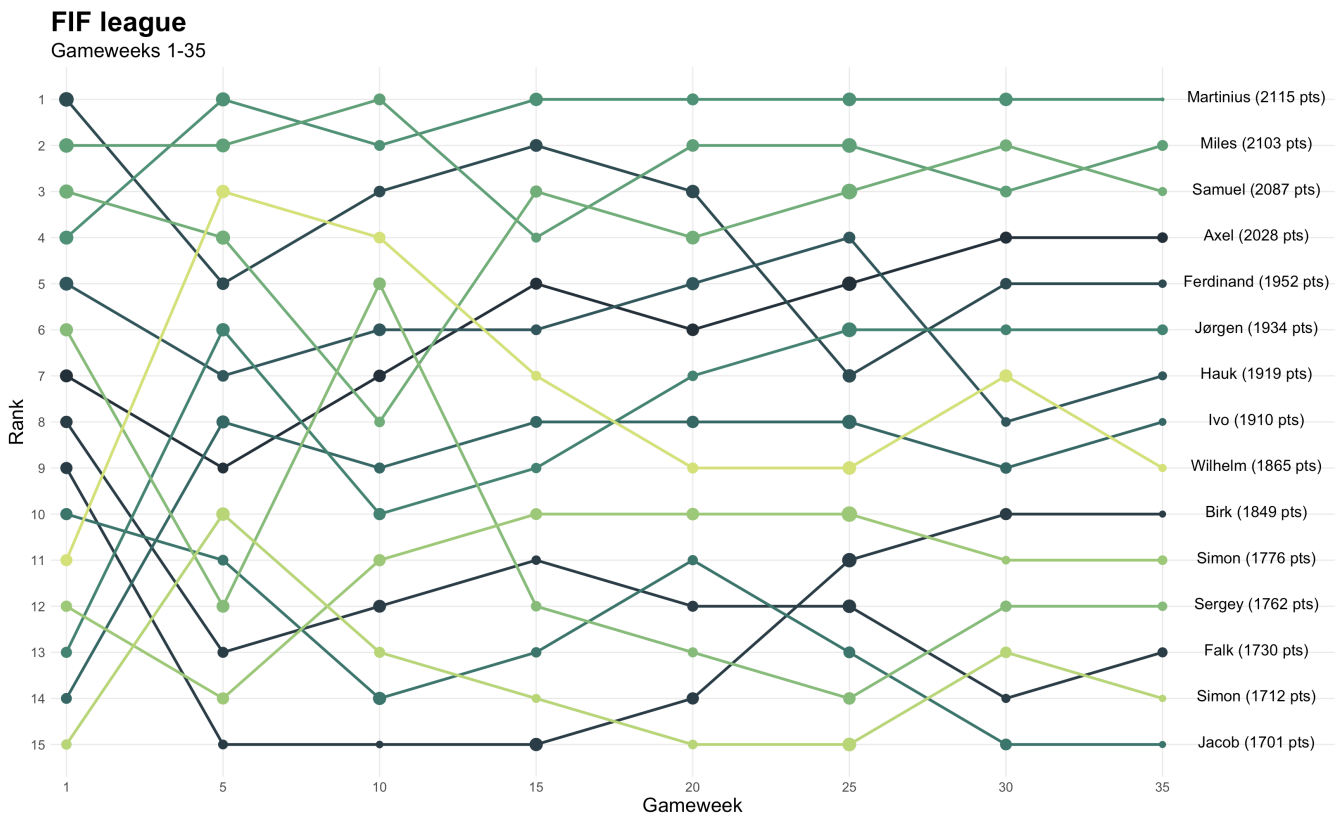
For illustrative purposes, I have subset the code to only show the events with an interval of 5 (5-10-15…) except from the first interval (1-5). This is also to make it more intuitive for the reader to track the changes in the performance. Had I chosen to have an interval of 1, in addition to the amount of users included (14) it would be too difficult for the reader to track the long-term decisions of the players.

There was a problem, however. When first attempted to have an interval of 5, the chart itself did not change accordingly, only the x-axis was changed in terms of only displaying gameweeks with 5 in interval. This was, however, solved, as one can see in the code below. For some reason, it is not possible to run the code in

rmarkdown, however, it did work after several attempts in the script, and I have included an image (screenshot) below of the result.

I have also chosen the colors in the chart through a color-picker, in line with the football theme.Subtitles, names of axes Y and X and headings were also named here. Below the code, one may see the end result.

```
library(ggplot2)
ggplot(subset(myleague_df, event %in% c(1, 5, 10, 15, 20, 25, 30, 35)), aes(x = as.factor
(event), y = week_rank, group = name)) +
  geom_line(aes(colour = name), size = 1.2) +
  geom_point(aes(colour = name, size = points)) + scale_y_reverse(breaks = seq(1,15,1), la
bels = seq(1,15,1)) + theme_minimal() +
  geom_text(data = myleague_df[myleague_df$event == max(myleague_df$event),
  ], aes(label = paste0(first_name,' (',total_points,' pts)')), vjust = 0.25, nudge_x = 0.
6, size = 5) +
  scale_x_discrete(breaks = c(1, 5, 10, 15, 20, 25, 30, 35), expand = expansion(add = c(0.
1, 1.1))) +
  theme(legend.position = 'none',
        panel.grid.minor = element_blank(),
        axis.ticks.y = element_blank(),
        plot.title = element_text(size = 25, face = 'bold'),
        plot.subtitle = element_text(size = 18),
        axis.title = element_text(size = 18),
        axis.text = element_text(size = 12),
        plot.margin = unit(c(1, 1, 1, 1), "cm")) +
  labs(title = paste0(league_title = 'FIF league'), x = 'Gameweek', y = 'Rank', subtitle =
"Gameweeks 1-35") +
  scale_colour_manual(values = c('#25303A','#2A3D46','#2A3D46','#2D4B51','#31585C','#35676
5','#3B756C','#448472','#509276','#60A078','#72AE79','#87BC7A','#9EC979','#B8D679','#D4E27
A'))
```



Although many alterations were made, the skeleton of the chart (meaning the colors, structure) came from this website. (https://rforjournalists.com/2019/11/12/how-to-track-your-fantasy-football-league-using-r/)

5.0. Using Wilhelm to track his decision-making

So: What are the main trends among the users? First of all, we have to ask: Who had the most dynamic changes? There are especially three players with the most changes according to the performance chart. Sergey, Wilhelm and Birk had all decisions that did and did not pay off during these 35 weeks.

I am most interested in examining Wilhelm's decisions in this paper. By extracting data on Wilhelm, we may track his decision history. We can thereby compare different choices made to the chart, and worldwide. We can use the function `fpl_get_user` to extract information on users via `fplscrapR` and the package `fplr`. First, let's load the packages necessary.

5.1. Tracking the decision-making of a FPL-user

```
remotes::install_github("wiscostret/fplscrapR")
library(fplscrapR)
library(fplr)
```

First, I would like to know information on the 5th gameweek.

```
fplscrapR::get_round_info(round = 5)
```

Second, the most captained player's name.

```
fplscrapR::get_player_name(playerid = 390)
```

The most captained was Heung-Ming Son (Tottenham). Lets see if that matched with the captain choice of Wilhelm in gameweek 5. Wilhelm's user ID is 4667552.

```
get_entry_player_picks(entryid = 4667552, gw = 5)
```

Wilhelm captained Son like most users did, in addition to vice cap on Vardy. Lets now see what changes. Things somehow went south, so lets do the same with the gameweeks 6-15.

```
library(fplscrapR)
fplscrapR::get_round_info(round = 6)
fplscrapR::get_player_name(playerid = 254)
```

The tables have turned, as Son is now the most vice-captained and Salah the most captained by gameweek 6. As we know, Son had a killer gameweek in GW 6, meaning many users made mistake by not captaining him this week. However, he underperformed in GW 5, leading to the inevitable decision to vice-cap him. Salah, on the other hand, did well, and earned the cap by most players in gameweek 6. The question is: what did Wilhelm do? Did he make the same mistake as most did?

```
get_entry_player_picks(entryid = 4667552, gw = 6)
get_entry_player_picks(entryid = 4667552, gw = 7)
```

As it turns out, yes. Wilhelm captained Salah, discarded Son completely, vice-captained Kane. Then, he swapped back Son, who, due to injury, did not play in GW 7, perhaps leading to Wilhelms decline in the league.

5.2. Creating a data set for a FPL-user

What happened later on with Wilhelm? As I have shown, one can retrieve data on his performance by using the different functions, but this is highly inefficient and repetitive. Therefore, I save the results as R objects and give them names according to the week (Wilhelm 1/w1).

```
w1 <- get_entry_player_picks(entryid = 4667552, gw = 1)
w2 <- get_entry_player_picks(entryid = 4667552, gw = 2)
w3 <- get_entry_player_picks(entryid = 4667552, gw = 3)
w4 <- get_entry_player_picks(entryid = 4667552, gw = 4)
w5 <- get_entry_player_picks(entryid = 4667552, gw = 5)
# (...)
```

I will now create a complete data set that contains all the valuable information from Wilhelm's season so far. As the data sets have the same exact variables, I can use `bind_rows` to merge the data sets vertically. I join the data sets:

```
wilhelm_full <- bind_rows(w1, w2, w3, w4, w5, w6, w7,
                          w8, w9, w10, w11, w12, w13,
                          w14, w15, w16, w17, w18, w19,
                          w20, w21, w22, w23, w24, w25,
                          w26, w27, w28, w29, w30, w31,
                          w32, w33, w34, w35)
```

I still need to retrieve some more information. I use `get_entry_season`, also from `fplscrapR`, to save information about Wilhelm's score, rank, and transfers.

```
wilhelm_season <- get_entry_season(4667552)
```

Since the new data set contains different columns, I need to use `dplyr`'s merging functionality. This is the function `_join`, which can be either `full_join`, `left_join`, `right_join`, `inner_join` or even `anti_join`. In this case, I wish to retrieve all the observations from the `season` data set, and will thus have to use `full_join`.

```
wilhelm_full <- full_join(wilhelm_full, wilhelm_season, by = "event")
```

I `rename` two of the variables, and I "unselect" the variables that I don't need with the `select` function.

```
wilhelm_full <- wilhelm_full %>%
  rename("gameweek" = "event",
         "user_id" = "entry")

wilhelm_full <- wilhelm_full %>%
  select(-c("value", "rank", "rank_sort"))
```

I then create a new first name variable, and remove the last name variable.

```
library(dplyr)

wilhelm_full$first_name <- sub(" .*$", "", wilhelm_full$name)

wilhelm_full <- wilhelm_full %>%
  select(-c("name"))
```

Then, I reorder the variables:

```
wilhelm_full <- wilhelm_full[c(16, 8, 7, 2:6, 12:15, 9:11)]
```

… and visualize the results in a table by using the `kable` function.. I subsetted it and made a table more visually appealing.

```
wilhelm_full[c(1:10)] %>%
  head() %>%
  kable() %>%
  kable_styling()
```

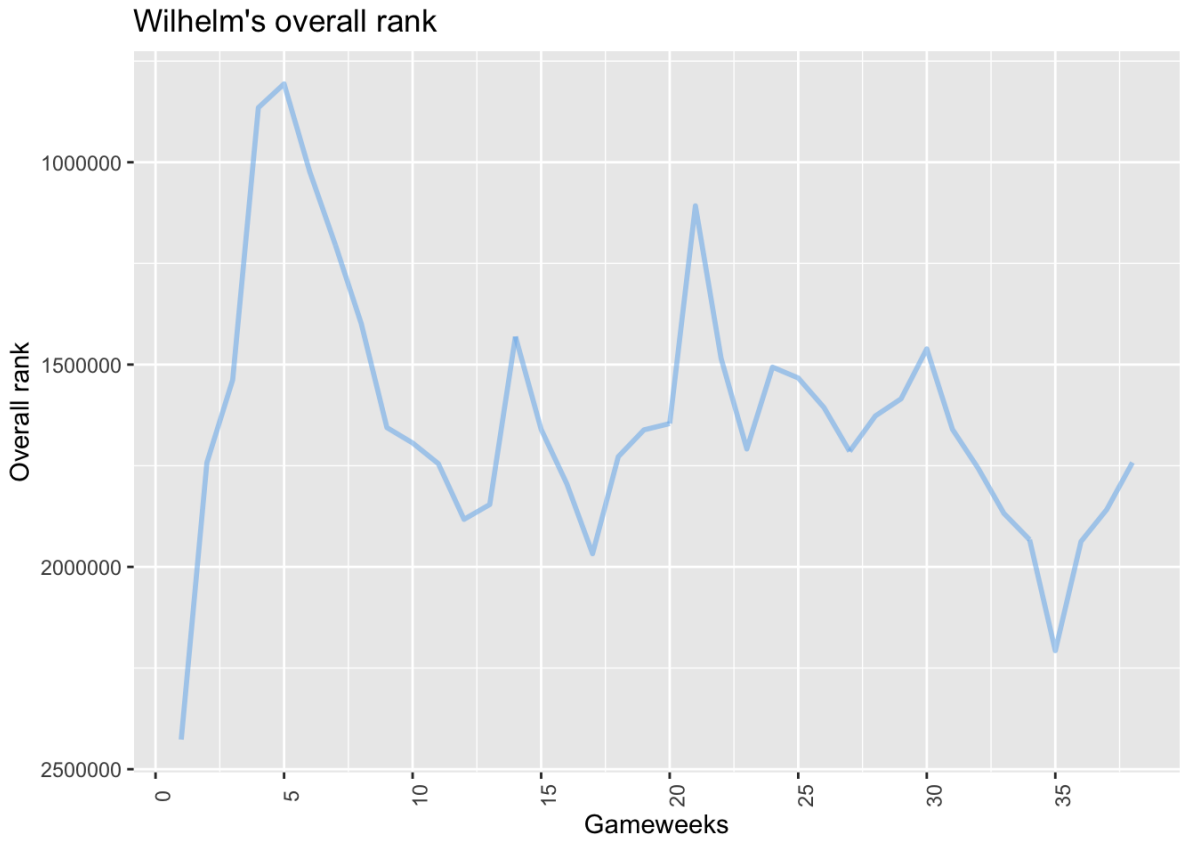
first_name	user_id	gameweek	playername	position	multiplier	is_captain	is_vice_captain	bank	event_transfers
Wilhelm	4667552	1	Mathew Ryan	12	0	FALSE	FALSE	0	0
Wilhelm	4667552	1	Tyrick Mitchell	3	1	FALSE	FALSE	0	0
Wilhelm	4667552	1	Aleksandar Mitrović	15	0	FALSE	FALSE	0	0
Wilhelm	4667552	1	Barry Douglas	13	0	FALSE	FALSE	0	0
Wilhelm	4667552	1	Jamie Vardy	11	1	FALSE	FALSE	0	0
Wilhelm	4667552	1	Alisson Ramses Becker	1	2	FALSE	TRUE	0	0

So, here we have the results. The table is tidy, and a data set has been created.

5.3. Plotting the data of a FPL-user

Let us try visualizing aspects of the data set.. I begin by using the data set to plot using the `geom_line` and `ggplot` functions. In addition, I've added titles by `ggtitle`, and defined the x- and y-axis by `scale_continuous`.

```
ggplot(data=wilhelm_full, aes(x=gameweek, y=overall_rank)) +
  geom_line(color="#80baed", size=1, alpha=0.6, linetype=1) +
  ggtitle("Wilhelm's overall rank") +
  labs(y= "Overall rank", x = "Gameweeks") +
  theme(axis.text.x = element_text(angle = 90)) +
  scale_x_continuous(breaks = scales::pretty_breaks(n = 10)) +
  scale_y_continuous(breaks = scales::pretty_breaks(n = 5), labels=scales::comma)+
  scale_y_reverse()
```



Here, I have a chart showing a user's overall ranking. At first, it was counter-intuitive when analyzing the y-axis. Had I had more time, I wanted to flip the chart, in other words, the y-axis would be decreasing to make it more intuitive. I therefore used `scale_y_reverse` function to do this, so now I have an intuitive chart.

Now, I want to see some overall statistics, showing how one can visualize transfer-statistics for all users of FPL.

5.4. Making a dataset for transfer-statistics from users of FPL

We begin loading the necessary packages, namely `tidyverse`, `ggplot2`, `dplyr`, `fplr` and `fplscrapR`.

```
library(tidyverse)
library(ggplot2)
library(dplyr)
library(fplr)
library(fplscrapR)
```

I will begin with creating a loop to produce only 10 gameweeks with the same information pertaining to each week.

```
for(i in 1:10) {
  weeks = fplscrapR::get_round_info(round = i)
  most_captained <- bind_rows(weeks)


  df <- weeks
}
```

As done above, however with a different function, I will create a data set containing the `get_round_info`, and define the rounds (gameweeks), from 1-10.

```
gw1 <- get_round_info(round=1)
gw2 <- get_round_info(round=2)
gw3 <- get_round_info(round=3)
gw4 <- get_round_info(round=4)
gw5 <- get_round_info(round=5)
gw6 <- get_round_info(round=6)
gw7 <- get_round_info(round=7)
gw8 <- get_round_info(round=8)
gw9 <- get_round_info(round=9)
gw10 <- get_round_info(round=10)
```

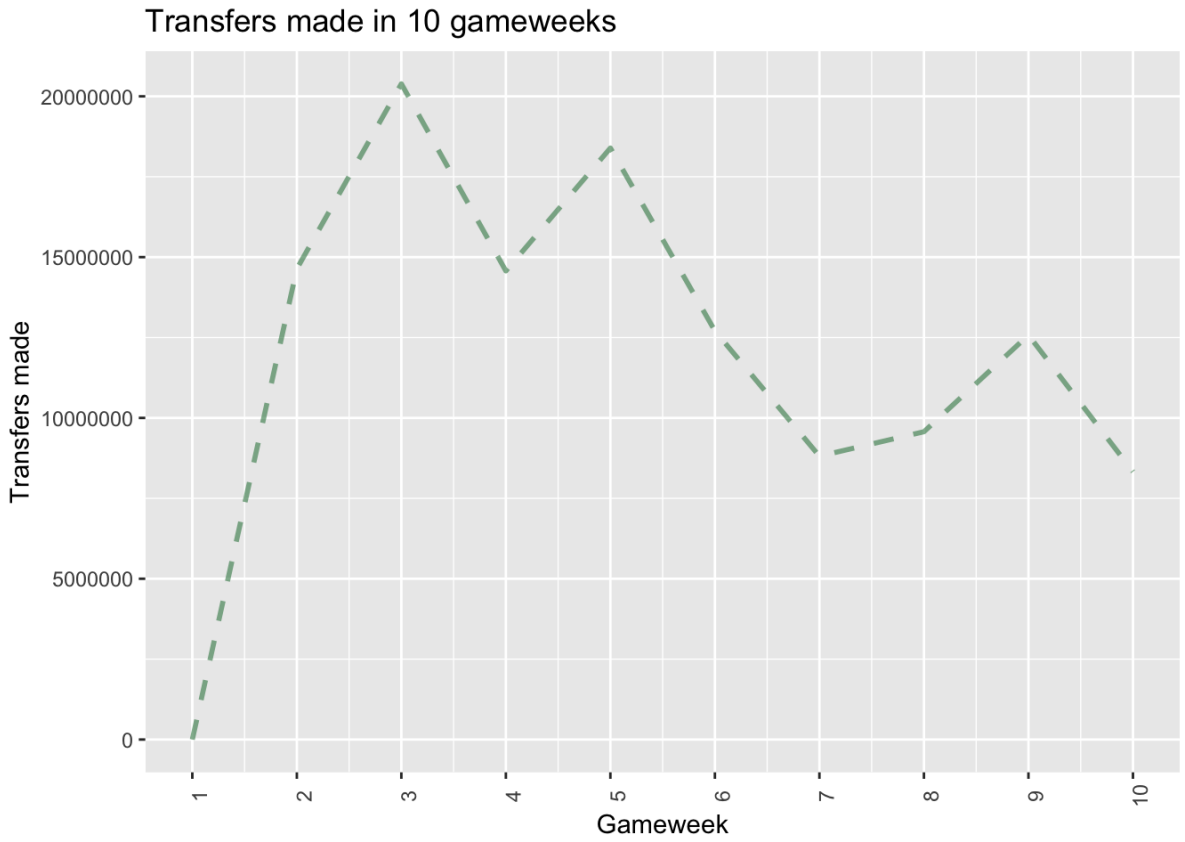
Then, I bind the rows using the `bind_rows` function.

```
weeks <- bind_rows(gw1, gw2, gw3, gw4, gw5, gw6, gw7,
                           gw8, gw9, gw10)


df <- weeks
```

5.5. Plotting overall-statistics from users of FPL

I want to visualize how many transfers were made over the course of 10 weeks by users worldwide. I have altered the axes, having an interval of 1. Additionally, I have made the chart more visually appealing by adding colors and using the `pretty_breaks` function.

```
library(ggplot2)

options(scipen = 999)

ggplot(data=weeks, aes(x=id, y=transfers_made)) +
  geom_line(color="#3a8551", size=1, alpha=0.6, linetype=8) +
  ggtitle("Transfers made in 10 gameweeks") +
  labs(y= "Transfers made", x = "Gameweek") +
  theme(axis.text.x = element_text(angle = 90)) +
  scale_x_continuous(breaks = scales::pretty_breaks(n = 10)) +
    scale_y_continuous(breaks = scales::pretty_breaks(n = 5))
```



As one can see, the number of transfers went down from gameweek 5 to 7, and many began transferring their players in weeks 2-3 and 4-5.

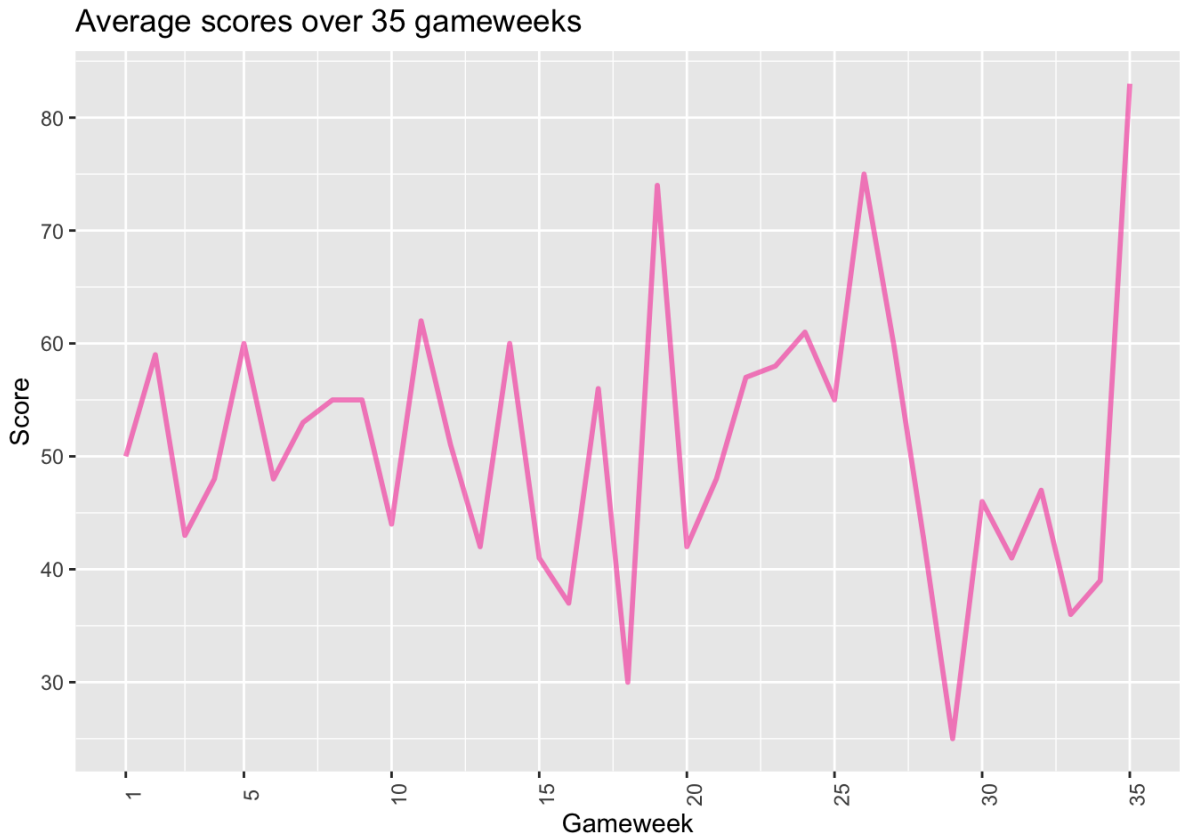
5.6. Comparing one user's score to average scores

Before comparing the scores of Wilhelm to what the average score among users was, I will begin by making two separate plots so that one can interpret the end result better. First, let's visualize how the average score looks like. I have already made a dataset named `weeks`. This only consists of 10 gameweeks. In the following code, I have expanded it to 35 weeks, to get the total.

```
gw11 <- get_round_info(round=11)
gw12 <- get_round_info(round=12)
gw13 <- get_round_info(round=13)
gw14 <- get_round_info(round=14)
gw15 <- get_round_info(round=15)
gw16 <- get_round_info(round=16)
gw17 <- get_round_info(round=17)

#(...)


weeks2 <- bind_rows(gw1, gw2, gw3, gw4, gw5, gw6, gw7,
                    gw8, gw9, gw10, gw11, gw12,
                    gw13, gw14, gw15, gw16, gw17,
                    gw18, gw19, gw20, gw21, gw22,
                    gw23, gw24, gw25, gw26, gw27,
                    gw28, gw29, gw30, gw31, gw32,
                    gw33, gw34, gw35)

df <- weeks2
```

Now, let's plot it, using `ggplot` and some tweaks for visualization purposes.

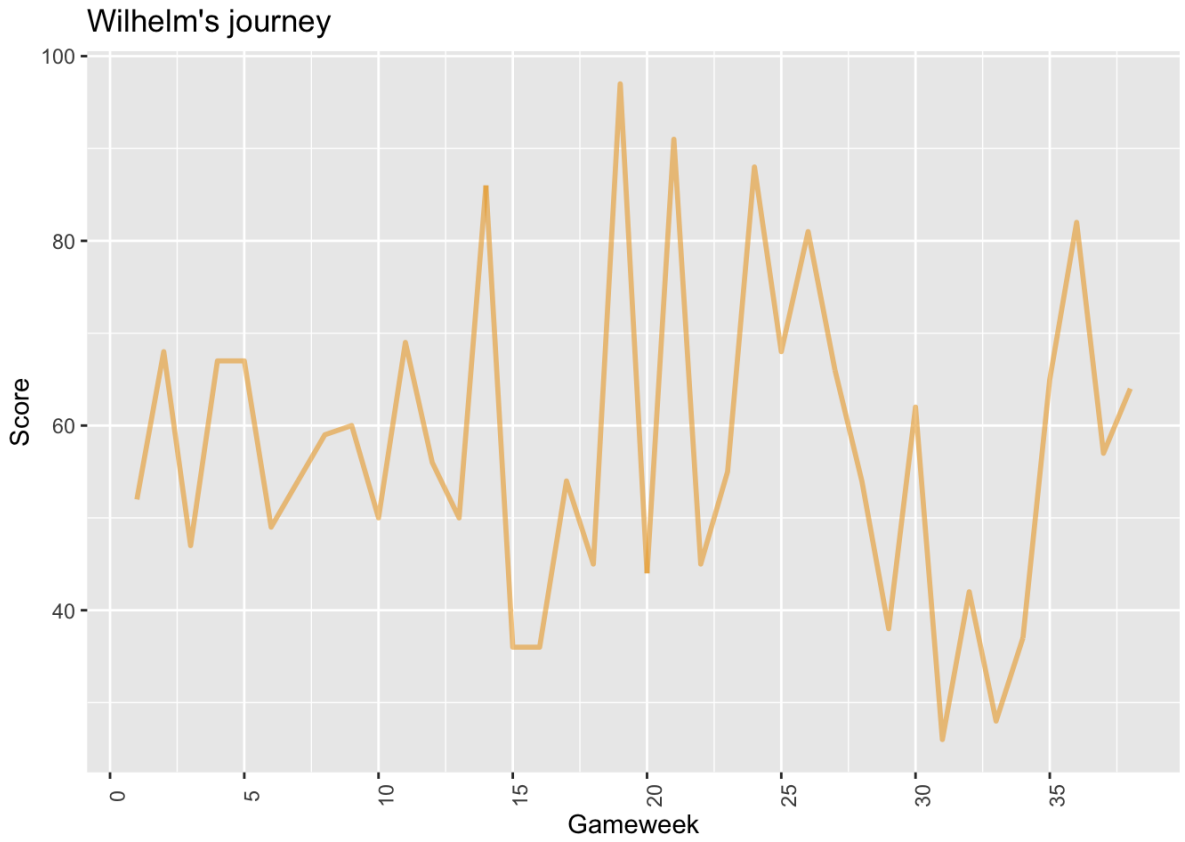
```
library(ggplot2)

ggplot(data=weeks2, aes(x=id, y=average_entry_score)) +
  geom_line(color="#f745a7", size=1, alpha=0.6, linetype=1) +
  ggtitle("Average scores over 35 gameweeks") +
  labs(y= "Score", x = "Gameweek") +
  theme(axis.text.x = element_text(angle = 90)) +
  scale_x_continuous(breaks = scales::pretty_breaks(n = 10)) +
  scale_y_continuous(breaks = scales::pretty_breaks(n = 5)) +
  scale_x_continuous(breaks = c(1, 5, 10, 15, 20, 25, 30, 35))
```



I have, as before, added the `scale_x_continuous` and specified that I want the graph to begin at gameweek 1, as there is no such thing as a gameweek 0. One should bear this in mind when plotting FPL-statistics.

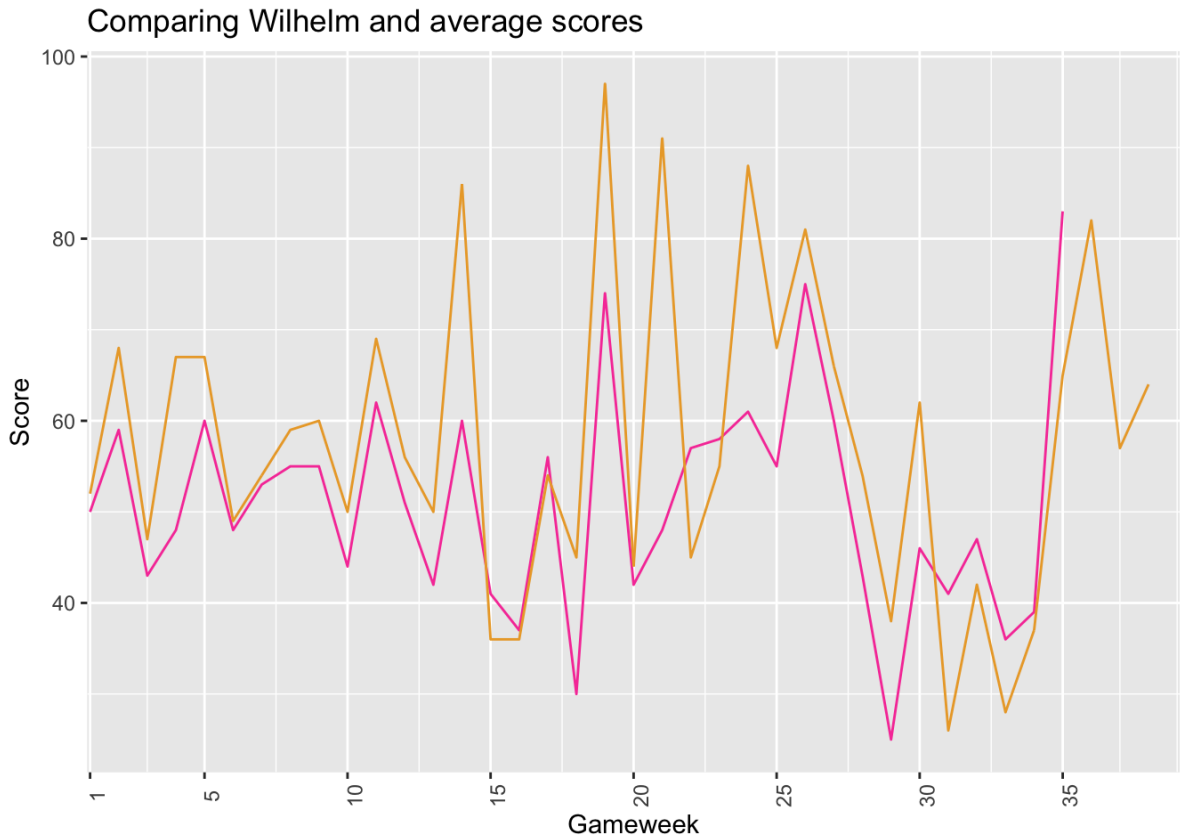
In sum, I have expanded the previously mentioned data set `weeks` to 35 gameweeks, and visualized it. Now, let's move to Wilhelm. I want to see how his scores were over the course of the 35 weeks, using the data set `wilhelm_full`.

```
ggplot(data=wilhelm_full, aes(x=gameweek, y=points))+
  geom_line(color="#eba834", size=1, alpha=0.6, linetype=1) +
  ggtitle("Wilhelm's journey") +
  labs(y= "Score", x = "Gameweek") +
  theme(axis.text.x = element_text(angle = 90)) +
  scale_x_continuous(breaks = scales::pretty_breaks(n = 10)) +
  scale_y_continuous(breaks = scales::pretty_breaks(n = 5))
```



One can, of course, let the eyes wander from one plot to another. However, I want to include both plots into one, combining two lines in one chart so that we can compare the two:

```
library(ggplot2)

ggplot() +
  geom_line(data=weeks2, aes(x=id, y=average_entry_score), color="#f745a7") +
  geom_line(data=wilhelm_full, aes(x=gameweek, y=points), color="#eba834")+
  ggtitle("Comparing Wilhelm and average scores") +
  labs(y= "Score", x = "Gameweek") +
  theme(axis.text.x = element_text(angle = 90)) +
  scale_x_continuous(breaks = scales::pretty_breaks(n = 10)) +
  scale_y_continuous(breaks = scales::pretty_breaks(n = 5))+
  scale_x_continuous(breaks = c(1, 5, 10, 15, 20, 25, 30, 35), expand = expansion(add = c(
0.1, 1.1)))
```



As one can see, the score of Wilhelm is consistently above the average score, with some variations. For instance, in gameweek 35, he obtained points under the average user of FPL. Additionally, he has underperformed where the average user has underperformed, as we can see by the pattern displayed. Thus, one can further conclude that the players ranked above him in the FIF-league have done this as well, to an even greater extent.

5.7. Comparing two users of the same mini league

We may further analyze Wilhelm's decision-making by comparing him to the top user of his mini league. According to the performance chart, Martinius has been consistently performing above Wilhelm. To wrap up the subtask regarding a comparison of two users within the same mini league, I will use two data sets created for the gameweeks of Martinius and Wilhelm respectively. We have already created the latter, so the same procedure follows for Martinius.

```
library(dplyr)
library(ggplot2)
library(fplscrapR)
library(kableExtra)
```

After loading required packages, we assign the function `get_entry_player_picks` and specify user ID (Martinius=4420185) and gameweek (1, 2, 3…).

```
m1 <- get_entry_player_picks(entryid = 4420185, gw = 1)
m2 <- get_entry_player_picks(entryid = 4420185, gw = 2)
m3 <- get_entry_player_picks(entryid = 4420185, gw = 3)
m4 <- get_entry_player_picks(entryid = 4420185, gw = 4)
m5 <- get_entry_player_picks(entryid = 4420185, gw = 5)
m6 <- get_entry_player_picks(entryid = 4420185, gw = 6)
m7 <- get_entry_player_picks(entryid = 4420185, gw = 7)
m8 <- get_entry_player_picks(entryid = 4420185, gw = 8)
m9 <- get_entry_player_picks(entryid = 4420185, gw = 9)

#(...)
```

Then, as I did before, I bind the rows using the `bind_rows` function, and furthermore I merge datasets by event by using the `full_join` function. Additionally, I clean the data set, style it and so on.

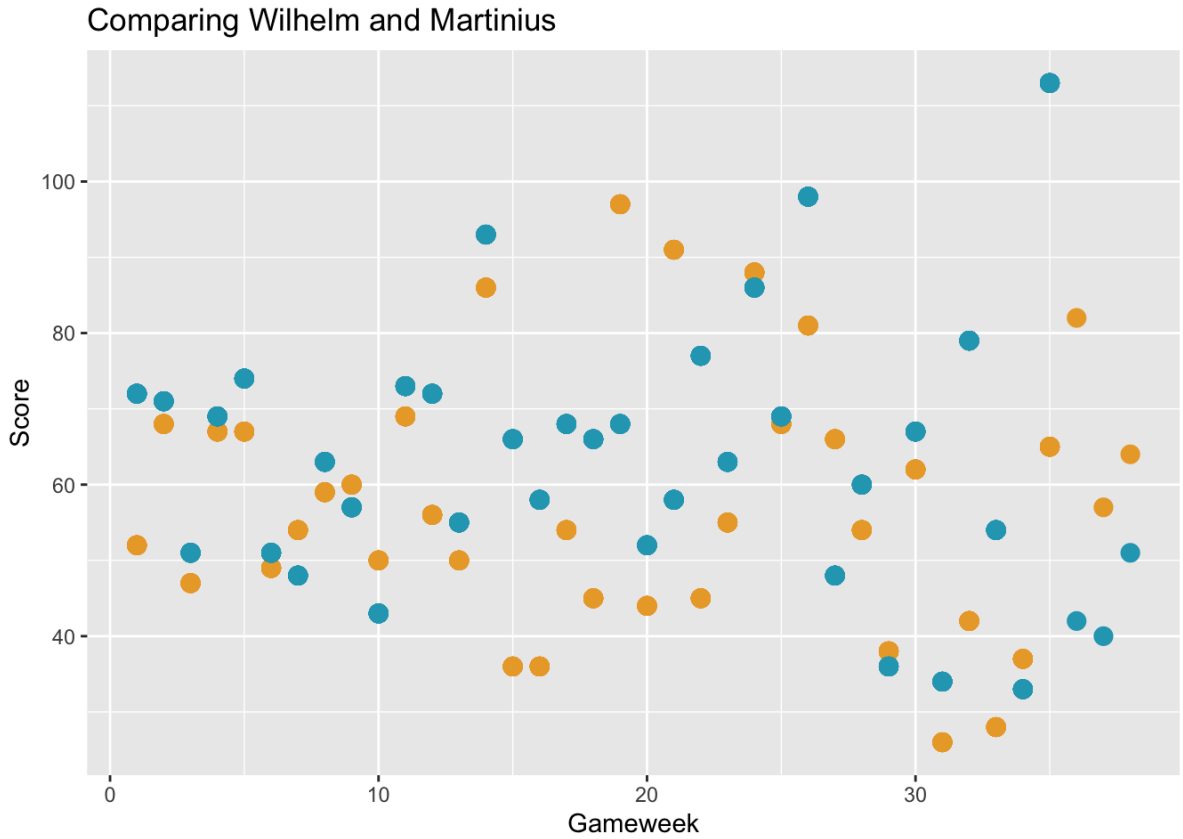
```
martinius_full <- bind_rows(m1, m2, m3, m4, m5, m6, m7,
                            m8, m9, m10, m11, m12, m13,
                            m14, m15, m16, m17, m18, m19,
                            m20, m21, m22, m23, m24, m25,
                            m26, m27, m28, m29, m30, m31,
                            m32, m33, m34, m35)

martinius_season <- get_entry_season(4420185)


martinius_full <- full_join(martinius_full, martinius_season, by = "event")

martinius_full <- martinius_full %>%
  rename("gameweek" = "event",
         "user_id" = "entry")

martinius_full <- martinius_full %>%
  select(-c("value", "rank", "rank_sort"))

martinius_full$first_name <- sub(" .*$", "", martinius_full$name)

martinius_full <- martinius_full %>%
  select(-c("name"))

martinius_full <- martinius_full[c(16, 8, 7, 2:6, 12:15, 9:11)]
```

```
martinius_full[c(1:10)] %>%
  head() %>%
  kable() %>%
  kable_styling()
```

first_name	user_id	gameweek	playername	position	multiplier	is_captain	is_vice_captain	bank	event_transfers
Martinius	4420185	1	Pierre-Emerick Aubameyang	8	1	FALSE	TRUE	0	0
Martinius	4420185	1	Jed Steer	12	0	FALSE	FALSE	0	0
Martinius	4420185	1	Keinan Davis	14	0	FALSE	FALSE	0	0
Martinius	4420185	1	Yves Bissouma	9	1	FALSE	FALSE	0	0
Martinius	4420185	1	Timo Werner	10	1	FALSE	FALSE	0	0
Martinius	4420185	1	Nathan Ferguson	15	0	FALSE	FALSE	0	0

Here is a fairly tidy data set created for Martinius. Now, let us make use of the `wilhelm_full` and `martinius_full` data sets by plotting a scatter plot using `geom_point` in the `ggplot2` package.

```
library(ggrepel)
library(ggplot2)

ggplot()+
  geom_point(data=wilhelm_full, aes(x=gameweek, y=points), color="#eba834", size=5, shape=
20) +
  geom_point(data=martinius_full, aes(x=gameweek, y=points), color="#25a5be", size=5, shap
e=20)+
ggtitle("Comparing Wilhelm and Martinius") +
  labs(y= "Score", x = "Gameweek")+
  geom_text_repel(label=c("Wilhelm", "Martinius"))
```



As we can see, in gameweek 19 and 21, Wilhelm performed better than Martinius. However, Martinius' team performed better in gameweeks 5, 11, 14, 26, 30 and so on. In gameweek 35, Martinius got over 120 points, probably being among some of the best scores worldwide. The blue dots for Martinius are overall placed higher in the chart than Wilhelm, showing us that Martinius has more points than Wilhelm, resulting in his overall higher performance in the FIF-league.

6.0. Summary

In this paper, I have used the case of the FIF-league and the user Wilhelm to show how many opportunities there are to analyze FPL-statistics. The data is accessible for all, transparent and with a lot of variations. For instance, one can scrape data of specific FPL-users, football players, football teams, mini leagues and average/overall statistics. This data has been used to create data sets, data frames, line charts, scatter plots, fixture tables, performance charts, etc.

It seems that one can do a lot: Map out performances to users in a mini-league and retrieve data from the overall historic of user data by using the available packages such as `fplr` and `fplscrapR` for comparisons. Additionally, I have shown that one can create a data set, with many aspects of Tidyverse preserved. Plotting has been done accordingly, to visualize the results. The conclusion is that the opportunities for analyzing FPL-statistics are **immense.**

7.0. References and inspiration

I was inspired by doing research on FPL statistics. Other than the occasional use of stackoverflow (https://stackoverflow.com/) and other people in STV2020, especially the teachers, I got help from visiting these websites:

- Fantasy Premier League (https://fantasy.premierleague.com/) (for data)
- wiscostret (https://wiscostret.github.io/fplscrapR/articles/fdrtable.html) (for fixture table and `fplscrapR`)
- rforjournalists (https://rforjournalists.com/2019/11/12/how-to-track-your-fantasy-football-league-using-r/) (for the performance chart)

In addition, I want to thank the members of the FIF-league for their help, co-student Jørgen (a R-wizard) and the teachers of STV2020.