StatsBombR

Claude REN

11/07/2020

Introduction

For user agreement and terms of use :

PACKAGE FROM STATSBOMBR : https://github.com/statsbomb/StatsBombR

In this document we are going to work on the package StatsBombR, which is well known for providing free data about football. The package contains various functions that facilitate the extraction of data and those functions are what we are going to study here with the help of the work of Ryo :

https://ryo-n7.github.io/2019-08-21-visualize-soccer-statsbomb-part-1/

We are going to change the dataset used, but all the graphical representation will be extracted from his work in order to show a better looking work for an easier reading. I wanted to show what you could do with those data even though the main purpose of this presentation is not to show results or anything but to explain how some of the package's functions work. Every chunk of code will be explain and you can go to my presentation document on ggplot2 to have further understanding.

So the dataset we are going to study here is the women premier league, and more specifically a match between Arsenal WFC and Chelsea FCW. We will study expected goal (xG), passes in the final third and the preferred link up by the players.

Packages presentation

- We are going to use different other package during this study case :
  - Tidyverse : A basic package containing diverse very popular and useful packages like, ggplot2 and dplyr.
    - ggplot2 : For data visualization, you can find more in my other presentation.
    - dplyr : For data manipulation, contain function like mutate(), filter().
  - ggrepel : Extension of ggplot2, more detail in my ggplot2 presentation.
  - ggupset : Extension of ggplot2, replace the x-axis of ggplot2 by a matrix.
  - tinytex : Make the conversion to .tex easier.

```
library(tidyverse)
library(StatsBombR)
library(ggrepel)
library(tinytex)
library(ggupset)
```

Choose you database

Before starting, and explaining the function I used for this study, it is important to precise that StatsBomb provides free data but if you have a premium access, you have the possibility to work on all the data they have and use ALL the function of the package. When you don't have that access, there are functions that you can't use because you need an account to use the function. So in the following study only free function are showed.

The first thing to do, is to look at the competition and for that you simply use the function FreeCompetitions(), after that in my case I wanted to study the Women Premier League, and the competition_id is 37. With the filter() function of dplyr you can filter all the competition to only keep the one you want. With that you can extract all free matches avalaible with FreeMatches().

When it is ready you apply the function StatsBombFreeEvents() which return all the data of matches selected. Then the function allclean() will format the data for R use by adding extra information.

The extraction of all the data takes time, so the better solution is to save the cleansed data, and to not run this chunk too often.

```
comps <- FreeCompetitions()

FAW_match <- comps %>%
  filter(competition_id == 37) %>%
  FreeMatches()

FAW <- StatsBombFreeEvents(MatchesDF = FAW_match)

FAW_clean <- FAW %>%
  allclean()

saveRDS(FAW_clean,
        file = ("C:/Users/claud/OneDrive/Documents/GitHub/REN_PSBx/
                Packages_presentation/Statsbomb/data/FAW_clean.RDS"))
```

Expected goal (xG)

Here we are going to see how to prepare the data and extract information for plotting the expected goal.

After loading the data, we can filter again in order to look at only one match, same system as the competition, you can just have a look at the data and find the match_id you want to study. Or if you study a larger database, you can do something similar by filtering by team.name and use the distinct() function on match_id to extract all the match id of the team you selected. The function mutate() is used to write off NA by 0 in the xG column with the function if_else(condition, true, false).

```
FAW_clean = readRDS("data/FAW_clean.RDS",
                    refhook = NULL)

Find_match_id <- FAW_clean %>%
  filter(team.name == "Arsenal WFC") %>%
  distinct(match_id)

head(Find_match_id)
```

```
## # A tibble: 6 x 1
##   match_id
##      <int>
## 1    19717
## 2    19722
## 3    19724
## 4    19736
## 5    19741
## 6    19744
```

```
AWFC_ <- FAW_clean %>%
  filter(match_id == 19736) %>%
  mutate(shot.statsbomb_xg =
           if_else(is.na(shot.statsbomb_xg),
                   0, shot.statsbomb_xg))
```

In the chunk below, we are creating a new data frame, we use the group_by() function and then summarize() which will create the new data frame according to the number of group, here two, with tot_xg as a new column with the total of xG of the two team. And with mutate we add a last column with text summarizing the data for the plot.

```
AWFC_xg <- AWFC_ %>%
  group_by(team.name) %>%
  summarize(tot_xg = sum(shot.statsbomb_xg) %>% signif(digits = 2)) %>%
  mutate(team_label = glue::glue("{team.name}: {tot_xg} xG"))

head(AWFC_xg)
```

```
## # A tibble: 2 x 3
##   team.name   tot_xg team_label
##   <chr>        <dbl> <glue>
## 1 Arsenal WFC   2.4  Arsenal WFC: 2.4 xG
## 2 Chelsea FCW   0.77 Chelsea FCW: 0.77 xG
```

We join the new data frame, by team name, and with mutate() we add the player name and the value of the xG when there is a goal which will be used for the plot too.

```
AWFC <- AWFC_ %>%
  left_join(AWFC_xg, by = "team.name") %>%
  mutate(player_label = case_when(
    shot.outcome.name == "Goal" ~ glue::glue(
      "{player.name}: {shot.statsbomb_xg %>% signif(digits = 2)} xG"),
    TRUE ~ ""))
```

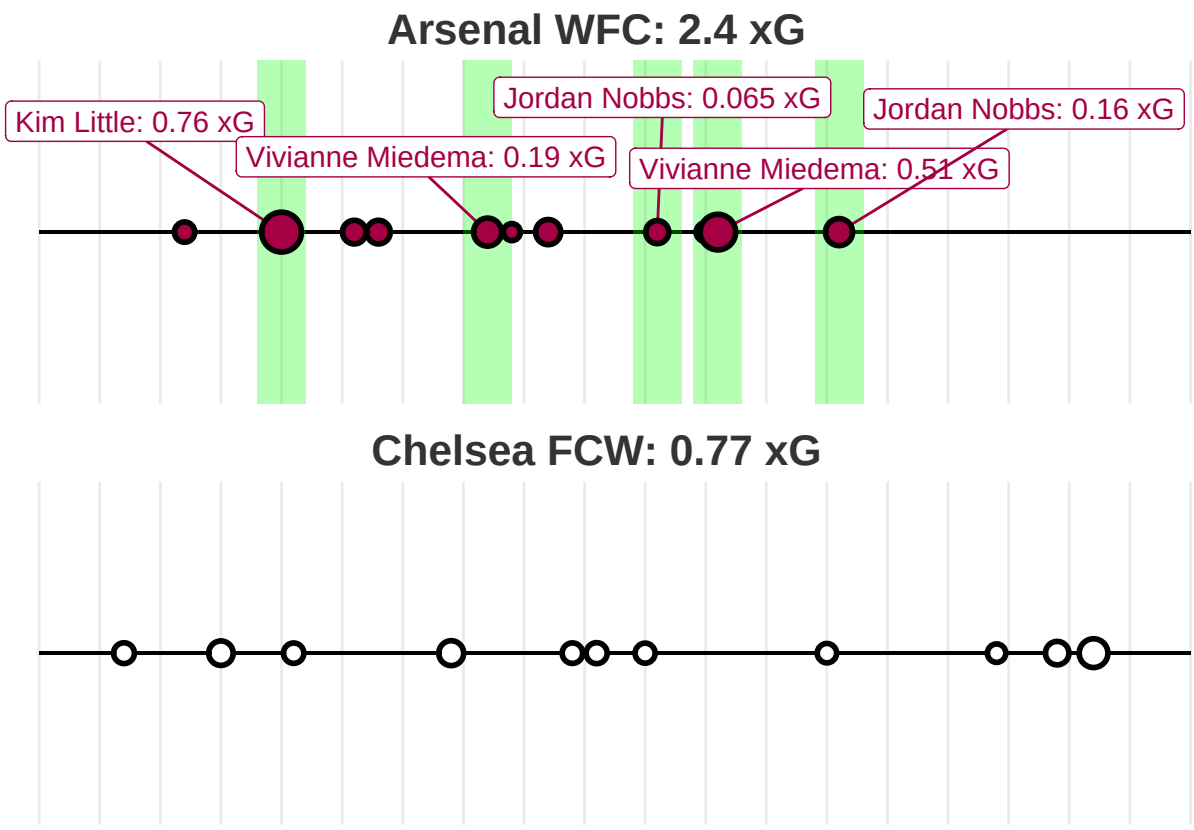
For the plot, we are using ggplot2, most of the function are explained on my other document, so I will briefly go through all the ggplot chunk of the study. So we use geom_rect() and geom_label_repel() (of ggrepel package), it shows the name of the player who scored with the xG associated with it because of the glue package. The function geom_point() represent all the xG value overtime, with the size showing the value of the xG.

We can note that we used facet_wrap() to show the two teams xG separately. The other function of customization are basic.

```
AWFC_xg_timelineplot <- AWFC %>%
  ggplot() +
  geom_segment(x = 0, xend = 95,
               y = 0, yend = 0) +
  geom_rect(data = AWFC %>% filter(shot.outcome.name == "Goal"),
            aes(xmin = minute - 2, xmax = minute + 2,
                ymin = -0.005, ymax = 0.005),
            alpha = 0.3, fill = "green") +
  geom_label_repel(data = AWFC %>% filter(shot.outcome.name == "Goal"),
             aes(x = minute, y = 0,
                 color = team.name, label = player_label),
             nudge_x = 4, nudge_y = 0.003,
             show.legend = FALSE) +
  geom_point(data = AWFC %>% filter(shot.statsbomb_xg != 0),
             shape = 21, stroke = 1.5,
```

```
            aes(x = minute, y = 0,
                size = shot.statsbomb_xg, fill = team.name)) +
  scale_color_manual(values = c("Arsenal WFC" = "#a50044",
                                "Chelsea FCW" = "black")) +
  scale_fill_manual(values = c("Arsenal WFC" = "#a50044",
                               "Chelsea FCW" = "white")) +
  facet_wrap(vars(team_label), ncol = 1) +
  scale_x_continuous(breaks = seq(0, 95, by = 5),
                     labels = c(seq(0, 40, by = 5), "HT",
                                seq(50, 90, by = 5), "FT"),
                     limits = c(-3, 95),
                     expand = c(0.01, 0)) +
  scale_y_continuous(limits = c(-0.005, 0.005),
                     expand = c(0, 0)) +
  scale_size(range = c(2, 6)) +
  theme_minimal() +
  theme(legend.position = "none",
        strip.text = element_text(size = 16,
                                  face = "bold", color = "grey20"),
        plot.caption = element_text(color = "grey20",
                                    hjust = 0),
        axis.title = element_blank(),
        axis.text = element_blank(),
        panel.grid.minor = element_blank(),
        panel.grid.major.y = element_blank())

AWFC_xg_timelineplot
```



In this part, we are still going to look at the xG, but we will plot the cumulative sum of it. For that the step are a similar to the previous plot, but here we need more information like the time and the sum at this instant and not only the total of xG. We start by using group_by() on minute, team.name and period and then summarize() to have the sum of xG at every minute of the two team in each period but what we want is the cumulated xG overtime so we need to do some more work. After ungroup() we group_by() team.name like in the previous chunk, and instead of adding a total xG column we add the accumulate sum. Lag() is used to initialize the rollsum value at 0 and then at every minute if the xG is not 0 we add it to the rollsum value.

After that, we prepare a the data frame for the plot, by joining the data and doing the same thing as previously we add columns for the goal event in order to display it in the plot in a good looking way.

```
AWFC_rollsum <- AWFC %>%
  group_by(minute, team.name, period) %>%
  summarize(sumxg = sum(shot.statsbomb_xg)) %>%
  ungroup() %>%
  group_by(team.name) %>%
  mutate(rollsum = lag(cumsum(sumxg)),
         rollsum = if_else(is.na(rollsum), 0, rollsum)) %>%
  select(team.name, minute, rollsum, sumxg) %>%
  mutate(rollsum = case_when(
```

```
    row_number() == n() & sumxg != 0 ~ rollsum + sumxg,
    TRUE ~ rollsum
  ))

AWFC_rollsum <- AWFC_rollsum %>%
  left_join(AWFC %>% filter(shot.outcome.name == "Goal")
            %>% select(minute, shot.outcome.name, team.name, player.name),
            by = c("minute", "team.name")) %>%
  mutate(rollsum_goal = rollsum + sumxg,
         minute_goal = minute + 1,
         player_label = case_when(
           shot.outcome.name == "Goal" ~ glue::glue(
             "{player.name}: {sumxg %>% signif(digits = 2)} xG"),
           TRUE ~ ""))
```

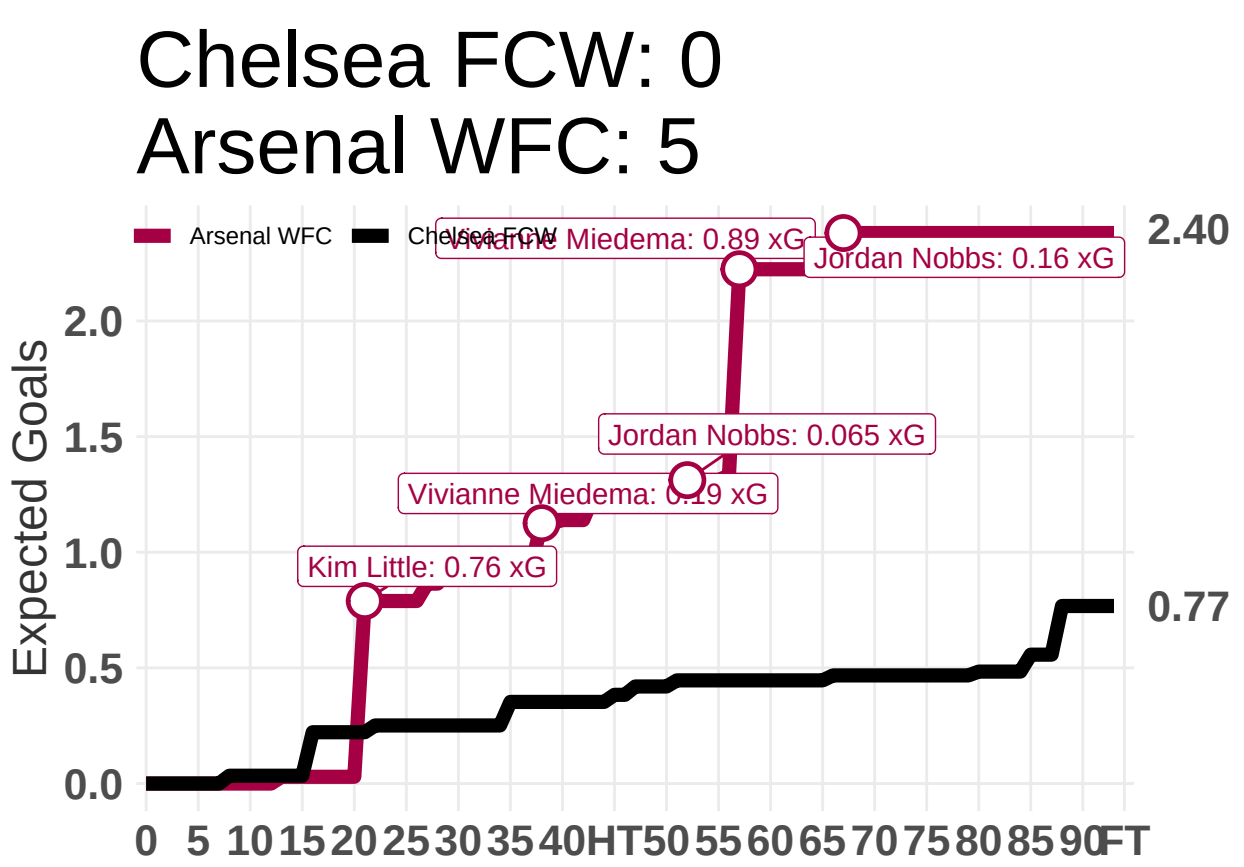
Now for the plot, pull() take the value of the previous data column tot_xg for the y-axis scale. The other element have already been explained in the previous chunk.

```
tot_awfc_df <- AWFC_xg %>%
  pull(tot_xg)

AWFC_rollsumxg_plot <- AWFC_rollsum %>%
  ggplot(aes(x = minute, y = rollsum,
             group = team.name, color = team.name)) +
  geom_line(size = 2.5) +
  geom_label_repel(data = AWFC_rollsum %>% filter(shot.outcome.name == "Goal"),
             aes(x = minute_goal, y = rollsum_goal,
                 color = team.name, label = player_label),
             nudge_x = 6, nudge_y = 0.15,
             show.legend = FALSE) +
  geom_point(data = AWFC_rollsum %>% filter(shot.outcome.name == "Goal"),
             aes(x = minute_goal, y = rollsum_goal, color = team.name), show.legend = FALSE,
             size = 5, shape = 21, fill = "white", stroke = 1.25) +
  scale_color_manual(values = c("Arsenal WFC" = "#a50044",
                                "Chelsea FCW" = "#000000"),
                     labels = c("Arsenal WFC",
                                "Chelsea FCW")) +
  scale_fill_manual(values = c("Arsenal WFC" = "#a50044",
                               "Chelsea FCW" = "#000000")) +
  scale_x_continuous(breaks = c(seq(0, 90, by = 5), 94),
                     labels = c(seq(0, 40, by = 5), "HT",
                                seq(50, 90, by = 5), "FT"),
                     expand = c(0.01, 0),
                     limits = c(0, 94)) +
  scale_y_continuous(sec.axis = sec_axis(~ ., breaks = tot_awfc_df)) +
  labs(title = "Chelsea FCW: 0 \nArsenal WFC: 5",
       x = NULL,
       y = "Expected Goals") +
```

```
  theme_minimal() +
  theme(plot.title = element_text(size = 30),
        plot.subtitle = element_text(size = 18, color = "grey20"),
        axis.title = element_text(size = 18, color = "grey20"),
        axis.text = element_text(size = 16, face = "bold"),
        panel.grid.minor = element_blank(),
        legend.position = c(0.2, 0.95),
        legend.direction = "horizontal",
        legend.title = element_blank())

AWFC_rollsumxg_plot
```



Number of pass in the final third

In football, expected goal alone is not a good way to apprehend the domination of a team during a game. So we have to look at other stat to find a better understanding of the game. For example, we can look at the number of pass in the final third of the opponent to see if before the goal or dangerous occasion (bigger xG) they were actually dominating the game or not. For that we are going to create a data frame containing the number of pass of each team in a 5 minutes window. Like before, we group by team and minute, then

we count everytime the action is labeled pass and that the location is further than 80m. So at every minute of the game, we have the number of pass for each team.

```
roll_final_pass <- AWFC %>%
  group_by(team.name, minute) %>%
  mutate(count = case_when(
    type.name == "Pass" & location.x >= 80 ~ 1L,
    TRUE ~ 0L
  )) %>%
  select(team.name, minute, count) %>%
  ungroup()
```

In a match there can be period when one team doesn't play any pass. In order to correct this problematic We create a data frame that contain all the combination of minutes during the match and we will use crossing() to do that.

Now that we have data for every minute of the game for each team we use rolling_sum(), a function with which we are going to sum the pass during a window of 5min.

Then with the same process we create a new data frame containing the number of pass of each team during a 5min window.

```
first_min <- AWFC$minute %>% unique() %>% first()
last_min <- AWFC$minute %>% unique() %>% last()
minute <- c(first_min:last_min)
team.name <- c("Chelsea FCW", "Arsenal WFC")

rolling_sum <- tibbletime::rollify(.f = sum, window = 5)

roll_awfc_pass <- crossing(minute, team.name) %>%
  left_join(roll_final_pass, by = c("minute", "team.name")) %>%
  group_by(team.name, minute) %>%
  summarize_all(sum) %>%
  ungroup() %>%
  mutate(count = ifelse(is.na(count), 0, count)) %>%
  group_by(team.name) %>%
  mutate(rollsum = rolling_sum(count),
         rollsum = ifelse(is.na(rollsum), 0, rollsum)) %>%
  group_by(team.name) %>%
  select(-count) %>%
  filter(row_number() %% 5 == 1 | row_number() == n())

roll_awfc_pass %>% head(5)
```

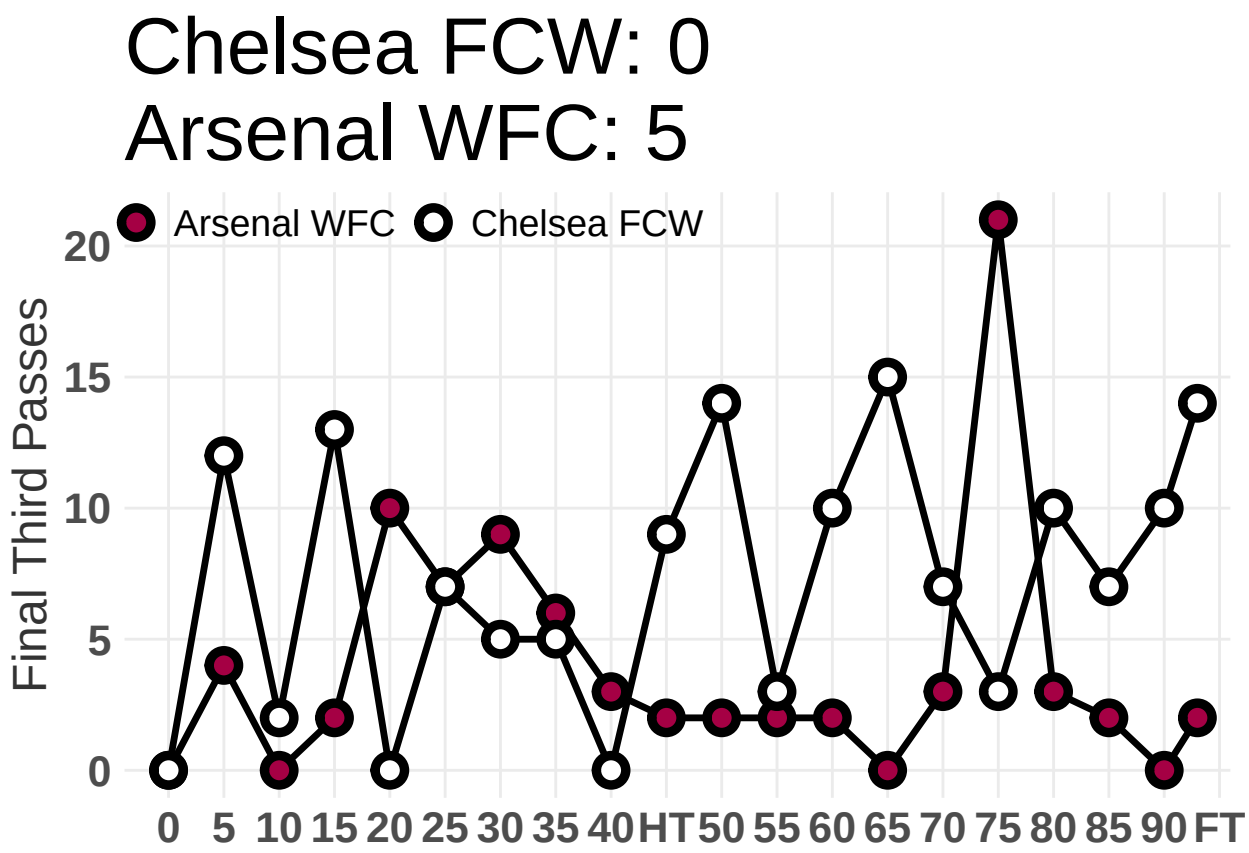
```
## # A tibble: 5 x 3
## # Groups:   team.name [1]
##   team.name   minute rollsum
##   <chr>        <int>   <dbl>
## 1 Arsenal WFC      0       0
```

```
## 2 Arsenal WFC      5       4
## 3 Arsenal WFC     10       0
## 4 Arsenal WFC     15       2
## 5 Arsenal WFC     20      10
```

Not much change for the plot. But in term of result we can see that, Chelsea dominated had the ball more time in the final third but, on the counter Arsenal managed to score 3 of their goals when they had not the control of the ball.

```
finalthird_rollingplot <- roll_awfc_pass %>%
  ggplot(aes(x = minute, y = rollsum,
             group = team.name)) +
  geom_line(data = roll_awfc_pass,
            size = 1.2) +
  geom_point(data = roll_awfc_pass,
             aes(fill = team.name),
             size = 3.5, shape = 21, stroke = 2.5) +
  scale_x_continuous(breaks = seq(0, 95, by = 5),
                     labels = c(seq(0, 40, by = 5), "HT",
                                seq(50, 90, by = 5), "FT"),
                     limits = c(-3, 95),
                     expand = c(0.01, 0)) +
  scale_y_continuous(breaks = seq(0, 30, by = 5),
                     labels = seq(0, 30, by = 5)) +
  scale_fill_manual(values = c("Arsenal WFC" = "#a50044",
                               "Chelsea FCW" = "white"),
                    labels = c("Arsenal WFC",
                               "Chelsea FCW")) +
  labs(title = "Chelsea FCW: 0 \nArsenal WFC: 5",
       x = NULL,
       y = "Final Third Passes") +
  theme_minimal() +
  theme(plot.title = element_text(size = 30),
        axis.title = element_text(size = 18, color = "grey20"),
        axis.text = element_text(size = 16, face = "bold"),
        panel.grid.minor = element_blank(),
        legend.text = element_text(size = 14),
        legend.position = c(0.25, 0.95),
        legend.direction = "horizontal",
        legend.title = element_blank())

finalthird_rollingplot
```



Prefered link up play

We can also look at the link up between players. We mutate the pass outcome column and clean the NA value. Then we filter the data by team and pass completed during open play ending inside the box. We select the information we want, player providing the pass, the receiver, season and location information.

We group by season name and we create a columns where we add the total of time the same association of players. For the plot we use ggupset, which mean we need to add a new variable of a list form. That's why we create the pass_duo variable containing the passer's name and the receiver's name.

```
pass_received_all_box <- AWFC_clean %>%
  mutate(pass.outcome.name = fct_explicit_na(pass.outcome.name, "Complete")) %>%
  filter(type.name == "Pass",
         team.name == "Arsenal WFC",
         pass.outcome.name == "Complete",
         !play_pattern.name %in% c("From Corner", "From Free Kick",
                                   "From Throw In"),
         pass.end_location.x >= 102 & pass.end_location.y <= 62 &
           pass.end_location.y >= 18) %>%
  select(player.name, pass.recipient.name,
```

```
        season_id, season_name,
        position.name, position.id,
        location.x, location.y,
        pass.end_location.x, pass.end_location.y,
        contains("pass")) %>%
  group_by(season_name) %>%
  add_count(player.name, pass.recipient.name, name = "pass_num") %>%
  ungroup() %>%
  mutate(player.name = glue::glue("{player.name}: {pass_num}")) %>%
  mutate(pass_duo = map2(player.name, pass.recipient.name, ~c(.x, .y))) %>%
  select(player.name, pass.recipient.name, pass_num,
        pass_duo,season_name)
```

For the last plot, we used the function nest() which create, with the help of group_by(), a column called data with all the information of the season.

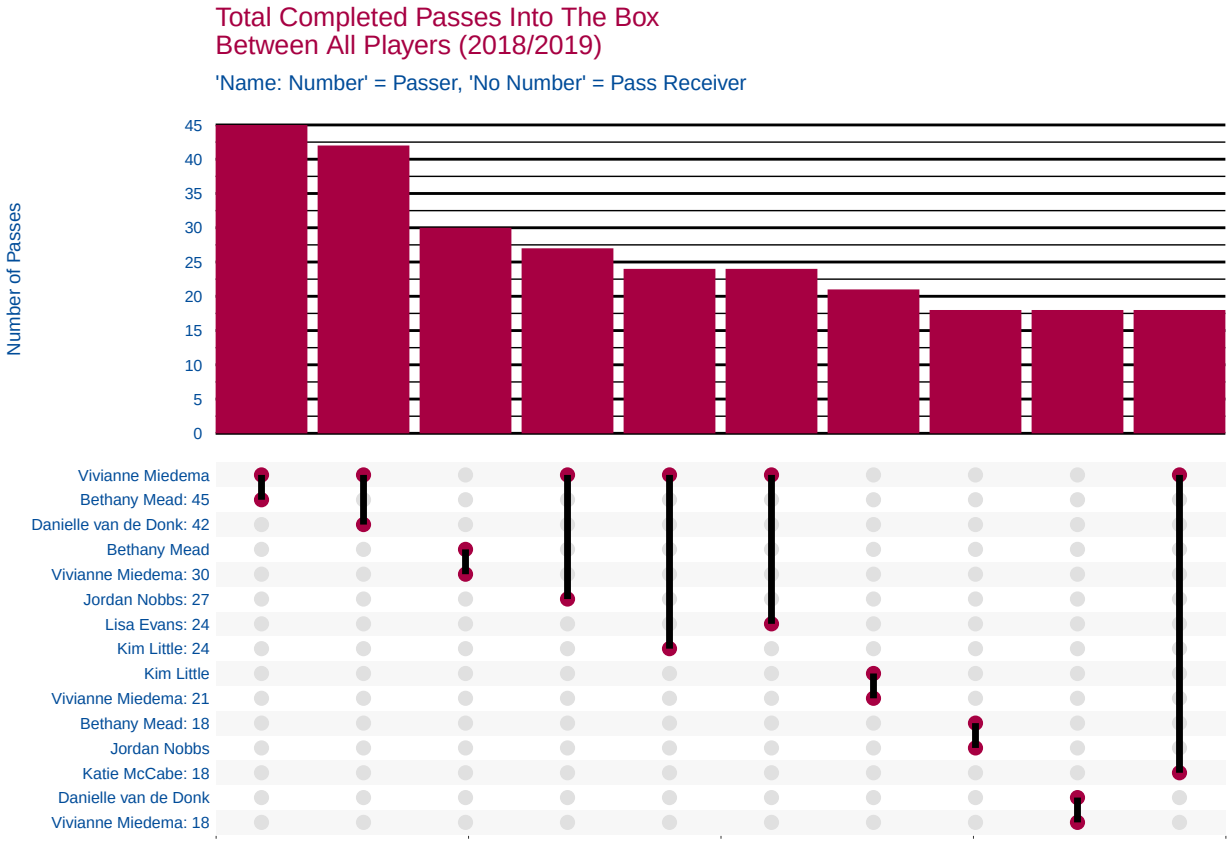
```
all_pass_nested_box <- pass_received_all_box %>%
  group_by(season_name) %>%
  nest() %>%
  mutate(plot = map2(
    .x = data, .y = season_name,
    ~ ggplot(data = .x, aes(x = pass_duo)) +
      geom_bar(fill = "#a70042") +
      scale_x_upset(n_intersections = 10,
                    expand = c(0.01, 0.01)) +
      scale_y_continuous(expand = c(0.04, 0.04)) +
      labs(title = glue::glue("
                              Total Completed Passes Into The Box
                              Between All Players ({.y})"),
           subtitle = "'Name: Number' = Passer, 'No Number' = Pass Receiver",
           x = NULL, y = "Number of Passes") +
      theme_combmatrix(
        text = element_text(
                            color = "#004c99"),
        plot.title = element_text( size = 10,
                                   color = "#a70042"),
        plot.subtitle = element_text( size = 8,
                                      color = "#004c99"),
        axis.title = element_text( size = 7,
                                   color = "#004c99"),
        axis.text.x = element_text( size = 6,
                                    color = "#004c99"),
        axis.text.y = element_text( size = 6,
                                    color = "#004c99"),
        panel.background = element_rect(fill = "white"),
        combmatrix.panel.point.size = 2,
        combmatrix.panel.point.color.fill = "#a70042",
        combmatrix.panel.line.color = "#a70042",
```

```
        panel.grid = element_line(color = "black"),
        panel.grid.major.x = element_blank(),
        axis.ticks = element_blank())))

all_pass_nested_1819 <- all_pass_nested_box$plot[[1]] +
  scale_y_continuous(labels = seq(0, 45, by = 5),
                     breaks = seq(0, 45, by = 5),
                     limits = c(0, 45))

all_pass_nested_1819
```

```
## Warning: Removed 342 rows containing non-finite values (stat_count).
```



Conclusion

This conclude the study of the Arsenal Women FC, of course there is many more function in the StatsBombR package. Most of them are function that simplify the extraction of data you look for like opposing team,

goalkeeper or lineups information. But I thought it was not worth showing if I hadn't results to display at the end. That's why I only explained the basic function that you need to use to start your analyze.

If you are looking for the other function you can do to : https://www.rdocumentation.org/packages/StatsBombR/versions/0.1.0