"StatsBombR Tutorial"

A tutorial on how to use the StatsBombR package to analyse football data.

Lorcán Mason

Table of contents

1 Introduction to the "StatsBombR" Package

All credit for this tutorial goes to StatsBomb and the webinar they recently hosted which went through this code. This tutorial is a step-by-step guide on how to use the StatsBombR package to analyse the 2024 Copa America data. You can check out all free data available by clicking here.

The "StatsBombR" package, developed by the innovative sports data company StatsBomb, represents a cutting-edge tool for sports analysts, data scientists, and football enthusiasts aiming to delve into comprehensive and high-quality football data. StatsBomb, renowned for its meticulous data collection and detailed analytical insights, provides an extensive dataset through this R package, enabling users to explore and analyse football matches with unparalleled depth.

1.1 Key Features

1.1.1 Comprehensive Data Access

The package provides access to detailed event-level data from numerous football leagues and competitions. This includes information on passes, shots, tackles, fouls, and other critical match events. The data is structured in a way that captures the complexity and nuances of football matches, allowing users to perform in-depth analyses.

1.1.2 User-Friendly Functions

"StatsBombR" offers a suite of functions that allow users to effortlessly query, filter, and manipulate the data. These functions are designed to be intuitive, making it easier for users to focus on their analysis rather than data handling. Key functions include:

- `get_match_data()`: Retrieves detailed data for a specific match.
- `get_team_data()`: Provides aggregated statistics for a particular team.
- `get_player_data()`: Accesses individual player statistics and performance metrics.
- `filter_events()`: Filters event data based on specified criteria (e.g., event type, player, location).

1.1.3 Visualization Tools

The package includes tools for creating insightful visualizations. Users can generate various types of maps and charts to better understand player and team behaviours. Some visualization functionalities include:

- **Pass Maps**: Visual representations of passing patterns and connections between players.
- **Shot Maps**: Detailed maps showing the locations and outcomes of shots taken during a match.
- **Heat Maps**: Visualizations that highlight areas of the pitch where players or teams are most active.
- **Event Plots**: Diagrams showing specific events, such as goals, assists, and key defensive actions.

1.1.4 Advanced Analytics

With the data and tools provided, users can perform advanced statistical analyses. This includes evaluating player performance, team strategies, and match outcomes using various statistical and machine learning methods. The package supports integration with other R libraries for statistical modelling, machine learning, and data visualization, such as `dplyr`, `ggplot2`, and `caret`.

The "StatsBombR" package stands out as a powerful resource for anyone interested in football analytics. By providing access to detailed and reliable football data along with user-friendly tools for analysis and visualization, StatsBomb empowers users to gain deeper insights into the beautiful game. Whether you are a seasoned analyst or a football fan with a penchant for data, "StatsBombR" offers the capabilities you need to explore football from a new perspective. With its extensive features and robust functionalities, "StatsBombR" is an essential tool for modern football analysis, enabling users to unlock the full potential of football data and drive informed decisions based on thorough and insightful analyses.

1.2 Install packages

```
install.packages("devtools", repos="http://cran.us.r-project.org")
install.packages("remotes", repos="http://cran.us.r-project.org")
remotes::install_version("SDMTools", "1.1-221", repos="http://cran.us.r-project.org")
devtools::install_github("statsbomb/StatsBombR", repos="http://cran.us.r-project.org")
devtools::install_github("FCrSTATS/SBpitch", repos="http://cran.us.r-project.org")
```

1.3 Load packages

```
library(tidyverse)
library(StatsBombR)
```

2 Pull the free StatsBomb competition data into the environment

```
Comps <- FreeCompetitions()
```

```
[1] "Whilst we are keen to share data and facilitate research, we also urge you to be responsible wit
```

This will pull all the free competitions made available by StatsBomb. Luck for us, there are 74 free competition datasets available. You can focus this data in anyway you want. To filter the data to a specific competition, you can filter by specifying any two column names from the dataset. To be more accurate, I prefer to use the numeric columns. You can check the structure of the data by running `str(df)` where `df` is the name you've saved the data frame. In this StatsBomb dataset, `competition_id` and `season_id` are the numeric columns.

```
Comps = Comps %>%
  filter(competition_id == "" & season_id == "")
```

For this example I am going to focus on the recent Copa America. The `competition_id` for the Copa America is 223 and the `season_id` is 282.

```
copa <- FreeCompetitions() %>%
  filter(competition_id == 223 & season_id == 282)

copa_matches <- FreeMatches(copa)
#1 This pulls all the matches for the desired competition.

Copa_Stats_Bomb_Data <- free_allevents(MatchesDF = copa_matches, Parallel = T)
#2 This pulls all the event data for the matches that are chosen.

copa_data_clean = allclean(Copa_Stats_Bomb_Data)
#3 Extracts lots of relevant information such as x/y coordinates.
# More information can be found in the package info.
```

Be sure to familiarise yourself with the columns it creates using `names(copa)`. This will help you to understand the data better and filter it to your needs.

```
names(copa_data_clean)
```

2.1 Filtering to team Shots and goals

For the next couple of examples, I will show you how to filter the data to get the shots and goals for teams and players. For this we will focus on the following columns: `team.name`, `type.name`, `shot.outcome.name`, and `player.name`. To check the values included in each of these columns, you can run the below code. This will help you to understand the variables better and make analysis easier to comprehend.

```
unique(copa_data_clean$team.name)

unique(copa_data_clean$type.name)

unique(copa_data_clean$shot.outcome.name)

unique(copa_data_clean$player.name)
```

2.1.1 Totals

```
shots_goals = copa_data_clean %>%
  group_by(team.name) %>%
  #1: This code groups the data by team, so that whatever operation we perform
  # on it will be done on a team by team basis. I.e, we will find the shots and
  # goals for every team one by one.
  summarise(shots = sum(type.name=="Shot", na.rm = TRUE),
            #2: Summarise takes whatever operation we give it and produces a
            # new, separate table out of it. The vast majority of summarise
            # uses come after group_by.
            goals = sum(shot.outcome.name=="Goal", na.rm = TRUE))
#3: shots = sum(type.name=="Shot", na.rm = TRUE) is telling it to create a new
# column called shots that sums up all the rows under the type.name column
# that contain the word "Shot". na.rm = TRUE tells it to ignore any NAs within
# that column. shot.outcome.name=="Goal", na.rm = TRUE)
# does the same but for goals.
```

2.1.2 Per match

```
# Adding in the n_distinct(match_id) means we are dividing the number of
# shots/goals by each distinct (or unique) instance of a match, for every team.
# I.e, we are dividing the numbers per game.

shots_goals = copa_data_clean %>%
  group_by(team.name) %>%
  summarise(shots = sum(type.name=="Shot", na.rm = TRUE)/n_distinct(match_id),
            goals = sum(shot.outcome.name=="Goal", na.rm = TRUE)/n_distinct(match_id))
```

2.2 Filter to player shots and key passes

Now lets filter the data to get the shots and key passes. For this we will focus on the following columns: `player.name`, `player.id`, `type.name`, and `pass.shot_assist`. To check the values included in each of these columns, you can run the below code like above. Again this will help you to understand the variables better and make analysis easier to comprehend.

```
unique(copa_data_clean$player.name)

unique(copa_data_clean$player.id)
```

```r
unique(copa_data_clean$type.name)

unique(copa_data_clean$pass.shot_assist)
```

2.2.1 Total

```r
player_shots_keypasses = copa_data_clean %>%
  group_by(player.name, player.id) %>%
  #1: This code groups the data by player,
  # so that whatever operation we perform on it will be done on a player by
  # player basis. I.e, we will find the shots and goals for every player one by one.
  summarise(shots = sum(type.name=="Shot", na.rm = TRUE),
            keypasses = sum(pass.shot_assist==TRUE, na.rm = TRUE))
```

2.2.2 Get minutes played data

```r
player_minutes = get.minutesplayed(copa_data_clean)
#1: This function gives us the minutes played in each match by ever
# player in the dataset.

player_minutes = player_minutes %>%
  group_by(player.id) %>%
  summarise(minutes = sum(MinutesPlayed))
#2: Now we group that by player and sum it altogether to get
# their total minutes played.
```

2.2.3 Join minutes played data to player shots dataframe

```r
player_shots_keypasses = left_join(player_shots_keypasses, player_minutes)
#1: left_join allows us to combine our shots and key passes table and our
# minutes table, with the the player.id acting as a reference point.

player_shots_keypasses = player_shots_keypasses %>%
  mutate(nineties = minutes/90)
#2: `mutate` is a `dplyr` function that creates a new column. In this instance
# we are creating a column that divides the minutes totals by 90,
# giving us each players number of 90s played.
```

```
player_shots_keypasses = player_shots_keypasses %>%
  mutate(shots_per90 = shots/nineties,
         kp_per90 = keypasses/nineties,
         shots_kp_per90 = shots_per90+kp_per90)
#3: Finally we divide our totals by our number of 90s to get our totals
# per 90s columns for shots and key passes.
# We also calculate the sum of these two columns.
```

2.2.4 Filter minutes for players who have played at least three 90 minutes (270 minutes)

```
player_shots_keypasses = player_shots_keypasses %>%
  filter(minutes>270)
#1: This code filters the data to only include players
# who have played at least 270 minutes for a fair comparison.
```

2.3 Passes into the final third

```
passes = copa_data_clean %>%
  filter(type.name=="Pass" & is.na(pass.outcome.name)) %>%
  #1: This code filters the data to only include passes that have
  # been completed. In this data, NA denotes a complete pass.
  filter(location.x<80 & pass.end_location.x>=80) %>%
  #2: This code filters the data to only include passes into the final third.
  group_by(player.name) %>%
  summarise(f3_passes = sum(type.name=="Pass"))
#3: This code groups the data by player and sums up the number of passes
# into the final third for each player.
```

2.3.1 Specific player passes into the final third (Valverde)

```
player_passes = copa_data_clean %>%
  filter(type.name=="Pass" & is.na(pass.outcome.name) &
           player.name=="Federico Santiago Valverde Dipetta") %>%
  filter(location.x<80 & pass.end_location.x>=80)
```

3 Plotting and Visualisations

Now that we have cleaned and filtered the data we want to investigate it is now time to communicate our findings. We can do this by plotting the data. Below are some examples of how you can plot the data.

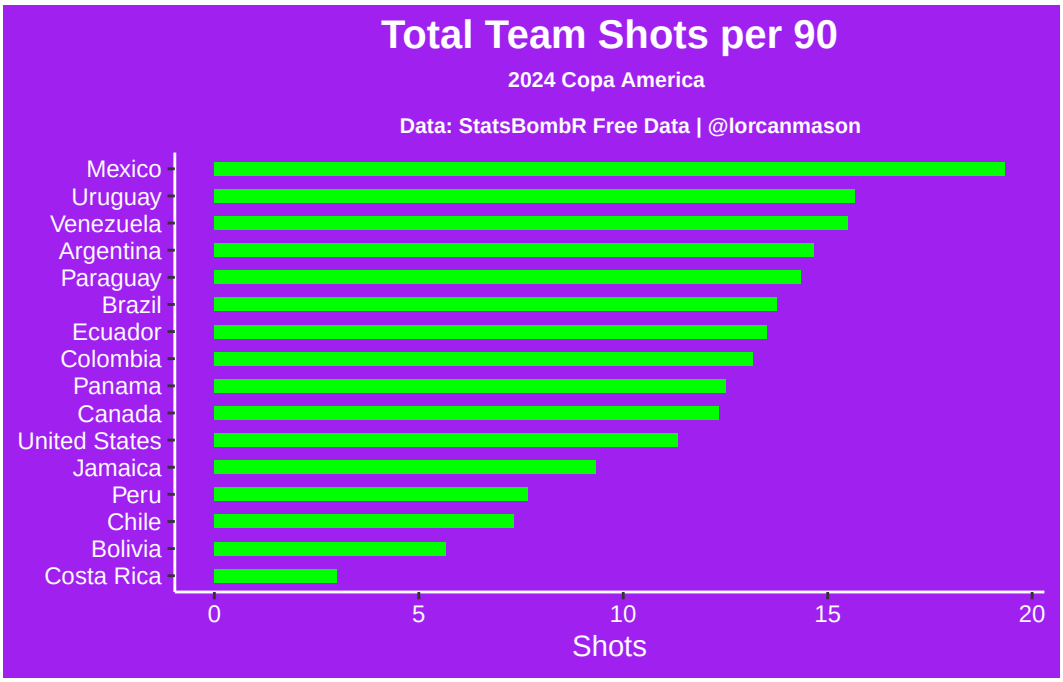
3.1 Load packages

```
library(ggplot2)
#1: This is the package we use to create our plots.
library(ggrepel)
#2: This is the package we use to modify our plots.
library(SBpitch)
#3: This is the package we use to create our pitch.
library(scales)
#4: This is the package we use to modify our scales.
library(prismatic)
#5: This is the package we use to modify our colours.
```

3.2 Plotting team shots and goals

```
ggplot(data = shots_goals,
       aes(x = reorder(team.name, shots), y = shots)) +
  #1: This code sets up the plot and tells it what data to use and
  # what columns to use for the x and y axis.
  geom_bar(stat = "identity", width = 0.5, fill="green") +
  #2: This code tells it to create a bar plot with the data we have given it.
  # stat = "identity" tells it to use the data as it is, width = 0.5 sets the
  # width of the bars, and fill = "green" sets the colour of the bars.
  labs(y="Shots")  +
  #3: This code sets the label for the y axis.
  coord_flip() +
  #4: This code flips the x and y axis.
  theme(
    axis.title.y = element_blank(),
    legend.position = "none",
    plot.background = element_rect(fill = "purple", colour = "purple"),
    panel.background = element_rect(fill = "purple", colour = "purple"),
    panel.grid.major = element_line(colour = "purple"),
    panel.grid.minor = element_blank(),
```

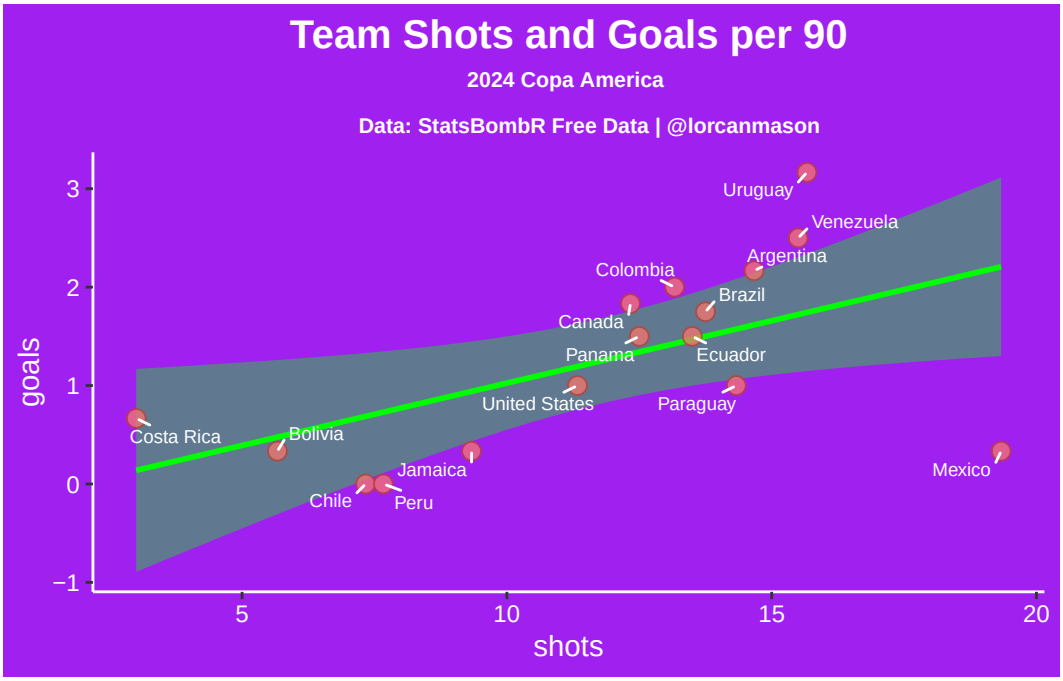
```
    axis.line = element_line(colour = "white"),
    axis.text = element_text(colour = "white"),
    axis.title = element_text(colour = "white"),
    plot.title = element_text(colour = "white", hjust=.5, face="bold", size = 15),
    plot.subtitle = element_text(colour = "white", hjust=.5, face="bold", size = 8)) +
  #5: This code sets the theme for the plot. It sets the background colour
  # grid lines, axis lines, axis text, axis title, plot title, and plot subtitle.
  labs(title = "Total Team Shots per 90",
       subtitle = "2024 Copa America

       Data: StatsBombR Free Data | @lorcanmason")
```




```
#6: This code sets the title and subtitle for the plot.
```

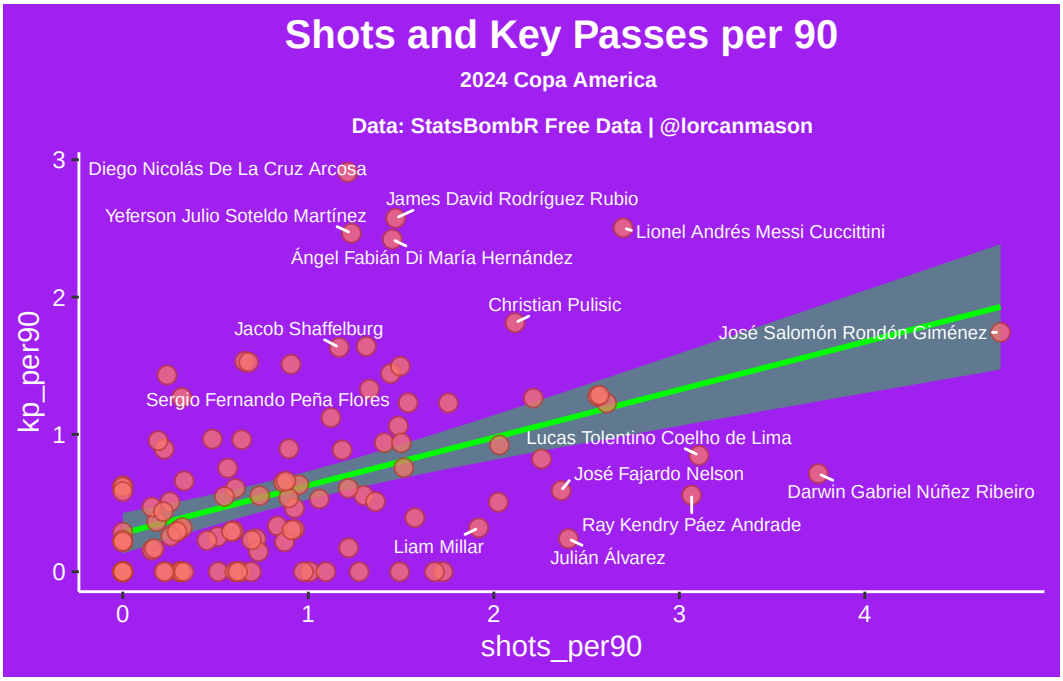
```
ggplot(shots_goals, aes(x = shots, y = goals, label = team.name)) +
  #1: This code sets up the plot and tells it what data to use and what
  # columns to use for the x and y axis.
  geom_smooth(method = "lm", color = "green", fill = "green") +
  #2: This code tells it to create a linear regression line on the plot.
  # method = "lm" tells it to use a linear model, colour = "green" sets the
  # colour of the line, and fill = "green" sets the fill colour of the line.
geom_point(aes(fill = "green", color = after_scale(clr_darken(fill, 0.3))),
             shape = 21,
             alpha = .75,
             size = 3) +
  #3: This code tells it to create points on the plot. aes(fill = "green",
```

```
  # color = after_scale(clr_darken(fill, 0.3))) sets the fill and colour of
  # the points, shape = 21 sets the shape of the points, alpha = .75 sets
  # the transparency of the points, and size = 3 sets the size of the points.
  geom_text_repel(size = 2.5, color = "white", min.segment.length = unit(0.1, "lines")) +
  #4: This code tells it to create text labels on the plot. size = 2.5 sets
  # the size of the text, colour = "white" sets the colour of the text,
  # and min.segment.length = unit(0.1, "lines") sets the minimum length of the segments.
  theme(
    legend.position = "none",
    plot.background = element_rect(fill = "purple", colour = "purple"),
    panel.background = element_rect(fill = "purple", colour = "purple"),
    panel.grid.major = element_line(colour = "purple"),
    panel.grid.minor = element_blank(),
    axis.line = element_line(colour = "white"),
    axis.text = element_text(colour = "white"),
    axis.title = element_text(colour = "white"),
    plot.title = element_text(colour = "white", hjust=.5, face="bold", size = 15),
    plot.subtitle = element_text(colour = "white", hjust=.5, face="bold", size = 8)) +
  labs(title = "Team Shots and Goals per 90",
       subtitle = "2024 Copa America

       Data: StatsBombR Free Data | @lorcanmason")
```



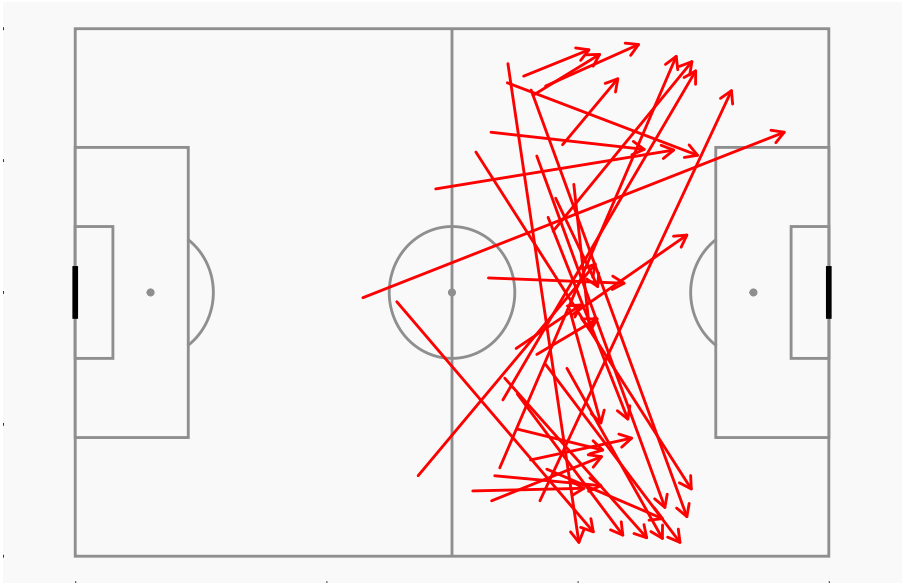
3.3 Plotting player shots and key passes

```
ggplot(player_shots_keypasses, aes(x = shots_per90, y = kp_per90, label = player.name)) +
  geom_smooth(method = "lm", color = "green", fill = "green") +
geom_point(aes(fill = "green", color = after_scale(clr_darken(fill, 0.3))),
             shape = 21,
             alpha = .75,
             size = 3) +
  geom_text_repel(size = 2.5, color = "white", min.segment.length = unit(0.1, "lines")) +
  theme(
    legend.position = "none",
    plot.background = element_rect(fill = "purple", colour = "purple"),
    panel.background = element_rect(fill = "purple", colour = "purple"),
    panel.grid.major = element_line(colour = "purple"),
    panel.grid.minor = element_blank(),
    axis.line = element_line(colour = "white"),
    axis.text = element_text(colour = "white"),
    axis.title = element_text(colour = "white"),
    plot.title = element_text(colour = "white", hjust=.5, face="bold", size = 15),
    plot.subtitle = element_text(colour = "white", hjust=.5, face="bold", size = 8)) +
  labs(title = "Shots and Key Passes per 90",
       subtitle = "2024 Copa America

       Data: StatsBombR Free Data | @lorcanmason")
```



3.4 Plotting passes into the final third

```
create_Pitch() +
  #1: This code creates a pitch.
  geom_segment(data = player_passes, aes(x = location.x, y = location.y,
                                         xend = pass.end_location.x, yend = pass.end_location.y),
               lineend = "round", linewidth = 0.5, colour = "red",
               arrow = arrow(length = unit(0.07, "inches"), ends = "last", type = "open")) +
  #2: This code creates arrows on the pitch to show the passes into the final third.
  labs(title = "Valverde",
       subtitle = "Copa America 2024 Final Third Passes") +
  scale_y_reverse() +
  #3: This code reverses the y axis.
  coord_fixed(ratio = 105/100)
```




```
#4: This code sets the aspect ratio of the pitch.
```