EPL 2023/24 Premier League Stats Analysis

Arjun Vijay Anup (24215155)

Table of Contents

Data Manipulation **2**

Part 1: Analysis **6**

Table: Top 10 Countries with the Most Players: 6

Table: Top Under-22 Players by Efficiency (Goal Contribution vs Expected Goal Contribution) 7

Plot: Expected Assists vs Expected Non-Penalty Goals per 90 Minutes - Liverpool Club 9

Plot: Club-Wise Distribution of Goals and Assists 12

Part 2: R Package **16**

Package Details 16

Purpose 16

Main Functions 17

Part 3: Functions/Programming **21**

Function: `playerStats()` 21

Method: `print()` 22

Method: `summary()` 23

Method: `plot()` - *Player Performance Comparison (Per 90 Minutes)* 25

Outputs 28

References **33**

- Necessary packages are loaded

```
# Loading necessary libraries
library(dplyr) # for table data manipulation
library(ggplot2) # for data visualisation
library(skimr) # for providing summary statistics
library(knitr) # for creating tables
library(tidyr) # for reshaping and tidying data
library(readxl) # for importing excel data
library(stringr) # for string manipulation
library(forcats) # for factor level manipulation
```

Data Manipulation

- The data utilized in this project was sourced from Player Standard Stats 2023-2024 Premier League. This dataset was accessed through one of links provided in the final project instructions (https://fbref.com/en/)
- The dataset includes a mixture of general and performance data of individual players in the English Premier League during the 2023-2024 season. Details of each column (variables) is given in the *Glossary* sheet in the excel workbook (*2023_24_PL_Player_Stats.xlsx*)

```
# First two lines is skipped during import as it contains the heading
# and other irrelevant rows
overallStats<-read_excel('2023_24_PL_Player_Stats.xlsx', sheet="Data",
                         skip = 2)
```

- As the columns names are all in short-handed form, we will be converting them into a more descriptive column name to give a better understanding for the viewer along with removing any rows with NA values

```
# Removing rows having NA values (if any)
overallStats<-na.omit(overallStats)

# Removing irrelavent columns and Renaming column names to meaningful names
overallStats <- overallStats |> select(-Rk,-Matches) |>
  rename("Country"="Nation", "Position"="Pos", "Club"="Squad",
         "Matches Played"="MP", "Minutes Played"="Min", "90s Played"="90s",
         "Goals"="Gls...12", "Assists"="Ast...13", "Goals+Assists"=
           "G+A...14", "Goals (Non-Penalty)"="G-PK...15", "Goals (Penalty)"=
           "PK", "Attempts (Penalty)"="PKatt", "Yellow Cards"="CrdY",
         "Red Cards"="CrdR", "Expected Goals"="xG...20",
         "Expected Goals (Non-Penalty)"= "npxG...21", "Expected Assists"=
           "xAG...22", "Expected Goals (Non-Penalty)+Expected Assists"=
           "npxG+xAG...23", "Progressive Carries"="PrgC",
         "Progressive Passes"="PrgP", "Progressive Passes Recieved"="PrgR",
         "Goals/90"="Gls...27", "Assists/90"="Ast...28",
         "Goals+Assists/90"="G+A...29",
         "Goals/90 (Non-Penalty)"="G-PK...30",
         "Goals (Non-Penalty)+Assists/90"="G+A-PK",
         "Expected Goals/90"="xG...32", "Expected Assists/90"="xAG...33",
         "Expected Goals+Expected Assists/90"="xG+xAG",
         "Expected Goals (Non-Penalty)/90"="npxG...35",
         "Expected Goals (Non-Penalty)+Expected Assists/90"="npxG+xAG...36")
```

- `gsub()` was used to search for values other than upper-case characters and replace replace them with “ ” to make the `Nationality` column consistent.
- As the `Nationality` values for the players are present in the abbreviated form. So we rename them to the real country names by mapping the country names.

```
# Removing all characters other than upper-case ones in the `Nationality`
# column. (includes whitespaces)
overallStats$Country<-gsub("[^A-Z]", "", overallStats$Country)

# Creating country maps values
country_maps<-c("ENG"="England", "USA"="USA", "CIV"="Ivory Coast",
                "MAR"="Morocco", "CRC"="Costa Rica", "FRA"="France",
                "BIH"="Bosnia and Herzegovina", "NGA"="Nigeria",
                "ALG"="Algeria", "NOR"="Norway", "SUI"="Switzerland",
                "NED"="Netherlands", "ESP"="Spain", "BRA"="Brazil",
                "PAR"="Paraguay", "MEX"="Mexico", "ARG"="Argentina",
                "DEN"="Denmark", "JAM"="Jamaica", "TOG"="Togo",
                "GHA"="Ghana", "GRE"="Greece", "CMR"="Cameroon",
                "SEN"="Senegal", "GRN"="Grenada", "WAL"="Wales",
                "TUN"="Tunisia",  "URU"="Uruguay", "ISL"="Iceland",
                "GNB"="Guinea-Bissau", "GER"="Germany", "MLI"="Mali",
                "NIR"="Northern Ireland", "CHI"="Chile", "ALB"="Albania",
                "SCO"="Scotland", "ECU"="Ecuador", "IRL"="Ireland",
                "ITA"="Italy", "POL"="Poland", "BEL"="Belgium",
                "POR"="Portugal", "ZIM"="Zimbabwe", "CZE"="Czech Republic",
                "COL"="Colombia", "ROU"="Romania", "SVK"="Slovakia",
                "SWE"="Sweden", "EGY"="Egypt",  "JPN"="Japan",
                "RSA"="South Africa", "IRN"="Iran", "CRO"="Croatia",
                "KOR"="South Korea","BFA"="Burkina Faso", "AUT"="Austria",
                "HUN"="Hungary", "GAB"="Gabon", "SRB"="Serbia","ANG"="Angola",
                "UKR"="Ukraine", "KVX"="Kosovo", "ISR"="Israel",
                "COD"="Democratic Republic of the Congo", "TUR"="Turkey",
                "NZL"="New Zealand")

# Mapping abbreviated country values with real country values
overallStats$Country<-country_maps[overallStats$Country]
```

- The variables `Position`, `Club` and `Country` were converted to factor type

```
overallStats<-overallStats |>
  mutate(Position=as.factor(Position),
         Club=as.factor(Club),
         Country=as.factor(Country))
```

- The dimension and structure of the dataset is displayed using the `dim()` and `str()` functions

```
# Dimension of the dataset
dim(overallStats)
```

```
[1] 580  35
```

- The dataset contains 580 rows and 35 columns

```
# Structure of the dataset
str(overallStats, width = 84, strict.width = "cut")
```

```
tibble [580 x 35] (S3: tbl_df/tbl/data.frame)
 $ Player                                          : chr [1:580] "Max Aarons" "Jo"..
 $ Country                                         : Factor w/ 66 levels "Albania"..
  ..- attr(*, "names")= chr [1:580] "ENG" "ENG" "USA" "ENG" ...
 $ Position                                        : Factor w/ 10 levels "DF","DF"..
 $ Club                                            : Factor w/ 20 levels "Arsenal"..
 $ Age                                             : num [1:580] 23 17 24 25 25 21..
 $ Born                                            : num [1:580] 2000 2006 1999 19..
 $ Matches Played                                  : num [1:580] 20 1 3 20 27 31 2..
 $ Starts                                          : num [1:580] 13 0 1 18 16 25 2..
 $ Minutes Played                                  : num [1:580] 1237 6 121 1617 1..
 $ 90s Played                                      : num [1:580] 13.7 0.1 1.3 18 1..
 $ Goals                                           : num [1:580] 0 0 0 2 10 6 1 0 ..
 $ Assists                                         : num [1:580] 1 0 0 0 0 1 0 0 0..
 $ Goals+Assists                                   : num [1:580] 1 0 0 2 10 7 1 0 ..
 $ Goals (Non-Penalty)                             : num [1:580] 0 0 0 2 10 6 1 0 ..
 $ Goals (Penalty)                                 : num [1:580] 0 0 0 0 0 0 0 0 0..
 $ Attempts (Penalty)                              : num [1:580] 0 0 0 0 0 0 0 0 0..
 $ Yellow Cards                                    : num [1:580] 1 0 0 2 1 3 3 0 4..
 $ Red Cards                                       : num [1:580] 0 0 0 0 0 0 1 0 1..
 $ Expected Goals                                  : num [1:580] 0 0 0 0.7 5.9 4.3..
 $ Expected Goals (Non-Penalty)                    : num [1:580] 0 0 0 0.7 5.9 4.3..
 $ Expected Assists                                : num [1:580] 0.8 0 0.1 0.1 0.7..
 $ Expected Goals (Non-Penalty)+Expected Assists   : num [1:580] 0.9 0 0.1 0.8 6.6..
 $ Progressive Carries                             : num [1:580] 22 0 4 10 16 111 ..
 $ Progressive Passes                              : num [1:580] 43 0 5 62 23 50 6..
 $ Progressive Passes Recieved                     : num [1:580] 26 0 1 5 110 278 ..
 $ Goals/90                                        : num [1:580] 0 0 0 0.11 0.63 0..
 $ Assists/90                                      : num [1:580] 0.07 0 0 0 0 0.04..
 $ Goals+Assists/90                                : num [1:580] 0.07 0 0 0.11 0.6..
 $ Goals/90 (Non-Penalty)                          : num [1:580] 0 0 0 0.11 0.63 0..
 $ Goals (Non-Penalty)+Assists/90                  : num [1:580] 0.07 0 0 0.11 0.6..
 $ Expected Goals/90                               : num [1:580] 0 0 0 0.04 0.37 0..
 $ Expected Assists/90                             : num [1:580] 0.06 0 0.06 0.01 ..
 $ Expected Goals+Expected Assists/90              : num [1:580] 0.06 0 0.06 0.05 ..
 $ Expected Goals (Non-Penalty)/90                 : num [1:580] 0 0 0 0.04 0.37 0..
 $ Expected Goals (Non-Penalty)+Expected Assists/90: num [1:580] 0.06 0 0.06 0.05 ..
```

- With the dataset now prepared by removing irrelevant variables and handling missing values, applying necessary changes to variable names, and renaming country names for clarity, we can proceed with the detailed analysis.

Part 1: Analysis

Table: Top 10 Countries with the Most Players:

- A table displaying the number of football players in the league from each country in the descending order of number of players is displayed with the help of `kable()` function.
- The purposes of the functions used in the code have been clearly highlighted through inline code comments.

```r
# Creating a new dataframe for table data
player_country_data <- overallStats |>
  # Grouping by the player country using group_by()

  group_by(Country) |>
  # Calculating the count of players for each country using summarise()
  summarise(count=n()) |>

  # Arranging by descending order of player count for each country
  arrange(desc(count)) |>

  # Display top 10 countries
  head(10)

# Displaying table with given column names
kable(player_country_data, col.names = c("Country", "No. of Players"),
      caption = "Top 10 Countries with the Most Players")
```

Table 1: Top 10 Countries with the Most Players

Country	No. of Players
England	200
Brazil	33
France	28
Spain	20
Netherlands	19
Portugal	19
Ireland	17
Belgium	16
Scotland	15
Argentina	14

- **Observations**:
  - Most number of players playing in the Premier League (2023/24 season) are of **English** nationality with a count of 200 players
  - **Brazil** ranks second with 33 players, followed by **France**, which comes third with 28 players
  - The difference between the number of English players and those from Brazil and France is **substantial**
  - Other countries in the top 10 include **Spain**, **Netherlands**, **Portugal**, **Ireland**, **Belgium**, **Scotland** and **Argentina**

Table: Top Under-22 Players by Efficiency (Goal Contribution vs Expected Goal Contribution)

- The table ranks the top under-22 players based on their Goal Contributions compared to their Expected Goal Contributions *i.e* Players with actual Goal Contributions exceeding their Expected Goal Contributions have a higher efficiency %, with the baseline efficiency set at 100% *(equal to the Expected Goals+Assists)*.
- Here, players who are atmost 21 years old, having made more than 10 appearances for the club during the season, and having scored a total of 10 goals or more are considered
- `kable()` function used for displaying the table. The purposes of the functions used in the code have been clearly highlighted through inline code comments.
- **NOTE**: *Goals and Expected Goals include Penalties*

```r
# Creating a new dataframe using pipe operators
under_21_data <- overallStats |>

  # Expected Goals+Assists is calculated and stored in a column using mutate()
  mutate(`Expected Goals+Assists`=`Expected Goals`+`Expected Assists`) |>

  # Performance Threshold % is calculated as the ratio of G+A and Expected G+A
  # multiplied by 100 and stored in a column using mutate()
  mutate(`Efficiency %`=(`Goals+Assists`/`Expected Goals+Assists`)*100) |>

  # Selecting relevant columns to display
  select(Player, Age, Position, Club,
         `Matches Played`, `Expected Goals+Assists`, `Goals+Assists`,
         `Efficiency %`) |>
  # Filtering players with age of 21 or less, atleast 10 matches played
  # this season and more than 10 goals scored
  filter(Age<=21, `Matches Played`>=10, `Goals+Assists`>10) |>

  # Arranging by descending order of Threshold %
  arrange(desc(`Efficiency %`)) |>

  # Displaying top 5 players
  head(5)

# Displaying table with given column names
kable(under_21_data, col.names = c("Player", "Age", "Position", "Club",
         "Matches Played", "Expected G+A", "G+A", "Efficiency %"),
      align = "lcccccc",
      caption = "Top Under-22 Players by Efficiency (G+A vs Expected G+A)")
```

Table 2: Top Under-22 Players by Efficiency (G+A vs Expected G+A)

Player	Age	Position	Club	Matches Played	Expected G+A	G+A	Efficiency %
Jeremy Doku	21	FW,MF	Manchester City	29	7.1	11	154.9296
Michael Olise	21	FW,MF	Crystal Palace	19	11.3	16	141.5929
Rasmus Højlund	20	FW	Manchester Utd	30	9.2	12	130.4348
Cole Palmer	21	FW,MF	Chelsea	33	29.3	33	112.6280
Anthony Elanga	21	FW,MF	Nott'ham Forest	36	13.3	14	105.2632

- **Observations**:
  - Among players who have atleast 10 matches played and scored above 10 goals in the 2023/24 season, **Jeremy Doku** is the highest performer among **Under-22 players**, outperforming his **Expected Goals+Assist** of **7.1** by **55%**, acheiving **11 Goals+Assist** (*Improvement% = Efficiency% - 100%*)
  - **Micheal Olise** came in at 2nd place, outperforming his **Expected Goals+Assist** of **11.3** by **41.6%**, acheiving **16 Goals+Assist**, indicating his strong performance in fewer matches. **Rasmus Højlund** followed him, demonstrating an efficiency of **130.43%** and outperforming his **Expected Goals+Assist** of **9.2** by **30.4%**,

  acheiving **12 Goals+Assists**. He is also the youngest of the lot at 20 years, which shows his potential for the future and makes him a prime target for other clubs to have him in thier roster.

  - **Cole Palmer** came in at 4th position, continuing his impressive run of form as he outperformed his already high **Expected Goals+Assist** of **29.3** by **12.6%**, acheiving an impressive **33 Goals+Assists**. Following him, **Anthony Elanga** also had an impressive 2023/24 season, where he outperformed his **Expected Goals+Assist** by **5.3%**
  - These players not only demonstrated thier exceptional ability in converting goal opportunities, but also highlighted thier potential as upcoming stars in the Premier League

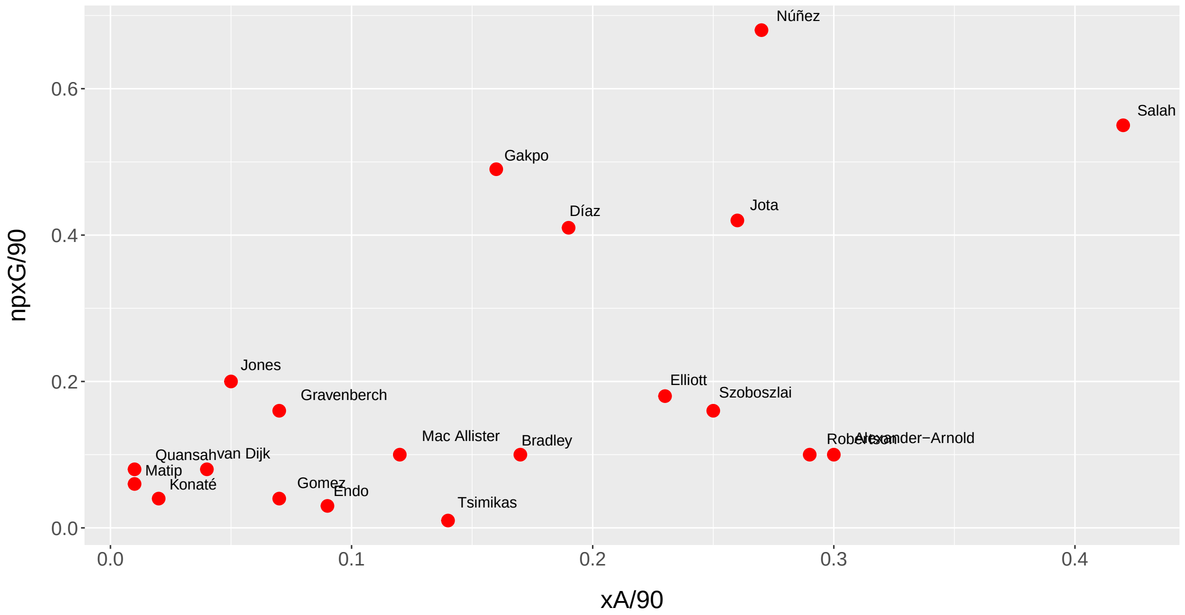
Plot: Expected Assists vs Expected Non-Penalty Goals per 90 Minutes - Liverpool Club

- The scatterplot below shows the relationship between **Expected Assists per 90 minutes (xA/90)** and **Expected Goals (Non-Penalty) per 90 minutes (npxG/90)** for **Liverpool players** in the 2023/24 Premier League season.
- The purposes of the functions used in the code have been clearly highlighted through inline code comments.
- **NOTE:** *Players with only positive xA/90 and npxG/90 have been reported in the plot.*

```r
# Preparing Dataframe for Liverpool players
liverpool_data<-overallStats|>

  # Selecting relevant columns
  select(Player, Club, `Expected Assists/90`,
         `Expected Goals (Non-Penalty)/90`) |>

  # Filtering data for players in Liverpool having more than
  # 0 Expected Assists and Expected Non-Penalty goals per 90 minutes
  filter(Club == "Liverpool" &
           `Expected Assists/90` > 0 &
           `Expected Goals (Non-Penalty)/90` > 0) |>

  # Creating new column `First Name` and `Last Name` and extracting first name
  # and Last Name of the players using word() from `stringr` package
  mutate(`First Name`= word(Player, start = 1),
         `Last Name`= word(Player, start = 2, end = -1))
```

```
# Plotting scatter plot of xA/90 vs npxG/90
ggplot(data=liverpool_data,
       aes(x=`Expected Assists/90`,y=`Expected Goals (Non-Penalty)/90`,
           label=`Last Name`))+

  # Point color and size are given
  geom_point(color="red", size=4)+

  # Adding plot title, caption and axis labels
  labs(
  title = "Expected Assists vs Expected Non Penalty Goals per 90 by Player
(Liverpool 2023/24)",
       caption = "Source: 2023_24_PL_Player_Stats.xlsx
(Player Standard Stats 2023-2024 Premier League)",
       x = "xA/90", y = "npxG/90")+

  # Labelling `Last Name` of players near points for clarity
  geom_text(position="jitter", vjust=-1, hjust=-0.2, size = 4, angle=0)+

  # Legend, Axes and Plot text and title position and styling customization
    theme(legend.position = "top",
        legend.text = element_text(size=12),
        legend.title = element_text(size=14),
        axis.title.x = element_text(size=18,margin = margin(t=15)),
        axis.title.y = element_text(size=18,margin = margin(r=15)),
        axis.text.x = element_text(size=14),
        axis.text.y = element_text(size=14,),
        plot.title = element_text(size=24,hjust=0.5,vjust=1,
                                  face="bold",margin = margin(b=30)),
        plot.title.position = "panel",
        plot.caption = element_text(size=12,hjust=0.5,vjust = 0,
                                    face="italic",margin=margin(t=20)),
        plot.caption.position = "panel")
```

Expected Assists vs Expected Non Penalty Goals per 90 by Player (Liverpool 2023/24)



Source: 2023_24_PL_Player_Stats.xlsx
(Player Standard Stats 2023−2024 Premier League)

- **Observations**:
  - **Mohamed Salah** and **Darwin Nunez** stands out from the rest of the players with high values for both **Expected Non-Penalty Goals per 90** (npxG/90) and **Expected Assists per 90** (xA/90), with **Salah** contributing significantly in Goals (0.55) and Assists (0.45), indicating that he is a complete attacking player being a prolific creator and consistent goal-scoring threat. **Nunez** had more goal-wise contribution across the season with a higher npxG/90 than **Salah** (around 0.65) but had a lower involvement in assists (around 0.275)
  - **Trent Alexander-Arnold** and **Andrew Robertson** display the highest xA/90 (around 0.3) with a considerably low npxG/90 (0.1) among all other remaining players, which emphasizes thier massive and key role in creating scoring opportunites for the team.
  - **Cody Gakpo**, **Luiz Diaz** and **Diego Jota** have notable contributions to both metrics, with slightly more contributions in goals rather than assists. Among them, **Gakpo** has the highest npxG/90 of around 0.5, while **Jota** leads in creating chances with an xA/90 at around 0.6
  - **Dominik Szoboszlai** and **Harvey Elliott** also have modest involvment in goals as well as assists, with them having the most balanced ratios between xA/90 and

    npxG/90 among the lot. The lower metrics reflects thier deeper midfield roles, focusing less on direct goal involvement.

  - Players that are grouped up near the orgin (**Van Dijk, Konate, Matip etc**), have minimal involvment in attacking plays. This highlights thier primary roles as Defenders, focusing on defensive duties rather than contributing to offensive efforts.

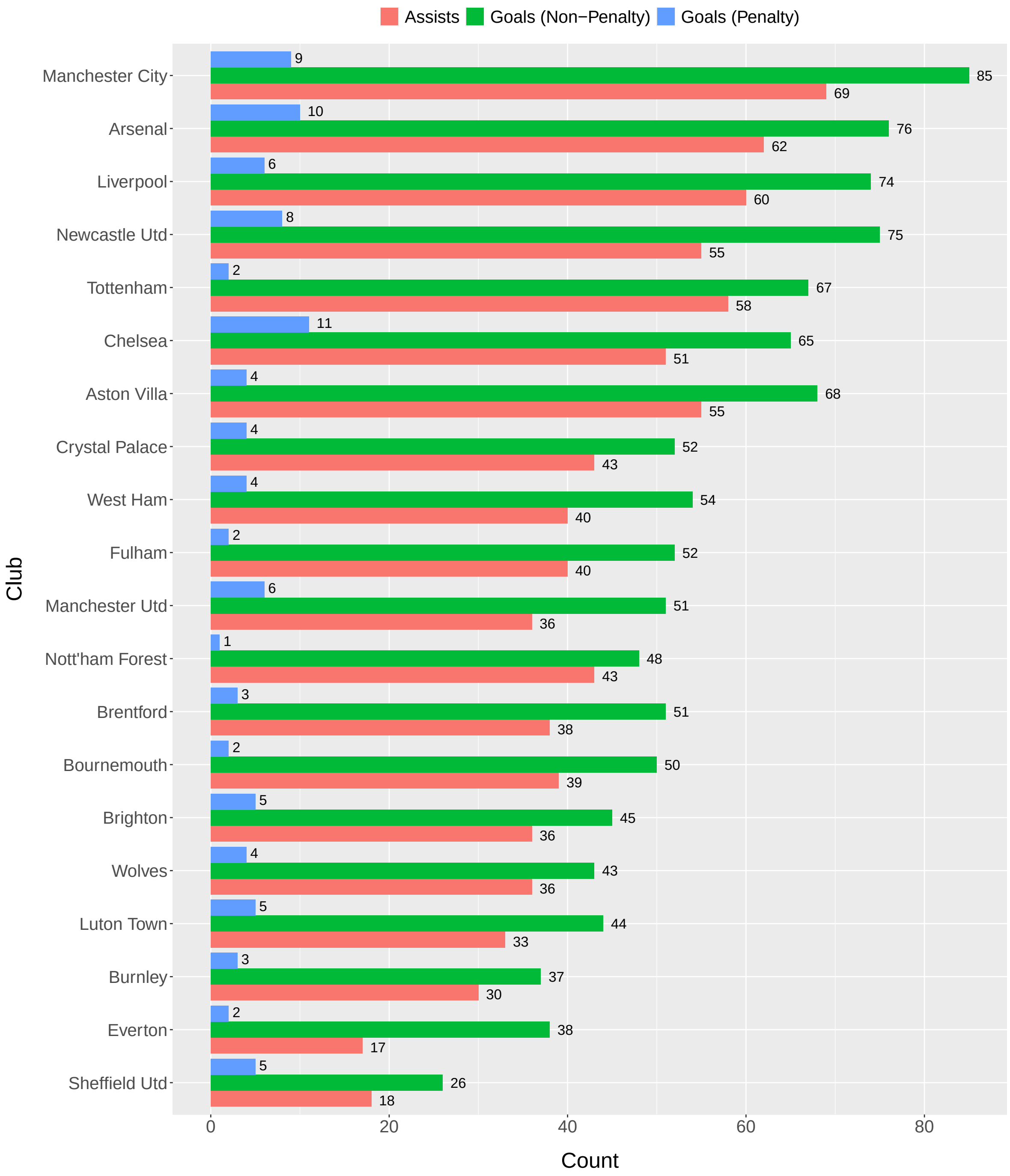
Plot: Club-Wise Distribution of Goals and Assists

- The barplot below shows the **Club-wise distribution of goals and assists** for the 2023/24 Premier League season. The distributions are categorized into **Non-Penalty Goals**, **Penalty Goals** and **Assists**
- The purposes of the functions used in the code have been clearly highlighted through inline code comments.

```r
# Preparing dataset for overall goal and assists count
goal_assist_data <- overallStats |>

  # Selecting relevant columns
  select(Player, Club, `Goals (Non-Penalty)`, `Goals (Penalty)`, Assists) |>

  # Using pivot_longer() and converting Goals and Assists into Long Format
  pivot_longer(cols = c(`Goals (Non-Penalty)`, `Goals (Penalty)`, Assists),
               names_to = "type", values_to = "count") |>

  # Grouping by Club and Type
  group_by(Club, type) |>

  # Taking sum of individual player contributions per Club per type and
  # then drop all levels of grouping using .groups argument after
  summarise(count=sum(count, na.rm=TRUE), .groups = "drop") |>

  # Grouping by Club and Summing total contributions (G+A) for each club
  group_by(Club) |>
  mutate(total = sum(count, na.rm=TRUE)) |>

  # Reordering Club by Total contributions (G+A) in descending order
  ungroup() |>
  mutate(Club = fct_reorder(Club, total, .fun = sum, na.rm = TRUE)) |>

  # Arranging in descending order of total contributions (Goals + Assists)
  arrange(desc(total))
```

```r
# Plotting barplot of Club and Total Goal contributions
ggplot(goal_assist_data, aes(x = Club, y = count,
                             fill = type, label = count)) +

  # Flipping coordinates for better view
  coord_flip() +

  # Using bar plot with dodged bars for Non-penalty, Penalty Goals and Assists
  geom_bar(stat = "identity", position = "dodge") +
  geom_text(position = position_dodge(width = 1),
            hjust=-0.5, size = 4.5)+

  # Adding plot title, caption and axis labels
  labs(title = "Club-Wise Distribution of Goals and Assists
(2023/24 Season)",
       caption = "Source: 2023_24_PL_Player_Stats.xlsx
(Player Standard Stats 2023-2024 Premier League)",
       x = "Club", y = "Count") +

  # Legend, Axes and Plot text and title position and styling customization
  theme(legend.position = "top",
        legend.text = element_text(size=16),
        legend.title = element_blank(),
        axis.title.x = element_text(size=20,margin = margin(t=15)),
        axis.title.y = element_text(size=20,margin = margin(r=15)),
        axis.text.x = element_text(size=16),
        axis.text.y = element_text(size=16),
        plot.title = element_text(size=26,hjust=0.5,vjust=1,
                                  face="bold",margin = margin(b=30)),
        plot.title.position = "panel",
        plot.caption = element_text(size=14,hjust=0.5,vjust = 0,
                                    face="italic",margin=margin(t=20)),
        plot.caption.position = "panel")
```

Club−Wise Distribution of Goals and Assists (2023/24 Season)



Source: 2023_24_PL_Player_Stats.xlsx
(Player Standard Stats 2023−2024 Premier League)

- **Observations**
  - **Manchester City** has the highest overall contributions in attack, with the highest number of Non-Penalty goals scored (85) and Highest number of assists (69), showcasing thier dominating presence in attacking output.
  - **Arsenal**, **Liverpool** and **Newcastle United** has the 2nd, 3rd and 4th highest total contributions in attack, among which **Arsenal** has a higher amount of penalty goals scored across the season at 10 goals. **Newcastle Utd** stands out with notable number of Non-Penalty goals but lower Assists, indicating more individual scoring efforts.
  - **Tottenham** showcases the most balanced distribution between Non-Penalty goals (67) and assists (58), highlighting thier cohesive team play.
  - Clubs such as **Chelsea**, **Manchester United** have higher attacking contribution from penalty goals when compared to thier total goals, showcasing thier reliance penalty kicks for scoring goals.
  - Clubs present in the bottom 4 (**Sheffield United**, **Everton**, **Burnley**, and **Luton Town**) have lowest overall contributions across all categories, denoting the difficulties they faced in attacking output during the season

Part 2: R Package

Package Details

- Name: **`summarytools` (v1.01)**
- Author: Dominic Comtois [aut, cre]
- Published: 2022-05-20
- Source: https://CRAN.R-project.org/package=summarytools

First, we check whether the package has already been installed. If not, the package is installed and loaded into the session. If the package is already installed, it is simply loaded into the session without going through the installation process.

```r
# Checking if `summarytools` package has been installed
# If not, it is installed and loaded in the current session
if(!require(summarytools)){
  install.packages("summarytools")
}
library(summarytools)
```

NOTE: *If you use macOS, please install XQuartz (https://www.xquartz.org) before running this package.*

If XQuartz is not installed on **macOS** you may face crashes when **rendering the PDF** (`summarytools` uses X11 windowing system and XQuartz is the the macOS implementation of X11)

Purpose

- **`summarytools`** package is used here to provide set of tools for concise, easy-to-read and user friendly summaries. It is also useful for exploring data, generating descriptive statistics, creating reports, as well as weighted frequency tables.

- Here are the some of the functionalities it offers:

  - Descriptive statistics for numeric as well as categorical variables
  - Frequency data/distributions (The number of times a value is repeated for a dataset)
  - Perform grouped analysis (Individual statistics or descriptive summaries)

Main Functions

1. **Descriptive Statistics: `descr()`**

    - `descr()` displays the descriptive statistics *(Mean, Standard deviation, 1st and 3rd quartile values, IQR, Min, Max etc)* for variables present in the dataset *(only numeric)*

    - The general syntax is `desc(data, stats="all")` where `data` is the dataset used and `stats` specifies the type of statistics to be displayed. Most common `stats` used are given below:

        - `"fivenum"` - displays Min, Q1, Median, Q3 and Max
        - `"common"` - displays Mean, Standard Deviation, Min, Median, Max, N.valid, Pct.Valid
        - `"all"` - It includes statistics from `"fivenum"` and `"common"`, along with skewness, SE. Skewness, Kurtosis, N.Valid and Pct. Valid

    - First, a dataset was prepared with only selected columns, as we have a large number of columns in the dataset which can lead to summary output spilling out of the pages.

    - The demonstration below provides a summary of selected columns `Minutes Played`, `Goals` and `Assists` from the dataset. (`descr()` excludes `Position` as it is a non-numerical variable). Stats value of `all` is used.

```r
# Dataset with selected columns and filtered on certain positions
descr_data <- overallStats |>
  filter(Position %in% c("FW", "MF", "DF")) |>
  select(`Minutes Played`, Goals, Assists, Position) |>

  # Unused factor levels dropped to prevent interference in stat data
  mutate(Position = droplevels(Position))

# Generating descriptive statistics for the dataset
descr_out<-descr(descr_data, stats = "all")
print(descr_out)
```

```
Non-numerical variable(s) ignored: Position
```

```
Descriptive Statistics
descr_data
N: 378
```

```
                    Assists    Goals   Minutes Played
----------------- --------- -------- ----------------
             Mean      1.45     2.05          1333.34
          Std.Dev      2.28     3.67          1023.98
              Min      0.00     0.00             1.00
               Q1      0.00     0.00           386.00
           Median      0.00     1.00          1240.00
               Q3      2.00     2.00          2150.00
              Max     13.00    27.00          3420.00
              MAD      0.00     1.48          1321.00
              IQR      2.00     2.00          1758.50
               CV      1.57     1.79             0.77
         Skewness      2.21     3.10             0.28
      SE.Skewness      0.13     0.13             0.13
         Kurtosis      5.11    11.77            -1.13
          N.Valid    378.00   378.00           378.00
                N    378.00   378.00           378.00
        Pct.Valid    100.00   100.00           100.00
```

2. **Frequency Distribution: `freq()`**

- `freq()` provides detailed information regarding the frequency of a specific numeric or categorical variable by displaying a frequency table.
- The table displays the following frquency data of each categorical or numerical variable:
  - Frequency
  - Valid %
  - Valid cumulative %
  - Total %
  - Total cumulative %
- General syntax is `freq(data, order = "name")`, where `data` is the dataset and `order` is the parameter by which the table values are ordered. Following are the `order` parameter values that can be used
  - `"name"`: This is the default ordering method in case of non-factor variables (Arranged in descending alphabetical names)
  - `"level"`: This is the default ordering method in case of factors (Arranged in descending alphabetical factor names)
  - `"freq"`: Arranges frequency values in the descending order (Highest count to least count)
- The demonstration of the function below provides the frequency distribution of the `Goals` variable in the dataset, arranged by the frequency count. (`order = "freq"`)

```r
# Generating frequency distribution of goals
freq_data<-freq(overallStats$`Goals`, order = "freq")
print(freq_data)
```

```
Frequencies
overallStats$Goals
Type: Numeric

              Freq   % Valid   % Valid Cum.   % Total   % Total Cum.
----------- ------ --------- -------------- --------- --------------
          0    282     48.62          48.62     48.62          48.62
          1     99     17.07          65.69     17.07          65.69
          2     55      9.48          75.17      9.48          75.17
          3     34      5.86          81.03      5.86          81.03
          4     24      4.14          85.17      4.14          85.17
          5     16      2.76          87.93      2.76          87.93
          6     14      2.41          90.34      2.41          90.34
          7     12      2.07          92.41      2.07          92.41
          8      9      1.55          93.97      1.55          93.97
         10      6      1.03          95.00      1.03          95.00
         11      6      1.03          96.03      1.03          96.03
          9      5      0.86          96.90      0.86          96.90
         12      4      0.69          97.59      0.69          97.59
         16      3      0.52          98.10      0.52          98.10
         19      3      0.52          98.62      0.52          98.62
         14      2      0.34          98.97      0.34          98.97
         13      1      0.17          99.14      0.17          99.14
         17      1      0.17          99.31      0.17          99.31
         18      1      0.17          99.48      0.17          99.48
         21      1      0.17          99.66      0.17          99.66
         22      1      0.17          99.83      0.17          99.83
         27      1      0.17         100.00      0.17         100.00
       <NA>      0                                0.00         100.00
      Total    580    100.00         100.00    100.00         100.00
```

3. **Grouped Data Analysis: `stby()`**

   - `stby()` applies a specific function (`descr()`, `freq()`, `mean()`, `sum()` etc) to grouped data, which is grouped by one or more factors. It then displays the corresponding function output, grouped by the corresponding factor(s).
   - General syntax is `stby(data, INDICES=..., FUN=...)`, where `data` is the dataset, `INDICES` is the variable by which grouping should be done and `FUN` is the

`summarytools` package function that should be applied to the grouped data.

- The demonstration of the function below displays the descriptive statistics data (using `FUN = descr`) of the `descriptive_data` dataset (created earlier in this section) for the number of minutes played for each `Position` value (using `INDICES = descriptive_data$Position`).

NOTE: *The position values were filtered to certain positions (DF, FW, MF) due to the high amount of unique `Position` values in the dataset and printing stats for all of them would result in the output spilling over the edges of the rendered PDF*

```r
# Grouped descriptive statistics, grouped by Positions
position_stby_data<-stby(data = descr_data$`Minutes Played`,
     INDICES = descr_data$Position, FUN = descr)
print(position_stby_data)
```

```
Descriptive Statistics
`Minutes Played` by Position
Data Frame: descr_data
N: 378

                         DF          FW          MF
----------------- --------- --------- ---------
             Mean   1508.80   1089.31   1274.54
          Std.Dev    975.01   1015.22   1062.67
              Min      1.00      1.00      1.00
               Q1    739.00     63.00    212.00
           Median   1384.50    991.00   1161.00
               Q3   2331.00   2019.00   2120.00
              Max   3420.00   3325.00   3263.00
              MAD   1136.41   1414.40   1421.81
              IQR   1587.75   1923.00   1908.00
               CV      0.65      0.93      0.83
         Skewness      0.23      0.43      0.32
      SE.Skewness      0.19      0.25      0.23
         Kurtosis     -1.03     -1.16     -1.28
          N.Valid    170.00     95.00    113.00
                N    170.00     95.00    113.00
        Pct.Valid    100.00    100.00    100.00
```

Part 3: Functions/Programming

The code chunks in this section demonstrates the use of S3 classes with appropriate print(), summary() and plot() methods. Detailed explanation of how the function and its methods work have been provided below:

Function: `playerStats()`

- `playerStats()` function defines an **S3 Class** `playerStats` which analyses player performance data. Below are the operations done in the function.
  - It checks and validates if all required columns are present within the dataset and passed into the function.
  - Creates an object with selected Variables (Player, Country, Position, Club, Age, Minutes Played , Goals (Non-Penalty, Penalty and Total), Assists, Expected Goals per 90, Expected Assists per 90)
- The purposes of the other functions used in the code have been clearly highlighted through the inline code comments.

Example:

```
# Defining the S3 Class
playerStats<-function(p){

  # Checking if the given set of columns are present in the dataset
  cols<-c("Player", "Country", "Position", "Club", "Age", "Minutes Played",
          "Goals", "Assists", "Goals (Non-Penalty)", "Goals (Penalty)",
          "Expected Goals/90", "Expected Goals (Non-Penalty)/90",
          "Expected Assists/90", "Goals/90 (Non-Penalty)", "Goals/90",
          "Assists/90")

  # If not, error message is printed and function exited
  if (!all(cols %in% names(p))) {
    stop("Data must contain the following columns: ",
         paste(cols, collapse = ", "))
  }

  # Creating the object
  p<-list(
    player=p$Player, country=p$Country, pos=p$Position, club=p$Club,
    age=p$Age, min_played=p$`Minutes Played`, goals=p$Goals,
    assists=p$Assists, pen_goals=p$`Goals (Penalty)`,
    n_pen_goals=p$`Goals (Non-Penalty)`, xG=p$`Expected Goals/90`,
    xA=p$`Expected Assists/90`, npxG=p$`Expected Goals (Non-Penalty)/90`,
    G_90=p$`Goals/90`, npG_90=p$`Goals/90 (Non-Penalty)`,
    A_90=p$`Assists/90`
    )

  # Assigning the S3 class
  class(p)<-"playerStats"
  return(p)
}
```

Method: `print()`

- The `print.playerStats()` method displays basic information about the object created by the class `playerStats`. The `print.playerStats()` method displays the following :
  - Total number of players
  - Positions
  - First 10 entries of the object with the *Player Name*, *Position*, *Club*, *Age*, *Minutes Played*, *Goals* and *Assists*
- The purposes of the other functions used in the code have been clearly highlighted through the inline code comments.

Example:

```
# Defining print method for the above S3 Class `playerStats`
print.playerStats<-function(x, ...){
  cat(" Player Performance Data\n")
  cat("|----------------------------------------------------------|\n")
  cat("Total No. of Players:", length(x$player), "\n")
  cat("Positions:\n", paste(unique(x$pos), collapse="\n "), "\n\n")
  cat("-----Data (First 10 Players)-----\n\n")

  # Creating dataframe to store basic player information
  p_data<-data.frame(
    Player=x$player,
    Position=x$pos,
    Club=x$club,
    Age=x$age,
    `Minutes Played`=x$min_played,
    Goals=x$goals,
    Assists=x$assists
  )

  # Print the first 10 entries of the dataframe
  print(head(p_data, 10))
}
```

Method: `summary()`

- The `summary.playerStats()` methods displays summary statistics of the each numerical and categorical variables.
    - Numerical Summary: Min, Mean and Max for `Goals`, `Assists`, `Goals (Non-Penalty)`, `Goals (Penalty)`, `Expected Goals/90` etc
    - Categorical Summary: Total number of players, countries, clubs and age range of players present in the dataset
- The purposes of the other functions used in the code have been clearly highlighted through the inline code comments.

Example:

```
# Defining summary method for S3 Class `playerStats`
summary.playerStats <- function(x, ...) {
  cat("Summary of Player Performance Statistics\n")
  cat("|---------------------------------------|\n\n")

  # Calculate min, average, and max of the selected variables into a dataframe
  num_summary <- data.frame(
    `Minutes Played`=c(min(x$min_played, na.rm = TRUE),
                       mean(x$min_played, na.rm = TRUE),
                       max(x$min_played, na.rm = TRUE)),

    Goals=c(min(x$goals, na.rm = TRUE),
            mean(x$goals, na.rm = TRUE),
            max(x$goals, na.rm = TRUE)),

    Assists=c(min(x$assists, na.rm = TRUE),
              mean(x$assists, na.rm = TRUE),
              max(x$assists, na.rm = TRUE)),
    `Goals(Penalty)`=c(min(x$pen_goals, na.rm = TRUE),
                       mean(x$pen_goals, na.rm = TRUE),
                       max(x$pen_goals, na.rm = TRUE)),

    `Goals(Non-Penalty)`=c(min(x$n_pen_goals, na.rm = TRUE),
                           mean(x$n_pen_goals, na.rm = TRUE),
                           max(x$n_pen_goals, na.rm = TRUE)),

    `Expected Goals/90`=c(min(x$xG, na.rm = TRUE),
                          mean(x$xG, na.rm = TRUE),
                          max(x$xG, na.rm = TRUE)),

    `Expected Assists/90`=c(min(x$xA, na.rm = TRUE),
                            mean(x$xA, na.rm = TRUE),
                            max(x$xA, na.rm = TRUE)),

    `Expected Goals (Non-Penalty)/90`=c(min(x$npxG, na.rm = TRUE),
                                        mean(x$npxG, na.rm = TRUE),
                                        max(x$npxG, na.rm = TRUE)),

    `Goals/90`=c(min(x$G_90, na.rm = TRUE),
                 mean(x$G_90, na.rm = TRUE),
                 max(x$G_90, na.rm = TRUE)),

    `Goals/90 (Non-Penalty)`=c(min(x$npG_90, na.rm = TRUE),
                               mean(x$npG_90, na.rm = TRUE),
                               max(x$npG_90, na.rm = TRUE)),

    `Assists/90`=c(min(x$A_90, na.rm = TRUE),
                   mean(x$A_90, na.rm = TRUE),
                   max(x$A_90, na.rm = TRUE))
  )

  # Taking transpose of the dataframe for better readability and cleaner view
  num_summary<-t(num_summary)

  # Setting column names for Minimum, Maximum and Average
  colnames(num_summary)<-c("Min", "Avg", "Max")

  # Displaying the numeric summary without row numbers for cleaner view
  cat("Numerical Summary:\n")
  cat("-----------------\n")
```

```
  print(num_summary, quote=FALSE)

  # Displaying categorical summary with Unique values and counts
  cat("\nCategorical Summary:\n")
  cat("--------------------\n")
  cat("Total Players:", length(x$player), "\n")
  cat("No. of Countries represented:", length(unique(x$country)), "\n")
  cat("No. of Clubs Represented:", length(unique(x$club)), "\n")
  cat("Positions:", "(", paste(unique(x$pos), collapse = "),("), ")", "\n")
  cat("Age Range:",
      min(x$age, na.rm = TRUE), "-", max(x$age, na.rm = TRUE), "\n")
}
```

Method: `plot()` - *Player Performance Comparison (Per 90 Minutes)*

- The `plot.playerStats()` methods displays a **Radarchart** for **Player Performance Comparison (Per 90 Minutes)** - using `fmsb` package for plotting - which compares the following performance attributes for upto 4 players.
  - Expected Goals/90
  - Expected Goals (Non-Penalty)/90
  - Expected Assists/90
  - Goals/90
  - Assists/90
  - Goals (Non-Penalty)/90
- Each player is represented by different colors for a cleaner view (`Blue, Red, Green` and `Brown`)
- The method has an inbuilt check to ensure that atleast 2 players are selected when the `plot.playerStats()` method is called and limits the player selection for upto 4.
- The plotting package `fmsb` has been used here as this approach offers a better visualization for performance data comparisons within players than traditional table data, scatterplots or dodged bar plots.
- Details of plotting package used:
  - Name: **`fmsb` (v0.7.6)**
  - Author: Minato Nakazawa
  - Published: 2024-01-19
  - Source: https://CRAN.R-project.org/package=fmsb

- The purposes of the other functions used in the code have been clearly highlighted through the inline code comments.

Example:

```
# Defining plot method for S3 Class `playerStats`
plot.playerStats<-function(x, p, ...){

  # Check to see if player list has more than 4 players
  # If yes, plotting function is exited with an error message
  if(length(p)>4){
    stop(
      "\nOnly a maximum of 4 players are allowed for
      Player Performance Radarchart!\n"
      )

  # Check to see if player list has less than 2 players
  # If yes, the plotting function is exited with an error message
  } else if(length(p)<2){
    stop(
      "\nEnter names of atleast 2 players to generate their
      Performance Comparison Radarchart!\n"
      )
  }

  # Check and quietly install the package if not installed
  # Load the package
  if(!require(fmsb, quietly = TRUE)){
  install.packages("fmsb", quiet = TRUE)
  }
  library(fmsb)

  # Creating dataframe for the radarchart
  player_radar_data <- data.frame(
    Player=x$player,
    xG=x$xG, xA=x$xA, npxG=x$npxG,
    Goals=x$G_90, `Goals (Non-Penalty)` = x$npG_90,
    Assists=x$A_90)

  # Filtering for the selected players for comparison
  player_radar_data<-player_radar_data[player_radar_data$Player %in% p,]

  # Check if the set of the 3 players are available in the dataset
  # If not, error is thrown and function exited
  if(nrow(player_radar_data)==0){
    stop("No matching players found with the given names.
         Please check and give correct names!\n")
  } else if(nrow(player_radar_data)<2){
    stop("Please enter valid names of atleast 2 players to
         generate their Performance Comparison Radarchart.\n")
  }

  # Normalizing data for the radar chart
  max_row<-apply(player_radar_data[-1], 2, max, na.rm = TRUE)
  min_row<-apply(player_radar_data[-1], 2, min, na.rm = TRUE)
  norm_radar_data<-rbind(max_row, min_row, player_radar_data[,-1])

  # Assigning appropriate row names
  rownames(norm_radar_data) <- c("Max", "Min", player_radar_data$Player)

  # Plotting radar chart using radarchart()
  radarchart(df = norm_radar_data,
             # Axis Type (Labelled axes)
             axistype = 1,
             # Radar Line color
             pcol = c("blue", "red", "green", "brown"),
             # Setting Line width
             plwd = 2,
             # Radar area fill colors with alpha = 0.4
             pfcol = c(rgb(0, 0, 1, 0.4),
                       rgb(1, 0, 0, 0.4),
                       rgb(0, 1, 0, 0.4),
                       rgb(0.65, 0.16, 0.16, 0.4)
                       ),
             # Setting title of plot
             title = "Player Performance Comparison (Per 90 Minutes)"
             )

  # Adding legend
  legend("topright",
         legend = rownames(norm_radar_data)[3:nrow(norm_radar_data)],
         col = c("blue", "red", "green", "brown"),
         lty =1, lwd =2, bty ="n")

}
```

- As we are working with Quarto, we explicitly register the S3 methods using `registerS3methods()` to avoid potential conflicts with packages having the same generic function or methods.

```
# Registering the method - needed when working with Quarto or RMarkdown.
registerS3method("print", "playerStats", print.playerStats)
registerS3method("summary", "playerStats", summary.playerStats)
registerS3method("plot", "playerStats", plot.playerStats)
```

- Printing the list of created methods for `playerStats` class.

```
# Examining the created methods for the `playerStats` class
methods(class = "playerStats")
```

```
[1] plot    print   summary
see '?methods' for accessing help and source code
```

Outputs

- The `playerStats()` function is called to create the `playerStats` object. It then organizes the dataset into a structured object containing player general attributes (eg: `player`, `country`) and performance variables (eg: `Goals`, `xG`, `xA`) and assigns it to the S3 class `playerStats`
- The structured object is then stored in `player_performance`

```
player_performance<-playerStats(overallStats)
```

- The `print.playerStats()` method is called for the `playerStats` class
- It displays the general data (Total number of players, Positions and First 10 data entries of the object with General player details (eg: `Name`, `Position`, `Club`, `Age`) and basic performance metrics (`Goals`, `Assists`, `Minutes Played`))

```
print(player_performance)
```

```
 Player Performance Data
|------------------------------------------------------------|
Total No. of Players: 580
Positions:
 DF
 MF
 FW
 MF,FW
 DF,MF
 GK
 MF,DF
 FW,MF
 DF,FW
 FW,DF

-----Data (First 10 Players)-----

              Player Position           Club Age Minutes.Played Goals Assists
1         Max Aarons       DF     Bournemouth  23           1237     0       1
2  Joshua Acheampong       DF        Chelsea  17              6     0       0
3        Tyler Adams       MF    Bournemouth  24            121     0       0
4   Tosin Adarabioyo       DF         Fulham  25           1617     2       0
5     Elijah Adebayo       FW     Luton Town  25           1419    10       0
6      Simon Adingra       FW       Brighton  21           2222     6       1
7       Nayef Aguerd       DF       West Ham  27           1857     1       0
8   Brandon Aguilera       FW Nott'ham Forest  20              5     0       0
9   Naouirou Ahamada    MF,FW  Crystal Palace  21            349     0       0
10  Anel Ahmedhodžić       DF   Sheffield Utd  24           2649     2       0
```

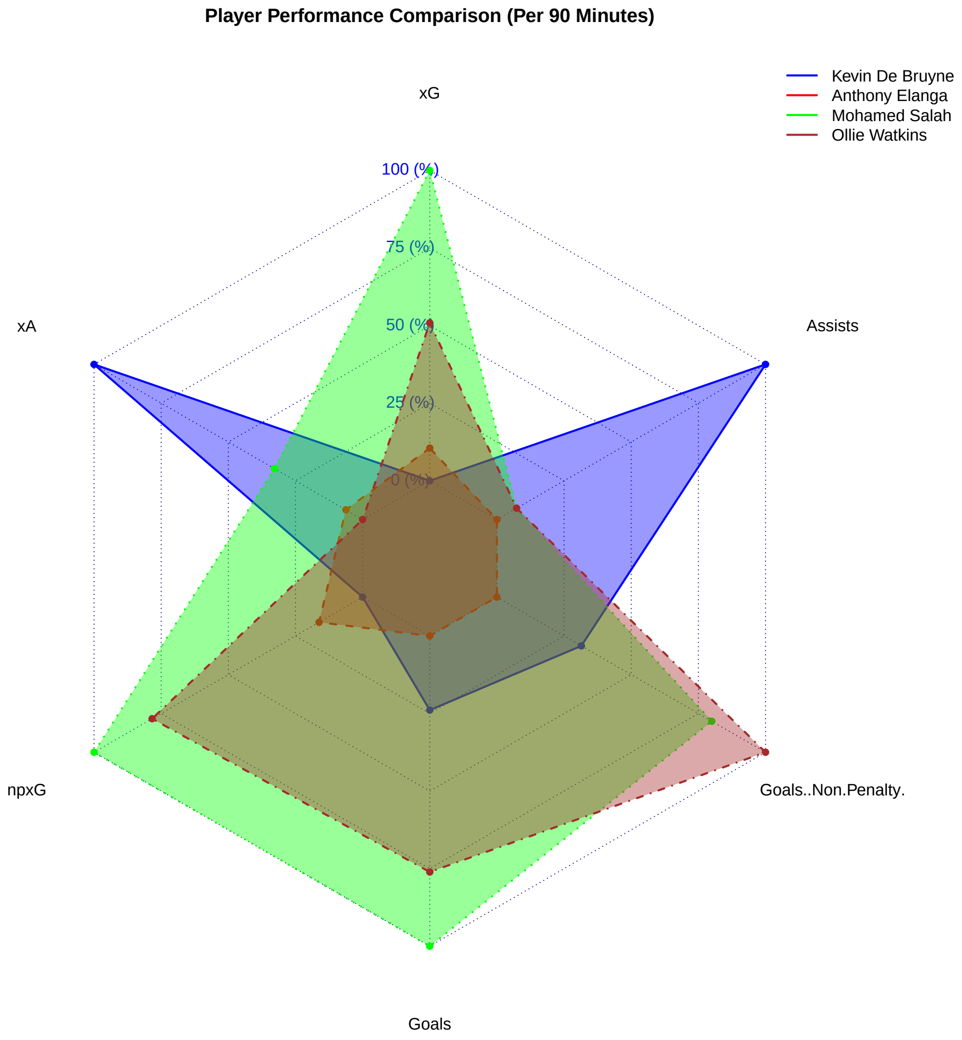
- The `summary.playerStats()` method is then called for the `playerStats` class
- Statistical summary of the player performance metrics present in the object is displayed
  - Numerical Summary: `Min`, `Avg`, `Max`,
  - Categorical Summary: `Total No. of Players`, `Countries`, `Clubs`, `Age range`, `Positions`

```r
summary(player_performance)
```

```

Summary of Player Performance Statistics
|---------------------------------------|

Numerical Summary:
-----------------
                                Min          Avg    Max
Minutes.Played                    1 1.294584e+03 3420.00
Goals                             0 2.063793e+00   27.00
Assists                           0 1.481034e+00   13.00
Goals.Penalty.                    0 1.655172e-01    9.00
Goals.Non.Penalty.                0 1.898276e+00   20.00
Expected.Goals.90                 0 1.449828e-01    3.23
Expected.Assists.90               0 1.007069e-01    4.44
Expected.Goals..Non.Penalty..90   0 1.384310e-01    3.23
Goals.90                          0 1.252586e-01    2.65
Goals.90..Non.Penalty.            0 1.181552e-01    2.65
Assists.90                        0 9.162069e-02    1.70

Categorical Summary:
--------------------
Total Players: 580
No. of Countries represented: 66
No. of Clubs Represented: 20
Positions: ( DF),(MF),(FW),(MF,FW),(DF,MF),(GK),(MF,DF),(FW,MF),(DF,FW),(FW,DF )
Age Range: 15 - 38
```

- The `plot.playerStats()` method is then called for the `playerStats` class
- A radar chart is created for selected 4 players (`"Ollie Watkins", "Mohamed Salah", "Kevin De Bruyne", "Anthony Elanga"`), comparing their per 90 minute performance metrics to visualize how each player excels in different metrics.
- This approach provides a clearer and better understanding of comparisons between attacking and creative metrics for each player, offering valuable insights for club managers, scouts and other personnel to decide which player is a better fit for their club.

```r
plot(player_performance, c("Ollie Watkins", "Mohamed Salah",
                           "Kevin De Bruyne", "Anthony Elanga"))
```



- **Observations:**
  - **Kevin De Bruyne (Blue)** exhibits the best playmaking abilities per 90 minutes with the highest amount of Assists and Expected Assists (xA/90) among the four players.
  - **Anthony Elanga (Red)** has the lowest performance per 90 minutes across all metrics compared to others in the radar chart, indicating a lower impact compared to others.

- **Mohamed Salah (Green)** is the best overall finisher of the lot and having balanced metrics all around, excelling per 90 minutes in Goals, Expected Non-Penalty Goals (npxG) and Expected Goals (xG)
- **Ollie Watkins (Brown)** showed solid finishing capabilities amongst the others, with the highest Non-Penalty Goals scored per 90 minutes. He has lower contributions in creating chances, making him a reliable scorer with minimal creative output.

References

- Player Standard Stats 2023-2024 Premier League
- CRAN: fmsb package

```r
citation("fmsb")
```

```
To cite package 'fmsb' in publications use:

  Nakazawa M (2024). _fmsb: Functions for Medical Statistics Book with
  some Demographic Data_. R package version 0.7.6,
  <https://CRAN.R-project.org/package=fmsb>.

A BibTeX entry for LaTeX users is

  @Manual{,
    title = {fmsb: Functions for Medical Statistics Book with some Demographic Data},
    author = {Minato Nakazawa},
    year = {2024},
    note = {R package version 0.7.6},
    url = {https://CRAN.R-project.org/package=fmsb},
  }

ATTENTION: This citation information has been auto-generated from the
package DESCRIPTION file and may need manual editing, see
'help("citation")'.
```

- CRAN: summarytools package

```r
citation("summarytools")
```

```
To cite package 'summarytools' in publications use:

  Comtois D (2025). _summarytools: Tools to Quickly and Neatly
  Summarize Data_. R package version 1.1.3,
  <https://CRAN.R-project.org/package=summarytools>.

A BibTeX entry for LaTeX users is

  @Manual{,
    title = {summarytools: Tools to Quickly and Neatly Summarize Data},
    author = {Dominic Comtois},
    year = {2025},
    note = {R package version 1.1.3},
    url = {https://CRAN.R-project.org/package=summarytools},
  }
```