STAT30340/STAT40730 Data Programming with R- Final Project

Sudhanshu Naresh Nerkar 24226989

Part 1: Analysis

1.1 Dataset Description

This dataset contains detailed statistics of **Premier League players for the 2018–2019 season**. It provides information on players' demographics, playing time, performance metrics, and rankings, both overall and in specific contexts (home, away, etc.).

The dataset is ideal for analyzing player performance, team comparisons, and league trends.

Source: The dataset can be found on footystats.org.

1.1.1 Variable Categorization

Categorical Variables

- `full_name` - Name of the player.
- `league` - The league the player is part of (e.g., Premier League).
- `season` - Season of the statistics (e.g., 2018-2019).
- `position` - Player's position (e.g., defender, forward).
- `Current Club` - The club the player represents.
- `nationality` - Player's nationality.
- `rank_in_league_top_attackers` - Rank among top attackers in the league.
- `rank_in_league_top_midfielders` - Rank among top midfielders in the league.
- `rank_in_league_top_defenders` - Rank among top defenders in the league.
- `rank_in_club_top_scorer` - Rank as the club's top scorer.

Numerical Variables

- `age` - Age of the player.
- `minutes_played_overall` - Total minutes played.
- `minutes_played_home` - Minutes played in home matches.
- `minutes_played_away` - Minutes played in away matches.
- `appearances_overall` - Total appearances.
- `appearances_home` - Appearances in home matches.
- `appearances_away` - Appearances in away matches.
- `goals_overall` - Total goals scored.
- `goals_home` - Goals scored in home matches.
- `goals_away` - Goals scored in away matches.
- `assists_overall` - Total assists.
- `assists_home` - Assists in home matches.
- `assists_away` - Assists in away matches.
- `penalty_goals` - Goals scored from penalties.
- `penalty_misses` - Penalties missed.
- `clean_sheets_overall` - Total clean sheets.
- `clean_sheets_home` - Clean sheets in home matches.
- `clean_sheets_away` - Clean sheets in away matches.
- `conceded_overall` - Total goals conceded.
- `conceded_home` - Goals conceded in home matches.
- `conceded_away` - Goals conceded in away matches.
- `yellow_cards_overall` - Total yellow cards.
- `red_cards_overall` - Total red cards.
- `goals_involved_per_90_overall` - Goals contributed per 90 minutes.
- `assists_per_90_overall` - Assists per 90 minutes.
- `goals_per_90_overall` - Goals scored per 90 minutes.
- `goals_per_90_home` - Goals scored per 90 minutes in home matches.
- `goals_per_90_away` - Goals scored per 90 minutes in away matches.
- `min_per_goal_overall` - Minutes per goal scored.
- `conceded_per_90_overall` - Goals conceded per 90 minutes.
- `min_per_conceded_overall` - Minutes per goal conceded.
- `min_per_match` - Average minutes per match.
- `min_per_card_overall` - Minutes per card received.
- `min_per_assist_overall` - Minutes per assist.
- `cards_per_90_overall` - Cards received per 90 minutes.

1.2 Data Preparation

Starting by loading the dataset and examining its structure. You'll need to load the necessary libraries and read the data.

1.2.1 Loading the necessary libraries

```
# Loading necessary libraries
library(ggplot2)
library(readxl)
library(knitr)
library(tidyr)
library(RColorBrewer)
library(dplyr)
```

```
Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

1.2.2 Loading the data

```
# Loading the dataset
players_data <- read.csv("england-premier-league-players-2018-to-2019-stats.csv")
```

1.2.3 Handling missing values

```r
# Standardizing missing values (blank cells and "N/A") to NA
players_data <- players_data %>%
  mutate(across(everything(), ~ na_if(as.character(.x), "N/A")))

# Removing columns with any NA values
players_data <- players_data %>% select(where(~ all(!is.na(.))))

# Removing rows with any NA values
players_data <- players_data %>% drop_na()
```

1.2.4 Inspecting the data

```r
glimpse(players_data)    #Printing the Structure of the data
```

```
Rows: 570
Columns: 45
$ full_name                 <chr> "Aaron Cresswell", "Aaron Lennon", "Aar~
$ age                       <chr> "32", "35", "32", "31", "22", "24", "25~
$ league                    <chr> "Premier League", "Premier League", "Pr~
$ season                    <chr> "2018/2019", "2018/2019", "2018/2019", ~
$ position                  <chr> "Defender", "Midfielder", "Midfielder",~
$ Current.Club              <chr> "West Ham United", "Burnley", "Huddersf~
$ minutes_played_overall    <chr> "1589", "1217", "2327", "1327", "69", "~
$ minutes_played_home       <chr> "888", "487", "1190", "689", "14", "160~
$ minutes_played_away       <chr> "701", "730", "1137", "638", "55", "153~
$ nationality               <chr> "England", "England", "Australia", "Wal~
$ appearances_overall       <chr> "20", "16", "29", "28", "2", "35", "2",~
$ appearances_home          <chr> "11", "7", "15", "14", "1", "18", "0", ~
$ appearances_away          <chr> "9", "9", "14", "14", "1", "17", "2", "~
$ goals_overall             <chr> "0", "1", "3", "4", "0", "0", "0", "5",~
$ goals_home                <chr> "0", "1", "1", "2", "0", "0", "0", "3",~
$ goals_away                <chr> "0", "0", "2", "2", "0", "0", "0", "2",~
$ assists_overall           <chr> "1", "1", "1", "6", "0", "3", "0", "6",~
$ assists_home              <chr> "1", "1", "0", "5", "0", "1", "0", "2",~
$ assists_away              <chr> "0", "0", "1", "1", "0", "2", "0", "4",~
$ penalty_goals             <chr> "0", "0", "1", "0", "0", "0", "0", "0",~
$ penalty_misses            <chr> "0", "0", "0", "0", "0", "0", "0", "0",~
$ clean_sheets_overall      <chr> "3", "4", "4", "7", "0", "12", "0", "5"~
$ clean_sheets_home         <chr> "2", "2", "3", "6", "0", "7", "0", "3",~
$ clean_sheets_away         <chr> "1", "2", "1", "1", "0", "5", "0", "2",~
```

```
$ conceded_overall               <chr> "22", "20", "46", "12", "1", "41", "3",~
$ conceded_home                  <chr> "12", "8", "20", "2", "0", "17", "0", "~
$ conceded_away                  <chr> "10", "12", "26", "10", "1", "24", "3",~
$ yellow_cards_overall           <chr> "1", "1", "4", "0", "0", "5", "0", "7",~
$ red_cards_overall              <chr> "0", "0", "0", "0", "0", "1", "0", "0",~
$ goals_involved_per_90_overall  <chr> "0.06", "0.15", "0.15", "0.68", "0", "0~
$ assists_per_90_overall         <chr> "0.06", "0.07", "0.04", "0.41", "0", "0~
$ goals_per_90_overall           <chr> "0", "0.07", "0.12", "0.27", "0", "0", ~
$ goals_per_90_home              <chr> "0", "0.18", "0.08", "0.26", "0", "0", ~
$ goals_per_90_away              <chr> "0", "0", "0.16", "0.28", "0", "0", "0"~
$ min_per_goal_overall           <chr> "0", "1217", "776", "332", "0", "0", "0~
$ conceded_per_90_overall        <chr> "1.25", "1.48", "1.78", "0.81", "1.3", ~
$ min_per_conceded_overall       <chr> "72", "61", "51", "111", "69", "76", "1~
$ min_per_match                  <chr> "79", "76", "80", "47", "35", "90", "25~
$ min_per_card_overall           <chr> "1589", "1217", "582", "0", "0", "523",~
$ min_per_assist_overall         <chr> "1589", "1217", "2327", "221", "0", "10~
$ cards_per_90_overall           <chr> "0.06", "0.07", "0.15", "0", "0", "0.17~
$ rank_in_league_top_attackers   <chr> "290", "196", "144", "69", "-1", "312",~
$ rank_in_league_top_midfielders <chr> "191", "187", "233", "8", "-1", "160", ~
$ rank_in_league_top_defenders   <chr> "80", "-1", "-1", "-1", "-1", "-1", "-1~
$ rank_in_club_top_scorer        <chr> "20", "10", "3", "5", "31", "22", "22",~
```

1.2.5 Convert all of the numerical variables into numeric format

```r
# Converting all relevant numerical columns to numeric type
numerical_columns <- c(
  "age", "minutes_played_overall", "minutes_played_home",
  "minutes_played_away",
  "appearances_overall", "appearances_home", "appearances_away",
  "goals_overall", "goals_home", "goals_away",
  "assists_overall", "assists_home", "assists_away",
  "penalty_goals", "penalty_misses",
  "clean_sheets_overall", "clean_sheets_home", "clean_sheets_away",
  "conceded_overall", "conceded_home", "conceded_away",
  "yellow_cards_overall", "red_cards_overall",
  "goals_involved_per_90_overall", "assists_per_90_overall",
  "goals_per_90_overall",
  "goals_per_90_home", "goals_per_90_away", "min_per_goal_overall",
  "conceded_per_90_overall", "min_per_conceded_overall", "min_per_match",
  "min_per_card_overall", "min_per_assist_overall", "cards_per_90_overall"
)
```

```r
# Applying as.numeric to all selected columns
players_data[numerical_columns] <- players_data[numerical_columns] %>%
  lapply(as.numeric)
```

1.3 Exploratory Data Analysis (EDA)

1.3.1 Numerical Summaries

```r
#Numerical Summary for Key Varibales

numerical_summary <- players_data %>%
  summarize(
    Age_mean = mean(age, na.rm = TRUE),
    Age_median = median(age, na.rm = TRUE),
    Age_sd = sd(age, na.rm = TRUE),
    Age_q25 = quantile(age, probs = 0.25, na.rm = TRUE),
    Age_q75 = quantile(age, probs = 0.75, na.rm = TRUE),

    Minutes_Played_mean = mean(minutes_played_overall,
    na.rm = TRUE),
    Minutes_Played_median = median(minutes_played_overall,
    na.rm = TRUE),
    Minutes_Played_sd = sd(minutes_played_overall, na.rm = TRUE),
    Minutes_Played_q25 = quantile(minutes_played_overall, probs = 0.25,
    na.rm = TRUE),
    Minutes_Played_q75 = quantile(minutes_played_overall, probs = 0.75,
    na.rm = TRUE),

    Appearances_mean = mean(appearances_overall, na.rm = TRUE),
    Appearances_median = median(appearances_overall, na.rm = TRUE),
    Appearances_sd = sd(appearances_overall, na.rm = TRUE),
    Appearances_q25 = quantile(appearances_overall, probs = 0.25,
    na.rm = TRUE),
    Appearances_q75 = quantile(appearances_overall, probs = 0.75,
    na.rm = TRUE),

    Goals_mean = mean(goals_overall, na.rm = TRUE),
    Goals_median = median(goals_overall, na.rm = TRUE),
    Goals_sd = sd(goals_overall, na.rm = TRUE),
    Goals_q25 = quantile(goals_overall, probs = 0.25, na.rm = TRUE),
```

```
    Goals_q75 = quantile(goals_overall, probs = 0.75, na.rm = TRUE),

    Assists_mean = mean(assists_overall, na.rm = TRUE),
    Assists_median = median(assists_overall, na.rm = TRUE),
    Assists_sd = sd(assists_overall, na.rm = TRUE),
    Assists_q25 = quantile(assists_overall, probs = 0.25, na.rm = TRUE),
    Assists_q75 = quantile(assists_overall, probs = 0.75, na.rm = TRUE),

    Penalty_Goals_mean = mean(penalty_goals, na.rm = TRUE),
    Penalty_Goals_median = median(penalty_goals, na.rm = TRUE),
    Penalty_Goals_sd = sd(penalty_goals, na.rm = TRUE),
    Penalty_Goals_q25 = quantile(penalty_goals, probs = 0.25, na.rm = TRUE),
    Penalty_Goals_q75 = quantile(penalty_goals, probs = 0.75, na.rm = TRUE),

    Clean_Sheets_mean = mean(clean_sheets_overall, na.rm = TRUE),
    Clean_Sheets_median = median(clean_sheets_overall, na.rm = TRUE),
    Clean_Sheets_sd = sd(clean_sheets_overall, na.rm = TRUE),
    Clean_Sheets_q25 = quantile(clean_sheets_overall, probs = 0.25, na.rm = TRUE),
    Clean_Sheets_q75 = quantile(clean_sheets_overall, probs = 0.75, na.rm = TRUE)
  )

# Creating the output data frame

output_df <- data.frame(
  Metric = rep(c("Age", "Minutes Played", "Appearances", "Goals",
                 "Assists", "Penalty Goals", "Clean Sheets"), each = 1),
  Mean = c(numerical_summary$Age_mean,
           numerical_summary$Minutes_Played_mean,
           numerical_summary$Appearances_mean,
           numerical_summary$Goals_mean,
           numerical_summary$Assists_mean,
           numerical_summary$Penalty_Goals_mean,
           numerical_summary$Clean_Sheets_mean),
  Median = c(numerical_summary$Age_median,
             numerical_summary$Minutes_Played_median,
             numerical_summary$Appearances_median,
             numerical_summary$Goals_median,
             numerical_summary$Assists_media,
             numerical_summary$Penalty_Goals_median,
             numerical_summary$Clean_Sheets_median),
  SD = c(numerical_summary$Age_sd,
         numerical_summary$Minutes_Played_sd,
```

```
          numerical_summary$Appearances_sd,
          numerical_summary$Goals_sd,
          numerical_summary$Assists_sd,
          numerical_summary$Penalty_Goals_sd,
          numerical_summary$Clean_Sheets_sd),

  Q25 = c(numerical_summary$Age_q25,
          numerical_summary$Minutes_Played_q25,
          numerical_summary$Appearances_q25,
          numerical_summary$Goals_q25,
          numerical_summary$Assists_q25,
          numerical_summary$Penalty_Goals_q25,
          numerical_summary$Clean_Sheets_q25),

  Q75 = c(numerical_summary$Age_q75,
          numerical_summary$Minutes_Played_q75,
          numerical_summary$Appearances_q75,
          numerical_summary$Goals_q75,
          numerical_summary$Assists_q75,
          numerical_summary$Penalty_Goals_q75,
          numerical_summary$Clean_Sheets_q75)
)

# Creating the table with kable
output_df %>%
  kable(
    col.names = c("Metric", "Mean", "Median", "SD", "25th%", "75th%"),
    caption = "Numerical Summary of Key Player Metrics",
    align = "r",
    digits = 2
  )
```

Table 1: Numerical Summary of Key Player Metrics

Metric	Mean	Median	SD	25th%	75th%
Age	29.74	30.0	4.39	26.00	33.0
Minutes Played	1317.58	1115.5	1096.27	212.25	2185.5
Appearances	18.38	19.0	12.86	6.00	30.0
Goals	1.82	0.0	3.48	0.00	2.0
Assists	1.30	0.0	2.18	0.00	2.0
Penalty Goals	0.15	0.0	0.75	0.00	0.0

Metric	Mean	Median	SD	25th%	75th%
Clean Sheets	4.99	4.0	4.66	1.00	7.0

Interpretation:

In summary, the data reveals the following key insights:

- **Right-Skewed Distributions:**

  Most metrics (Minutes Played, Goals, Assists, Penalty Goals, Clean Sheets) exhibit right-skewed distributions, indicating that a few players have significantly higher values than the rest.

- **Variability:**

  High variability is observed in Minutes Played, Appearances, Goals, Assists, and Clean Sheets, suggesting a wide range of player performance.

- **Age and Appearances:**

  Age and Appearances show more symmetrical distributions, indicating a more even spread of values.

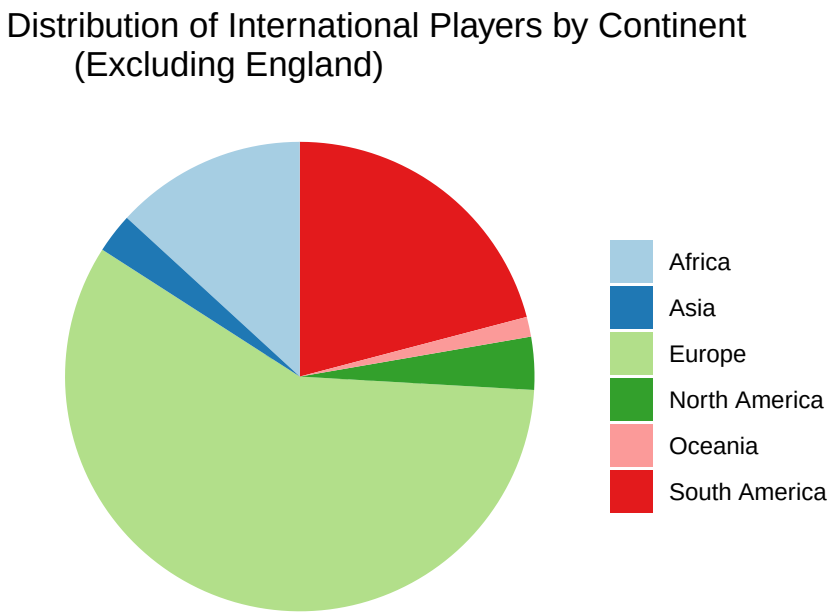
These insights highlight the importance of considering both central tendency (mean, median) and variability (SD, IQR) when analyzing player performance data.

1.3.2 Graphical Summaries

(i) Distribution of International Players by Continent (Excluding England):

```
# Creating a complete mapping from nationality to continent (expand as needed)
continent_mapping <- c(
"England" = "Europe", "France" = "Europe", "Germany" = "Europe",
"Italy" = "Europe", "Spain" = "Europe",
"Brazil" = "South America", "Argentina" = "South America",
"Colombia" = "South America", "Chile" = "South America",
"Nigeria" = "Africa", "Senegal" = "Africa", "Egypt" = "Africa",
"Cameroon" = "Africa", "Morocco" = "Africa",
"Japan" = "Asia", "South Korea" = "Asia", "China" = "Asia",
"Iran" = "Asia", "Australia" = "Oceania",
"USA" = "North America", "Mexico" = "North America",
"Canada" = "North America",
"Belgium" = "Europe", "Portugal" = "Europe", "Holland" = "Europe",
"Poland" = "Europe","Peru" = "South America",
```

```
"Uruguay" = "South America", "Venezuela" = "South America",
"Ghana" = "Africa", "South Africa" = "Africa",
"Ivory Coast" = "Africa","Tunisia" = "Africa",
"India" = "Asia", "Saudi Arabia" = "Asia", "Qatar" = "Asia",
"United Arab Emirates" = "Asia","New Zealand" = "Oceania",
"Fiji" = "Oceania"
)

# Adding a continent column to the dataset
players_data <- players_data %>%
  mutate(continent = continent_mapping[nationality])
# Assign continent based on nationality mapping



# Filtering out English players and group by continent
international_players <- players_data %>%
  filter(nationality != "England" & !is.na(continent)) %>%
  count(continent) %>%
  arrange(desc(n))

# Generating a color palette for all continents using "Paired" palette
colors <- brewer.pal(length(unique(international_players$continent)),"Paired")

# Plotting pie chart with custom color palette
ggplot(international_players, aes(x = "", y = n, fill = continent)) +
  geom_bar(stat = "identity", width = 1) +
  coord_polar(theta = "y") +
  scale_fill_manual(values = colors) +  # Apply the custom color palette
  labs(title = "Distribution of International Players by Continent
       (Excluding England)",
       x = "", y = "") +
  theme_void() +
  theme(legend.title = element_blank())
```



Interpretation of Pie Chart:

The pie chart titled **"Distribution of International Players by Continent (Excluding England)"** shows the breakdown of international players across different continents, excluding players from England.

Here's what we can interpret from the chart:

Dominance of Europe:

- The largest slice of the pie is occupied by **Europe**, indicating that a significant portion of international players come from this continent.

Other Notable Continents:

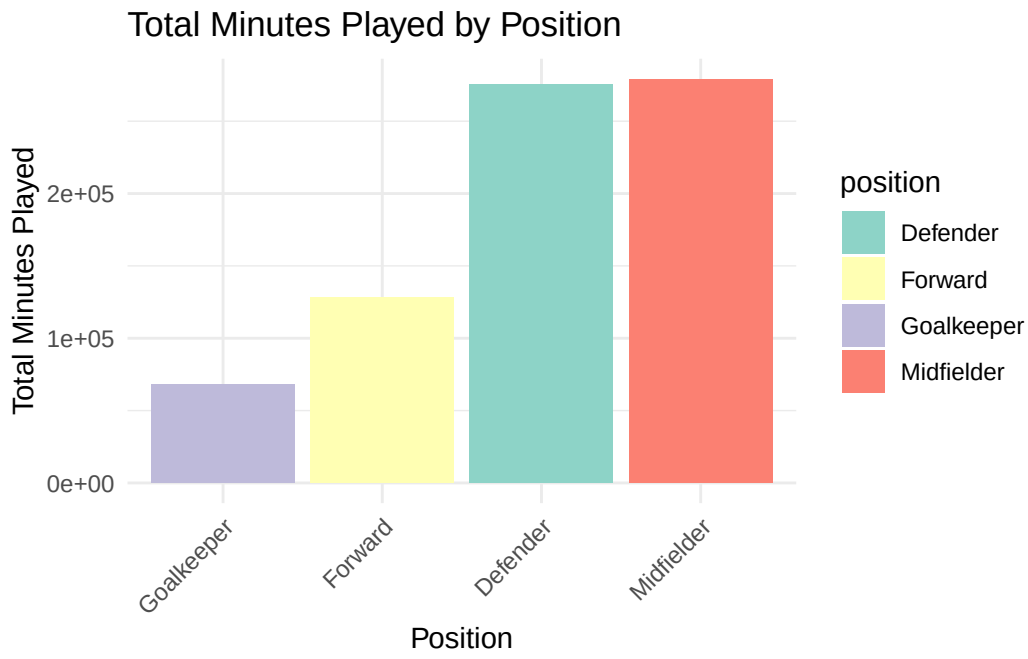
- **South America** has the second-largest share, suggesting a strong presence of players from this region.
- **Africa** and **Asia** have noticeable segments, indicating a growing number of players from these continents.
- **North America** and **Oceania** have smaller slices, suggesting fewer players from these regions.

Overall:

The chart highlights the global nature of international players, with a strong focus on European and South American players. It also suggests an increasing representation of players from Africa and Asia in the international scene.

(ii) Minutes played by position (Bar Plot)

```
# Summarizing total minutes played by position
minutes_by_position <- players_data %>%
  group_by(position) %>%
  summarise(total_minutes = sum(minutes_played_overall, na.rm = TRUE)) %>%
  arrange(desc(total_minutes))  # Sort by total minutes played

# Creating a bar plot to visualize the total minutes played by position
ggplot(minutes_by_position, aes(x = reorder(position, total_minutes),
y = total_minutes, fill = position)) +
  geom_bar(stat = "identity") +
  labs(title = "Total Minutes Played by Position",
       x = "Position",
       y = "Total Minutes Played") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  # Rotating x-axis labels for better readability
  scale_fill_brewer(palette = "Set3")
```



Interpretation:

Distribution of Playing Time:

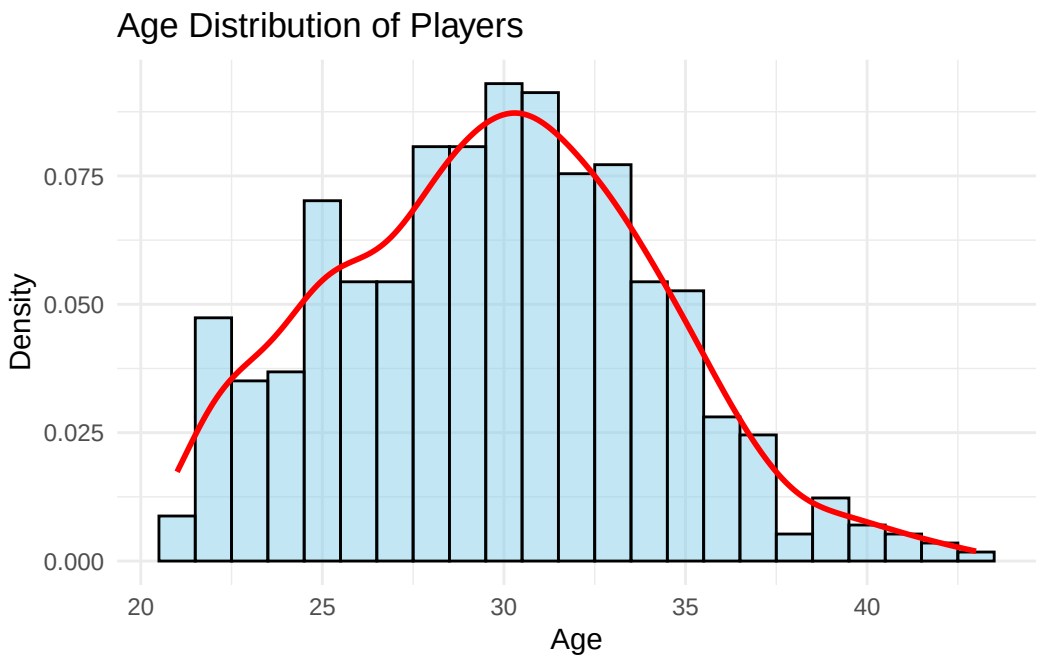
- **Midfielders** have the highest total minutes played, indicating that they are heavily relied upon for playing time.
- **Defenders** have the second-highest minutes played, suggesting a significant role in the team's strategy.
- **Forwards** have a moderate amount of playing time.
- **Goalkeepers** have the lowest total minutes played, which is expected as they are typically substituted less frequently.

Overall:

The chart highlights that the team's playing time is heavily concentrated on midfielders and defenders, with goalkeepers playing the least. This suggests a tactical approach that prioritizes possession and defensive stability.

(iii) Age distribution of players (Histogram)

```
# Converting age to numeric if it's not already
players_data$age <- as.numeric(players_data$age)

# Plotting the distribution of player ages with a density line
ggplot(players_data, aes(x = age)) +
  geom_histogram(aes(y = after_stat(density)), binwidth = 1, fill = "skyblue",
                 color = "black", alpha = 0.5) +
  geom_density(color = "red", linewidth = 1) +  # Add a density curve (line)
  labs(title = "Age Distribution of Players", x = "Age", y = "Density") +
  theme_minimal()
```



Interpretation:

Distribution of Ages:

- The distribution is **skewed to the right**, meaning there are more players in the younger age groups (20s and early 30s) compared to the older age groups.
- The peak of the distribution lies between 25 and 30, indicating that this age range has the highest concentration of players.
- There is a gradual decline in the number of players as the age increases, suggesting that fewer older players are present in the premier league.

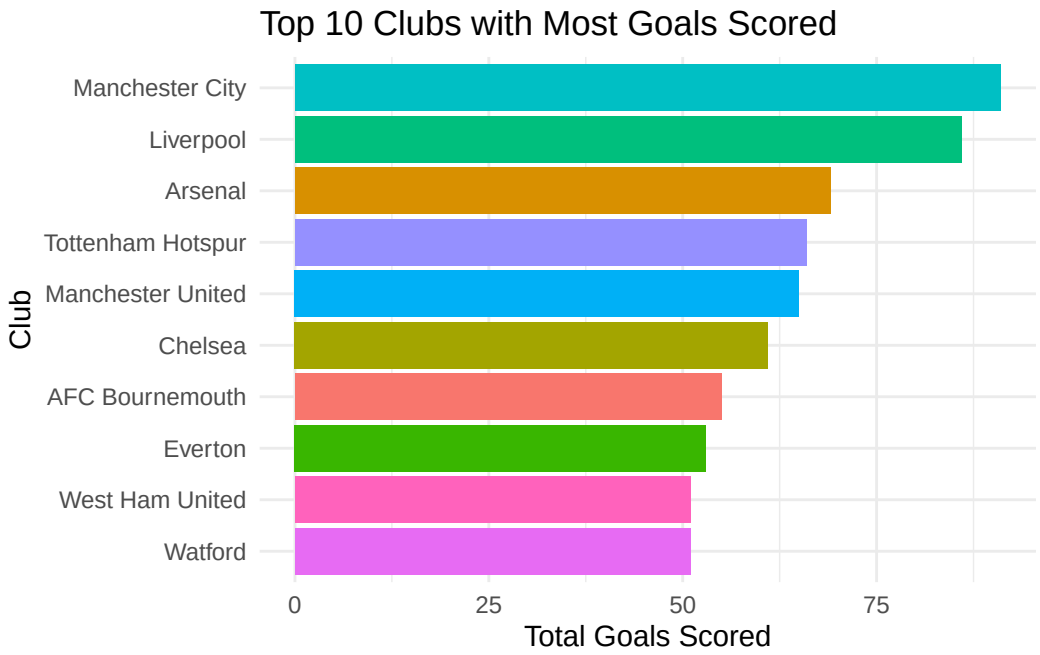
Overall:

The chart highlights that the premier league has a younger player base, with a majority of players in their late 20s and early 30s. This suggests a focus on youth and potentially a strategy for long-term development.

(iv) Top Scoring Clubs (Bar chart)

```
# Summarizing the total goals scored by each club
goals_by_club <- players_data %>%
  group_by(`Current.Club`) %>%
  summarise(total_goals = sum(goals_overall, na.rm = TRUE)) %>%
  arrange(desc(total_goals))
```

```r
# Selecting top clubs (e.g., top 10) for better visualization
top_clubs <- goals_by_club %>%
  head(10)

# Plotting the bar chart
ggplot(top_clubs, aes(x = reorder(`Current.Club`, total_goals),
  y = total_goals, fill = `Current.Club`)) +
  geom_bar(stat = "identity") +
  coord_flip() +  # Flip the coordinates to make it horizontal
  labs(title = "Top 10 Clubs with Most Goals Scored",
       x = "Club", y = "Total Goals Scored") +
  theme_minimal() +
  theme(legend.position = "none")  # Remove legend since it's not needed here
```



Interpretation:

Top-Scoring Clubs:

- **Manchester City** leads the chart with the highest number of goals scored.
- **Liverpool** and **Arsenal** follow closely behind, occupying the second and third positions, respectively.

Other Notable Clubs:

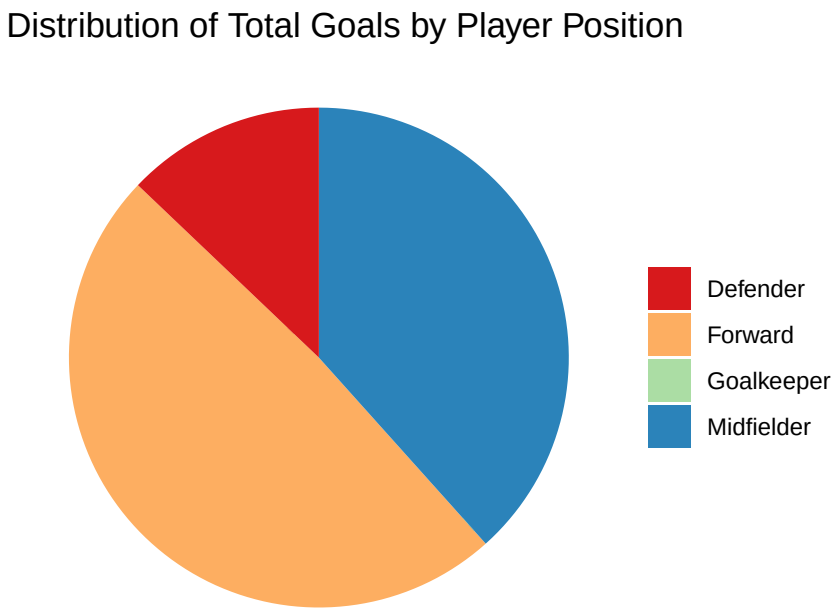
- **Tottenham Hotspur**, **Manchester United**, and **Chelsea** also have a significant number of goals scored, placing them in the top 5.
- Clubs like **AFC Bournemouth**, **Everton**, **West Ham United**, and **Watford** have scored fewer goals compared to the top teams.

Overall:

The chart highlights the goal-scoring prowess of Manchester City and Liverpool, with Arsenal also showcasing strong attacking capabilities. The other clubs in the top 10 have also contributed significantly to the overall goals scored in the league.

(v) Distribution of Total Goals by Player Position

```
# Summarizing total goals by position
position_goals <- players_data %>%
  group_by(position) %>%
  summarise(total_goals = sum(goals_overall, na.rm = TRUE)) %>%
  arrange(desc(total_goals))

# Plotting pie chart of total goals by position
ggplot(position_goals, aes(x = "", y = total_goals, fill = position)) +
  geom_bar(stat = "identity", width = 1) +
  coord_polar(theta = "y") +
  labs(title = "Distribution of Total Goals by Player Position", x = "", y = "") +
  theme_void() +
  theme(legend.title = element_blank()) +
  scale_fill_brewer(palette = "Spectral")
```



Interpretation:

The pie chart titled "Distribution of Total Goals by Player Position" shows the proportion of goals scored by players in different positions.

Here's what we can interpret from the chart:

Goal-Scoring Distribution:

- **Midfielders** have scored the highest proportion of goals, with a significant portion of the pie chart dedicated to them.
- **Forwards** have the second-highest proportion of goals, indicating their key role in scoring.
- **Defenders** and **Goalkeepers** have scored a much smaller proportion of goals, as expected.

Overall:

The chart highlights the significant contribution of midfielders and forwards to the team's goal-scoring tally. Defenders and goalkeepers, while crucial for other aspects of the game, have a limited impact on goal-scoring.

1.3.3 Tabular Summaries

(i) Top 10 Most Impactful Players (based on the sum of goals and assists)

```
# Ensuring that goals_overall and assists_overall are numeric
players_data$goals_overall <- as.numeric(players_data$goals_overall)
players_data$assists_overall <- as.numeric(players_data$assists_overall)

# Calculating total impact (goals + assists) for each player and select top 10 players
top_impactful_players <- players_data %>%
  mutate(total_impact = goals_overall + assists_overall) %>%
  # Calculate total impact
  arrange(desc(total_impact)) %>%  # Sort by total impact in descending order
  select(full_name, `Current.Club`, position, goals_overall,
  assists_overall, total_impact) %>%  # Select required columns
  head(10)  # Get the top 10 players

# Printing the top 10 impactful players as a formatted table using knitr's kable()
kable(top_impactful_players,
      col.names = c("Player Name", "Club", "Position", "Goals",
                    "Assists", "Goals + Assists"),
      caption = "Top 10 Most Impactful Players (Goals + Assists)")
```

Table 2: Top 10 Most Impactful Players (Goals + Assists)

Player Name	Club	Position	Goals	Assists	Goals + Assists
Eden Hazard	Chelsea	Midfielder	16	15	31
Mohamed Salah	Liverpool	Forward	22	8	30
Sergio Aguero	Manchester City	Forward	21	8	29
Pierre-Emerick Aubameyang	Arsenal	Forward	22	5	27
Raheem Sterling	Manchester City	Forward	17	10	27
Callum Wilson	AFC Bournemouth	Forward	14	9	23
Sadio Mané	Liverpool	Forward	22	1	23
Jamie Vardy	Leicester City	Forward	18	4	22
Paul Pogba	Manchester United	Midfielder	13	9	22
Alexandre Lacazette	Arsenal	Forward	13	8	21

(ii) Top 5 most fit players based on minutes played

```r
# Filtering out goalkeepers and calculate the top 5 outfield players based on minutes played
top_fit_players <- players_data %>%
  filter(position != "Goalkeeper") %>%  # Exclude goalkeepers
  arrange(desc(minutes_played_overall)) %>%
  # Sort by minutes played in descending order
  select(full_name, `Current.Club`, position, minutes_played_overall) %>%
  # Select relevant columns
  head(5)  # Get the top 5 players

# Printing the top 5 most fit players as a formatted table using knitr's kable()
kable(top_fit_players,
      col.names = c("Player Name", "Club", "Position", "Minutes Played"),
      caption = "Top 5 Most Fit Outfield Players (Based on Minutes Played)")
```

Table 3: Top 5 Most Fit Outfield Players (Based on Minutes Played)

Player Name	Club	Position	Minutes Played
Ben Mee	Burnley	Defender	3420
Conor Coady	Wolverhampton Wanderers	Midfielder	3420
Luka Milivojević	Crystal Palace	Midfielder	3420
Nathan Aké	AFC Bournemouth	Defender	3412
Cesar Azpilicueta	Chelsea	Defender	3403

(iii) Top 5 Most Indisciplined Clubs Based on Cards (Yellow + Red Cards)

```r
# Ensuring that yellow_cards_overall and red_cards_overall are numeric
players_data$yellow_cards_overall <- as.numeric(players_data$yellow_cards_overall)
players_data$red_cards_overall <- as.numeric(players_data$red_cards_overall)

# Calculating the total indiscipline (yellow cards + red cards) for each club
indisciplined_clubs <- players_data %>%
  group_by(`Current.Club`) %>%  # Group by club
  summarise(total_yellow_cards = sum(yellow_cards_overall, na.rm = TRUE),
# Sum yellow cards
            total_red_cards = sum(red_cards_overall, na.rm = TRUE),
# Sum red cards
            total_indiscipline = total_yellow_cards + total_red_cards) %>%
# Calculating total indiscipline
  arrange(desc(total_indiscipline)) %>%
  # Sorting by total indiscipline in descending order
```

```
  select(`Current.Club`, total_yellow_cards, total_red_cards, total_indiscipline)

# Selecting relevant columns

# Printing the most indisciplined clubs as a formatted table using knitr's kable()
kable(head(indisciplined_clubs),
      col.names = c("Club", "Total Yellow Cards", "Total Red Cards", "Total Cards"),
      caption = "Most Indisciplined Clubs (Yellow + Red Cards)")
```

Table 4: Most Indisciplined Clubs (Yellow + Red Cards)

Club	Total Yellow Cards	Total Red Cards	Total Cards
Watford	82	4	86
Manchester United	79	4	83
Southampton	76	3	79
Arsenal	75	2	77
Burnley	76	1	77
Fulham	72	2	74

(iv) Best Goalkeepers based on Number of Clean Sheets

```
# Filtering out goalkeepers and sort by clean sheets
top_clean_sheet_goalkeepers <- players_data %>%
  filter(position == "Goalkeeper") %>%  # Filter for goalkeepers only
  arrange(desc(clean_sheets_overall)) %>%
  # Sorting by total clean sheets in descending order
  select(full_name, `Current.Club`, clean_sheets_overall)
  # Selecting relevant columns

# Printing the top goalkeepers with most clean sheet using knitr's kable()
kable(head(top_clean_sheet_goalkeepers),
      col.names = c("Player Name", "Club", "Clean Sheets"),
      caption = "Goalkeepers with Most Clean Sheets")
```

Table 5: Goalkeepers with Most Clean Sheets

Player Name	Club	Clean Sheets
Alisson Becker	Liverpool	21
Ederson	Manchester City	20

Player Name	Club	Clean Sheets
Jordan Pickford	Everton	14
Kepa Arrizabalaga	Chelsea	14
Hugo Lloris	Tottenham Hotspur	12
Martin Dúbravka	Newcastle United	11

(v) Top 5 Youngest and Oldest Players in the Premier League

```
# Getting the top 5 youngest players
top_5_youngest_players <- players_data %>%
  arrange(age) %>%  # Sort by age in ascending order (youngest first)
  select(full_name, `Current.Club`, position, age) %>%  # Select relevant columns
  head(5)  # Get the top 5 youngest players

# Getting the top 5 oldest players
top_5_oldest_players <- players_data %>%
  arrange(desc(age)) %>%  # Sort by age in descending order (oldest first)
  select(full_name, `Current.Club`, position, age) %>%  # Select relevant columns
  head(5)  # Get the top 5 oldest players

# Printing the top 5 youngest players as a formatted table using knitr's kable()
kable(top_5_youngest_players,
      col.names = c("Player Name", "Club", "Position", "Age"),
      caption = "Top 5 Youngest Players")
```

Table 6: Top 5 Youngest Players

Player Name	Club	Position	Age
Callum Hudson-Odoi	Chelsea	Midfielder	21
Curtis Jones	Liverpool	Midfielder	21
James Garner	Manchester United	Defender	21
Mason Greenwood	Manchester United	Midfielder	21
Matty Daly	Huddersfield Town	Midfielder	21

```
# Printing the top 5 oldest players as a formatted table using knitr's kable()
kable(top_5_oldest_players,
      col.names = c("Player Name", "Club", "Position", "Age"),
      caption = "Top 5 Oldest Players")
```

Table 7: Top 5 Oldest Players

Player Name	Club	Position	Age
Julián Speroni	Crystal Palace	Goalkeeper	43
Artur Boruc	AFC Bournemouth	Goalkeeper	42
Bruno Saltor Grau	Brighton & Hove Albion	Defender	42
Heurelho da Silva Gomes	Watford	Goalkeeper	41
Peter Crouch	Burnley	Forward	41

(vi) Highest Penalty Conversion Rate (Minimum 5 Attempts)

```
# Filtering players with at least 5 penalty attempts
penalty_data <- players_data %>%
  filter(penalty_goals + penalty_misses >= 5) %>%
  mutate(penalty_conversion_rate = penalty_goals / (penalty_goals +
                                                      penalty_misses)) %>%
  arrange(desc(penalty_conversion_rate)) %>%
  select(full_name, `Current.Club`, penalty_goals,
         penalty_misses, penalty_conversion_rate)

# Displaying the top penalty converters in a formatted table
penalty_data %>%
  head(10) %>%
  mutate(penalty_conversion_rate = scales::percent(penalty_conversion_rate)) %>%
  kable(
    caption = "Top Players with Highest Penalty Conversion Rate (Minimum 5 Attempts)",
    col.names = c("Player Name", "Club", "Penalty Goals",
                  "Penalty Misses", "Conversion Rate"),
    align = "lccc"
  )
```

Table 8: Top Players with Highest Penalty Conversion Rate (Minimum 5 Attempts)

Player Name	Club	Penalty Goals	Penalty Misses	Conversion Rate
Luka Milivojević	Crystal Palace	10	1	90.9%
Joshua King	AFC Bournemouth	5	1	83.3%
Jamie Vardy	Leicester City	4	1	80.0%
Pierre-Emerick Aubameyang	Arsenal	4	1	80.0%

Player Name	Club	Penalty Goals	Penalty Misses	Conversion Rate
Paul Pogba	Manchester United	7	3	70.0%
Gylfi Sigurdsson	Everton	2	3	40.0%

(vii) Top 5 Impactful Young players based on Goals+Assists (Under the Age 23)

```
# Creating a table with top 5 impactful young players (U23)
top_young_players_kable <- players_data %>%
  filter(age < 23) %>%  # Filter for young players under 23
  select(full_name, age, Current.Club, goals_overall, assists_overall) %>%
  mutate(GA = goals_overall + assists_overall) %>%
  rename(Name = full_name, Age = age, Club = Current.Club,
  Goals = goals_overall, Assists = assists_overall, `Goals+Assists` = GA) %>%
  arrange(desc(`Goals+Assists`)) %>%  # Sort by descending G+A
  head(5) %>%  # Select top 5 players
  kable(caption = "Top 5 Impactful Young Players (U23) - Goals+Assists")

# Printing the table
top_young_players_kable
```

Table 9: Top 5 Impactful Young Players (U23) - Goals+Assists

Name	Age	Club	Goals	Assists	Goals+Assists
Dwight McNeil	22	Burnley	3	5	8
Ryan Sessegnon	22	Fulham	2	6	8
Michael Obafemi	22	Southampton	1	1	2
Callum Hudson-Odoi	21	Chelsea	0	1	1
Domingos Quina	22	Watford	1	0	1

1.4: Conclusion

The analysis provided comprehensive insights into **player performance**, positional importance, and team trends in the Premier League. Europe was found to dominate in **player representation**, with South America following closely. Midfielders played the most minutes, highlighting their importance, while players aged 25–30 made up the majority, emphasizing the league's focus on **younger talent**. Eden Hazard stood out as the top performer in **combined goals and assists**, while Watford emerged as the most **indisciplined club** with the highest total cards.

Part 2: R Package (DataExplorer)

2.1: Purpose of DataExplore Package

The **DataExplorer** package in R is a comprehensive tool designed to automate and simplify the process of **exploratory data analysis (EDA)**, which is a critical first step in any data science or statistical project. The package helps users quickly understand the structure, content, and relationships within their datasets, reducing the time and effort required for manual exploration. Its primary goal is to make EDA more efficient, systematic, and accessible, even for users with limited programming experience.

Key Features and Purpose:

1. **Automated EDA Reports**:
    - The package allows users to generate a complete EDA report with a single command using the `create_report()` function. This report includes detailed information about the dataset's structure, missing values, distributions, correlations, and outliers.
    - This functionality is particularly valuable for quickly summarizing large and complex datasets.
2. **Visualization of Data**:
    - The package includes functions to visualize missing data patterns (`plot_missing()`), data distributions (`plot_histogram()`), categorical frequencies (`plot_bar()`), and correlations between numerical variables (`plot_correlation()`).
    - These visualizations are customizable and provide clear insights into the underlying patterns of the dataset.
3. **Data Profiling**:
    - The package provides detailed profiling of both numerical and categorical variables. This includes summaries like mean, median, and standard deviation for numerical variables, as well as frequency distributions and unique counts for categorical variables.
    - Users can also profile individual variables to gain a deeper understanding of specific features in the dataset.
4. **Streamlining Data Cleaning**:
    - DataExplorer helps identify missing data, highly correlated variables, and outliers, allowing users to address these issues before proceeding with further analysis or modeling.

5. **Ease of Use**:
    - Its functions are user-friendly and require minimal coding, making it accessible to analysts and data scientists at all skill levels. By automating repetitive tasks, it helps save time and ensures consistency in analysis.
6. **Customizable Outputs**:
    - While the package is designed for automation, it also provides flexibility. Users can specify the type of outputs they want or customize visualizations according to their needs.

When to Use DataExplorer:

- **Initial Data Exploration**: Quickly understand the dataset when starting a project, particularly with large or unfamiliar datasets.
- **Quality Assessment**: Identify issues such as missing values, imbalanced categories, or highly correlated variables that may impact further analysis.
- **Communication**: Generate automated reports to share insights with collaborators or stakeholders in a clear and concise format.

Summary:

The **DataExplorer** package is an invaluable tool for automating and standardizing EDA workflows. By enabling users to quickly assess and visualize their data, it supports better decision-making in the early stages of data analysis. Its ability to generate comprehensive reports, coupled with its visual and numerical summaries, makes it a powerful resource for both beginners and experienced data analysts. The package not only accelerates the exploration phase but also ensures that critical insights and potential data quality issues are not overlooked, making it a cornerstone for effective data analysis workflows.

2.2: Functionality of Main Functions

DataExplorer package to explore the Premier League dataset. I'll demonstrate three key functions of the package on the Premier League Players dataset:

1. `create_report()`: To generate a comprehensive EDA report.
2. `plot_bar()`: To create bar plots for categorical variables.
3. `plot_density()`: To visualize density of numerical variables.

2.2.1: Generating an Automated EDA Report

The `create_report()` function generates a comprehensive report for the dataset, including summaries, visualizations, and key insights.

```
# Loading DataExplorer library
library(DataExplorer)

# Generating an automated EDA report
create_report(players_data)
```

Explanation:

- This function creates a self-contained HTML report, which includes information about data structure, missing values, variable distributions, correlations, and other EDA insights.
- The generated report can be opened in any browser for further exploration.

Output:

We will get an HTML file with visualizations like histograms, bar charts, missing data heatmaps, and correlation matrices, giving a complete overview of the dataset.

Here are few screenshots of the report generated:

Data Profiling Report

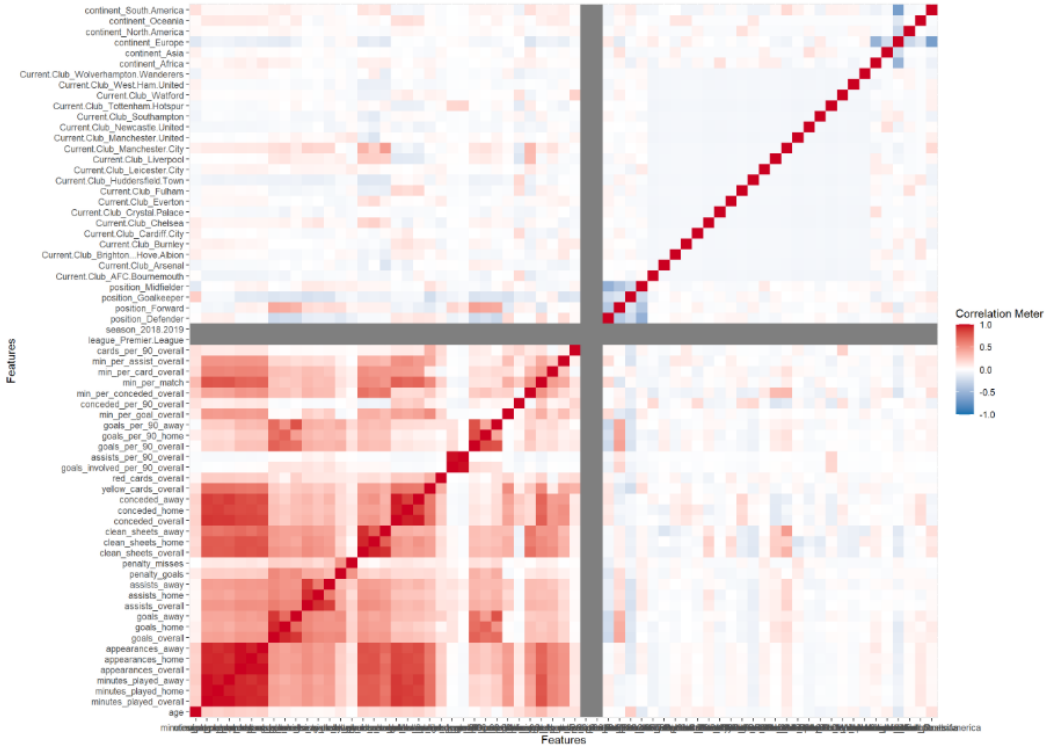
Basic Statistics

Raw Counts

Name	Value
Rows	570
Columns	46
Discrete columns	11
Continuous columns	35
All missing columns	0
Missing observations	135
Complete Rows	435
Total observations	26,220
Memory allocation	316.5 Kb

Correlation Analysis

```
## 6 features with more than 20 categories ignored!
## full_name: 435 categories
## nationality: 29 categories
## rank_in_league_top_attackers: 314 categories
## rank_in_league_top_midfielders: 314 categories
## rank_in_league_top_defenders: 129 categories
## rank_in_club_top_scorer: 32 categories
```

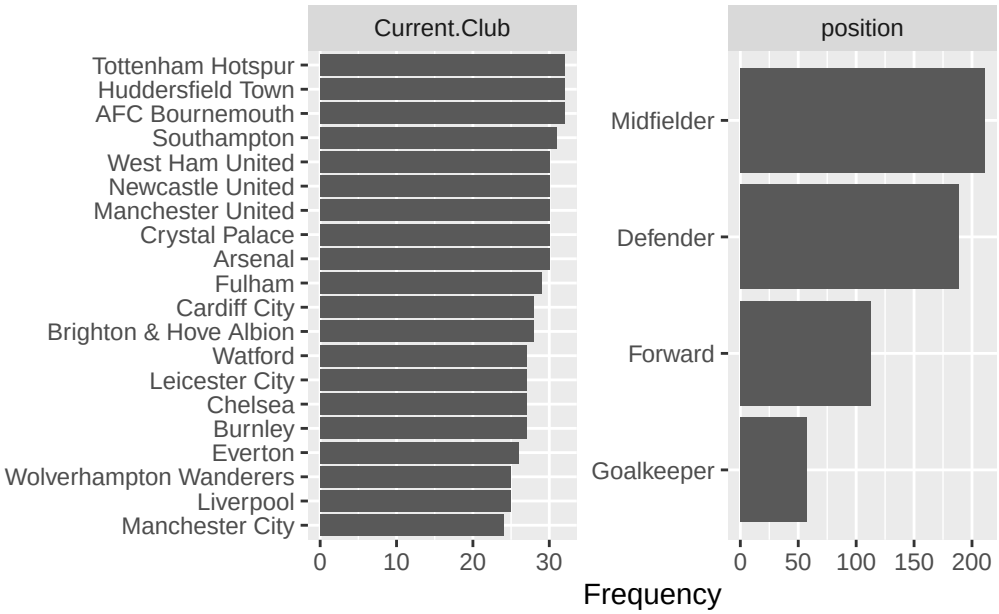


Here is the PDF version of full HTML Report File generated : bit.ly/DataExplorerReport

2.2.3: Bar Plot for Categorical Variables

The `plot_bar()` function creates bar plots for categorical variables, showing the frequency of each category.

```
# Creating a new data frame with few categorical variables
categorical_vars <- players_data %>% select('Current.Club', 'position')

# Plotting bar charts for categorical variables
plot_bar(categorical_vars)
```



Explanation:

- `plot_bar()` will create separate bar charts for each categorical variable in the `categorical_vars` data frame.
- This helps visualize the frequency of each category within each variable (e.g., how many players belong to each team, or the number of players in each position).

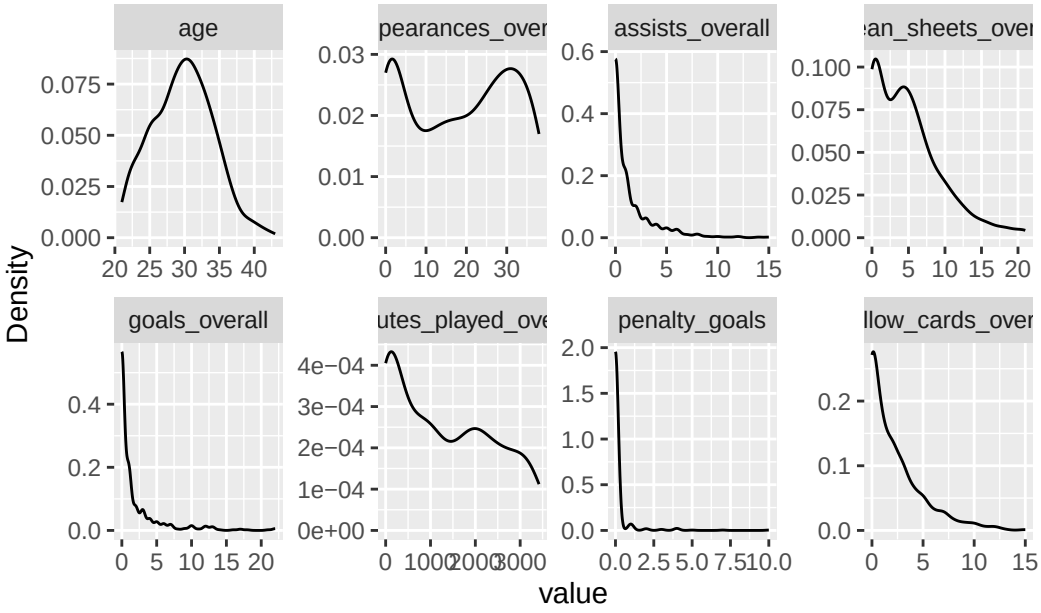
Output:

- A bar plot showing the distribution of players across teams (e.g., how many players are in `Arsenal`, `Liverpool`, etc.).
- Another bar plot showing the distribution of player positions (e.g., Forward, Midfielder, Defender, etc.).

2.2.3: Density Plot for Numerical Variables

```
# Creating a new data frame with few numerical variables
numerical_vars <- players_data %>% select(goals_overall, appearances_overall,
assists_overall,age,minutes_played_overall,yellow_cards_overall,
penalty_goals,clean_sheets_overall)
```

```r
# Visualizing the density of numerical variables
plot_density(numerical_vars)
```



Explanation:

- `plot_density()` will generate smooth density plots for each of the selected numerical variables, helping you visualize the distribution (whether the data is normal, skewed, etc.).
- This is useful for identifying patterns, such as whether most players scored only a few goals or played a similar number of matches.

Output:

The density plot shows the distribution of various numerical variables:

1. **`age`**: Most players are aged 25–30, typical for professional footballers.
2. **`appearances_overall`**: Most players have 0–10 appearances, with fewer having many appearances.
3. **`assists_overall`**: The majority have 0 assists, with only a few players having more.
4. **`clean_sheets_overall`**: Most players have 0 clean sheets, as this is mainly relevant for goalkeepers and defenders.

5. `goals_overall`: Most players score few or no goals, with only a few scoring many.
6. `minutes_played_overall`: Most players play fewer minutes, with only a few playing significant time.
7. `penalty_goals`: Only a small number of players score penalty goals.
8. `yellow_cards_overall`: Most players have few yellow cards, with a few accumulating many.

Overall, the data is right-skewed, with most players having low values in these categories, and a few standout performers.

2.3: Conclusion

The `DataExplorer` package greatly streamlined the **exploratory data analysis** process, automating key tasks like generating reports and visualizing trends. It provided a holistic view of the dataset by identifying **missing values**, correlations, and distributions with minimal manual effort. By automating repetitive tasks, the package enhanced **efficiency and clarity**, making it a powerful tool for data profiling.

Part 3: Functions/Programming

3.1: Writing the Analysis Function

This function should:

1. Accept the dataset and a player's name as input.
2. Extract data for the selected player.
3. Perform calculations like total goals, total assists, goals per 90 minutes, etc.
4. Return results in a structured format (S3 class object).

```
# Defining a function to analyze the performance of a specific player
player_analysis <- function(players_data, player_name) {
  # Checking if the player's name exists in the dataset
  # If not found, stop execution and return an error message
  if (!player_name %in% players_data$full_name) {
    stop("Player not found in the dataset.")
  }

  # Filtering the dataset to only include data for the specified player
```

```
  players_data <- players_data[players_data$full_name == player_name, ]

  # Extracting and calculate performance metrics for the player
  total_goals <- players_data$goals_overall     # Total goals
  total_assists <- players_data$assists_overall # Total assists
  minutes_played <- players_data$minutes_played_overall  # Total minutes played
  goals_per_90 <- players_data$goals_per_90_overall # Goals scored per 90min
  assists_per_90 <- players_data$assists_per_90_overall # Assists provided per 90min

  # Storing all calculated metrics in a list
  result <- list(
    player_name = player_name,       # The name of the player being analyzed
    total_goals = total_goals,       # Total goals scored
    total_assists = total_assists,   # Total assists provided
    minutes_played = minutes_played, # Total minutes played
    goals_per_90 = goals_per_90,     # Goals scored per 90 minutes
    assists_per_90 = assists_per_90  # Assists provided per 90 minutes
  )

  # Assigning the S3 class "player_analysis" to the result object
  class(result) <- "player_analysis"

  # Returning the result object
  return(result)
}
```

Function Structure: `player_analysis`

The `player_analysis` function analyzes a player's performance based on a dataset of player statistics. Here's a structured breakdown:

Inputs:

- **players_data**: A dataset containing player performance statistics (e.g., goals, assists, minutes played).
- **player_name**: The specific name of the player to analyze.

Steps:

1. **Check Player Existence**:
    - Verify if the given player exists in the dataset.
    - If the player is not found, raise an error with a descriptive message.

2. **Filter Dataset**:
    - Filter the dataset to include only data related to the specified player.
3. **Calculate Metrics**:
    - **Total Goals**: The total number of goals scored by the player.
    - **Total Assists**: The total number of assists made by the player.
    - **Minutes Played**: The total minutes the player has played.
    - **Goals per 90 Minutes**: The rate at which the player scores goals per 90 minutes.
    - **Assists per 90 Minutes**: The rate at which the player provides assists per 90 minutes.
4. **Store Results**:
    - Store these metrics along with the player's name in a list.
5. **Assign Class**:
    - Assign the S3 class `player_analysis` to the result list, which facilitates specialized handling through methods.
6. **Return Result**:
    - Return the list containing the player's performance data.

Output:

- A list containing:
    - `player_name`: The player's name.
    - `total_goals`: Total goals scored.
    - `total_assists`: Total assists made.
    - `minutes_played`: Total minutes played.
    - `goals_per_90`: Goals scored per 90 minutes.
    - `assists_per_90`: Assists provided per 90 minutes.

This structured approach ensures that the function accurately processes player data and formats it for further analysis using S3 methods like printing, summarizing, and plotting.

3.2: Create Custom Methods

To make the results informative, defining these methods for the `player_analysis` class:

1. **Print Method**

   Shows a quick summary of the player's performance.

```r
# Defining a custom print method for the "player_analysis" S3 class
print.player_analysis <- function(x, ...) {
  # Printing the player's name as part of the analysis summary
  cat("Player Analysis for:", x$player_name, "\n")

  # Printing the total number of goals scored by the player
  cat("Total Goals:", x$total_goals, "\n")

  # Printing the total number of assists made by the player
  cat("Total Assists:", x$total_assists, "\n")
}
```

   This code defines a custom `print` method for the `player_analysis` S3 class. It provides a **basic summary** of a player's performance with key highlights.

   Key Details:

   1. **`cat("Player Analysis for:", x$player_name, "\n")`**: Displays the player's name to identify the analysis.
   2. **`cat("Total Goals:", x$total_goals, "\n")`**: Outputs the total number of goals scored by the player.
   3. **`cat("Total Assists:", x$total_assists, "\n")`**: Displays the total number of assists made by the player.

   This `print` method offers a quick and straightforward overview of the player's key statistics, focused on goals and assists.

2. **Summary Method**

   Displays detailed statistics for the player.

```r
# Defining a custom summary method for the "player_analysis" S3 class
summary.player_analysis <- function(object, ...) {
  # Printing a header to indicate that this is a detailed player analysis
  cat("Detailed Player Analysis:\n")

  # Printing the player's name for identification
  cat("Player:", object$player_name, "\n")
```

```
  # Printing the total minutes played by the player
  cat("Minutes Played:", object$minutes_played, "\n")

  # Printing the number of goals scored by the player per 90 minutes
  cat("Goals per 90 minutes:", object$goals_per_90, "\n")

  # Printing the number of assists made by the player per 90 minutes
  cat("Assists per 90 minutes:", object$assists_per_90, "\n")
}
```

This code defines a custom `summary` method for the `player_analysis` S3 class. It provides a **detailed textual summary** of a player's performance, including key metrics.

Key Details:

1. **`cat("Detailed Player Analysis:\n")`**: Prints a header to indicate the start of the summary.
2. **`cat("Player:", object$player_name, "\n")`**: Displays the player's name.
3. **`cat("Minutes Played:", object$minutes_played, "\n")`**: Outputs the total minutes played by the player.
4. **`cat("Goals per 90 minutes:", object$goals_per_90, "\n")`**: Shows the player's efficiency in scoring goals per 90 minutes.
5. **`cat("Assists per 90 minutes:", object$assists_per_90, "\n")`**: Displays the player's efficiency in providing assists per 90 minutes.

This method is designed to give a **comprehensive summary** of a player's performance metrics, going beyond the basic information provided by the `print` method.

3. **Plot Method**

Creates a bar chart for goals and assists.

```
# Function to create a bar plot for player analysis
plot.player_analysis <- function(x, ...) {

  # Create a bar plot using player's total goals and assists
  barplot(
    c(x$total_goals, x$total_assists),
    names.arg = c("Goals", "Assists"),
    col = c("blue", "green"),
    main = paste("Performance of", x$player_name),
    ylab = "Count"      # Label for the y-axis
```

```
  )
}
```

This code defines a custom `plot` method for the `player_analysis` S3 class. It creates a **bar plot** to visually compare a player's **total goals** and **total assists**.

Key Details:

1. `barplot`: Generates a bar chart.
   - `c(x$total_goals, x$total_assists)`: Sets the bar heights (goals and assists).
   - `names.arg = c("Goals", "Assists")`: Labels the bars.
   - `col = c("blue", "green")`: Colors the bars (blue for goals, green for assists).
   - `main`: Sets the plot title (includes the player's name dynamically).
   - `ylab`: Labels the y-axis as "Count."

The plot gives a clear comparison of the player's contribution in terms of goals and assists.

3.3: Example Usage: Analyzing Players

Analyzing a Player: **Eden Hazard**

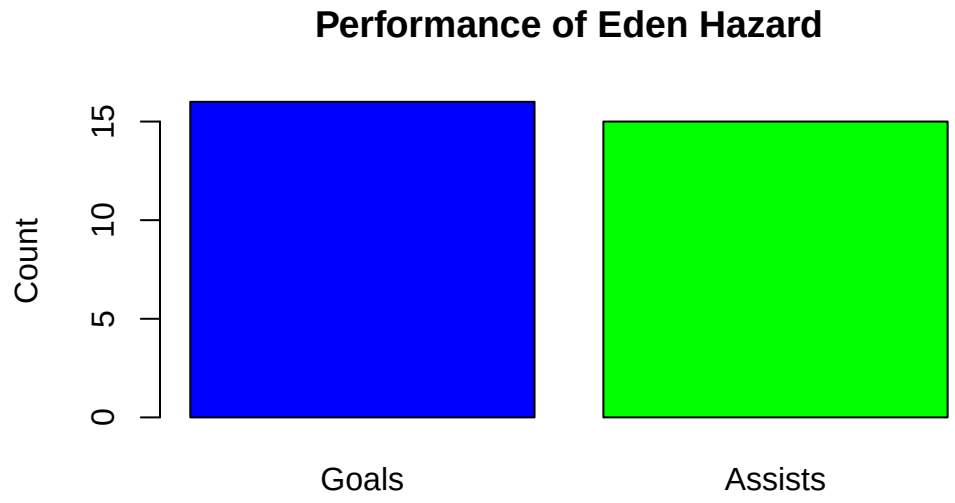
```
# Analyzing a specific player (Eden Hazard) using the player_analysis function
# This will extract and calculate performance metrics for the player
hazard_analysis <- player_analysis(players_data, "Eden Hazard")

# Print the basic performance summary for Eden Hazard
# Includes details like total goals and assists
print(hazard_analysis)
```

```
Player Analysis for: Eden Hazard
Total Goals: 16
Total Assists: 15
```

```
# Display a detailed summary of Eden Hazard's performance
# Includes additional metrics like minutes played, goals per 90, and assists per 90
summary(hazard_analysis)
```

```
Detailed Player Analysis:
Player: Eden Hazard
Minutes Played: 2926
Goals per 90 minutes: 0.49
Assists per 90 minutes: 0.46
```

```r
# Generate a bar plot visualizing Eden Hazard's performance
# Compares metrics like goals and assists
plot(hazard_analysis)
```



Analyzing a Player: **Sergio Aguero**

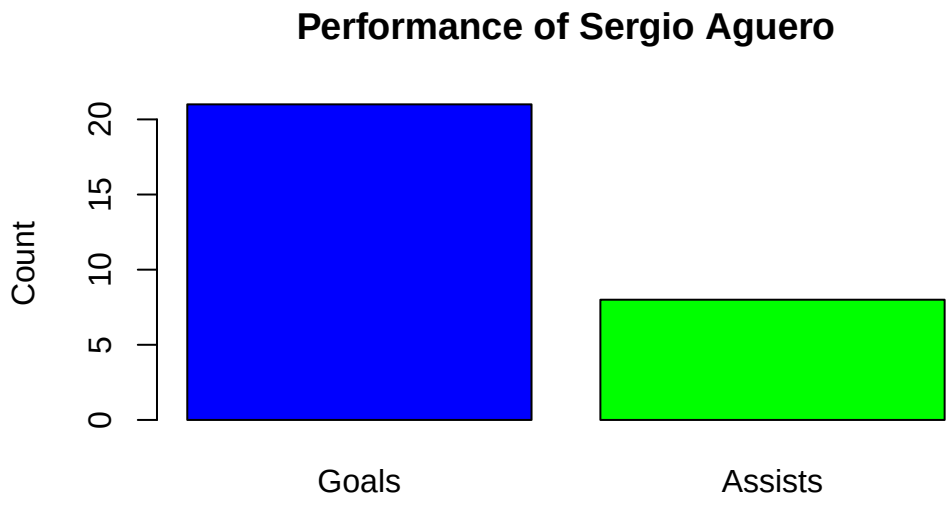
```r
# Analyzing a player named "Sergio Aguero"
sergio_analysis <- player_analysis(players_data, "Sergio Aguero")

# Printing results
print(sergio_analysis)
```

```
Player Analysis for: Sergio Aguero
Total Goals: 21
Total Assists: 8
```

```r
# Summary of results
summary(sergio_analysis)
```

```
Detailed Player Analysis:
Player: Sergio Aguero
Minutes Played: 2480
Goals per 90 minutes: 0.76
Assists per 90 minutes: 0.29
```

```r
# Plotting results
plot(sergio_analysis)
```



Sergio Aguero's Performance:

- **Goals:** Aguero scored **20 goals**. The blue bar represents the number of goals.
- **Assists:** He provided **8 assists**. The green bar represents the number of assists.

Comparison with Eden Hazard:

- **Goals:** Aguero scored 5 more goals than Hazard.
- **Assists:** Hazard provided 7 more assists than Aguero.

Overall:

- Aguero was more prolific in scoring goals, while Hazard was more involved in creating goal-scoring opportunities for his teammates.

3.4: Conclusion

The custom `player_analysis` function delivered a structured framework for analyzing **individual player performance**, calculating key metrics like goals, assists, and efficiency per 90 minutes. Custom methods for `print`, `summary`, and `plot` enabled detailed summaries and **visual comparisons** of performance metrics. Analyses of players like Eden Hazard and Sergio Aguero showcased its ability to provide **actionable insights**, emphasizing the utility of modular and reusable programming practices in R.