DOCUMENTATION

Part 1: Data Importation and Initial Exploration

Description:

This section covers the importation of the dataset, displaying the first and last rows, and selecting key attributes.

---

1. Load Necessary Libraries

```
library(readr)
library(dplyr)
library(tidyr)
```

Explanation:

The `readr` library is used for efficient reading of CSV files. `dplyr` is for data manipulation, and `tidyr` helps in tidying the data (although it is not actively used in this part).

---

2. Load the Dataset

```
file_path <- "E:/Desktop/LANG-R/R PROJECT/data_minimized.csv"
data <- read_csv(file_path)
```

Explanation:

The dataset is loaded from the specified path (`data_minimized.csv`). The `read_csv()` function from `readr` reads the CSV file into a tibble (a modern version of a data frame).

---

3. Display the First Rows

```
data_head <- head(data)
print("Premières lignes du jeu de données :")
```

```
print(data_head)
```

Explanation:

The `head()` function displays the first six rows of the dataset to give an overview of the data's structure.

Result:

```
# A tibble: 6 × 54
  `Unnamed: 0`     ID Name        Age Photo Nationality Flag
Overall Potential Club
         <dbl>  <dbl> <chr>     <dbl> <chr> <chr>       <chr>
<dbl>     <dbl> <chr>
1            0 158023 L. Messi     31 http… Argentina   http…
94        94 FC B…
2            1  20801 Cristian…    33 http… Portugal    http…
94        94 Juve…
3            2 190871 Neymar Jr    26 http… Brazil      http…
92        93 Pari…
4            3 193080 De Gea       27 http… Spain       http…
91        93 Manc…
5            4 192985 K. De Br…    27 http… Belgium     http…
91        92 Manc…
6            5 183277 E. Hazard    27 http… Belgium     http…
91        91 Chel…
```

4. Display the Last Rows

```
data_tail <- tail(data)
print("Dernières lignes du jeu de données :")
print(data_tail)
```

Explanation:

The `tail()` function displays the last six rows of the dataset to ensure there are no issues with the data at the end.

Result:

```
# A tibble: 6 × 54
  `Unnamed: 0`     ID Name        Age Photo Nationality Flag
Overall Potential Club
         <dbl>  <dbl> <chr>     <dbl> <chr> <chr>       <chr>
<dbl>     <dbl> <chr>
1        18201 243413 D. Walsh     18 http… Republic o… http…
47         68 Wate…
2        18202 238813 J. Lunds…    19 http… England     http…
47         65 Crew…
3        18203 243165 N. Chris…    19 http… Sweden      http…
47         63 Trel…
4        18204 241638 B. Worman    16 http… England     http…
47         67 Camb…
5        18205 246268 D. Walke…    17 http… England     http…
47         66 Tran…
6        18206 246269 G. Nugent    16 http… England     http…
46         66 Tran…
```

5. Display Dimensions (Rows and Columns)

```
data_dimensions <- dim(data)
print(paste("Dimensions du jeu de données :", data_dimensions
[1], "lignes et", data_dimensions[2], "colonnes"))
```

Explanation:

The `dim()` function returns the dimensions of the dataset (rows and columns). This helps you understand the overall size of the data.

Result:

```
"Dimensions du jeu de données : 18207 lignes et 54 colonnes"
```

6. Select Players by Position

```
unique_positions <- unique(data$Position)
print("Postes disponibles :")
print(unique_positions)
```

Explanation:

This step identifies all the unique player positions in the dataset using the `unique()` function.

Result:

```
"Postes disponibles :"
[1] "RF"  "ST"  "LW"  "GK"  "RCM" "LF"  "RS"  "RCB" "LCM" "C
B"  "LDM" "CAM" "CDM"
[14] "LS"  "LCB" "RM"  "LAM" "LM"  "LB"  "RDM" "RW"  "CM"  "R
B"  "RAM" "CF"  "RWB"
[27] "LWB" NA
```

Summary:

This section focused on loading and inspecting the dataset, providing an overview of its structure and unique player positions. The next steps will likely involve more detailed analysis and manipulation of this data.

Data Manipulation with `dplyr` and `tidyr`

Objective:

This section focuses on utilizing the `dplyr` and `tidyr` packages to manipulate and transform a dataset of football players. The dataset contains various details such as player names, clubs, positions, age, nationality, and many other attributes. The operations aim to filter, group, and reshape the data for analysis.

Required Libraries:

```
# Load necessary libraries
library(dplyr)
library(tidyr)
```

Code & Explanation

1. Filter Players by Position (ST)

The first operation filters the dataset to extract only the players who play as Strikers (Position = "ST").

```
# Filter players with the position "ST" (Striker)
selected_position <- data %>% filter(Position == "ST")

# Print the names of selected players
print("Joueurs sélectionnés pour le poste ST :")
print(head(selected_position$Name))
```

Explanation:

- `filter(Position == "ST")`: Filters the rows where the `Position` column is equal to "ST" (Striker).
- `head(selected_position$Name)`: Displays the first few names of the players who have the position "ST".

Result:

```
Joueurs sélectionnés pour le poste ST :
[1] "Cristiano Ronaldo" "R. Lewandowski"    "H. Kane"
"S. Agüero"
[5] "G. Bale"           "M. Icardi"
```

2. Filter Players by Club (Real Madrid)

Next, we filter the dataset to list all players who belong to the club "Real Madrid".

```
# Filter players belonging to "Real Madrid"
real_madrid_players <- data %>% filter(Club == "Real Madrid")

# Print the names of the Real Madrid players
print("Joueurs du Real Madrid :")
print(real_madrid_players$Name)
```

Explanation:

- `filter(Club == "Real Madrid")`: Filters the rows where the `Club` column is equal to "Real Madrid".
- `real_madrid_players$Name`: Extracts and displays the names of the Real Madrid players.

Result:

```
Joueurs du Real Madrid :
[1] "L. Modrić"       "Sergio Ramos"    "T. Kroos"        "T.
Courtois"
 [5] "Casemiro"        "Isco"            "Marcelo"
"G. Bale"
 [9] "K. Navas"        "R. Varane"       "Marco Asensio"
"K. Benzema"
[13] "Carvajal"        "Lucas Vázquez"   "Nacho Fernández" "D
ani Ceballos"
...
```

3. Analyze Players by Position

This part groups the players by their position and calculates the average age and the number of players in each position.

```
# Analyze players by position: average age and count of playe
rs
position_analysis <- data %>%
  group_by(Position) %>%
```

```
  summarise(Average_Age = mean(Age, na.rm = TRUE), Count = n
())

# Print the result of the analysis
print("Analyse par poste (moyenne d'âge et nombre de joueurs)
:")
print(head(position_analysis))
```

Explanation:

- `group_by(Position)`: Groups the dataset by the `Position` column.
- `summarise()`: Calculates the mean age (`Average_Age`) and the number of players (`Count`) in each group, excluding any `NA` values with `na.rm = TRUE`.

Result:

```
Analyse par poste (moyenne d'âge et nombre de joueurs) :
# A tibble: 6 × 3
  Position Average_Age Count
  <chr>          <dbl> <int>
1 CAM             24.6   958
2 CB              24.8  1778
3 CDM             25.2   948
4 CF              23.5    74
5 CM              23.5  1394
6 GK              26.0  2025
```

4. Filter Players by Age (> 25)

Here, we filter players who are older than 25 years.

```
# Filter players with age > 25
filtered_data <- data %>% filter(Age > 25)

# Print the filtered data
```

```
print("Données filtrées (Age > 25) :")
print(head(filtered_data))
```

Explanation:

- `filter(Age > 25)`: Filters the dataset to only include players whose age is greater than 25.

Result:

```
Données filtrées (Age > 25) :
# A tibble: 6 × 54
  `Unnamed: 0`     ID Name        Age Photo Nationality Flag
Overall Potential Club
         <dbl>  <dbl> <chr>     <dbl> <chr> <chr>       <chr>
<dbl>     <dbl> <chr>
1            0 158023 L. Messi     31 http… Argentina   http…
94        94 FC B…
2            1  20801 Cristian…    33 http… Portugal    http…
94        94 Juve…
3            2 190871 Neymar Jr    26 http… Brazil      http…
92        93 Pari…
4            3 193080 De Gea       27 http… Spain       http…
91        93 Manc…
5            4 192985 K. De Br…    27 http… Belgium     http…
91        92 Manc…
6            5 183277 E. Hazard    27 http… Belgium     http…
91        91 Chel…
```

5. Select Specific Columns

In this step, we select only the `Name` and `Age` columns from the dataset.

```
# Select only the 'Name' and 'Age' columns
selected_data <- data %>% select(Name, Age)

# Print the selected columns
```

```
print("Colonnes sélectionnées :")
print(head(selected_data))
```

Explanation:

- `select(Name, Age)`: Selects the `Name` and `Age` columns from the dataset.

Result:

```
Colonnes sélectionnées :
# A tibble: 6 × 2
  Name                Age
  <chr>             <dbl>
1 L. Messi             31
2 Cristiano Ronaldo    33
3 Neymar Jr            26
4 De Gea               27
5 K. De Bruyne         27
6 E. Hazard            27
```

6. Calculate Average Age by Club

Here, we calculate the average age of players for each club.

```
# Calculate average age per club
average_age_per_club <- data %>%
  group_by(Club) %>%
  summarise(Average_Age = mean(Age, na.rm = TRUE)) %>%
  ungroup()

# Print the result
print("Âge moyen par club :")
print(average_age_per_club)
```

Explanation:

- `group_by(Club)`: Groups the data by the `Club` column.

- `summarise()`: Calculates the average age (`Average_Age`) for each club.
- `ungroup()`: Removes the grouping after the calculation.

Result:

```
Âge moyen par club :
# A tibble: 652 × 2
   Club                  Average_Age
   <chr>                       <dbl>
1 1. FC Heidenheim 1846        24
2 1. FC Kaiserslautern         23.8
3 1. FC Köln                   24.3
...
```

7. Pivot Data to Wider Format (Club Columns)

In this operation, we pivot the data to a wider format where each club becomes a separate column containing the average age of players in that club.

```
# Pivot the data to a wider format: each club becomes a colum
n
wider_data <- average_age_per_club %>%
  pivot_wider(names_from = Club, values_from = Average_Age)

# Print the result
print("Données après pivot_wider :")
print(wider_data)
```

Explanation:

- `pivot_wider(names_from = Club, values_from = Average_Age)`: Converts the dataset into a wider format where the clubs are the column names and the values are the average ages of players in those clubs.

Result:

```
Données après pivot_wider :
# A tibble: 1 × 652
  `1. FC Heidenheim 1846` `1. FC Kaiserslautern` `1. FC Köln`
`1. FC Magdeburg`
                    <dbl>                   <dbl>        <dbl>
<dbl>
1                      24                    23.8         24.3
24.7
...
```

8. Pivot Data Back to Long Format

Finally, we pivot the dataset back to a long format where each row corresponds to a club and its average age.

```
# Pivot the data back to long format
long_data <- wider_data %>%
  pivot_longer(cols = everything(), names_to = "Club", values
_to = "Average_Age")

# Print the result
print("Données après pivot_longer :")
print(long_data)
```

Explanation:

- `pivot_longer(cols = everything(), names_to = "Club", values_to = "Average_Age")`: Converts the dataset from the wide format back to a long format, where each club and its average age is represented in a single row.

Result:

```
Données après pivot_longer :
# A tibble: 652 × 2
   Club                  Average_Age
   <chr>                       <dbl>
```

```
1 1. FC Heidenheim 1846        24
2 1. FC Kaiserslautern         23.8
...
```

Conclusion

The operations performed in this script demonstrate how to filter, group, and reshape data using `dplyr` and `tidyr`. The goal was to analyze player data by position, club, and age, while reshaping the data for easier analysis and presentation.

PART 3 : Documentation for the Data Visualization Code

1. Libraries Used

- `ggplot2`: This is a powerful package in R for creating static visualizations. It allows users to build plots in a layered, declarative way using `ggplot()` as the main function.
- `plotly`: This package allows you to make interactive plots, enhancing the user experience by enabling features such as zoom, hover, and clickable elements.

2. Code Breakdown

2.1 Creating an Histogram of Age

```
ggplot(data, aes(x = Age)) +
  geom_histogram(binwidth = 1, fill = "blue", color = "blac
k") +
  labs(title = "Histogramme de l'âge", x = "Age", y = "Nombre
de joueurs")
```

- **Objective**: The histogram shows the distribution of player ages.
- `ggplot(data, aes(x = Age))`: Initializes the plot with the dataset `data` and sets `Age` as the x-axis.

- `geom_histogram()`: Plots a histogram. The `binwidth` controls the width of each bar, `fill` is used for coloring the bars, and `color` is used for the border of each bar.
- `labs()`: Adds titles to the plot and axis labels.

2.2 Boxplot of Age by Position

```
ggplot(data, aes(x = Position, y = Age, fill = Position)) +
  geom_boxplot() +
  labs(title = "Diagramme à boîtes de l'âge par position", x
= "Position", y = "Âge") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```

- **Objective**: This boxplot visualizes the distribution of ages for each position.
- `ggplot(data, aes(x = Position, y = Age, fill = Position))`: Sets the x-axis to the `Position` and the y-axis to `Age`. The `fill` aesthetic colors the boxes by position.
- `geom_boxplot()`: Draws a boxplot, summarizing the distribution of age values for each position.
- `labs()`: Adds a title and axis labels.
- `theme(axis.text.x = element_text(angle = 90, hjust = 1))`: Rotates the x-axis labels for better readability, especially when positions have long names.

2.3 Boxplot of Value by Position

```
ggplot(data, aes(x = Position, y = Value, fill = Position)) +
  geom_boxplot() +
  labs(title = "Diagramme à boîtes de la valeur par positio
n", x = "Position", y = "Valeur") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```

- **Objective**: This boxplot shows the distribution of player values by position.

- `ggplot(data, aes(x = Position, y = Value, fill = Position))`: Similar to the previous boxplot but visualizing `Value` on the y-axis.
- `geom_boxplot()`: Creates the boxplot for `Value`.
- `labs()`: Adds title and labels.
- `theme(axis.text.x = element_text(angle = 90, hjust = 1))`: Rotates the x-axis labels to make them easier to read.

2.4 Bar Plot for Average Age by Position

```
avg_age_by_position <- data %>%
  group_by(Position) %>%
  summarise(Average_Age = mean(Age, na.rm = TRUE))

ggplot(avg_age_by_position, aes(x = Position, y = Average_Ag
e, fill = Position)) +
  geom_bar(stat = "identity") +
  labs(title = "Âge moyen par position", x = "Position", y =
"Âge moyen") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```

- **Objective**: This bar plot shows the average age for each position.
- `avg_age_by_position <- data %>% ...`: Uses the `dplyr` package to group the data by `Position` and calculate the average `Age` for each position.
- `geom_bar(stat = "identity")`: Plots the data as a bar plot where the height of the bars corresponds to the `Average_Age`.
- `labs()`: Adds a title and axis labels.
- `theme(axis.text.x = element_text(angle = 90, hjust = 1))`: Rotates the x-axis labels for better readability.

2.5 Interactive Boxplot for Age by Position Using `plotly`

```
p1 <- ggplot(data, aes(x = Position, y = Age, fill = Positio
n)) +
```

```
  geom_boxplot() +
  labs(title = "Diagramme à boîtes de l'âge par position", x
= "Position", y = "Âge") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))

# Convertir ggplot en plot interactif avec plotly
interactive_p1 <- ggplotly(p1)
interactive_p1
```

- **Objective**: This interactive boxplot allows users to zoom, hover, and explore the data more dynamically.
- `ggplot(data, aes(x = Position, y = Age, fill = Position))`: Sets up the plot with the same data and aesthetics as the previous static boxplot.
- `ggplotly(p1)`: Converts the static `ggplot` object (`p1`) into an interactive plot using the `plotly` package.
- `interactive_p1`: This is the interactive plot, which can be displayed in an R environment (RStudio, Jupyter Notebook, etc.).

Saving Images of the Plots

If you want to save the plots as images, you can use the following commands:

Save Static Plots:

```
# Save the histogram as an image
ggsave("histogram_age.png", plot = last_plot(), width = 8, he
ight = 6)

# Save the boxplot of age by position
ggsave("boxplot_age_by_position.png", plot = last_plot(), wid
th = 8, height = 6)

# Save the bar plot of average age by position
```

```
ggsave("barplot_avg_age_by_position.png", plot = last_plot(),
width = 8, height = 6)
```

Save Interactive Plots:

For interactive plots using `plotly`, you can save them as HTML files.

```
# Save the interactive plot
htmlwidgets::saveWidget(interactive_p1, "interactive_boxplot_
age_by_position.html")
```

- `ggsave()`: Saves the static plot as an image (e.g., PNG, PDF, etc.). You can adjust the `width` and `height` to control the size of the output image.
- `htmlwidgets::saveWidget()`: Saves the interactive plot as an HTML file that you can open in any web browser.

Summary of Visualization Outputs

- **Histogram**: Shows the distribution of ages of all players in the dataset.
- **Boxplot for Age by Position**: Displays how age varies across different player positions.
- **Boxplot for Value by Position**: Displays the distribution of player values by position.
- **Bar Plot for Average Age by Position**: Shows the average age of players grouped by position.
- **Interactive Boxplot**: A dynamic version of the boxplot for age by position, allowing user interaction such as zooming and hover to explore the data.

Conclusion

These visualizations help to understand various aspects of the FIFA 19 dataset, such as the distribution of player ages, their values, and average ages by position. The interactive plot enhances the experience by providing a more detailed and explorative way to analyze the data.

Documentation for the Age Mean and Median Calculation Code

1. Overview

This section of the code computes two important statistics: the **mean** and the **median** of the `Age` column in the dataset `data`. These measures provide insights into the central tendency of player ages in the dataset.

2. Code Breakdown

2.1 Calculating the Mean of Age

```
age_mean <- mean(data$Age, na.rm = TRUE)
```

- **Objective**: This line calculates the **mean** (average) age of the players in the dataset.
- `data$Age`: Accesses the `Age` column in the `data` dataframe.
- `mean()`: The `mean()` function calculates the average of the specified column (`Age`).
- `na.rm = TRUE`: This argument ensures that `NA` (missing) values are ignored during the calculation of the mean. If `NA` values were present and not removed, the result would be `NA`.
- **Result**: The computed mean age is stored in the variable `age_mean`.

2.2 Calculating the Median of Age

```
age_median <- median(data$Age, na.rm = TRUE)
```

- **Objective**: This line calculates the **median** age of the players in the dataset.
- `data$Age`: Accesses the `Age` column in the `data` dataframe.
- `median()`: The `median()` function calculates the median, which is the middle value when the ages are sorted in ascending order.
- `na.rm = TRUE`: Like the `mean()` function, this argument ensures that missing values (`NA`) are excluded from the calculation of the median.

- **Result**: The computed median age is stored in the variable `age_median`.

2.3 Displaying the Results

```
print(paste("Moyenne de l'âge :", age_mean))
```

- **Objective**: This line prints the mean age to the console.
- `paste()`: Combines the string `"Moyenne de l'âge :"` with the computed `age_mean` value.
- `print()`: Displays the concatenated string and computed mean age in the console.

The output for the mean age looks like this:

```
"Moyenne de l'âge : 25.1222057450431"
```

```
print(paste("Médiane de l'âge :", age_median))
```

- **Objective**: This line prints the median age to the console.
- `paste()`: Combines the string `"Médiane de l'âge :"` with the computed `age_median` value.
- `print()`: Displays the concatenated string and computed median age in the console.

The output for the median age looks like this:

```
"Médiane de l'âge : 25"
```

3. Explanation of Mean and Median

- **Mean**: The mean age is the average age of all the players in the dataset. It is calculated by summing all the ages and dividing by the number of players.
    - In this case, the mean age is approximately **25.12** years.

- **Median**: The median is the middle value in the dataset when all values are sorted. If there is an odd number of values, it is the middle value; if even, it is the average of the two middle values.
    - Here, the median age is **25**, meaning half of the players are younger than 25, and half are older than 25.

4. Use of `na.rm = TRUE`

Both `mean()` and `median()` functions include the argument `na.rm = TRUE`. This ensures that missing (`NA`) values in the `Age` column do not interfere with the calculations. Without this argument, if any `NA` values are present, both functions would return `NA` as the result.

5. Conclusion

- **Mean Age**: The average age of players in the dataset is approximately **25.12** years.
- **Median Age**: The middle value for age is **25** years, indicating that the dataset is fairly balanced around this age.