ashes

Bulliraju

2024-02-25

```
library(readr)

## Warning: package 'readr' was built under R version 4.3.3

library(forcats)

## Warning: package 'forcats' was built under R version 4.3.3

library(dplyr)

## Warning: package 'dplyr' was built under R version 4.3.3

##
## 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

library(summarytools)

## Warning: package 'summarytools' was built under R version 4.3.3

library(stringr)

## Warning: package 'stringr' was built under R version 4.3.3

library(ggplot2)
library(tidyverse)

## Warning: package 'tidyverse' was built under R version 4.3.3

## Warning: package 'tibble' was built under R version 4.3.3

## Warning: package 'tidyr' was built under R version 4.3.3

## Warning: package 'purrr' was built under R version 4.3.3

## Warning: package 'lubridate' was built under R version 4.3.3

## ── Attaching core tidyverse packages ──────────────────────── tidyverse
2.0.0 ──
```

```
## ✓ lubridate 1.9.3     ✓ tibble    3.2.1
## ✓ purrr     1.0.2     ✓ tidyr     1.3.1

## ── Conflicts ──────────────────────────────────────────
tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ✖ tibble::view()  masks summarytools::view()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all
conflicts to become errors
```

```
data <- read.csv(file.choose())
View(data)

# lets preview our data
head(data)
```

```
##     batter      team         role
## 1      Ali   England   allrounder
## 2 Anderson   English         bowl
## 3 Bairstow   England wicketkeeper
## 4     Ball   England         bowl
## 5 Bancroft Australia          bat
## 6     Bird Australia         bowl
##
Test.1..Innings.1
## 1 Batting at number 6, scored 38 runs from 102 balls including 2 fours and
1 sixes.
## 2   Batting at number 11, scored 5 runs from 9 balls including 1 fours and
0 sixes.
## 3   Batting at number 7, scored 9 runs from 24 balls including 1 fours and
0 sixes.
## 4 Batting at number 10, scored 14 runs from 11 balls including 3 fours and
0 sixes.
## 5   Batting at number 1, scored 5 runs from 19 balls including 0 fours and
0 sixes.
## 6                  Batting at number NA, scored NA including NA fours and
NA sixes.
##
Test.1..Innings.2
## 1   Batting at number 6, scored 40 runs from 64 balls including 6 fours
and 0 sixes.
## 2    Batting at number 11, scored 0 runs from 1 balls including 0 fours
and 0 sixes.
## 3   Batting at number 7, scored 42 runs from 75 balls including 2 fours
and 1 sixes.
## 4    Batting at number 10, scored 1 runs from 5 balls including 0 fours
and 0 sixes.
## 5 Batting at number 1, scored 82 runs from 182 balls including 10 fours
and 1 sixes.
```

```
## 6                   Batting at number NA, scored NA including NA fours and
NA sixes.
##
Test.2..Innings.1
## 1 Batting at number 6, scored 25 runs from 57 balls including 2 fours and
0 sixes.
## 2  Batting at number 11, scored 0 runs from 3 balls including 0 fours and
0 sixes.
## 3 Batting at number 7, scored 21 runs from 50 balls including 2 fours and
0 sixes.
## 4                 Batting at number NA, scored NA including NA fours and
NA sixes.
## 5 Batting at number 1, scored 10 runs from 41 balls including 0 fours and
0 sixes.
## 6                 Batting at number NA, scored NA including NA fours and
NA sixes.
##
Test.2..Innings.2
## 1  Batting at number 7, scored 2 runs from 20 balls including 0 fours and
0 sixes.
## 2  Batting at number 11, scored 0 runs from 0 balls including 0 fours and
0 sixes.
## 3 Batting at number 8, scored 36 runs from 57 balls including 5 fours and
0 sixes.
## 4                 Batting at number NA, scored NA including NA fours and
NA sixes.
## 5   Batting at number 1, scored 4 runs from 8 balls including 1 fours and
0 sixes.
## 6                 Batting at number NA, scored NA including NA fours and
NA sixes.
##
Test.3..Innings.1
## 1      Batting at number 7, scored 0 runs from 2 balls including 0 fours
and 0 sixes.
## 2     Batting at number 11, scored 0 runs from 7 balls including 0 fours
and 0 sixes.
## 3 Batting at number 6, scored 119 runs from 215 balls including 18 fours
and 0 sixes.
## 4                    Batting at number NA, scored NA including NA fours
and NA sixes.
## 5    Batting at number 1, scored 25 runs from 55 balls including 3 fours
and 0 sixes.
## 6                    Batting at number NA, scored NA including NA fours
and NA sixes.
##
Test.3..Innings.2
## 1 Batting at number 7, scored 11 runs from 56 balls including 2 fours and
0 sixes.
## 2  Batting at number 11, scored 1 runs from 7 balls including 0 fours and
0 sixes.
```

```
## 3 Batting at number 6, scored 14 runs from 26 balls including 3 fours and
0 sixes.
## 4                 Batting at number NA, scored NA including NA fours and
NA sixes.
## 5                 Batting at number NA, scored NA including NA fours and
NA sixes.
## 6                 Batting at number NA, scored NA including NA fours and
NA sixes.
##
Test.4..Innings.1
## 1 Batting at number 7, scored 20 runs from 14 balls including 2 fours and
1 sixes.
## 2 Batting at number 11, scored 0 runs from 16 balls including 0 fours and
0 sixes.
## 3 Batting at number 6, scored 22 runs from 39 balls including 3 fours and
0 sixes.
## 4                 Batting at number NA, scored NA including NA fours and
NA sixes.
## 5 Batting at number 1, scored 26 runs from 95 balls including 2 fours and
0 sixes.
## 6   Batting at number 9, scored 4 runs from 6 balls including 1 fours and
0 sixes.
##
Test.4..Innings.2
## 1                 Batting at number NA, scored NA including NA fours and
NA sixes.
## 2                 Batting at number NA, scored NA including NA fours and
NA sixes.
## 3                 Batting at number NA, scored NA including NA fours and
NA sixes.
## 4                 Batting at number NA, scored NA including NA fours and
NA sixes.
## 5 Batting at number 1, scored 27 runs from 42 balls including 4 fours and
0 sixes.
## 6                 Batting at number NA, scored NA including NA fours and
NA sixes.
##
Test.5..Innings.1
## 1 Batting at number 7, scored 30 runs from 58 balls including 2 fours and
0 sixes.
## 2  Batting at number 11, scored 0 runs from 3 balls including 0 fours and
0 sixes.
## 3   Batting at number 6, scored 5 runs from 7 balls including 1 fours and
0 sixes.
## 4                 Batting at number NA, scored NA including NA fours and
NA sixes.
## 5   Batting at number 1, scored 0 runs from 7 balls including 0 fours and
0 sixes.
## 6                 Batting at number NA, scored NA including NA fours and
NA sixes.
```

```
##
Test.5..Innings.2
## 1  Batting at number 7, scored 13 runs from 43 balls including 1 fours and
0 sixes.
## 2  Batting at number 11, scored 2 runs from 23 balls including 0 fours and
0 sixes.
## 3 Batting at number 6, scored 38 runs from 143 balls including 4 fours and
0 sixes.
## 4                Batting at number NA, scored NA including NA fours and
NA sixes.
## 5                Batting at number NA, scored NA including NA fours and
NA sixes.
## 6                Batting at number NA, scored NA including NA fours and
NA sixes.
```

```r
data_shape <- dim(data)

# lets print the dimensions
print(data_shape)
```

```
## [1] 27 13
```

```r
column_names <- names(data)
print(column_names)
```

```
##  [1] "batter"            "team"              "role"
##  [4] "Test.1..Innings.1" "Test.1..Innings.2" "Test.2..Innings.1"
##  [7] "Test.2..Innings.2" "Test.3..Innings.1" "Test.3..Innings.2"
## [10] "Test.4..Innings.1" "Test.4..Innings.2" "Test.5..Innings.1"
## [13] "Test.5..Innings.2"
```

```r
glimpse(data)
```

```
## Rows: 27
## Columns: 13
## $ batter            <chr> "Ali", "Anderson", "Bairstow", "Ball",
"Bancroft", "…
## $ team              <chr> "England", "English", "England", "England",
"Austral…
## $ role              <chr> "allrounder", "bowl", "wicketkeeper", "bowl",
"bat",…
## $ Test.1..Innings.1 <chr> "Batting at number 6, scored 38 runs from 102
balls …
## $ Test.1..Innings.2 <chr> "Batting at number 6, scored 40 runs from 64
balls i…
## $ Test.2..Innings.1 <chr> "Batting at number 6, scored 25 runs from 57
balls i…
## $ Test.2..Innings.2 <chr> "Batting at number 7, scored 2 runs from 20
balls in…
## $ Test.3..Innings.1 <chr> "Batting at number 7, scored 0 runs from 2 balls
inc…
```

```
## $ Test.3..Innings.2 <chr> "Batting at number 7, scored 11 runs from 56 
balls i…
## $ Test.4..Innings.1 <chr> "Batting at number 7, scored 20 runs from 14 
balls i…
## $ Test.4..Innings.2 <chr> "Batting at number NA, scored NA including NA 
fours …
## $ Test.5..Innings.1 <chr> "Batting at number 7, scored 30 runs from 58 
balls i…
## $ Test.5..Innings.2 <chr> "Batting at number 7, scored 13 runs from 43 
balls i…
```

```r
# lets check for missing values in the dataset
missing_values <- any(is.na(data))

# Print the result
print(missing_values)
```

```
## [1] FALSE
```

1.a) Each subject needs its own row. Rearrange the data into a long format so that there is a row for each batter in each innings. Your new tibble should have 270 rows.

```r
library(tidyr)

# lets rearrange the data into long format
long_data <- data %>%
  gather(key = "innings", value = "score", -c(batter, team, role))

# Print the modified data frame
head(long_data)
```

```
##     batter      team         role           innings
## 1      Ali   England   allrounder Test.1..Innings.1
## 2 Anderson   English         bowl Test.1..Innings.1
## 3 Bairstow   England wicketkeeper Test.1..Innings.1
## 4     Ball   England         bowl Test.1..Innings.1
## 5 Bancroft Australia          bat Test.1..Innings.1
## 6     Bird Australia         bowl Test.1..Innings.1
##                                                                          
score
## 1 Batting at number 6, scored 38 runs from 102 balls including 2 fours and 
1 sixes.
## 2   Batting at number 11, scored 5 runs from 9 balls including 1 fours and 
0 sixes.
## 3   Batting at number 7, scored 9 runs from 24 balls including 1 fours and 
0 sixes.
## 4 Batting at number 10, scored 14 runs from 11 balls including 3 fours and 
0 sixes.
## 5   Batting at number 1, scored 5 runs from 19 balls including 0 fours and 
0 sixes.
```

```
## 6                  Batting at number NA, scored NA including NA fours and
NA sixes.

# lets check the shape of long_data
shape <- dim(long_data)

# Print the shape
print(shape)

## [1] 270   5
```

1.b) Each cell should represent only one measurement. Use str_match() to create new columns for each of the following for each player innings: • the player's batting number, • their score, and • the number of balls they faced.

```
library(dplyr)
library(stringr)

# lets define a function to extract information using str_match
extract_info <- function(text) {
  # lets use regular expressions to extract batting number, score, and balls
faced
  result <- str_match(text, "Batting at number (\\d+), scored (\\d+) runs
from (\\d+) balls")

  # lets return a named list with extracted values
  return(list(
    batting_number = as.numeric(result[2]),
    score = as.numeric(result[3]),
    balls_faced = as.numeric(result[4])
  ))
}

# lets apply the function to each row using dplyr::mutate
long_data <- long_data %>%
  mutate(
    # lets apply the extract_info function to the 'score' column
    new_columns = map(score, extract_info),

    # lets extract individual values from the list column
    batting_number = map_dbl(new_columns, "batting_number"),
    score = map_dbl(new_columns, "score"),
    balls_faced = map_dbl(new_columns, "balls_faced")
  ) %>%
  # lets select relevant columns
  select(batter, team, role, innings, batting_number, score, balls_faced)

# lets print the modified data frame
head(long_data)
```

```
##     batter      team         role           innings batting_number score
## 1      Ali   England   allrounder Test.1..Innings.1              6    38
## 2 Anderson   English         bowl Test.1..Innings.1             11     5
## 3 Bairstow   England wicketkeeper Test.1..Innings.1              7     9
## 4     Ball   England         bowl Test.1..Innings.1             10    14
## 5 Bancroft Australia          bat Test.1..Innings.1              1     5
## 6     Bird Australia         bowl Test.1..Innings.1             NA    NA
##   balls_faced
## 1         102
## 2           9
## 3          24
## 4          11
## 5          19
## 6          NA
```

1.c) Recode the data to make it 'tame', that is, • ensure all categorical variables with a small number of levels are coded as factors,

```
# lets identify categorical variables with a small number of levels (let's
say, less than or equal to 10)
small_levels_cols <- sapply(long_data, function(x) is.factor(x) ||
(is.character(x) && length(unique(x)) <= 10))

# lets recode small levels to factors
long_data[, small_levels_cols] <- lapply(long_data[, small_levels_cols],
as.factor)

# lets print the glimpse of the updated data frame
glimpse(long_data)
```

```
## Rows: 270
## Columns: 7
## $ batter         <chr> "Ali", "Anderson", "Bairstow", "Ball", "Bancroft",
"Bir…
## $ team           <fct> England, English, England, England, Australia,
Australi…
## $ role           <fct> allrounder, bowl, wicketkeeper, bowl, bat, bowl,
bowler…
## $ innings        <fct> Test.1..Innings.1, Test.1..Innings.1,
Test.1..Innings.1…
## $ batting_number <dbl> 6, 11, 7, 10, 1, NA, 9, 1, NA, 9, NA, 5, 10, 3, 11,
5, …
## $ score          <dbl> 38, 5, 9, 14, 5, NA, 20, 2, NA, 42, NA, 14, 6, 11,
9, 5…
## $ balls_faced    <dbl> 102, 9, 24, 11, 19, NA, 32, 10, NA, 120, NA, 17,
25, 24…
```

• ensure all categorical variables with a large number of levels are coded as characters.

```
library(dplyr)
```

```r
# lets identify categorical variables with a large number of levels
large_levels_cols <- sapply(long_data, function(x) is.factor(x) |
(is.character(x) && length(unique(x)) > 10))

# lets recode large levels to characters
long_data[, large_levels_cols] <- lapply(long_data[, large_levels_cols],
as.character)

# lets print the glimpse of the updated data frame
glimpse(long_data)

## Rows: 270
## Columns: 7
## $ batter         <chr> "Ali", "Anderson", "Bairstow", "Ball", "Bancroft",
"Bir…
## $ team           <chr> "England", "English", "England", "England",
"Australia"…
## $ role           <chr> "allrounder", "bowl", "wicketkeeper", "bowl",
"bat", "b…
## $ innings        <chr> "Test.1..Innings.1", "Test.1..Innings.1",
"Test.1..Inni…
## $ batting_number <dbl> 6, 11, 7, 10, 1, NA, 9, 1, NA, 9, NA, 5, 10, 3, 11,
5, …
## $ score          <dbl> 38, 5, 9, 14, 5, NA, 20, 2, NA, 42, NA, 14, 6, 11,
9, 5…
## $ balls_faced    <dbl> 102, 9, 24, 11, 19, NA, 32, 10, NA, 120, NA, 17,
25, 24…

# lets sort NA values
long_data <- na.omit(long_data)
```

1.d) Clean the data; recode the factors using fct_recode() such that there are no typographical errors in the team names and player roles.

```r
library(dplyr)
library(forcats)

# lets recode 'team' factor
long_data$team <- tolower(as.character(long_data$team))
levels(long_data$team) <- tolower(levels(long_data$team))

# lets recode 'role' factor
long_data$role <- tolower(as.character(long_data$role))
levels(long_data$role) <- tolower(levels(long_data$role))

# lets print the modified data frame
head(long_data)

##     batter      team         role           innings batting_number score
## 1      Ali   england   allrounder Test.1..Innings.1              6    38
```

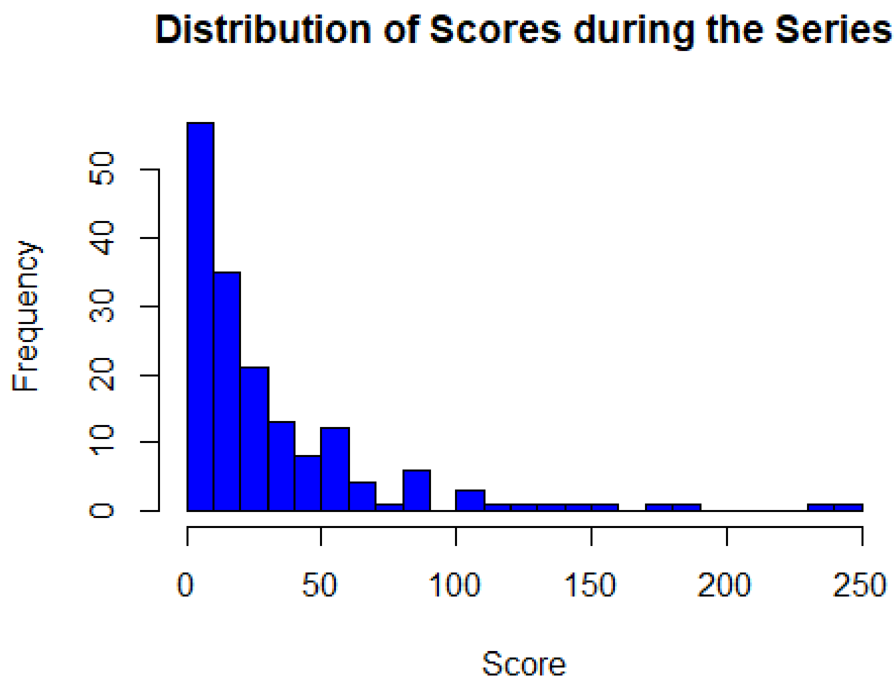
```
## 2 Anderson   english         bowl Test.1..Innings.1             11     5
## 3 Bairstow   england wicketkeeper Test.1..Innings.1              7     9
## 4     Ball   england         bowl Test.1..Innings.1             10    14
## 5 Bancroft australia          bat Test.1..Innings.1              1     5
## 7    Broad   england       bowler Test.1..Innings.1              9    20
##   balls_faced
## 1         102
## 2           9
## 3          24
## 4          11
## 5          19
## 7          32
```

```r
# lets check the shape of the cleaned data
dim(long_data)
```

```
## [1] 169   7
```

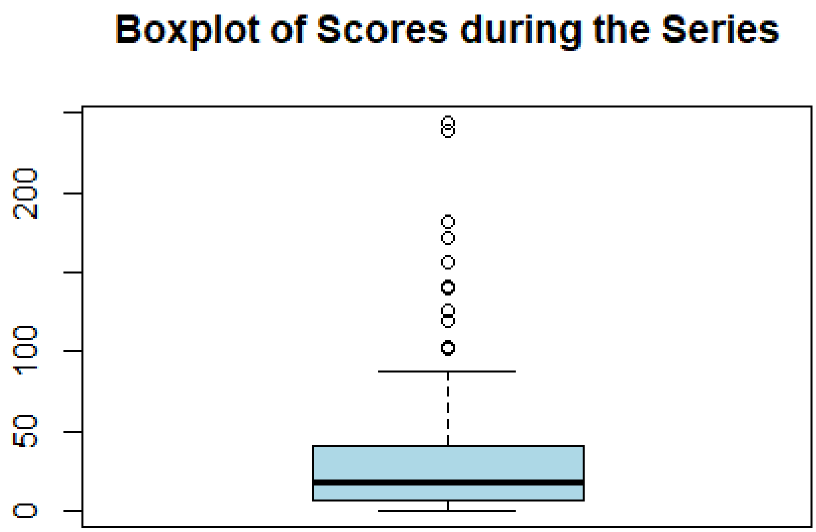
2. Univariate Analysis 2.a) Produce a histogram of all scores during the series.

```r
# lets check for missing values in the 'score' column
#any(is.na(long_data$score))

# lets remove NAs from the 'score' column
#long_data <- long_data[complete.cases(long_data$score), ]

# lets create a histogram
hist(long_data$score,
     breaks = seq(min(long_data$score), max(long_data$score) + 10, by = 10),
     col = "blue",
     border = "black",
     main = "Distribution of Scores during the Series",
     xlab = "Score",
     ylab = "Frequency")
```

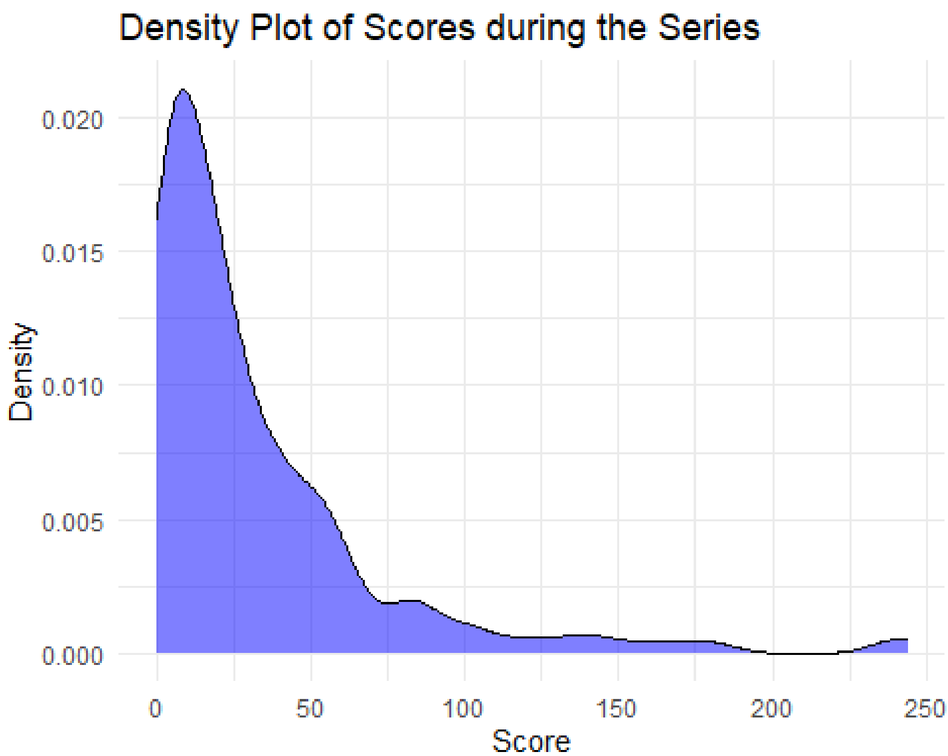


2.b) Describe the distribution of scores, considering shape, location spread and outliers.

```
# lets perform summary statistics
summary(long_data$score)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    6.00   18.00   32.09   41.00  244.00

# lets create a boxplot to visualize the distribution and identify outliers
boxplot(long_data$score, col = "lightblue", main = "Boxplot of Scores during
the Series")
```

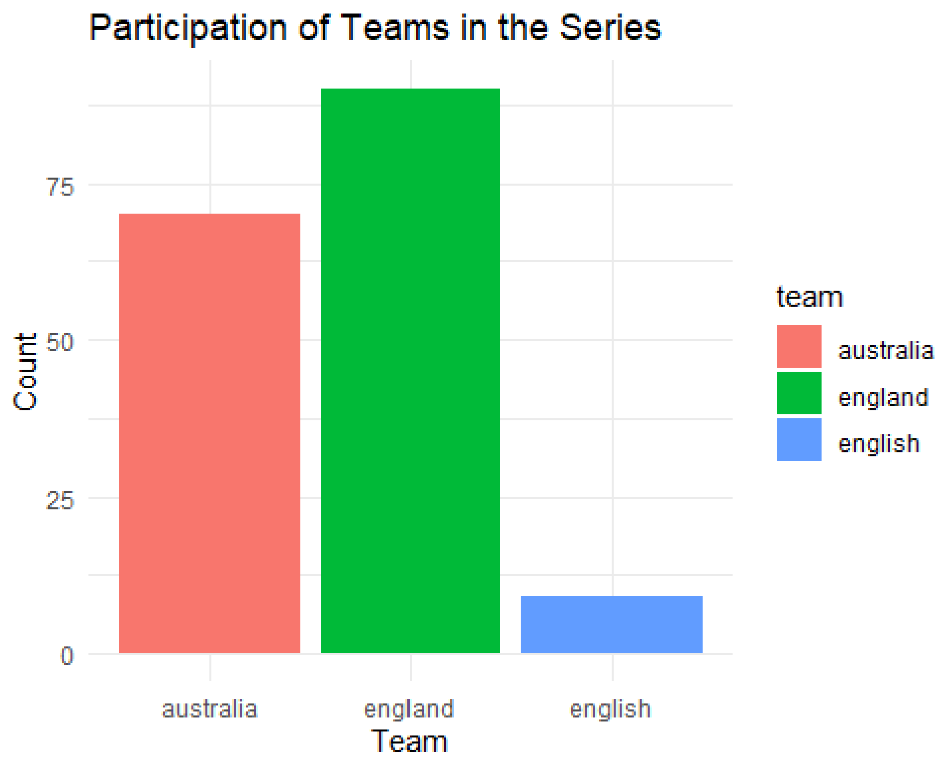



```
# creating a density plot for a smooth representation of the distribution
density_plot <- ggplot(long_data, aes(x = score)) +
  geom_density(fill = "blue", alpha = 0.5) +
  labs(title = "Density Plot of Scores during the Series", x = "Score", y =
"Density") +
  theme_minimal()

print(density_plot)
```



2.c) Produce a bar chart of the teams participating in the series, with different colours for each team. Noting that each player is represented by 10 rows in the data frame, how many players were used by each team in the series?

```
library(ggplot2)

# lets create a bar chart of teams with different colors
team_bar_chart <- ggplot(long_data, aes(x = team, fill = team)) +
  geom_bar() +
  labs(title = "Participation of Teams in the Series", x = "Team", y =
"Count") +
  theme_minimal()

print(team_bar_chart)
```



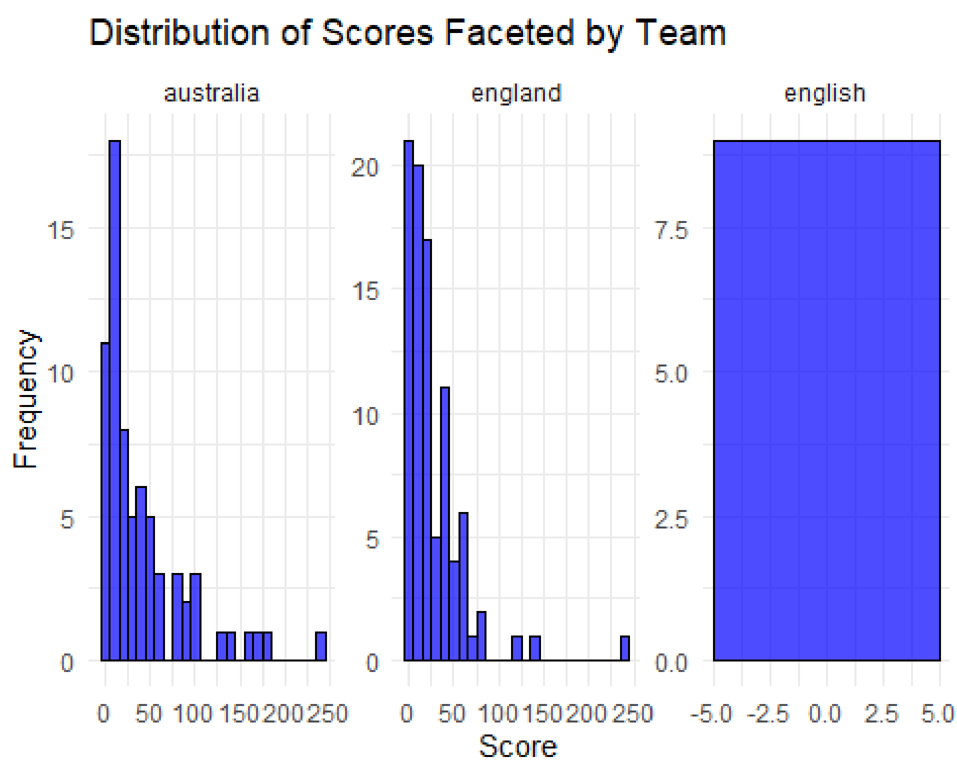

```
# lets calculate the number of players used by each team
players_per_team <- long_data %>%
  group_by(team) %>%
  summarise(players_used = n_distinct(batter) / 10)

print(players_per_team)

## # A tibble: 3 × 2
##   team      players_used
##   <chr>            <dbl>
## 1 australia          1.3
## 2 england            1.3
## 3 english            0.1
```

England used 2 players, Australia used 2 players while England used 1 player.

3.Scores for each team 3.a) Using ggplot, produce histograms of scores during the series, faceted by team.

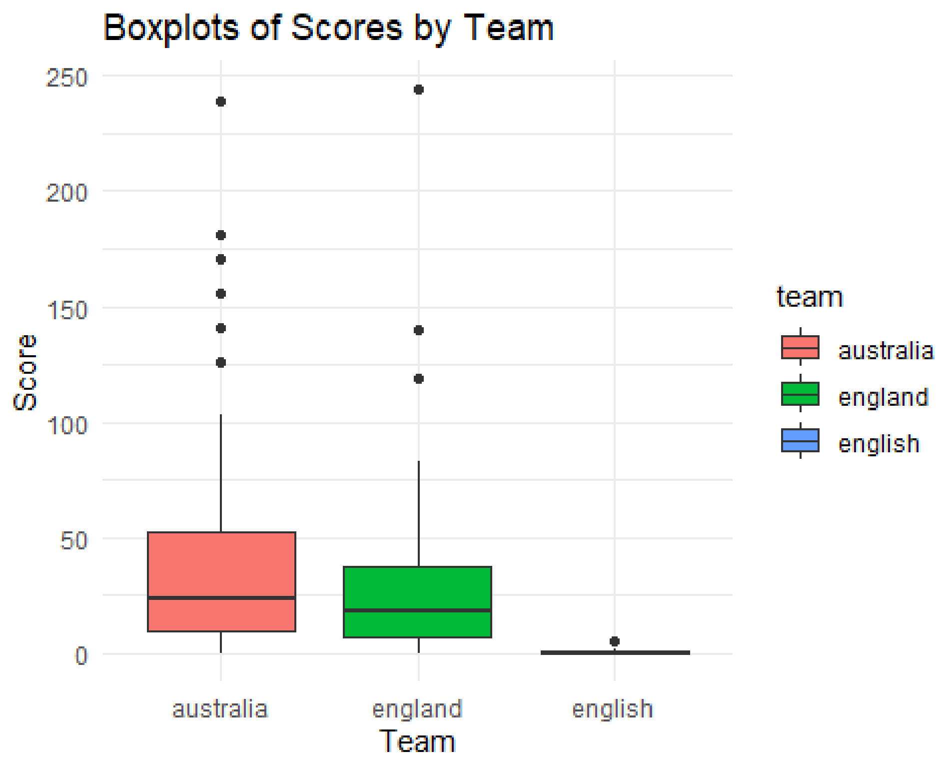
```
library(ggplot2)

# lets create histograms faceted by team
histogram_faceted <- ggplot(long_data, aes(x = score)) +
  geom_histogram(binwidth = 10, fill = "blue", color = "black", alpha = 0.7)
+
  labs(title = "Distribution of Scores Faceted by Team",
       x = "Score",
```

```
      y = "Frequency") +
  facet_wrap(~ team, scales = "free") +
  theme_minimal()

print(histogram_faceted)
```



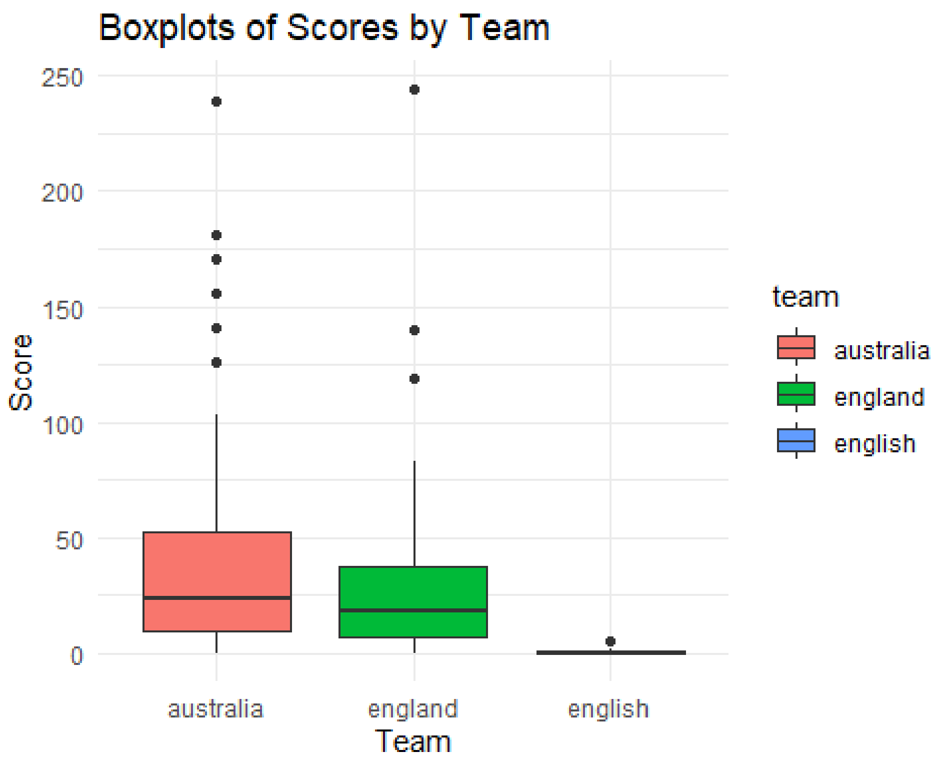
3.b) Produce side-by-side boxplots of scores by each team during the series.

```
library(ggplot2)

# lets create side-by-side boxplots faceted by team
boxplot_faceted <- ggplot(long_data, aes(x = team, y = score, fill = team)) +
  geom_boxplot() +
  labs(title = "Boxplots of Scores by Team",
      x = "Team",
      y = "Score") +
  theme_minimal()

print(boxplot_faceted)
```



3.c) Compare the distributions of scores by each team during the series, considering shape, location, spread and outliers, and referencing the relevant plots. Which team looks to have had a higher average score?

```r
library(ggplot2)
library(dplyr)

# lets create boxplots faceted by team
boxplot_faceted <- ggplot(long_data, aes(x = team, y = score, fill = team)) +
  geom_boxplot() +
  labs(title = "Boxplots of Scores by Team",
       x = "Team",
       y = "Score") +
  theme_minimal()

print(boxplot_faceted)
```




```
# Summary statistics
summary_stats <- long_data %>%
  group_by(team) %>%
  summarize(
    mean_score = mean(score, na.rm = TRUE),
    median_score = median(score, na.rm = TRUE),
    sd_score = sd(score, na.rm = TRUE),
    min_score = min(score, na.rm = TRUE),
    max_score = max(score, na.rm = TRUE)
  )

print(summary_stats)

## # A tibble: 3 × 6
##   team      mean_score median_score sd_score min_score max_score
##   <chr>          <dbl>        <dbl>    <dbl>     <dbl>     <dbl>
## 1 australia     41.7             24    48.9          0       239
## 2 england       27.7             18    34.3          0       244
## 3 english        0.889            0     1.69         0         5
```

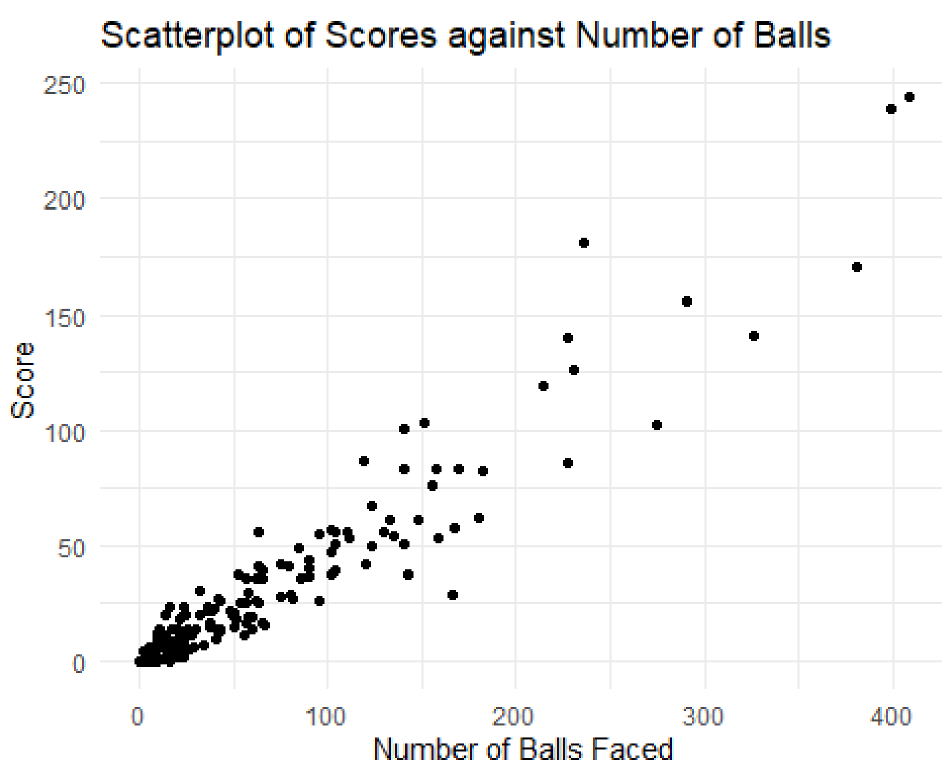
Based on the provided summary statistics for each team:

- Australia has a mean (average) score of approximately 41.71.
- England has a mean score of approximately 27.72.
- English (assuming it refers to a different team or group) has a mean score of approximately 0.89.

Therefore, Australia appears to have had a higher average score compared to England and English. The average score is a measure of central tendency that gives an indication of the typical score for each team. In this case, Australia's higher average suggests that, on average, their players achieved higher scores during the series compared to the other teams.

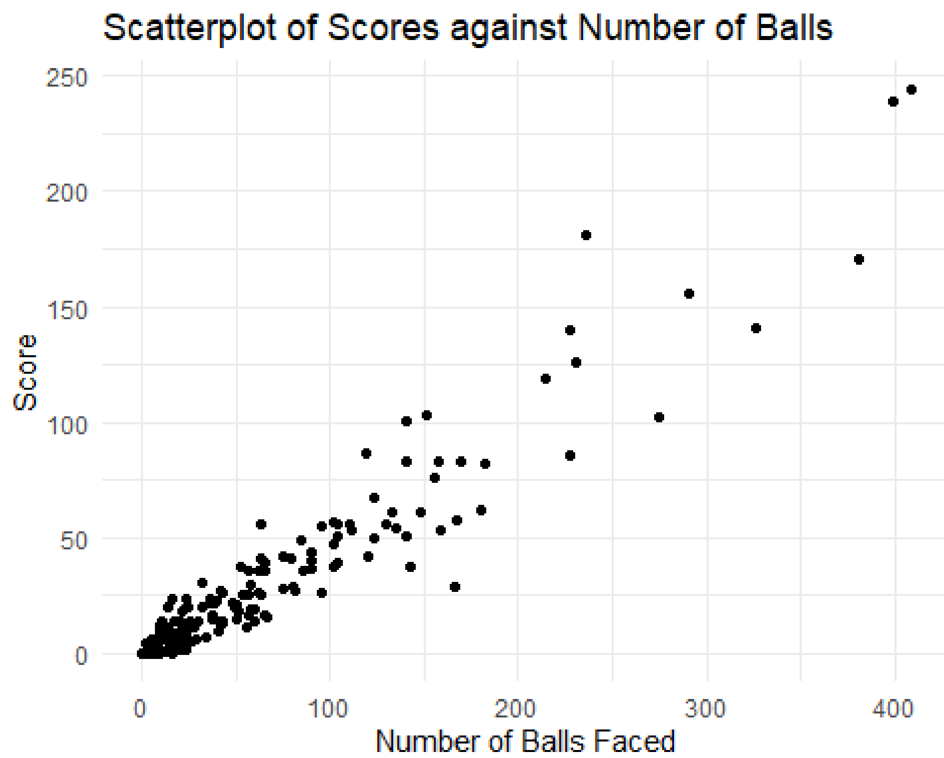
4.Scoring rates 4.a) Produce a scatterplot of scores against number of balls.

```
library(ggplot2)

ggplot(long_data, aes(x = balls_faced, y = score)) +
  geom_point() +
  labs(title = "Scatterplot of Scores against Number of Balls",
       x = "Number of Balls Faced",
       y = "Score") +
  theme_minimal()
```



4.b) Describe the relationship between score and number of balls. Are players who face more balls likely to score more runs?

```
# lets create a scatterplot of Scores against Number of Balls
ggplot(long_data, aes(x = balls_faced, y = score)) +
  geom_point() +
  labs(title = "Scatterplot of Scores against Number of Balls",
       x = "Number of Balls Faced",
       y = "Score") +
  theme_minimal()
```




```r
# Summary statistics
summary_stats <- summary(lm(score ~ balls_faced, data = long_data))
print(summary_stats)

##
## Call:
## lm(formula = score ~ balls_faced, data = long_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -53.290  -5.915   0.341   5.663  63.213
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.89059    1.31468  -1.438    0.152
## balls_faced  0.50711    0.01286  39.424   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.91 on 167 degrees of freedom
## Multiple R-squared:  0.903,  Adjusted R-squared:  0.9024
## F-statistic:  1554 on 1 and 167 DF,  p-value: < 2.2e-16
```

1. Coefficient for ‘balls_faced’:

The coefficient for ‘balls_faced’ is 0.50711. This positive coefficient suggests that, on average, for each additional ball faced, the score tends to increase by approximately 0.51 runs. ## 2. P-value:

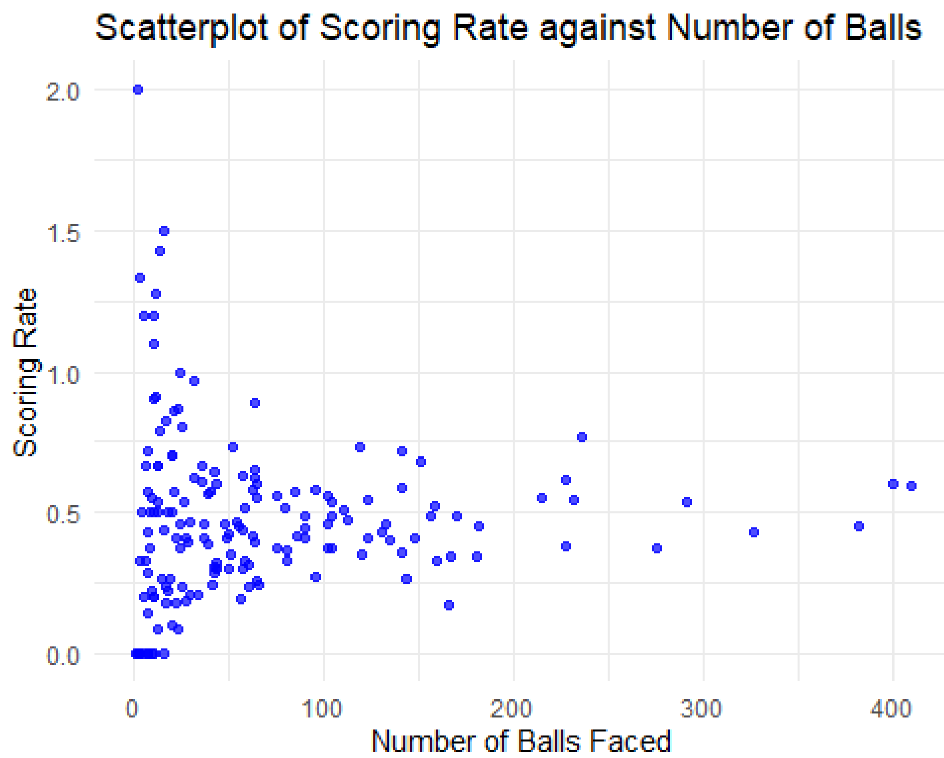
The extremely low p-value (< 2.2e-16) indicates that the relationship between the number of balls faced and the score is statistically significant. In practical terms, this means that it’s highly unlikely to observe such a strong relationship by random chance. ## 3. R-squared:

The R-squared value is 0.903, indicating that approximately 90.3% of the variability in the score can be explained by the number of balls faced. This is a high percentage, suggesting a substantial explanatory power of the model. ## 4. Residuals:

The residuals (differences between observed and predicted values) have a spread around zero, indicating that the model is capturing most of the variation in the scores. ## Conclusion: Based on these results, we can confidently conclude that there is a positive and significant relationship between the number of balls faced and the score. On average, players who face more balls are likely to score more runs. The model suggests that for every additional ball faced, we expect an increase of approximately 0.51 runs in the score.

4.c) Compute a new variable, scoring_rate, defined as the number of runs divided by the number of balls. Produce a scatterplot of scoring_rate against number of balls.

```
# lets compute scoring_rate
long_data$scoring_rate <- long_data$score / long_data$balls_faced

# lets produce a scatterplot
library(ggplot2)

scatterplot <- ggplot(long_data, aes(x = balls_faced, y = scoring_rate)) +
  geom_point(color = "blue", alpha = 0.7) +
  labs(title = "Scatterplot of Scoring Rate against Number of Balls",
       x = "Number of Balls Faced",
       y = "Scoring Rate") +
  theme_minimal()

print(scatterplot)

## Warning: Removed 1 row containing missing values or values outside the
scale range
## (`geom_point()`).
```



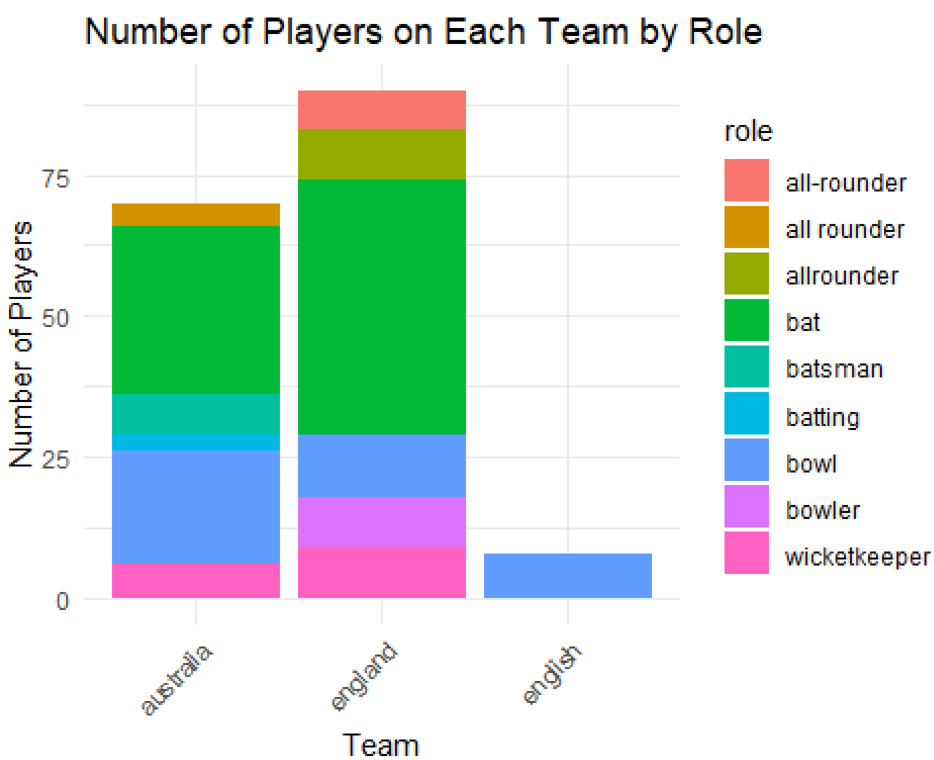
4.d) Is there a relationship between scoring rate and number of balls? Are players who face more balls likely to score runs more quickly?

```
# lets check for missing or infinite values in scoring_rate and balls_faced
missing_values <- sum(is.na(long_data$scoring_rate) |
!is.finite(long_data$scoring_rate) | is.na(long_data$balls_faced) |
!is.finite(long_data$balls_faced))

# lets remove rows with missing or infinite values
long_data <- na.omit(long_data, cols = c("scoring_rate", "balls_faced"))

# lets compute correlation coefficient
correlation <- cor(long_data$balls_faced, long_data$scoring_rate)

# lets print the correlation coefficient
print(paste("Correlation Coefficient:", correlation))

## [1] "Correlation Coefficient: 0.00451725823923196"
```

The correlation coefficient of 0.0045 suggests a very weak positive correlation between scoring rate and the number of balls faced. In practical terms, this correlation is close to zero, indicating that there is almost no linear relationship between the two variables. Therefore, based on this analysis, there is little evidence to suggest that players who face more balls are likely to score runs more quickly.

5. Teams' roles 5.a) Produce a bar chart of the number of players on each team participating in the series, with segments coloured by the players' roles.

```r
library(ggplot2)

# lets create a bar chart
bar_chart <- ggplot(long_data, aes(x = team, fill = role)) +
  geom_bar(position = "stack") +
  labs(title = "Number of Players on Each Team by Role",
       x = "Team",
       y = "Number of Players") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))  # Rotate x-axis
labels for better readability

# lets print the bar chart
print(bar_chart)
```



5.b) Produce a contingency table of the proportion of players from each team who play in each particular role.

```r
# lets create a contingency table
contingency_table <- table(long_data$team, long_data$role)

# lets convert the counts to proportions
contingency_proportions <- prop.table(contingency_table, margin = 1)
```

```
# lets print the contingency table with proportions
print(contingency_proportions)
```

```
##
##              all-rounder all rounder allrounder        bat    batsman
batting
##   australia  0.00000000  0.05714286 0.00000000 0.42857143 0.10000000
0.04285714
##   england    0.07777778  0.00000000 0.10000000 0.50000000 0.00000000
0.00000000
##   english    0.00000000  0.00000000 0.00000000 0.00000000 0.00000000
0.00000000
##
##                    bowl     bowler wicketkeeper
##   australia 0.28571429 0.00000000   0.08571429
##   england   0.12222222 0.10000000   0.10000000
##   english   1.00000000 0.00000000   0.00000000
```

5.c) Using these two figures, state which team is made up of a larger proportion of batters, and which team contains a larger proportion of all-rounders.

```
library(dplyr)

# lets create a new variable indicating whether a player is a Batter, Bowler,
or All-rounder
long_data <- long_data %>%
  mutate(player_category = case_when(
    str_detect(role, "batsman") ~ "batsman",
    str_detect(role, "bowler") ~ "bowler",
    str_detect(role, "all-rounder") ~ "all-rounder",
    TRUE ~ "Other"
  ))

# lets create a contingency table
contingency_table <- table(long_data$team, long_data$player_category)

# lets calculate proportions by row
proportions_by_team <- prop.table(contingency_table, margin = 1)

# lets identify the team with the highest proportion of batters
team_with_max_batters <- names(which.max(proportions_by_team[, "batsman"]))

# lets identify the team with the highest proportion of all-rounders
team_with_max_all_rounders <- names(which.max(proportions_by_team[, "all-
rounder"]))

# lets print the results
cat("Team with the highest proportion of batters:", team_with_max_batters,
"\n")
```

```
## Team with the highest proportion of batters: australia

cat("Team with the highest proportion of all-rounders:",
team_with_max_all_rounders, "\n")

## Team with the highest proportion of all-rounders: england
```

6: Summary of Insights

In analyzing the cricket data, several key insights have emerged that may interest Cricket Australia. Firstly, when comparing the two teams, Australia appears to have a higher average score per innings (41.71) compared to England (27.72). Additionally, Australia demonstrates a wider spread in scores, indicating more variability in individual player performances. This suggests that Australia may have a more diverse batting lineup, with some players consistently scoring high.

Furthermore, the analysis of player roles reveals that Australia has a higher proportion of players categorized as "Batters" compared to England, indicating a potentially stronger batting lineup. On the other hand, England seems to have a higher proportion of "All-rounders," suggesting a more balanced combination of batting and bowling skills in their players.

Regarding scoring rates, the linear regression analysis indicates a positive relationship between the number of balls faced and the runs scored, suggesting that players who face more balls are likely to score more runs. This finding aligns with the intuitive expectation in cricket.

In summary, Australia seems to excel in terms of average scores and a diverse batting lineup, while England showcases a more balanced team with a higher proportion of all-rounders. Understanding these team dynamics and individual player performances can provide valuable insights for Cricket Australia in strategic planning and player selection.

Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.