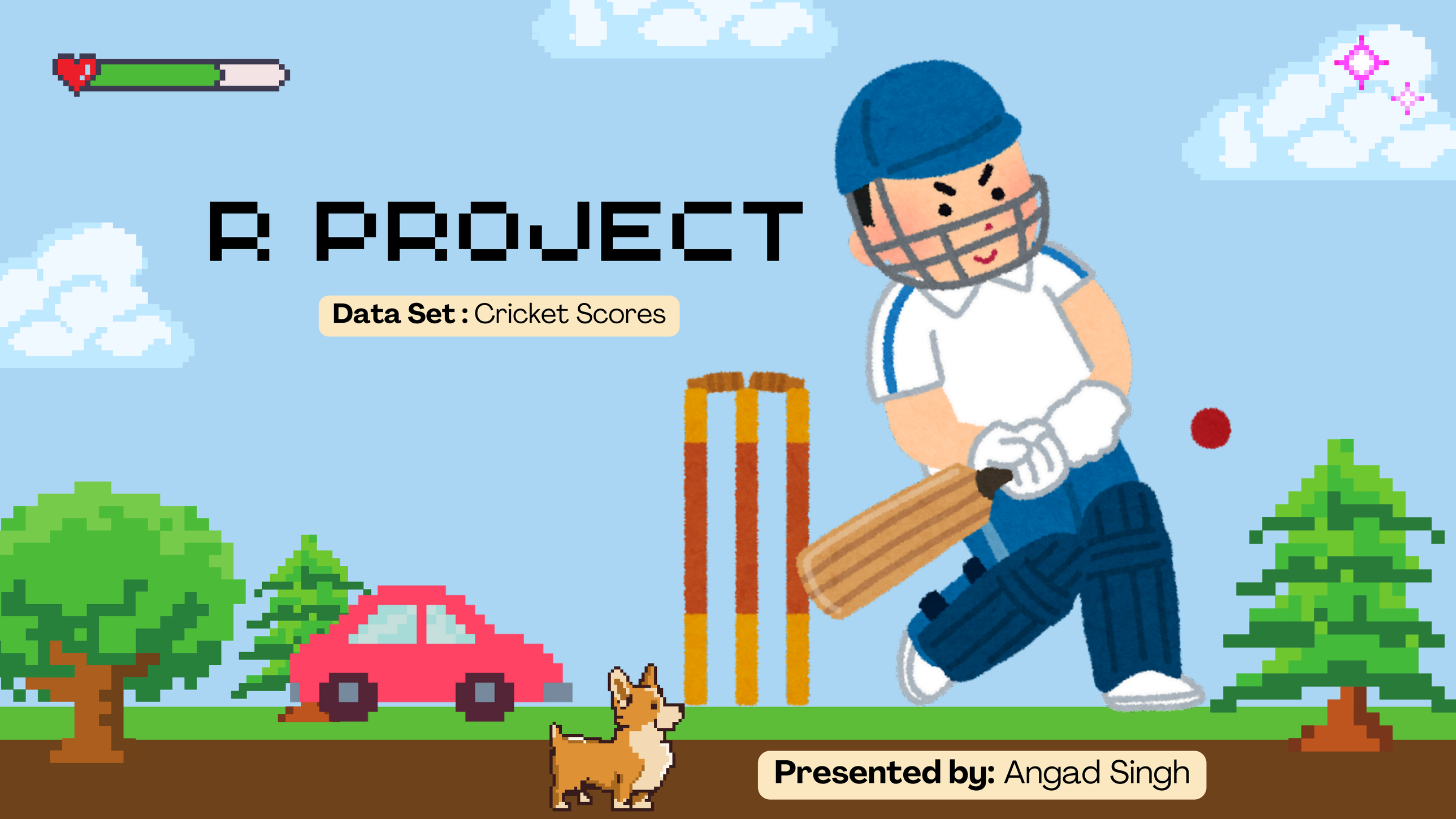
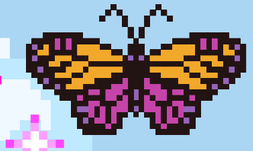
R PROJECT

Data Set : Cricket Scores



Presented by: Angad Singh

TOPIC OUTLINE



MEAN VALUES

MAX VALUES

MEAN

MODE

PERCENTILES

VARIANCE

SD

MEDIAN

COVARIANCE

CORRELATION

MIN VALUES

Project 3 R.R | Angad_Cricket

Source on Save | Run | Source

```r
min_Matches_Batted <- min(Angad_Cricket$Matches_Batted)
min_Not_Outs <- min(Angad_Cricket$Not_Outs)
min_Runs_Scored <- min(Angad_Cricket$Runs_Scored)
min_Highest_Score <- min(Angad_Cricket$Highest_Score)
min_Batting_Average <- min(Angad_Cricket$Batting_Average)
min_Balls_Faced <- min(Angad_Cricket$Balls_Faced)
min_Batting_Strike_Rate <- min(Angad_Cricket$Batting_Strike_Rate)
min_Centuries <- min(Angad_Cricket$Centuries)
min_Half_Centuries <- min(Angad_Cricket$Half_Centuries)
min_Fours <- min(Angad_Cricket$Fours)
min_Sixes <- min(Angad_Cricket$Sixes)
min_Catches_Taken <- min(Angad_Cricket$Catches_Taken)

# Print the minimum value for each column
print(paste("Minimum Matches Batted:", min_Matches_Batted))
print(paste("Minimum Not Outs:", min_Not_Outs))
print(paste("Minimum Runs Scored:", min_Runs_Scored))
print(paste("Minimum Highest Score:", min_Highest_Score))
print(paste("Minimum Batting Average:", min_Batting_Average))
print(paste("Minimum Balls Faced:", min_Balls_Faced))
print(paste("Minimum Batting Strike Rate:", min_Batting_Strike_Rate))
print(paste("Minimum Centuries:", min_Centuries))
print(paste("Minimum Half Centuries:", min_Half_Centuries))
print(paste("Minimum Fours:", min_Fours))
print(paste("Minimum Sixes:", min_Sixes))
print(paste("Minimum Catches Taken:", min_Catches_Taken))
```

21:39 (Top Level) R Script

MIN VALUES (OUTPUT)

R 4.3.2 · ~/

```
> # Print the minimum value for each column
> print(paste("Minimum Matches Batted:", min_Matches_Batted))
[1] "Minimum Matches Batted: 2"

> print(paste("Minimum Not Outs:", min_Not_Outs))
[1] "Minimum Not Outs: 0"

> print(paste("Minimum Runs Scored:", min_Runs_Scored))
[1] "Minimum Runs Scored: 0"

> print(paste("Minimum Highest Score:", min_Highest_Score))
[1] "Minimum Highest Score: 0"

> print(paste("Minimum Batting Average:", min_Batting_Average))
[1] "Minimum Batting Average: 0"

> print(paste("Minimum Balls Faced:", min_Balls_Faced))
[1] "Minimum Balls Faced: 0"

> print(paste("Minimum Batting Strike Rate:", min_Batting_Strike_Rate))
[1] "Minimum Batting Strike Rate: 0"

> print(paste("Minimum Centuries:", min_Centuries))
[1] "Minimum Centuries: 0"

> print(paste("Minimum Half Centuries:", min_Half_Centuries))
[1] "Minimum Half Centuries: 0"

> print(paste("Minimum Fours:", min_Fours))
[1] "Minimum Fours: 0"

> print(paste("Minimum Sixes:", min_Sixes))
```

MAX VALUES

Project 3 R.R | Angad_Cricket

Source on Save | Run | Source

```r
# 2. Find the maximum value for each column
max_Matches_Batted <- max(Angad_Cricket$Matches_Batted)
max_Not_Outs <- max(Angad_Cricket$Not_Outs)
max_Runs_Scored <- max(Angad_Cricket$Runs_Scored)
max_Highest_Score <- max(Angad_Cricket$Highest_Score)
max_Batting_Average <- max(Angad_Cricket$Batting_Average)
max_Balls_Faced <- max(Angad_Cricket$Balls_Faced)
max_Batting_Strike_Rate <- max(Angad_Cricket$Batting_Strike_Rate)
max_Centuries <- max(Angad_Cricket$Centuries)
max_Half_Centuries <- max(Angad_Cricket$Half_Centuries)
max_Fours <- max(Angad_Cricket$Fours)
max_Sixes <- max(Angad_Cricket$Sixes)
max_Catches_Taken <- max(Angad_Cricket$Catches_Taken)

# Print the maximum value for each column
print(paste("Maximum Matches Batted:", max_Matches_Batted))
print(paste("Maximum Not Outs:", max_Not_Outs))
print(paste("Maximum Runs Scored:", max_Runs_Scored))
print(paste("Maximum Highest Score:", max_Highest_Score))
print(paste("Maximum Batting Average:", max_Batting_Average))
print(paste("Maximum Balls Faced:", max_Balls_Faced))
print(paste("Maximum Batting Strike Rate:", max_Batting_Strike_Rate))
print(paste("Maximum Centuries:", max_Centuries))
print(paste("Maximum Half Centuries:", max_Half_Centuries))
print(paste("Maximum Fours:", max_Fours))
print(paste("Maximum Sixes:", max_Sixes))
print(paste("Maximum Catches Taken:", max_Catches_Taken))
```

15:3 (Top Level) R Script

MAX VALUES (OP)

Project 3 R.R | Angad_Cricket

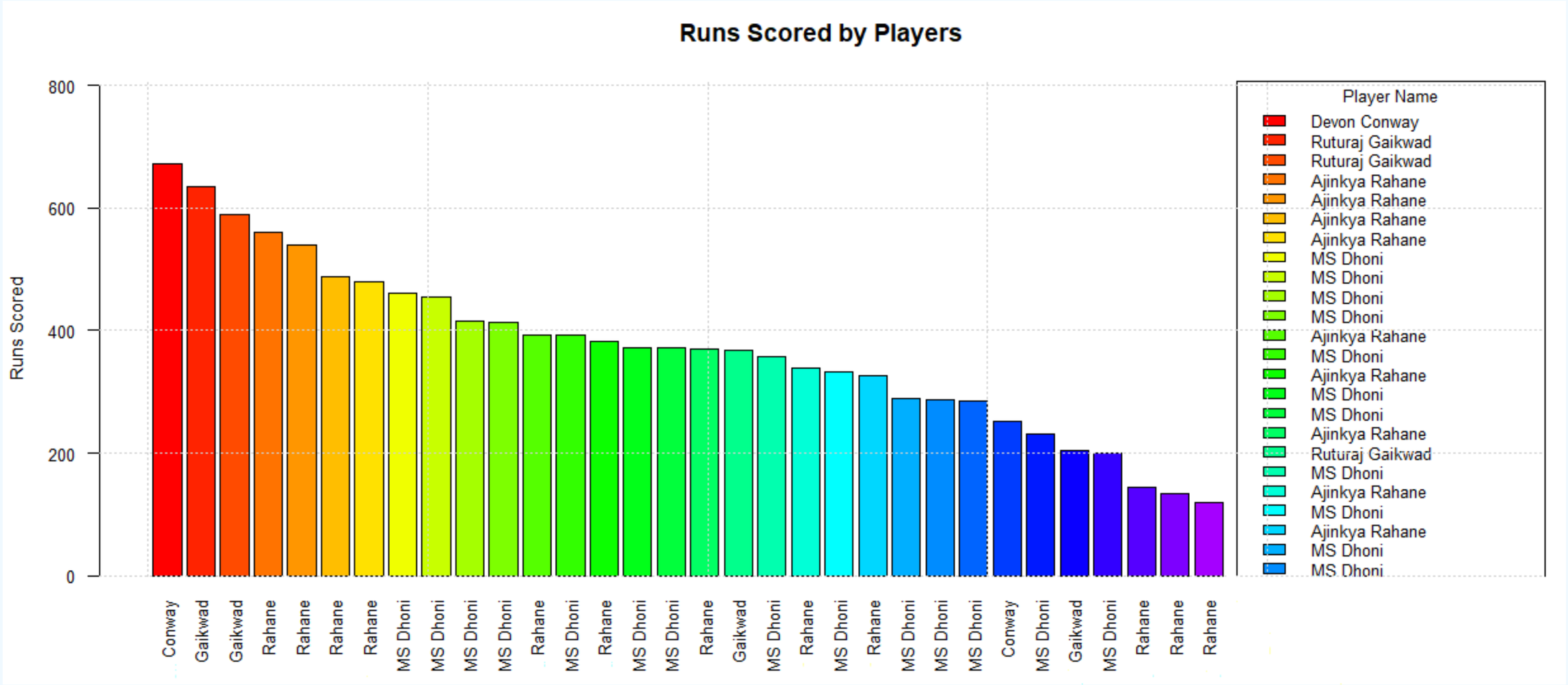
Source on Save | Run | Source

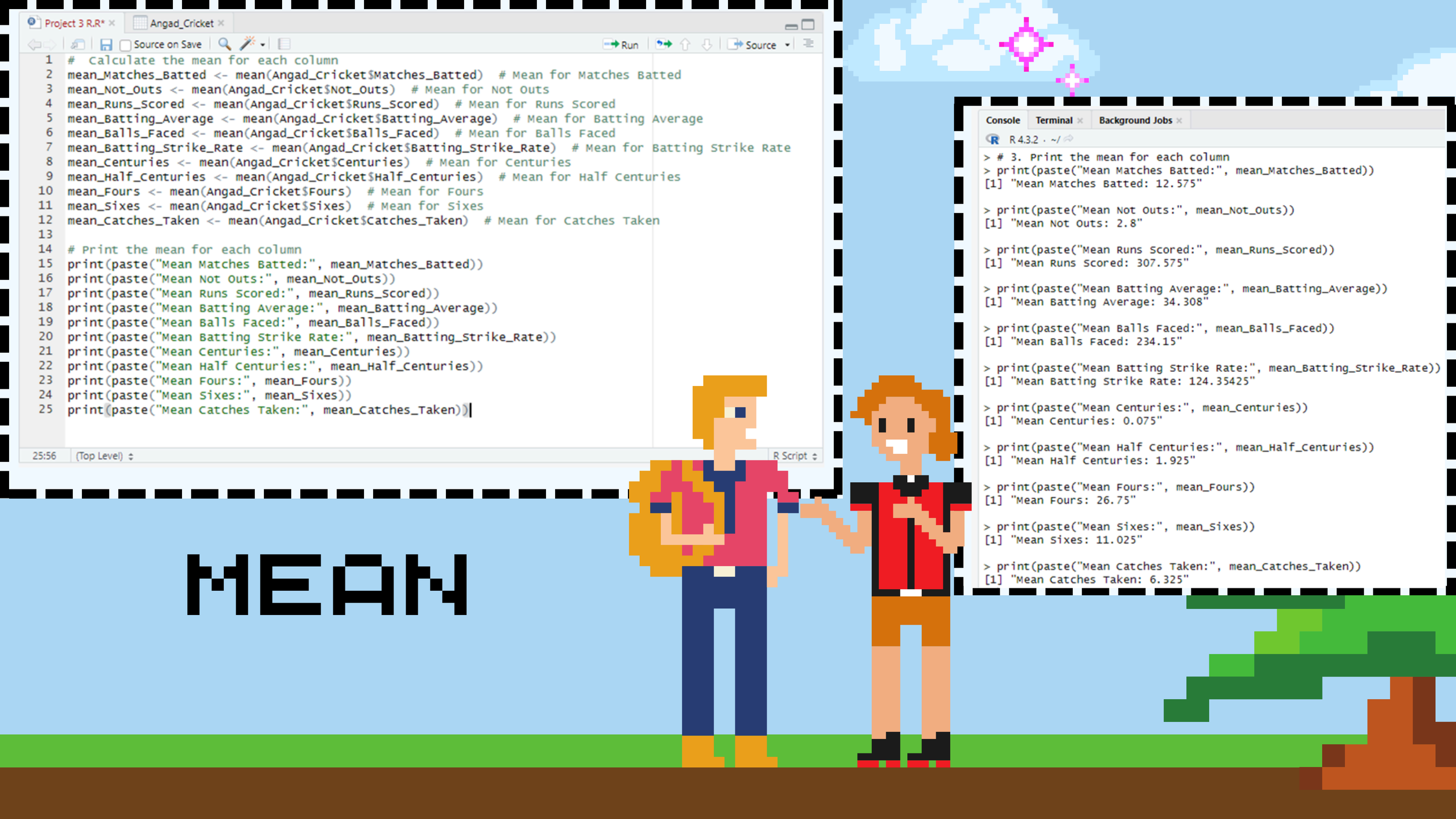
```r
# 2. Find the maximum value for each column
max_Matches_Batted <- max(Angad_Cricket$Matches_Batted)
max_Not_Outs <- max(Angad_Cricket$Not_Outs)
max_Runs_Scored <- max(Angad_Cricket$Runs_Scored)
max_Highest_Score <- max(Angad_Cricket$Highest_Score)
max_Batting_Average <- max(Angad_Cricket$Batting_Average)
max_Balls_Faced <- max(Angad_Cricket$Balls_Faced)
max_Batting_Strike_Rate <- max(Angad_Cricket$Batting_Strike_Rate)
max_Centuries <- max(Angad_Cricket$Centuries)
max_Half_Centuries <- max(Angad_Cricket$Half_Centuries)
max_Fours <- max(Angad_Cricket$Fours)
max_Sixes <- max(Angad_Cricket$Sixes)
max_Catches_Taken <- max(Angad_Cricket$Catches_Taken)

# Print the maximum value for each column
print(paste("Maximum Matches Batted:", max_Matches_Batted))
print(paste("Maximum Not Outs:", max_Not_Outs))
print(paste("Maximum Runs Scored:", max_Runs_Scored))
print(paste("Maximum Highest Score:", max_Highest_Score))
print(paste("Maximum Batting Average:", max_Batting_Average))
print(paste("Maximum Balls Faced:", max_Balls_Faced))
print(paste("Maximum Batting Strike Rate:", max_Batting_Strike_Rate))
print(paste("Maximum Centuries:", max_Centuries))
print(paste("Maximum Half Centuries:", max_Half_Centuries))
print(paste("Maximum Fours:", max_Fours))
print(paste("Maximum Sixes:", max_Sixes))
print(paste("Maximum Catches Taken:", max_Catches_Taken))
```

15:3 (Top Level) R Script

VISUALIZATION




Project 3 R.R*
Angad_Cricket
Source on Save
Run
Source
# Calculate the mean for each column
mean_Matches_Batted <- mean(Angad_Cricket$Matches_Batted) # Mean for Matches Batted
mean_Not_Outs <- mean(Angad_Cricket$Not_Outs) # Mean for Not Outs
mean_Runs_Scored <- mean(Angad_Cricket$Runs_Scored) # Mean for Runs Scored
mean_Batting_Average <- mean(Angad_Cricket$Batting_Average) # Mean for Batting Average
mean_Balls_Faced <- mean(Angad_Cricket$Balls_Faced) # Mean for Balls Faced
mean_Batting_Strike_Rate <- mean(Angad_Cricket$Batting_Strike_Rate) # Mean for Batting Strike Rate
mean_Centuries <- mean(Angad_Cricket$Centuries) # Mean for Centuries
mean_Half_Centuries <- mean(Angad_Cricket$Half_Centuries) # Mean for Half Centuries
mean_Fours <- mean(Angad_Cricket$Fours) # Mean for Fours
mean_Sixes <- mean(Angad_Cricket$Sixes) # Mean for Sixes
mean_Catches_Taken <- mean(Angad_Cricket$Catches_Taken) # Mean for Catches Taken
# Print the mean for each column
print(paste("Mean Matches Batted:", mean_Matches_Batted))
print(paste("Mean Not Outs:", mean_Not_Outs))
print(paste("Mean Runs Scored:", mean_Runs_Scored))
print(paste("Mean Batting Average:", mean_Batting_Average))
print(paste("Mean Balls Faced:", mean_Balls_Faced))
print(paste("Mean Batting Strike Rate:", mean_Batting_Strike_Rate))
print(paste("Mean Centuries:", mean_Centuries))
print(paste("Mean Half Centuries:", mean_Half_Centuries))
print(paste("Mean Fours:", mean_Fours))
print(paste("Mean Sixes:", mean_Sixes))
print(paste("Mean Catches Taken:", mean_Catches_Taken))
25:56
(Top Level)
R Script
Console
Terminal
Background Jobs
R 4.3.2 · ~/
> # 3. Print the mean for each column
> print(paste("Mean Matches Batted:", mean_Matches_Batted))
[1] "Mean Matches Batted: 12.575"
> print(paste("Mean Not Outs:", mean_Not_Outs))
[1] "Mean Not Outs: 2.8"
> print(paste("Mean Runs Scored:", mean_Runs_Scored))
[1] "Mean Runs Scored: 307.575"
> print(paste("Mean Batting Average:", mean_Batting_Average))
[1] "Mean Batting Average: 34.308"
> print(paste("Mean Balls Faced:", mean_Balls_Faced))
[1] "Mean Balls Faced: 234.15"
> print(paste("Mean Batting Strike Rate:", mean_Batting_Strike_Rate))
[1] "Mean Batting Strike Rate: 124.35425"
> print(paste("Mean Centuries:", mean_Centuries))
[1] "Mean Centuries: 0.075"
> print(paste("Mean Half Centuries:", mean_Half_Centuries))
[1] "Mean Half Centuries: 1.925"
> print(paste("Mean Fours:", mean_Fours))
[1] "Mean Fours: 26.75"
> print(paste("Mean Sixes:", mean_Sixes))
[1] "Mean Sixes: 11.025"
> print(paste("Mean Catches Taken:", mean_Catches_Taken))
[1] "Mean Catches Taken: 6.325"
MEAN

Project 3 R.R | Angad_Cricket

Source on Save | Run | Source

```r
# Find the median value for each column
median_Matches_Batted <- median(Angad_Cricket$Matches_Batted)
median_Not_Outs <- median(Angad_Cricket$Not_Outs)
median_Runs_Scored <- median(Angad_Cricket$Runs_Scored)
median_Highest_Score <- median(Angad_Cricket$Highest_Score)
median_Batting_Average <- median(Angad_Cricket$Batting_Average)
median_Balls_Faced <- median(Angad_Cricket$Balls_Faced)
median_Batting_Strike_Rate <- median(Angad_Cricket$Batting_Strike_Rate)
median_Centuries <- median(Angad_Cricket$Centuries)
median_Half_Centuries <- median(Angad_Cricket$Half_Centuries)
median_Fours <- median(Angad_Cricket$Fours)
median_Sixes <- median(Angad_Cricket$Sixes)
median_Catches_Taken <- median(Angad_Cricket$Catches_Taken)

# Print the median value for each column
print(paste("Median Matches Batted:", median_Matches_Batted))
print(paste("Median Not Outs:", median_Not_Outs))
print(paste("Median Runs Scored:", median_Runs_Scored))
print(paste("Median Highest Score:", median_Highest_Score))
print(paste("Median Batting Average:", median_Batting_Average))
print(paste("Median Balls Faced:", median_Balls_Faced))
print(paste("Median Batting Strike Rate:", median_Batting_Strike_Rate))
print(paste("Median Centuries:", median_Centuries))
print(paste("Median Half Centuries:", median_Half_Centuries))
print(paste("Median Fours:", median_Fours))
print(paste("Median Sixes:", median_Sixes))
print(paste("Median Catches Taken:", median_Catches_Taken))
```

27:60 (Top Level) R Scrip

MEDIAN

Console | Terminal | Background Jobs

R 4.3.2 · ~/

```
> # Print the median value for each column
> print(paste("Median Matches Batted:", median_Matches_Batted))
[1] "Median Matches Batted: 14"

> print(paste("Median Not Outs:", median_Not_Outs))
[1] "Median Not Outs: 2"

> print(paste("Median Runs Scored:", median_Runs_Scored))
[1] "Median Runs Scored: 335.5"

> print(paste("Median Highest Score:", median_Highest_Score))
[1] "Median Highest Score: NA"

> print(paste("Median Batting Average:", median_Batting_Average))
[1] "Median Batting Average: 32.24"

> print(paste("Median Balls Faced:", median_Balls_Faced))
[1] "Median Balls Faced: 250"

> print(paste("Median Batting Strike Rate:", median_Batting_Strike_Rate))
[1] "Median Batting Strike Rate: 126.55"

> print(paste("Median Centuries:", median_Centuries))
[1] "Median Centuries: 0"

> print(paste("Median Half Centuries:", median_Half_Centuries))
[1] "Median Half Centuries: 2"
```

Project 3 R.R | Angad_Cricket

Source on Save

```r
# Custom function to find the mode
get_mode <- function(x) {
  uniq_x <- unique(x)
  tab <- tabulate(match(x, uniq_x))
  mode_index <- which.max(tab)
  return(uniq_x[mode_index])
}

# Find the mode for a specific column
mode_Matches_Batted <- get_mode(Angad_Cricket$Matches_Batted)
mode_Not_Outs <- get_mode(Angad_Cricket$Not_Outs)
mode_Runs_Scored <- get_mode(Angad_Cricket$Runs_Scored)

# Print the mode for each column
print(paste("Mode Matches Batted:", mode_Matches_Batted))
print(paste("Mode Not Outs:", mode_Not_Outs))
print(paste("Mode Runs Scored:", mode_Runs_Scored))
```

MODE

```
> # Print the mode for each column
> print(paste("Mode Matches Batted:", mode_Matches_Batted))
[1] "Mode Matches Batted: 16"

> print(paste("Mode Not Outs:", mode_Not_Outs))
[1] "Mode Not Outs: 0"

> print(paste("Mode Runs Scored:", mode_Runs_Scored))
[1] "Mode Runs Scored: 61"
```

PERCENTILE

Project 3 R.R* | Angad_Cricket

Source on Save | Run | Source

```
1 quantile(Angad_Cricket$Runs_Scored)
2 quantile(Angad_Cricket$Matches_Batted)
3 quantile(Angad_Cricket$Not_Outs)
4 quantile(Angad_Cricket$Batting_Strike_Rate)
5 quantile(Angad_Cricket$Balls_Faced)
6 quantile(Angad_Cricket$Fours)
7 quantile(Angad_Cricket$Sixes)
8
```

8:1 (Top Level)

```
> quantile(Angad_Cricket$Runs_Scored)
    0%    25%    50%    75%   100%
  0.00 141.25 335.50 414.50 672.00

> quantile(Angad_Cricket$Matches_Batted)
   0%   25%   50%   75%  100%
 2.00  9.75 14.00 16.00 19.00

> quantile(Angad_Cricket$Not_Outs)
   0%   25%   50%   75%  100%
 0.00  1.00  2.00  4.25 10.00

> quantile(Angad_Cricket$Batting_Strike_Rate)
      0%      25%      50%      75%     100%
  0.0000 116.9250 126.5500 136.9675 182.4600

> quantile(Angad_Cricket$Balls_Faced)
    0%    25%    50%    75%   100%
  0.00 126.75 250.00 309.25 481.00

> quantile(Angad_Cricket$Fours)
  0%  25%  50%  75% 100%
 0.0 13.5 23.0 38.0 77.0

> quantile(Angad_Cricket$Sixes)
   0%   25%   50%   75%  100%
 0.00  3.75  9.50 16.00 30.00
>
```

VARIANCE

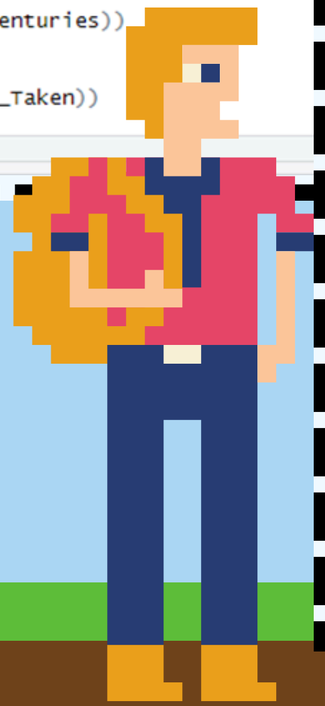
Project 3 R.R | Angad_Cricket

Source on Save | Run | Source

```r
# Calculate the variance for each column
variance_Matches_Batted <- var(Angad_Cricket$Matches_Batted)
variance_Not_Outs <- var(Angad_Cricket$Not_Outs)
variance_Runs_Scored <- var(Angad_Cricket$Runs_Scored)
variance_Highest_Score <- var(Angad_Cricket$Highest_Score)
variance_Batting_Average <- var(Angad_Cricket$Batting_Average)
variance_Balls_Faced <- var(Angad_Cricket$Balls_Faced)
variance_Batting_Strike_Rate <- var(Angad_Cricket$Batting_Strike_Rate)
variance_Centuries <- var(Angad_Cricket$Centuries)
variance_Half_Centuries <- var(Angad_Cricket$Half_Centuries)
variance_Fours <- var(Angad_Cricket$Fours)
variance_Sixes <- var(Angad_Cricket$Sixes)
variance_Catches_Taken <- var(Angad_Cricket$Catches_Taken)

#  Print the variance for each column
print(paste("Variance Matches Batted:", variance_Matches_Batted))
print(paste("Variance Not Outs:", variance_Not_Outs))
print(paste("Variance Runs Scored:", variance_Runs_Scored))
print(paste("Variance Highest Score:", variance_Highest_Score))
print(paste("Variance Batting Average:", variance_Batting_Average))
print(paste("Variance Balls Faced:", variance_Balls_Faced))
print(paste("Variance Batting Strike Rate:", variance_Batting_Strike_Rate))
print(paste("Variance Centuries:", variance_Centuries))
print(paste("Variance Half Centuries:", variance_Half_Centuries))
print(paste("Variance Fours:", variance_Fours))
print(paste("Variance Sixes:", variance_Sixes))
print(paste("Variance Catches Taken:", variance_Catches_Taken))
```

1:3 (Top Level)



Console | Terminal | Background Jobs

R 4.3.2 · ~/

```
> #  Print the variance for each column
> print(paste("Variance Matches Batted:", variance_Matches_Batted))
[1] "Variance Matches Batted: 25.5326923076923"

> print(paste("Variance Not Outs:", variance_Not_Outs))
[1] "Variance Not Outs: 7.13846153846154"

> print(paste("Variance Runs Scored:", variance_Runs_Scored))
[1] "Variance Runs Scored: 32290.3532051282"

> print(paste("Variance Highest Score:", variance_Highest_Score))
[1] "Variance Highest Score: NA"

> print(paste("Variance Batting Average:", variance_Batting_Average))
[1] "Variance Batting Average: 309.024744615385"

> print(paste("Variance Balls Faced:", variance_Balls_Faced))
[1] "Variance Balls Faced: 17580.6948717949"

> print(paste("Variance Batting Strike Rate:", variance_Batting_Strike_Rate))
[1] "Variance Batting Strike Rate: 1019.04663532051"

> print(paste("Variance Centuries:", variance_Centuries))
[1] "Variance Centuries: 0.0711538461538462"

> print(paste("Variance Half Centuries:", variance_Half_Centuries))
[1] "Variance Half Centuries: 2.68653846153846"

> print(paste("Variance Fours:", variance_Fours))
[1] "Variance Fours: 365.423076923077"
```

STANDARD DEVIATION

Project 3 R.R | Angad_Cricket

Source on Save | Run | Source

```r
# Calculate the standard deviation for each column
sd_Matches_Batted <- sd(Angad_Cricket$Matches_Batted)
sd_Not_Outs <- sd(Angad_Cricket$Not_Outs)
sd_Runs_Scored <- sd(Angad_Cricket$Runs_Scored)
sd_Highest_Score <- sd(Angad_Cricket$Highest_Score)
sd_Batting_Average <- sd(Angad_Cricket$Batting_Average)
sd_Balls_Faced <- sd(Angad_Cricket$Balls_Faced)
sd_Batting_Strike_Rate <- sd(Angad_Cricket$Batting_Strike_Rate)
sd_Centuries <- sd(Angad_Cricket$Centuries)
sd_Half_Centuries <- sd(Angad_Cricket$Half_Centuries)
sd_Fours <- sd(Angad_Cricket$Fours)
sd_Sixes <- sd(Angad_Cricket$Sixes)
sd_Catches_Taken <- sd(Angad_Cricket$Catches_Taken)

# Print the standard deviation for each column
print(paste("Standard Deviation Matches Batted:", sd_Matches_Batted))
print(paste("Standard Deviation Not Outs:", sd_Not_Outs))
print(paste("Standard Deviation Runs Scored:", sd_Runs_Scored))
print(paste("Standard Deviation Highest Score:", sd_Highest_Score))
print(paste("Standard Deviation Batting Average:", sd_Batting_Average))
print(paste("Standard Deviation Balls Faced:", sd_Balls_Faced))
print(paste("Standard Deviation Batting Strike Rate:", sd_Batting_Strike_Rate))
print(paste("Standard Deviation Centuries:", sd_Centuries))
print(paste("Standard Deviation Half Centuries:", sd_Half_Centuries))
print(paste("Standard Deviation Fours:", sd_Fours))
print(paste("Standard Deviation Sixes:", sd_Sixes))
print(paste("Standard Deviation Catches Taken:", sd_Catches_Taken))
```

27:68 (Top Level) R Script

Console | Terminal | Background Jobs

R 4.3.2 · ~/

```
> # Print the standard deviation for each column
> print(paste("Standard Deviation Matches Batted:", sd_Matches_Batted))
[1] "Standard Deviation Matches Batted: 5.05298845315248"

> print(paste("Standard Deviation Not Outs:", sd_Not_Outs))
[1] "Standard Deviation Not Outs: 2.6717899502883"

> print(paste("Standard Deviation Runs Scored:", sd_Runs_Scored))
[1] "Standard Deviation Runs Scored: 179.695167450681"

> print(paste("Standard Deviation Highest Score:", sd_Highest_Score))
[1] "Standard Deviation Highest Score: NA"

> print(paste("Standard Deviation Batting Average:", sd_Batting_Average))
[1] "Standard Deviation Batting Average: 17.5790996531502"

> print(paste("Standard Deviation Balls Faced:", sd_Balls_Faced))
[1] "Standard Deviation Balls Faced: 132.592212711738"

> print(paste("Standard Deviation Batting Strike Rate:", sd_Batting_Strike
[1] "Standard Deviation Batting Strike Rate: 31.9225098530882"

> print(paste("Standard Deviation Centuries:", sd_Centuries))
[1] "Standard Deviation Centuries: 0.266746782836918"

> print(paste("Standard Deviation Half Centuries:", sd_Half_Centuries))
[1] "Standard Deviation Half Centuries: 1.63906633835805"
```

COVARIANCE

Project 3 R.R* | Angad_Cricket

Source on Save | Run | Source

```r
# Calculate the covariance between specific pairs of variables
cov_Matches_Batted_Runs_Scored <- cov(Angad_Cricket$Matches_Batted, Angad_Cricket$Runs_Scored)

# Print the covariance between specific pairs of variables
print(paste("Covariance between Matches Batted and Runs Scored:", cov_Matches_Batted_Runs_Scored))


# Calculate the covariance between specific pairs of variables
cov_Matches_Batted_Runs_Scored <- cov(Angad_Cricket$Balls_Faced, Angad_Cricket$Batting_Average)

# Print the covariance between specific pairs of variables
print(paste("Covariance between Balls Faced and Batting Average:", cov_Matches_Batted_Runs_Scored))
```

```
> print(paste("Covariance between Matches Batted and Runs Scored:", cov_Matches_Batted_Ru ..
ED]
[1] "Covariance between Matches Batted and Runs Scored: 686.455769230769"

> # Calculate the covariance between specific pairs of variables
> cov_Matches_Batted_Runs_Scored <- cov(Angad_Cricket$Balls_Faced, Angad_Cricket$Batt .... |

> # Print the covariance between specific pairs of variables
> print(paste("Covariance between Balls Faced and Batting Average:", cov_Matches_Batted_R ..
ED]
[1] "Covariance between Balls Faced and Batting Average: 1380.92876923077"
```

CORRELATION

Project 3 R.R* | Angad_Cricket

Source on Save | Run | Source

```r
# Calculate the correlation between specific pairs of variables
cor_Matches_Batted_Runs_Scored <- cor(Angad_Cricket$Matches_Batted, Angad_Cricket$Runs_Scored)
cor_Not_Outs_Runs_Scored <- cor(Angad_Cricket$Not_Outs, Angad_Cricket$Runs_Scored)

# Print the correlation between specific pairs of variables
print(paste("Correlation between Matches Batted and Runs Scored:", cor_Matches_Batted_Runs_Scored))
print(paste("Correlation between Not Outs and Runs Scored:", cor_Not_Outs_Runs_Scored))
```

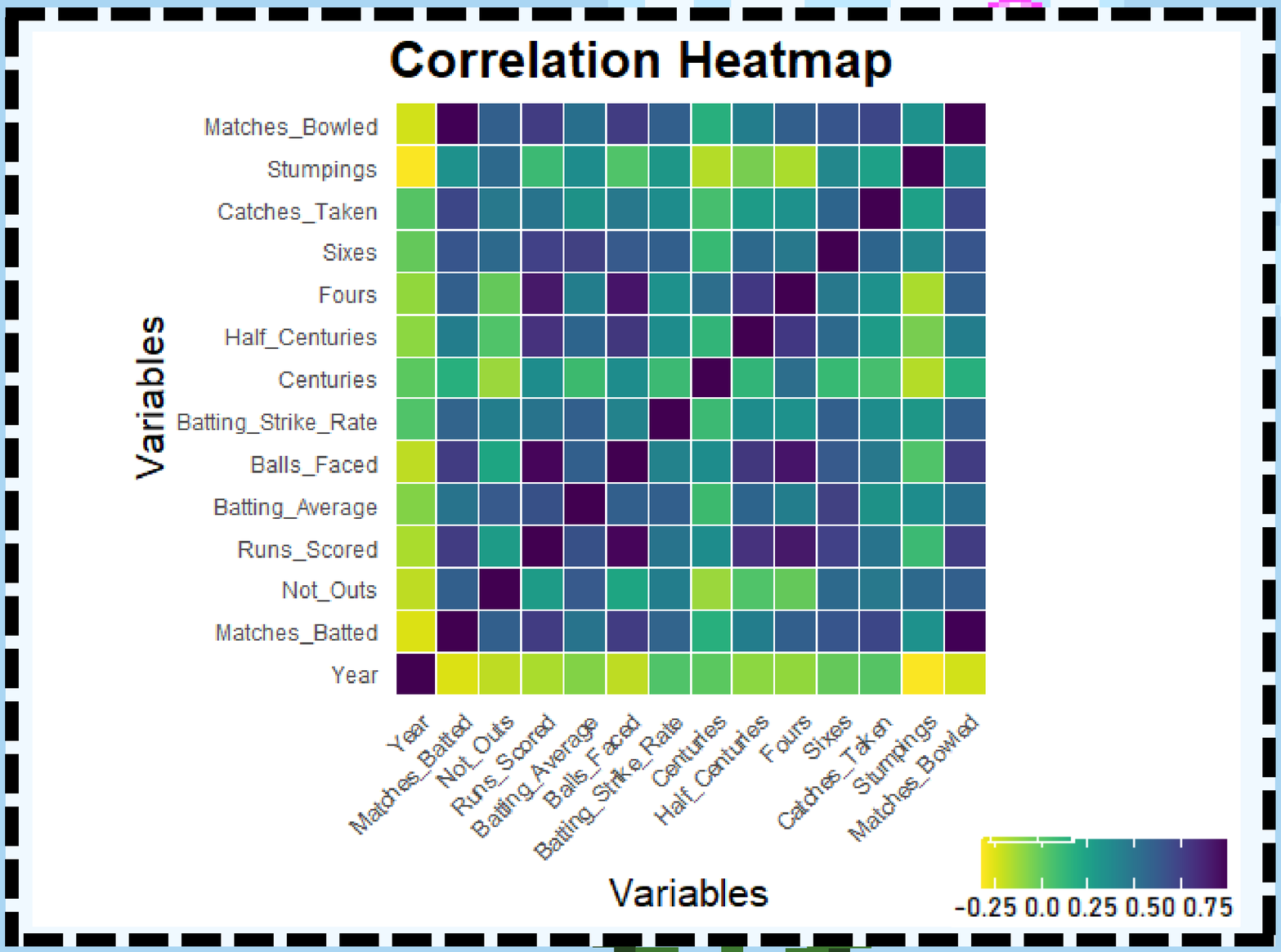
```
> # Calculate the correlation between specific pairs of variables
> cor_Matches_Batted_Runs_Scored <- cor(Angad_Cricket$Matches_Batted, Angad_Cricket$ ....

> cor_Not_Outs_Runs_Scored <- cor(Angad_Cricket$Not_Outs, Angad_Cricket$Runs_Scored)

> # Add more pairs as needed...
>
> # Print the correlation between specific pairs of variables
> print(paste("Correlation between Matches Batted and ..." ... [TRUNCATED]
[1] "Correlation between Matches Batted and Runs Scored: 0.756010549512153"

> print(paste("Correlation between Not Outs and Runs Scored:", cor_Not_Outs_Runs_Scored))
[1] "Correlation between Not Outs and Runs Scored: 0.263434458306537"
```

CORRELATION (VIS)



EXPLAINATION

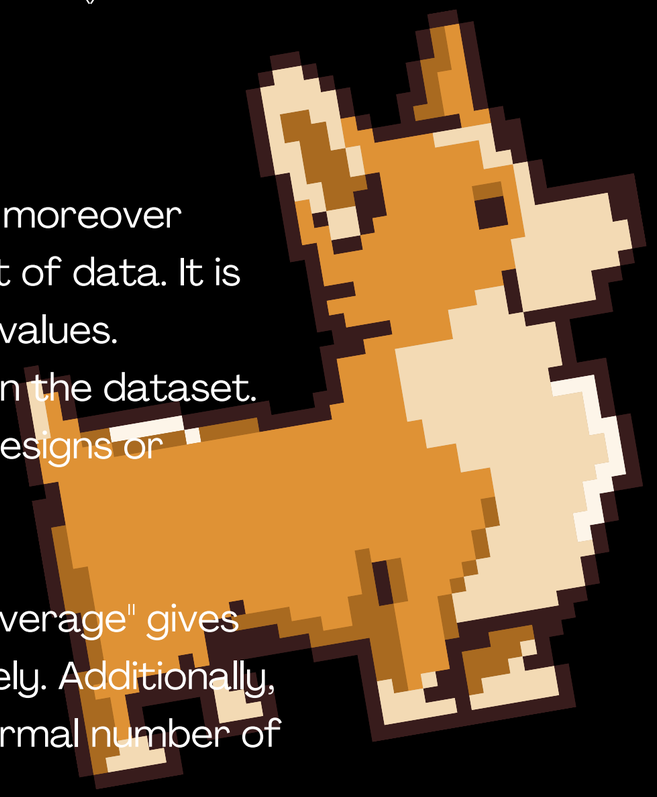
a) Mean

1. Importing the dataset:

The primary step is to consequence the dataset into R. Bringing in data allows you to analyze and control it utilizing different measurable strategies and capacities in R. In this case, the dataset is accepted to be put away in a CSV (Comma Separated Values) file, which could be a common arrange for tabular information. The `read.csv()` work is utilized to perused the dataset from the record and stack it into R as a information outline.

2. Calculating the mean for each column:

After bringing in the dataset, the another step is to calculate the mean for each column. The mean, moreover known as the normal, may be a degree of central inclination that speaks to the normal value of a set of data. It is calculated by summing up all the values in a column and after that dividing by the overall number of values. Calculating the mean for each column gives experiences into the normal value of each variable within the dataset. This data is valuable for understanding the central inclination of the information and distinguishing designs or patterns.



For case, in a cricket dataset, calculating the mean for factors such as "Runs_Scored" or "Batting_Average" gives data almost the normal number of runs scored or the normal batting execution of players, separately. Additionally, calculating the mean for factors like "Matches_Batted" or "Catches_Taken" gives insights into the normal number of matches played or the normal number of catches taken by players, separately.

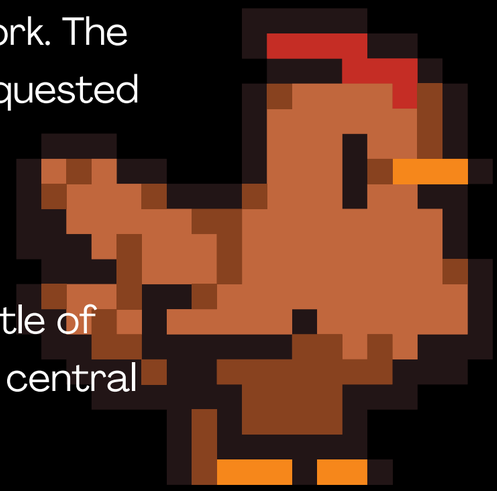
Generally, calculating the mean for each column is an fundamental step in information investigation because it makes a difference in summarizing the information and understanding the typical values of diverse factors within the dataset.

Median

This code calculates the median value for each column within the dataset `Angad_Cricket` and after that prints out the comes about. Let's break down each step and examine its centrality:

1. Importing the dataset:
The code starts by importing the dataset `Angad_Cricket` into R. This is often accomplished utilizing the `read.csv()` work, which peruses information from a CSV record and makes a information outline in R.

2. Finding the median for each column:
After importing the dataset, the code calculates the median value for each column utilizing the `median()` work. The middle could be a measure of central inclination that represents the center value of a dataset when it is requested from smallest to largest.



3. Printing the median value for each column:
Finally, the code prints out the calculated median value for each column. Each print explanation shows the title of the column along with its comparing median value. This step is significant for visualizing and interpreting the central inclination of each variable within the dataset.

In summary, the given code effectively calculates and prints the middle value for each column within the dataset `Angad_Cricket`, giving important experiences into the central position of the information distribution for each variable. Altering the record way and column names suitably will empower you to utilize this code along with your particular dataset.

Standard Deviation

This code calculates the standard deviation for each column in the dataset Angad_Cricket and then prints out the results. Let's break down each step and discuss its significance:

1. Importing the dataset:

The code starts by importing the dataset Angad_Cricket into R. This is achieved using the read.csv() function, which reads data from a CSV file and creates a data frame in R.

2. Calculating the standard deviation for each column:

After importing the dataset, the code calculates the standard deviation for each column using the sd() function. The standard deviation is a measure of the amount of variation or dispersion in a set of values. It quantifies how much the values in a dataset differ from the mean value.

3. Printing the standard deviation for each column:

Finally, the code prints out the calculated standard deviation for each column. Each print statement displays the name of the column along with its corresponding standard deviation value. This step is crucial for understanding the variability or spread of values within each variable in the dataset.

COV

This code calculates the correlation between specific sets of variables within the dataset Angad_Cricket and prints out the comes about. Let's examine each part:

Calculating the correlation between specific pairs of variables:
The code calculates the relationship between each indicated match of factors utilizing the cor() work. Relationship measures the quality and course of the straight relationship between two factors. A relationship value near to 1 shows a solid positive direct relationship, a esteem near to -1 shows a solid negative straight relationship, and a esteem near to demonstrates no straight relationship.

Printing the relationship between particular sets of factors:
After calculating the relationship for each combine of factors, the code prints out the relationship coefficients. Each print explanation shows the name of the variables within the combine at the side their comparing relationship coefficient. This step is essential for visualizing and deciphering the connections between different factors within the dataset.

Additional pairs:
we willinclude more sets of factors by rehashing the same handle. Basically calculate the relationship using the cor() function for each extra combine of factors and print out the comes about.

In outline, this code proficiently calculates and prints the relationship between particular sets of factors within the dataset Angad_Cricket, giving profitable bits of knowledge into the connections between distinctive factors. Altering the column names fittingly will empower you to analyze the relationship between any desired sets of variables in our dataset.

Variance

This code calculates the variance for each column within the dataset `Angad_Cricket` and after that prints out the results. Let's break down each step and talk about its significance:

1. Importing the dataset:

The code starts by importing the dataset `Angad_Cricket` into R. Typically accomplished using the `read.csv()` work, which reads data from a CSV record and makes a information outline in R.

2. Calculating the variance for each column:

After bringing in the dataset, the code calculates the fluctuation for each column utilizing the `var()` work. Change measures the spread or scattering of information focuses around the cruel. It measures how much person perceptions deviate from the mean value.

3. Printing the variance for each column:

At last, the code prints out the calculated change for each column. Each print articulation shows the title of the column along side its comparing variance value. This step is significant for understanding the changeability or spread of values inside each variable within the dataset.

In outline, the given code efficiently calculates and prints the fluctuation for each column within the dataset `Angad_Cricket`, giving profitable experiences into the spread of values for each variable. Altering the record way and column names suitably will empower you to utilize this code together with your particular dataset.

Mode

1. Custom function to find the mode:

The code defines a custom function named get_mode(). This function takes a vector as input and returns its mode. To discover the mode, the function to begin with extracts the one of a kind components from the input vector. At that point, it makes a frequency table for these special components and recognizes the element(s) with the most elevated frequency, which speak to the mode(s) of the input vector.

2. Importing the dataset:

The dataset, named Angad_Cricket, is imported into R utilizing the read.csv() work. This work reads information from a CSV record and makes a information outline in R. Alterations ought to be made to the record way and title to coordinate the area and title of the dataset record.

3. Finding the mode for each column:

The get_mode() function is connected to each column within the dataset Angad_Cricket to discover the mode for each column. For each column, the mode is calculated and stored in a isolated variable.

4. Printing the mode for each column:

At last, the code prints out the mode for each column. Each print statement shows the name of the column at the side its comparing mode value. This step permits for simple visualization and translation of the foremost habitually happening values in each variable.

In summary, the given code defines a custom function to discover the mode, applies this work to each column within the dataset, and after that prints out the mode for each column. Altering the record way and column names will empower you to utilize this handle together with your particular dataset.

Max

This code snippet imports a dataset named your_dataset.csv, calculates the maximum value for each column, and then prints out these maximum values. Let's talk about the process without including the real code:

1. Importing the dataset:
The dataset your_dataset.csv is imported into R utilizing the read.csv() function. This work peruses information from a CSV record and makes a information outline in R. Ensure to alter the record way and title inside the cites to match the area and name of your dataset record.

2. Finding the maximum value for each column:
For each column within the dataset Angad_Cricket, the code calculates the greatest value using the max() function. This function returns the biggest element in a vector or column. The maximum value for each column is put away in a isolated variable.

3. Printing the maximum value for each column:
The code prints out the maximum value for each column. Each print explanation concatenates the title of the column with its comparing maximum value. This step encourages the visualization and interpretation of the highest value observed in each variable.

In summary, this code proficiently calculates and prints the maximum value for each column within the dataset Angad_Cricket, giving experiences into the highest value watched for each variable. Altering the record way and column names will permit you to utilize this handle together with your particular dataset.

Min

This code snippet imports a dataset named your_dataset.csv, calculates the minimum value for each column, and after that prints out these minimum values. Let's talk about the method without counting the genuine code:

1. Importing the dataset:

The dataset your_dataset.csv is imported into R utilizing the read.csv() function. This function reads information from a CSV record and makes a information outline in R. Guarantee to alter the record way and name within the cites to match the area and name of your dataset record.

2. Finding the minimum value for each column:

For each column within the dataset Angad_Cricket, the code calculates the minimum value using the min() work. This work returns the littlest component in a vector or column. The minimum value for each column is put away in a isolated variable.

3. Printing the minimum value for each column:

The code prints out the minimum value for each column. Each print statement concatenates the title of the column with its comparing least value. This step facilitates the visualization and interpretation of the lowest value observed in each variable.

In summary, this code effectively calculates and prints the least esteem for each column within the dataset Angad_Cricket, giving bits of knowledge into the least esteem watched for each variable. Altering the record way and column names will permit you to utilize this handle along with your particular dataset.

THANK YOU FOR LISTENING!

Don't hesitate to ask any questions!