Data Wrangling Assessment Task 3: Dataset challenge

Thaddeus Lee, S3933533

Setup

Insert and load the packages you need to produce the report here:

```
# This is a chunk where you can load the packages required for producing the report
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
library(magrittr)
library(dplyr) # For Wrangling Data
```

```
##
## 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(tidyr) # For Reading and Writing Data.
```

```
##
## Attaching package: 'tidyr'

## The following object is masked from 'package:magrittr':
##
##     extract
```

```
library(outliers)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x lubridate::as.difftime() masks base::as.difftime()
## x lubridate::date()        masks base::date()
## x tidyr::extract()         masks magrittr::extract()
## x dplyr::filter()          masks stats::filter()
## x lubridate::intersect()   masks base::intersect()
## x dplyr::lag()             masks stats::lag()
## x purrr::set_names()       masks magrittr::set_names()
## x lubridate::setdiff()     masks base::setdiff()
## x lubridate::union()       masks base::union()
```

```
library(deducorrect)
```

```
## Loading required package: editrules

## Loading required package: igraph

##
## Attaching package: 'igraph'

## The following objects are masked from 'package:purrr':
##
##     compose, simplify

## The following object is masked from 'package:tibble':
##
##     as_data_frame

## The following object is masked from 'package:tidyr':
##
##     crossing

## The following objects are masked from 'package:dplyr':
##
##     as_data_frame, groups, union

## The following objects are masked from 'package:lubridate':
##
##     %--%, union

## The following objects are masked from 'package:stats':
##
##     decompose, spectrum
```

```
## The following object is masked from 'package:base':
##
##     union
```

```
##
## Attaching package: 'editrules'
```

```
## The following objects are masked from 'package:igraph':
##
##     blocks, normalize
```

```
## The following object is masked from 'package:purrr':
##
##     reduce
```

```
## The following objects are masked from 'package:tidyr':
##
##     contains, separate
```

```
## The following object is masked from 'package:dplyr':
##
##     contains
```

```
library(deductive)
library(validate)
```

```
##
## Attaching package: 'validate'
```

```
## The following objects are masked from 'package:igraph':
##
##     compare, hierarchy
```

```
## The following object is masked from 'package:ggplot2':
##
##     expr
```

```
## The following object is masked from 'package:dplyr':
##
##     expr
```

```
library(Hmisc)
```

```
## Loading required package: lattice
```

```
## Loading required package: survival
```

```
## Loading required package: Formula
```

```
##
## Attaching package: 'Hmisc'

## The following objects are masked from 'package:validate':
##
##     label, label<-

## The following objects are masked from 'package:dplyr':
##
##     src, summarize

## The following objects are masked from 'package:base':
##
##     format.pval, units
```

```
library(MVN)
library(readr)
library(openxlsx)
library(tinytex)
library(stringr)
```

Data Description

For Assignment 3, I have chosen two data sets covering Indian Premier League (IPL) Cricket. I find cricket interesting as the way it is played, and scored make it a very statistics heavy sport.

Both datasets were obtained from Kaggle, a website where individuals and organisations can provide datasets on a wide range of topics. The first dataset covers the player auction that takes place in February each year where teams bid on players to join their team(*Cricket Mastery, 2022*). The dataset was put uploaded to KAGGLE by user VINITSHAH0110, with the data being scraped off pubicly available information surch as Wikipedia and various news websites covering the auction (*VINITSHAH0110, 2022*). The auction has become a major event in India.

Our second data set, covers IPL player statistics and contains data on players such as their bowling and batting statistics (*Vora, S 2022*). Both datasets were available from Kaggle as CSV files.

Data analysis plays a major part in sports like cricket, particularly when yearly player auctions are held where a lot of money is involved. With this information in mind, I thought it would be interesting to merge both auction and player datasets together to create a dataset that might give insight into the sorts of statistics that might attract highest bids.

Below we export the data sets into R as follows:

```
# This is a chunk for importing/reading/scraping datasets and then merging them.
# Code for Importing Data sets
Player_DF <- read.csv("IPL_Data.csv", header = TRUE, sep=",")
Auction_DF <- read.csv("IPL_Auction_2022_FullList.csv", header = TRUE, sep=",")
```

Understand Our Datasets

Player List Data Frame

Initial inpection of Player data frame allows us to observe a shape of 237 observations and 39 variables.

Using the **str()** function, we obtain our column names and their respective data types. Along with a brief description of each variable, I have included the variable names and datatypes below:

- **Name:** *Character Data* lists name of cricket players in dataset.
- **Team:** *Character Data* that lists team the cricket player plays for.
- **Url:** *Character Data* providing URL that directs to cricket player's statistics.
- **Type:** *Character Data* noting role of player in team/sport
- **ValueinCR:** *Numeric Data* notes each player's networth in "Crores" which is a unit of ten million rupees.
- **Full.Name:** *Character Data* row of full name for each player.
- **Born:** *Character Data* provides data on each player's date of birth and birth place.
- **Age:** *Character Data* provides data on players age in years, months and days.
- **National.Side:** *Character Data* data listing the country each player represents and plays for.
- **Batting.Style:** *Character Data* data on cricket player's batting style, whether player is left-handed or right-handed.
- **Bowling:** *Character Data* provides information on players bowling type or style.
- **Sport:** *Character Data* provides data on game format played. "Cricket" or "IPL"
- **MatchPlayed:** *Integer Data* on number of matches played by cricketer.
- **InningsBatted:** *Integer Data* The number of Innings batted in IPL
- **NotOuts:** *Integer Data* The number of times cricket player has not been outted or dismissed by the end of an inning.
- **RunsScored:** *Integer Data* Total number of runs scored in IPL
- **HighestInnScore:** *Character Data*
- **X100s:** *Integer Data* number of times a run of 100 or more has been made in a single Inning. Also known as a century (*Harris, M 2022*)
- **X50s:** *Integer Data* number of times a run of 50 has been made in a single Inning.
- **X4s:** *Integer Data* number of times when 4 runs are scored by the batting team (*Luke, 2022*).
- **X6s:** *Integer Data* number of times when 6 runs are scored by the batting team (*Luke, 2022*).
- **BattingAVG:** *Numeric Data* showing player's batting average
- **BattingS.R:** *Numeric Data* showing player's strike rate for batting
- **CatchesTaken:** *Integer Data* number of catches made
- **StumpingsMade:** *Integer Data* number of stumpings made in IPL
- **Ducks:** *Integer Data* Number of duck outs. Where a batter has not scored any runs before being dismissed in an inning (*Harris, M 2022*).
- **R.O:** *Integer Data* number of times dismissed based on run outs
- **InningsBowled:** *Integer Data* number of innings bowled
- **Overs:** *Numeric Data* number of overs bowled. An over consists of six legitimate bowls (*Harris, M, 2022*).
- **Maidens:** *Integer Data* number of maidens bowled. A maiden is where no runs are scored by the batting side (*Harris, M 2022*.
- **RunsConceded:** *Integer Data* runs conceded in IPL
- **Wickets:** *Integer Data* number of wickets taken
- **Best:** *Character Data* Best bowling figure over runs conceded for player in IPL
- **X3s:** *Integer Data* number of three wicket hauls scored
- **X5s:** *Integer Data* number of five wicket hauls scored
- **BowlingAVG:** *Numeric Data* noting bowling average of player.
- **EconomyRate:** *Numeric Data*noting players economy rate. The economy rate is the number of runs conceded per over bowled. Hence a lower rate is better (*Wikipedia, 2022*).
- **S.R:** *Numeric Data* Bowling Strike Rate
- **Mtc:** *Integer Data* number of matches played in IPL.

IPL Auction Data Frame

str() tells us that out auction data set is shaped with 589 observations and 17 variables. Each variable is listed below with a brief description and their data type.

- **Set.No.:** *Integer* set number for player
- **Set.Name:** *Character* player's set name. Set name relates to player's specialty
- **Player:** *Character* player's name
- **Country:** *Character* player's country of
- **State.Association:** *Character* state association.
- **Age:** *Integer* player's age
- **Specialism:** *Character* provides data on players specialisation
- **Batting:** *Character* information on batting style. Right-handed, Left-handed.
- **Bowling:** *Character* player's bowling style.
- **IPL:** *Integer* number of IPL matches played.
- **Previous.IPLTeam.s.:** *Character* player's previous teams
- **X2021.Team:** *Character* team played for in previous year
- **C.U.A:** *Character* information on player cap. "Capped" means player has played for national team. "Uncapped" has not played for national team. Associate Nation.
- **Base.Price:** *Integer* player's base auction price.
- **Sold.Price:** *Character* players sold price
- **New.Franchise:** *Character* player's new team or franchise
- **Bid:** *Character* player's bid status. Two responses. "Sold" and "Unsold".

Data Type Conversions and Duplicate Values

Below we use the **as.factor()** function to turn our character variables *Player*, *Team*, *Type*, *National.Side*, *Batting.Style*, *Bowling*, *Sport*, *Set.Name*, *Country*, *State.Association*, *Specialism*, *Batting*, *Bowling*, *X2021.Team*, *C.U.A*, *New.Franchise*, and *Bid*

We convert all character values to uppercase and use the **unique()** to check for any duplicates in these variables to make sure we don't get any duplicates due to spelling mistakes.

The **rename()** function is used to change the *Name* column in our player dataset to *Player* for joining in the next step.

```r
# This is a chunk where you inspect the types of variables, data structures, check the attributes in th

#Check structure of Player dataframe.
str(Player_DF)
```

```
## 'data.frame':    237 obs. of  39 variables:
##  $ Name           : chr  "Mayank Agarwal" "Liam Livingstone" "Kagiso Rabada" "Shahrukh Khan" ...
##  $ Team           : chr  "PBKS" "PBKS" "PBKS" "PBKS" ...
##  $ Url            : chr  "https://sports.ndtv.com/cricket/players/1430-mayank-agarwal-playerprofile"
##  $ Type           : chr  "Batsman " "All-Rounder " "Bowler " "All-Rounder " ...
##  $ ValueinCR      : num  12 11.5 9.25 9 8.25 6.75 6 5.25 4 3.8 ...
##  $ Full.Name      : chr  "Mayank Anurag Agarwal" "Liam Stephen Livingstone" "Kagiso Rabada" "Masood S
##  $ Born           : chr  "February 16, 1991 Bangalore, Karnataka" "August 4, 1993 Barrow-in-Furness,
##  $ Age            : chr  "31 Years, 0 Months, 28 Days" "28 Years, 7 Months, 11 Days" "26 Years, 9 Mon
##  $ National.Side  : chr  "India" "England" "South Africa" "India" ...
##  $ Batting.Style  : chr  "Right Handed" "Right Handed" "Left Handed" "Right Handed" ...
##  $ Bowling        : chr  "Off break" "Leg break" "Right-arm fast" "Off break" ...
##  $ Sport          : chr  "" "IPL" "IPL" "" ...
```

```
##  $ MatchPlayed    : int  100 9 50 11 192 28 NA 42 23 10 ...
##  $ InningsBatted  : int  95 9 18 10 191 28 NA 11 3 6 ...
##  $ NotOuts        : int  4 1 8 3 25 3 NA 4 2 6 ...
##  $ RunsScored     : int  2131 112 138 153 5784 1038 NA 31 2 84 ...
##  $ HighestInnScore: chr  "106 v RR" "44 v SRH" "44 v MI" "47 v CSK" ...
##  $ X100s          : int  1 0 0 0 2 1 NA 0 0 0 ...
##  $ X50s           : int  11 0 0 0 44 7 NA 0 0 0 ...
##  $ X4s            : int  203 9 11 9 654 99 NA 3 0 5 ...
##  $ X6s            : int  85 6 4 10 124 46 NA 0 0 3 ...
##  $ BattingAVG     : num  23.4 14 13.8 21.9 34.8 ...
##  $ BattingS.R     : num  135 126 103 134 127 ...
##  $ CatchesTaken   : int  40 7 23 4 82 18 NA 11 6 2 ...
##  $ StumpingsMade  : int  0 0 0 0 0 4 NA 0 0 0 ...
##  $ Ducks          : int  6 0 5 1 11 2 NA 4 1 0 ...
##  $ R.O            : int  4 0 2 1 16 0 NA 1 0 0 ...
##  $ InningsBowled  : int  NA 1 50 NA 6 NA NA 41 23 10 ...
##  $ Overs          : num  NA 1 190 NA 8 NA NA 150 76.1 32 ...
##  $ Maidens        : int  NA 0 2 NA 0 NA NA 0 1 1 ...
##  $ RunsConceded   : int  NA 13 1560 NA 66 NA NA 1117 669 228 ...
##  $ Wickets        : int  NA 0 76 NA 4 NA NA 43 30 5 ...
##  $ Best           : chr  "" "0/13 v MI" "4/21 v RCB" "" ...
##  $ X3s            : int  NA 0 4 NA 0 NA NA 3 3 1 ...
##  $ X5s            : int  NA 0 0 NA 0 NA NA 0 1 0 ...
##  $ BowlingAVG     : num  NA NA 20.5 NA 16.5 ...
##  $ EconomyRate    : num  NA 13 8.21 NA 8.25 NA NA 7.44 8.78 7.12 ...
##  $ S.R            : num  NA NA 15 NA 12 ...
##  $ Mtc            : int  NA 1 50 NA 6 NA NA 41 23 10 ...
```

```
head(Player_DF)
```

```
##                Name Team
## 1   Mayank Agarwal PBKS
## 2 Liam Livingstone PBKS
## 3    Kagiso Rabada PBKS
## 4    Shahrukh Khan PBKS
## 5   Shikhar Dhawan PBKS
## 6   Jonny Bairstow PBKS
##                                                                             Url
## 1            https://sports.ndtv.com/cricket/players/1430-mayank-agarwal-playerprofile
## 2 https://sports.ndtv.com/cricket/players/64363-liam-stephen-livingstone-playerprofile
## 3            https://sports.ndtv.com/cricket/players/64042-kagiso-rabada-playerprofile
## 4           https://sports.ndtv.com/cricket/players/113433-shahrukh-khan-playerprofile
## 5             https://sports.ndtv.com/cricket/players/737-shikhar-dhawan-playerprofile
## 6            https://sports.ndtv.com/cricket/players/1551-jonny-bairstow-playerprofile
##             Type ValueinCR                Full.Name
## 1        Batsman     12.00    Mayank Anurag Agarwal
## 2    All-Rounder     11.50 Liam Stephen Livingstone
## 3         Bowler      9.25            Kagiso Rabada
## 4    All-Rounder      9.00     Masood Shahrukh Khan
## 5        Batsman      8.25           Shikhar Dhawan
## 6 Wicket-Keeper      6.75   Jonathan Marc Bairstow
##                                            Born                         Age
## 1       February 16, 1991 Bangalore, Karnataka 31 Years, 0 Months, 28 Days
## 2 August 4, 1993 Barrow-in-Furness, Cumberland 28 Years, 7 Months, 11 Days
```

```
## 3                    May 25, 1995 Johannesburg 26 Years, 9 Months, 22 Days
## 4             May 27, 1995 Chennai, Tamil Nadu 26 Years, 9 Months, 20 Days
## 5                       December 5, 1985 Delhi 36 Years, 3 Months, 10 Days
## 6       September 26, 1989 Bradford, Yorkshire 32 Years, 5 Months, 19 Days
##   National.Side Batting.Style          Bowling Sport MatchPlayed InningsBatted
## 1         India  Right Handed        Off break              100            95
## 2       England  Right Handed        Leg break   IPL          9             9
## 3  South Africa   Left Handed  Right-arm fast    IPL         50            18
## 4         India  Right Handed        Off break               11            10
## 5         India   Left Handed        Off break   IPL        192           191
## 6       England  Right Handed Right-arm medium               28            28
##   NotOuts RunsScored HighestInnScore X100s X50s X4s X6s BattingAVG BattingS.R
## 1       4       2131        106 v RR     1   11 203  85      23.41     135.47
## 2       1        112        44 v SRH     0    0   9   6      14.00     125.84
## 3       8        138         44 v MI     0    0  11   4      13.80     102.98
## 4       3        153        47 v CSK     0    0   9  10      21.85     134.21
## 5      25       5784     106* v PBKS     2   44 654 124      34.84     126.64
## 6       3       1038       114 v RCB     1    7  99  46      41.52     142.19
##   CatchesTaken StumpingsMade Ducks R.O InningsBowled Overs Maidens RunsConceded
## 1           40             0     6   4            NA    NA      NA           NA
## 2            7             0     0   0             1     1       0           13
## 3           23             0     5   2            50   190       2         1560
## 4            4             0     1   1            NA    NA      NA           NA
## 5           82             0    11  16             6     8       0           66
## 6           18             4     2   0            NA    NA      NA           NA
##   Wickets       Best X3s X5s BowlingAVG EconomyRate S.R Mtc
## 1      NA             NA  NA         NA          NA  NA  NA
## 2       0   0/13 v MI   0   0         NA       13.00  NA   1
## 3      76 4/21 v RCB   4   0      20.52        8.21  15  50
## 4      NA             NA  NA         NA          NA  NA  NA
## 5       4   1/7 v DC   0   0      16.50        8.25  12   6
## 6      NA             NA  NA         NA          NA  NA  NA
```

```r
#Check structure of Auction dataframe.
str(Auction_DF)
```

```
## 'data.frame':    589 obs. of  17 variables:
##  $ Set.No.            : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ Set.Name           : chr  "M" "M" "M" "M" ...
##  $ Player             : chr  "Trent Boult" "Pat Cummins" "Shikhar Dhawan" "Shreyas Iyer" ...
##  $ Country            : chr  "New Zealand" "Australia" "India" "India" ...
##  $ State.Association  : chr  "" "" "DDCA" "MCA" ...
##  $ Age                : int  32 28 36 27 26 32 35 29 37 35 ...
##  $ Specialism         : chr  "BOWLER" "ALL-ROUNDER" "BATSMAN" "BATSMAN" ...
##  $ Batting            : chr  "RHB" "RHB" "LHB" "RHB" ...
##  $ Bowling            : chr  "LEFT ARM Fast Medium" "RIGHT ARM Fast" "-" "RIGHT ARM Leg Spin" ...
##  $ IPL                : int  62 37 192 87 50 77 167 77 100 150 ...
##  $ Previous.IPLTeam.s.: chr  "SRH, KKR, DD,MI" "DD, MI, KKR" "DCH, MI, SRH, DC" "DC" ...
##  $ X2021.Team         : chr  "MI" "KKR" "DC" "DC" ...
##  $ C.U.A              : chr  "Capped" "Capped" "Capped" "Capped" ...
##  $ Base.Price         : int  200 200 200 200 200 200 200 200 200 200 ...
##  $ Sold.Price         : chr  "8 CR" "7.25 CR" "8.25 CR" "12.25 CR" ...
##  $ New.Franchise      : chr  "Rajasthan Royals" "Kolkata Knight Riders" "Punjab Kings" "Kolkata Knig
##  $ Bid                : chr  "Sold" "Sold" "Sold" "Sold" ...
```

```r
head(Player_DF)
```

```
##                Name Team
## 1    Mayank Agarwal PBKS
## 2 Liam Livingstone PBKS
## 3     Kagiso Rabada PBKS
## 4     Shahrukh Khan PBKS
## 5    Shikhar Dhawan PBKS
## 6    Jonny Bairstow PBKS
##                                                                              Url
## 1            https://sports.ndtv.com/cricket/players/1430-mayank-agarwal-playerprofile
## 2 https://sports.ndtv.com/cricket/players/64363-liam-stephen-livingstone-playerprofile
## 3            https://sports.ndtv.com/cricket/players/64042-kagiso-rabada-playerprofile
## 4           https://sports.ndtv.com/cricket/players/113433-shahrukh-khan-playerprofile
## 5             https://sports.ndtv.com/cricket/players/737-shikhar-dhawan-playerprofile
## 6            https://sports.ndtv.com/cricket/players/1551-jonny-bairstow-playerprofile
##            Type ValueinCR               Full.Name
## 1       Batsman     12.00    Mayank Anurag Agarwal
## 2   All-Rounder     11.50 Liam Stephen Livingstone
## 3        Bowler      9.25            Kagiso Rabada
## 4   All-Rounder      9.00     Masood Shahrukh Khan
## 5       Batsman      8.25           Shikhar Dhawan
## 6 Wicket-Keeper      6.75   Jonathan Marc Bairstow
##                                           Born                          Age
## 1       February 16, 1991 Bangalore, Karnataka 31 Years, 0 Months, 28 Days
## 2 August 4, 1993 Barrow-in-Furness, Cumberland 28 Years, 7 Months, 11 Days
## 3                     May 25, 1995 Johannesburg 26 Years, 9 Months, 22 Days
## 4              May 27, 1995 Chennai, Tamil Nadu 26 Years, 9 Months, 20 Days
## 5                        December 5, 1985 Delhi 36 Years, 3 Months, 10 Days
## 6       September 26, 1989 Bradford, Yorkshire 32 Years, 5 Months, 19 Days
##   National.Side Batting.Style         Bowling Sport MatchPlayed InningsBatted
## 1         India  Right Handed        Off break               100            95
## 2       England  Right Handed        Leg break   IPL           9             9
## 3  South Africa   Left Handed   Right-arm fast   IPL          50            18
## 4         India  Right Handed        Off break                11            10
## 5         India   Left Handed        Off break   IPL         192           191
## 6       England  Right Handed Right-arm medium                28            28
##   NotOuts RunsScored HighestInnScore X100s X50s X4s X6s BattingAVG BattingS.R
## 1       4       2131        106 v RR     1   11 203  85      23.41     135.47
## 2       1        112        44 v SRH     0    0   9   6      14.00     125.84
## 3       8        138         44 v MI     0    0  11   4      13.80     102.98
## 4       3        153        47 v CSK     0    0   9  10      21.85     134.21
## 5      25       5784     106* v PBKS     2   44 654 124      34.84     126.64
## 6       3       1038       114 v RCB     1    7  99  46      41.52     142.19
##   CatchesTaken StumpingsMade Ducks R.O InningsBowled Overs Maidens RunsConceded
## 1           40             0     6   4            NA    NA      NA           NA
## 2            7             0     0   0             1     1       0           13
## 3           23             0     5   2            50   190       2         1560
## 4            4             0     1   1            NA    NA      NA           NA
## 5           82             0    11  16             6     8       0           66
## 6           18             4     2   0            NA    NA      NA           NA
##   Wickets       Best X3s X5s BowlingAVG EconomyRate S.R Mtc
## 1      NA             NA  NA         NA          NA  NA  NA
```

```
## 2       0  0/13 v MI   0   0         NA       13.00  NA   1
## 3      76 4/21 v RCB   4   0      20.52        8.21  15  50
## 4      NA              NA  NA         NA          NA  NA  NA
## 5       4   1/7 v DC   0   0      16.50        8.25  12   6
## 6      NA              NA  NA         NA          NA  NA  NA
```

```r
################################################################################
# Convert values in each column to uppercase for ease of analysis.

# Convert variables in data frame to uppercase, if Variable is a character data type.
Player_DF <- data.frame(lapply(Player_DF, function(v) {
  if (is.character(v)) return(toupper(v))
  else return(v)
}))

# Convert variables in data frame to uppercase, if Variable is a character data type.
Auction_DF <- data.frame(lapply(Auction_DF, function(v) {
  if (is.character(v)) return(toupper(v))
  else return(v)
}))

# Rename "Name" Variable in Player dataframe to "Player" so that it matches with "Player"
Player_DF <- rename(Player_DF, "Player" = "Name")

###############################################################################

#Player Data Frame - Data Type Conversions.
Player_DF$Player <- as.factor(Player_DF$Player)
Player_DF$Team <- as.factor(Player_DF$Team)
Player_DF$Type <- as.factor(Player_DF$Type)
Player_DF$National.Side <- as.factor(Player_DF$National.Side)
Player_DF$Batting.Style <- as.factor(Player_DF$Batting.Style)
Player_DF$Bowling <- as.factor(Player_DF$Bowling)
Player_DF$Sport <- as.factor(Player_DF$Sport)

#Auction Data Frame - Data Type Conversions.
Auction_DF$Player <- as.factor(Auction_DF$Player)
Auction_DF$Set.Name <- as.factor(Auction_DF$Set.Name)
Auction_DF$Country <- as.factor(Auction_DF$Country)
Auction_DF$State.Association <- as.factor(Auction_DF$State.Association)
Auction_DF$Specialism <- as.factor(Auction_DF$Specialism)
Auction_DF$Batting <- as.factor(Auction_DF$Batting)
Auction_DF$Bowling <- as.factor(Auction_DF$Bowling)
Auction_DF$X2021.Team <- as.factor(Auction_DF$X2021.Team)
Auction_DF$C.U.A <- as.factor(Auction_DF$C.U.A)
Auction_DF$New.Franchise <- as.factor(Auction_DF$New.Franchise)
Auction_DF$Bid <- as.factor(Auction_DF$Bid)


###############################################################################
##              Check for errors and duplicate values.
unique(Player_DF$Team)
```

```
##  [1] PBKS SRH  RR   RCB  MI   CSK  KKR  DC   LSG  GT
```

```
## Levels: CSK DC GT KKR LSG MI PBKS RCB RR SRH
```

```
unique(Player_DF$Type)
```

```
## [1] BATSMAN        ALL-ROUNDER    BOWLER         WICKET-KEEPER
## Levels: ALL-ROUNDER  BATSMAN  BOWLER  WICKET-KEEPER
```

```
unique(Player_DF$National.Side)
```

```
##  [1] INDIA        ENGLAND      SOUTH AFRICA WEST INDIES  AUSTRALIA
##  [6] SRI LANKA    NEW ZEALAND  AFGHANISTAN               SINGAPORE
## [11] BANGLADESH
## 11 Levels:  AFGHANISTAN AUSTRALIA BANGLADESH ENGLAND INDIA ... WEST INDIES
```

```
unique(Player_DF$Batting.Style)
```

```
## [1] RIGHT HANDED LEFT HANDED
## Levels:  LEFT HANDED RIGHT HANDED
```

```
unique(Player_DF$Bowling)
```

```
##  [1] OFF BREAK               LEG BREAK              RIGHT-ARM FAST
##  [4] RIGHT-ARM MEDIUM        LEG BREAK GOOGLY       LEFT-ARM MEDIUM FAST
##  [7] SLOW LEFT-ARM ORTHODOX RIGHT-ARM FAST MEDIUM  RIGHT-ARM MEDIUM FAST
## [10]                        LEFT-ARM FAST          LEFT-ARM MEDIUM
## [13] LEFT-ARM FAST MEDIUM   SLOW LEFT-ARM CHINAMAN
## 14 Levels:  LEFT-ARM FAST LEFT-ARM FAST MEDIUM ... SLOW LEFT-ARM ORTHODOX
```

```
unique(Player_DF$Sport)
```

```
## [1]         IPL     CRICKET
## Levels:  CRICKET IPL
```

```
unique(Auction_DF$Set.Name)
```

```
##  [1] M     BA1   AL1   WK1   FA1   SP1   UBA1  UAL1  UWK1  UFA1  USP1  BA2
## [13] AL2   FA2   SP2   UBA2  UAL2  UFA2  BA3   AL3   WK2   FA3   UBA3  UAL3
## [25] UWK2  UFA3  USP2  AL4   FA4   UBA4  UAL4  UWK3  UFA4  AL5   FA5   UBA5
## [37] UAL5  UBA6  UAL6  AL7   UAL7  UFA7  UAL8  UFA8  UAL10 UAL12 UAL13 SP3
## [49] BA4   USP3  BA5   UWK4  UFA5  USP4  AL6   FA6   UFA6  UAL9  UFA9  UAL11
## [61] UAL14 UAL15
## 62 Levels: AL1 AL2 AL3 AL4 AL5 AL6 AL7 BA1 BA2 BA3 BA4 BA5 FA1 FA2 FA3 ... WK2
```

```
unique(Auction_DF$Country)
```

```
##  [1] NEW ZEALAND  AUSTRALIA    INDIA        SOUTH AFRICA WEST INDIES
##  [6] ENGLAND      SRI LANKA    BANGLADESH   AFGHANISTAN  NEPAL
## [11] IRELAND      NAMIBIA      ZIMBABWE     USA          SCOTLAND
## 15 Levels: AFGHANISTAN AUSTRALIA BANGLADESH ENGLAND INDIA IRELAND ... ZIMBABWE
```

```r
unique(Auction_DF$State.Association)
```

```
##  [1]        DDCA   MCA    CAB    TNCA   KSCA   KCA    BCA    HCA    JSCA
## [11] ACA    RCA    UPCA   VCA    PCA    MACA   ASCA   SCA    MPCA   UTCA
## [21] RSPB   GUCA   HPCA   OCA    GCA    JKCA   HYCA   CSCSCA BICA   NCA
## [31] SSCB   CAP    CAU    TCA    MECA
## 35 Levels:  ACA ASCA BCA BICA CAB CAP CAU CSCSCA DDCA GCA GUCA HCA ... VCA
```

```r
unique(Auction_DF$Specialism)
```

```
## [1] BOWLER       ALL-ROUNDER  BATSMAN      WICKETKEEPER
## Levels: ALL-ROUNDER BATSMAN BOWLER WICKETKEEPER
```

```r
unique(Auction_DF$Batting)
```

```
## [1] RHB LHB
## Levels: LHB RHB
```

```r
unique(Auction_DF$Bowling)
```

```
##  [1] LEFT ARM FAST MEDIUM     RIGHT ARM FAST           -
##  [4] RIGHT ARM LEG SPIN       RIGHT ARM OFF SPIN       RIGHT ARM FAST MEDIUM
##  [7] LEFT ARM SLOW ORTHODOX   LEFT ARM FAST            LEFT ARM SLOW UNORTHODOX
## [10] RIGHT ARM MEDIUM         LEFT ARM MEDIUM
## 11 Levels: - LEFT ARM FAST LEFT ARM FAST MEDIUM ... RIGHT ARM OFF SPIN
```

```r
unique(Auction_DF$X2021.Team)
```

```
## [1] MI   KKR  DC   PBKS CSK  SRH  RR   RCB
## Levels:  CSK DC KKR MI PBKS RCB RR SRH
```

```r
unique(Auction_DF$C.U.A)
```

```
## [1] CAPPED    UNCAPPED  ASSOCIATE
## Levels: ASSOCIATE CAPPED UNCAPPED
```

```r
unique(Auction_DF$New.Franchise)
```

```
##  [1] RAJASTHAN ROYALS            KOLKATA KNIGHT RIDERS
##  [3] PUNJAB KINGS                GUJARAT TITANS
##  [5] LUCKNOW SUPER GIANTS        ROYAL CHALLENGERS BANGALORE
##  [7] DELHI CAPITALS              CHENNAI SUPER KINGS
##  [9] SUNRISERS HYDERABAD         MUMBAI INDIANS
## [11] LUKNOW SUPER GIANTS         GUJARAT TITAN
## [13]
## 13 Levels:  CHENNAI SUPER KINGS DELHI CAPITALS GUJARAT TITAN ... SUNRISERS HYDERABAD
```

```r
unique(Auction_DF$New.Franchise)
```

```
##  [1] RAJASTHAN ROYALS            KOLKATA KNIGHT RIDERS
##  [3] PUNJAB KINGS                GUJARAT TITANS
##  [5] LUCKNOW SUPER GIANTS        ROYAL CHALLENGERS BANGALORE
##  [7] DELHI CAPITALS              CHENNAI SUPER KINGS
##  [9] SUNRISERS HYDERABAD         MUMBAI INDIANS
## [11] LUKNOW SUPER GIANTS         GUJARAT TITAN
## [13]
## 13 Levels:  CHENNAI SUPER KINGS DELHI CAPITALS GUJARAT TITAN ... SUNRISERS HYDERABAD
```

Merge Player and Auction Dataframes

Our Player dataset has fewer observations than our Auction dataset, Using the common variable of *Player* which consists of player names, we use a left-join using **merge()** to combine our datasets. Any unmatching rows from our larger Auction dataset are dropped.

Using **str()** on our merged dataset shows we have a dataframe with 237 observation and 42 variables.

```r
###############################################################################
##############################     MERGE DATA SETS    #########################

# Data sets are merged using the merge function.
IPLDATA <- merge(x = Player_DF, y = Auction_DF, by = "Player", all.x = TRUE)

str(IPLDATA)
```

```
## 'data.frame':    237 obs. of  55 variables:
##  $ Player            : Factor w/ 237 levels "ABDUL SAMAD",..: 1 2 3 4 5 6 7 8 9 10 ...
##  $ Team              : Factor w/ 10 levels "CSK","DC","GT",..: 10 4 3 10 1 10 4 8 4 3 ...
##  $ Url               : chr  "HTTPS://SPORTS.NDTV.COM/CRICKET/PLAYERS/113179-ABDUL-SAMAD-PLAYERPROFI
##  $ Type              : Factor w/ 4 levels "ALL-ROUNDER ",..: 2 2 2 1 3 2 2 3 2 3 ...
##  $ ValueinCR         : num  4 0.4 2.6 6.5 1.9 2.6 1 0.2 1.5 2.4 ...
##  $ Full.Name         : chr  "ABDUL SAMAD FAROOQ" "ABHIJEET TOMAR" "ABHINAV MANOHAR SADARANGANI" "AB
##  $ Born              : chr  "OCTOBER 28, 2001 KALA KOT, JAMMU & KASHMIR" "MARCH 14, 1995 JAIPUR, RA
##  $ Age.x             : chr  "20 YEARS, 4 MONTHS, 18 DAYS" "27 YEARS, 0 MONTHS, 2 DAYS" "27 YEARS, 6
##  $ National.Side     : Factor w/ 11 levels "","AFGHANISTAN",..: 6 6 6 6 7 9 6 6 5 11 ...
##  $ Batting.Style     : Factor w/ 3 levels "","LEFT HANDED",..: 3 3 3 2 3 3 3 3 3 3 ...
##  $ Bowling.x         : Factor w/ 14 levels "","LEFT-ARM FAST",..: 6 8 7 14 9 8 11 11 11 10 ...
##  $ Sport             : Factor w/ 3 levels "","CRICKET","IPL": 3 1 1 3 3 3 3 1 1 3 ...
##  $ MatchPlayed       : int  23 NA NA 22 9 6 151 NA 6 3 ...
##  $ InningsBatted     : int  18 NA NA 20 6 6 141 NA 6 2 ...
##  $ NotOuts           : int  4 NA NA 6 2 1 16 NA 0 2 ...
##  $ RunsScored        : int  222 NA NA 241 23 146 3941 NA 148 15 ...
##  $ HighestInnScore   : chr  "33 V DC" "" "" "46* V RCB" ...
##  $ X100s             : int  0 NA NA 0 0 0 2 NA 0 0 ...
##  $ X50s              : int  0 NA NA 0 0 0 28 NA 0 0 ...
##  $ X4s               : int  12 NA NA 17 0 12 417 NA 13 2 ...
##  $ X6s               : int  14 NA NA 12 1 4 76 NA 6 0 ...
##  $ BattingAVG        : num  15.85 NA NA 17.21 5.75 ...
##  $ BattingS.R        : num  146.1 NA NA 139.3 79.3 ...
##  $ CatchesTaken      : int  13 NA NA 5 7 3 58 NA 2 1 ...
```

```
##  $ StumpingsMade      : int  0 NA NA 0 0 0 0 NA 0 0 ...
##  $ Ducks              : int  2 NA NA 0 2 0 13 NA NA 1 ...
##  $ R.O                : int  1 NA NA 1 0 0 7 NA NA 0 ...
##  $ InningsBowled      : int  4 NA NA 14 9 2 1 NA NA 3 ...
##  $ Overs              : num  8 NA NA 22 32 4 1 NA NA 8.4 ...
##  $ Maidens            : int  0 NA NA 0 0 0 0 NA NA 1 ...
##  $ RunsConceded       : int  105 NA NA 176 308 23 5 NA NA 87 ...
##  $ Wickets            : int  2 NA NA 7 7 0 1 NA NA 6 ...
##  $ Best               : chr  "1/9 V PBKS" "" "" "2/4 V MI" ...
##  $ X3s                : int  0 NA NA 0 0 0 0 NA NA 0 ...
##  $ X5s                : int  0 NA NA 0 0 0 0 NA NA 1 ...
##  $ BowlingAVG         : num  52.5 NA NA 25.1 44 ...
##  $ EconomyRate        : num  13.12 NA NA 8 9.62 ...
##  $ S.R                : num  24 NA NA 18.9 27.4 ...
##  $ Mtc                : int  4 NA NA 14 9 2 1 NA NA 3 ...
##  $ Set.No.            : int  NA 40 NA 8 22 12 12 10 19 31 ...
##  $ Set.Name           : Factor w/ 62 levels "AL1","AL2","AL3",..: NA 42 NA 23 15 9 9 44 10 16 ...
##  $ Country            : Factor w/ 15 levels "AFGHANISTAN",..: NA 5 NA 5 9 11 5 5 4 14 ...
##  $ State.Association  : Factor w/ 35 levels "","ACA","ASCA",..: NA 27 NA 26 1 1 21 6 1 1 ...
##  $ Age.y              : int  NA 27 NA 21 29 27 33 25 33 25 ...
##  $ Specialism         : Factor w/ 4 levels "ALL-ROUNDER",..: NA 2 NA 1 3 2 2 3 2 3 ...
##  $ Batting            : Factor w/ 2 levels "LHB","RHB": NA 2 NA 1 2 2 2 2 2 2 ...
##  $ Bowling.y          : Factor w/ 11 levels "-","LEFT ARM FAST",..: NA 11 NA 5 7 11 1 8 1 7 ...
##  $ IPL                : int  NA NA NA 22 9 6 151 0 6 3 ...
##  $ Previous.IPLTeam.s.: chr  NA "" NA "SRH" ...
##  $ X2021.Team         : Factor w/ 9 levels "","CSK","DC",..: NA 1 NA 9 5 6 3 7 1 1 ...
##  $ C.U.A              : Factor w/ 3 levels "ASSOCIATE","CAPPED",..: NA 3 NA 3 2 2 2 3 2 2 ...
##  $ Base.Price         : int  NA 20 NA 20 150 100 100 20 150 75 ...
##  $ Sold.Price         : chr  NA "40 L" NA "6.5 CR" ...
##  $ New.Franchise      : Factor w/ 13 levels "","CHENNAI SUPER KINGS",..: NA 6 NA 13 2 13 6 12 6 5 ..
##  $ Bid                : Factor w/ 2 levels "SOLD","UNSOLD": NA 1 NA 1 1 1 1 1 1 1 ...
```

```
head(IPLDATA)
```

```
##            Player Team
## 1     ABDUL SAMAD  SRH
## 2  ABHIJEET TOMAR  KKR
## 3 ABHINAV MANOHAR   GT
## 4 ABHISHEK SHARMA  SRH
## 5      ADAM MILNE  CSK
## 6   AIDEN MARKRAM  SRH
##                                                                    Url
## 1       HTTPS://SPORTS.NDTV.COM/CRICKET/PLAYERS/113179-ABDUL-SAMAD-PLAYERPROFILE
## 2    HTTPS://SPORTS.NDTV.COM/CRICKET/PLAYERS/108580-ABHIJEET-TOMAR-PLAYERPROFILE
## 3    HTTPS://SPORTS.NDTV.COM/CRICKET/PLAYERS/64145-ABHINAV-MANOHAR-PLAYERPROFILE
## 4   HTTPS://SPORTS.NDTV.COM/CRICKET/PLAYERS/108562-ABHISHEK-SHARMA-PLAYERPROFILE
## 5          HTTPS://SPORTS.NDTV.COM/CRICKET/PLAYERS/1510-ADAM-MILNE-PLAYERPROFILE
## 6 HTTPS://SPORTS.NDTV.COM/CRICKET/PLAYERS/64634-AIDEN-KYLE-MARKRAM-PLAYERPROFILE
##          Type ValueinCR                   Full.Name
## 1     BATSMAN       4.0           ABDUL SAMAD FAROOQ
## 2     BATSMAN       0.4               ABHIJEET TOMAR
## 3     BATSMAN       2.6 ABHINAV MANOHAR SADARANGANI
## 4 ALL-ROUNDER       6.5              ABHISHEK SHARMA
## 5      BOWLER       1.9            ADAM FRASER MILNE
```

```
## 6     BATSMAN        2.6          AIDEN KYLE MARKRAM
##                                         Born                        Age.x
## 1 OCTOBER 28, 2001 KALA KOT, JAMMU & KASHMIR 20 YEARS, 4 MONTHS, 18 DAYS
## 2           MARCH 14, 1995 JAIPUR, RAJASTHAN  27 YEARS, 0 MONTHS, 2 DAYS
## 3               SEPTEMBER 16, 1994 BANGALORE 27 YEARS, 6 MONTHS, -1 DAYS
## 4         SEPTEMBER 4, 2000 AMRITSAR, PUNJAB 21 YEARS, 6 MONTHS, 11 DAYS
## 5            APRIL 13, 1992 PALMERSTON NORTH 29 YEARS, 11 MONTHS, 2 DAYS
## 6                  OCTOBER 4, 1994 CENTURION 27 YEARS, 5 MONTHS, 11 DAYS
##   National.Side Batting.Style             Bowling.x Sport MatchPlayed
## 1         INDIA  RIGHT HANDED             LEG BREAK   IPL          23
## 2         INDIA  RIGHT HANDED             OFF BREAK                NA
## 3         INDIA  RIGHT HANDED      LEG BREAK GOOGLY                NA
## 4         INDIA   LEFT HANDED SLOW LEFT-ARM ORTHODOX   IPL          22
## 5   NEW ZEALAND  RIGHT HANDED        RIGHT-ARM FAST   IPL           9
## 6  SOUTH AFRICA  RIGHT HANDED             OFF BREAK   IPL           6
##   InningsBatted NotOuts RunsScored HighestInnScore X100s X50s X4s X6s
## 1            18       4        222         33 V DC     0    0  12  14
## 2            NA      NA         NA                    NA   NA  NA  NA
## 3            NA      NA         NA                    NA   NA  NA  NA
## 4            20       6        241       46* V RCB     0    0  17  12
## 5             6       2         23        15 V CSK     0    0   0   1
## 6             6       1        146         42 V MI     0    0  12   4
##   BattingAVG BattingS.R CatchesTaken StumpingsMade Ducks R.O InningsBowled
## 1      15.85     146.05           13             0     2   1             4
## 2         NA         NA           NA            NA    NA  NA            NA
## 3         NA         NA           NA            NA    NA  NA            NA
## 4      17.21     139.30            5             0     0   1            14
## 5       5.75      79.31            7             0     2   0             9
## 6      29.20     122.68            3             0     0   0             2
##   Overs Maidens RunsConceded Wickets       Best X3s X5s BowlingAVG EconomyRate
## 1     8       0          105       2 1/9 V PBKS   0   0      52.50       13.12
## 2    NA      NA           NA      NA             NA  NA         NA          NA
## 3    NA      NA           NA      NA             NA  NA         NA          NA
## 4    22       0          176       7   2/4 V MI   0   0      25.14        8.00
## 5    32       0          308       7 2/21 V CSK   0   0      44.00        9.62
## 6     4       0           23       0  0/5 V RCB   0   0         NA        5.75
##     S.R Mtc Set.No. Set.Name      Country State.Association Age.y  Specialism
## 1 24.00   4      NA     <NA>         <NA>              <NA>    NA        <NA>
## 2    NA  NA      40     UBA5        INDIA               RCA    27     BATSMAN
## 3    NA  NA      NA     <NA>         <NA>              <NA>    NA        <NA>
## 4 18.85  14       8     UAL1        INDIA               PCA    21 ALL-ROUNDER
## 5 27.42   9      22      FA3  NEW ZEALAND                      29      BOWLER
## 6    NA   2      12      BA2 SOUTH AFRICA                      27     BATSMAN
##   Batting              Bowling.y IPL Previous.IPLTeam.s. X2021.Team    C.U.A
## 1    <NA>                   <NA>  NA                <NA>       <NA>     <NA>
## 2     RHB     RIGHT ARM OFF SPIN  NA                                UNCAPPED
## 3    <NA>                   <NA>  NA                <NA>       <NA>     <NA>
## 4     LHB LEFT ARM SLOW ORTHODOX  22                 SRH        SRH UNCAPPED
## 5     RHB         RIGHT ARM FAST   9             RCB, MI         MI   CAPPED
## 6     RHB     RIGHT ARM OFF SPIN   6                PBKS       PBKS   CAPPED
##   Base.Price Sold.Price         New.Franchise  Bid
## 1         NA       <NA>                  <NA> <NA>
## 2         20       40 L KOLKATA KNIGHT RIDERS SOLD
## 3         NA       <NA>                  <NA> <NA>
```

```
## 4          20     6.5 CR   SUNRISERS HYDERABAD SOLD
## 5         150     1.90 L   CHENNAI SUPER KINGS SOLD
## 6         100    2.60 CR   SUNRISERS HYDERABAD SOLD
```

Tidy & Manipulate Data I

Untidy Born Column

Inspecting our data, we can observe that the **Born** column from our *Player dataset* combines the player's date of birth with their place of birth into the same column. As such, it does not conform to Hadley Wickham's 'Tidy Data Principles'.

To amend this, we use the **str_replace_all** function to first replace the *","* with an empty space, we then use **separate** function which is part of **tidyr** to split the player's date of birth and place of birth. Split creates for columns which we name **Day**, **Month**, **Year** and **PLACEOFBIRTH**.

We will combine our newly created **Day**, **Month**, and **Year** columns together. First we convert **Month** to numbers using the **str_replace_all** function. Then we use the **as.numeric** function to convert our **Day**, **Month** and **Year** from characters into numeric data types.

Lastly, we use **mutate()** to combine our separate **Day**, **Month** and **Year** into a single column which we assign **DATEOFBIRTH**.

PLACEOFBIRTH does not contain consistent information, certain rows only contain cities or provinces. It is not necessary for the purposes of our dataset so it is dropped below.

Untidy Best Column

Our *Player Dataset* also has a **Best** column which combines a players best Bowls/RunsConceded with the team played agains, we will have to give this stat it's own column in case the stat needs to be analysed.

We again use **separate** function to split the **Best** column in **Highest.Runs.Scored**, **V**, **VTeam**. Again to drop the "*" so that we are left with the bowls over runs conceded figure, we then use the separate function again so that they have separate columns **Best.Bowling.Figure.BOWLS** and **Best.Bowling.Figure.RunsConceded**. These are then converted to numeric data types.

Duplicate Columns

inspecting our merged data set, we can observe that there are a number of variables where information is shared and repeated.

Variables such as name age, and country stand out. Other variables taken from our **Auction_DF** such as **Country**, **Specialism**, **Batting**, **Bowling**, **IPL**, **SoldPrice** and **NewFranchise**, are repeated in **Player_DF** in the following respective columns: **NationalSide**, **Type**, **BattingStyle**, **Bowling**, **MatchPlayed**, **ValueinCR** and **Team**.

We can run a match or subset the columns and view them side by side:

Country Check - NationalSide vs Country We can see that NationalSide from our *Player Dataset* is the more complete of the two. Countries match and there are fewer missing values.

Specialism Check - Specialism vs Type - We can see that information across these variables match. Type variable from the auction set is more complete and has no missing values.

Batting Check - Batting vs Batting Style - Both columns note info on whether the player is right-handed or left-handed. Batting uses short hand RHB and LHB to note right-handed and left-handed respectively.

The columns match, however the **BattingStyle** variable from the *Player Dataset* is the more complete of the two with fewer missing values.

Bowling Check - Column from *Player Dataset* is more complete of the two.

Matches Check - IPL vs Match Played - IPL are matches played. We can see that both columns are similar with the rows of data that they share, but **MatchPlayed** from the *Player Dataset* is more complete with fewer missing values.

Sold Price Check - SoldPrice vs ValueinCR - With Indian currency, a Crore is equal to ten million rupees, a lakh is equal to a hundred thousand rupees (*Wikipedia, 2022*). **SoldPrice** from the *auction data set* is inconsistent as it notes the bid or sale price for the player in different units, C for Crore, and L for lakh. **ValueinCR** from our *Player Dataset* lists the sold price consistently in units of Crore and is more complete with less missing values.

Team Check - NewFranchise vs Team - Team from our *Player Dataset* is more complete even though the team names are abbreviated.

Tidying up our new Data Frame

Data used in the **Player__DF** according to Kaggle has been updated more recently with data well after the IPL 2022 auction had taken place. We can observe that the data between the **Player** and **Auction** data sets are similar.

In creating our tidy data set we will then drop the duplicate columns that are less complete as well as junk columns created from the string splitting we did.

```
# This is a chunk where you check whether the data conforms to the tidy data principles and reshape you


# Born Column
IPLDATA$Born = str_replace_all(IPLDATA$Born, ",", "")

###STR_SPLIT_FIXED DOES NOT WORK
#IPLDATA$Born <- str_split_fixed(IPLDATA$Born, " ", n = 4)

#IPLDATA$DA <- IPLDATA[,c(7,8,9)]
#IPLDATA



IPLDATA <-
  tidyr::separate(
  IPLDATA,
  Born,
  into = c("MONTH", "DAY", "YEAR", "PLACEOFBIRTH"),
  sep = " ",
  remove = TRUE,
  extra = "merge",
  fill = "warn",
)
```

```
## Warning: Expected 4 pieces. Missing pieces filled with 'NA' in 10 rows [35, 37,
## 39, 40, 58, 60, 156, 164, 167, 194].
```

```r
# Replace Month with number
IPLDATA$MONTH = str_replace_all(IPLDATA$MONTH, "JANUARY", "01")
IPLDATA$MONTH = str_replace_all(IPLDATA$MONTH, "FEBRUARY", "02")
IPLDATA$MONTH = str_replace_all(IPLDATA$MONTH, "MARCH", "03")
IPLDATA$MONTH = str_replace_all(IPLDATA$MONTH, "APRIL", "04")
IPLDATA$MONTH = str_replace_all(IPLDATA$MONTH, "MAY", "05")
IPLDATA$MONTH = str_replace_all(IPLDATA$MONTH, "JUNE", "06")
IPLDATA$MONTH = str_replace_all(IPLDATA$MONTH, "JULY", "07")
IPLDATA$MONTH = str_replace_all(IPLDATA$MONTH, "AUGUST", "07")
IPLDATA$MONTH = str_replace_all(IPLDATA$MONTH, "SEPTEMBER", "09")
IPLDATA$MONTH = str_replace_all(IPLDATA$MONTH, "OCTOBER", "10")
IPLDATA$MONTH = str_replace_all(IPLDATA$MONTH, "NOVEMBER", "11")
IPLDATA$MONTH = str_replace_all(IPLDATA$MONTH, "DECEMBER", "12")


# Convert Day, Month and YEAR to Numeric
IPLDATA$DAY <- as.numeric(IPLDATA$DAY)
IPLDATA$MONTH <- as.numeric(IPLDATA$MONTH)
IPLDATA$YEAR <- as.numeric(IPLDATA$YEAR)

IPLDATA <- IPLDATA %>%
  mutate(DATEOFBIRTH = make_date(YEAR, MONTH, DAY))


# Highest Inning Column - separate numeric figure from characters.
# After separating, we name the numeric component Highest Runs Scored.

IPLDATA <-
  tidyr::separate(
  IPLDATA,
  HighestInnScore,
  into = c("Highest.Runs.Scored", "V", "VTeam"),
  sep = " ",
  remove = TRUE,
  extra = "merge",
  fill = "warn",
)
```

```
## Warning: Expected 3 pieces. Missing pieces filled with `NA` in 84 rows [2, 3, 8,
## 11, 14, 18, 20, 21, 22, 23, 25, 26, 27, 28, 31, 32, 33, 35, 36, 37, ...].
```

```r
# Remove '*' in column
IPLDATA$Highest.Runs.Scored = str_replace_all(IPLDATA$Highest.Runs.Scored, "\\*", "")
# Change column to numeric
IPLDATA$Highest.Runs.Scored <- as.numeric(IPLDATA$Highest.Runs.Scored)


# Best - separate numeric score from characters and assign numeric column new variable name Best.Bowlin

IPLDATA <-
  tidyr::separate(
  IPLDATA,
  Best,
```

```r
  into = c("Best.Bowling.Figure", "VB", "VBest"),
  sep = " ",
  remove = TRUE,
  extra = "merge",
  fill = "warn",
)
```

```
## Warning: Expected 3 pieces. Missing pieces filled with `NA` in 115 rows [2, 3,
## 8, 9, 11, 12, 14, 16, 18, 19, 21, 22, 23, 25, 26, 27, 28, 31, 32, 33, ...].
```

```r
IPLDATA <-
  tidyr::separate(
  IPLDATA,
  Best.Bowling.Figure,
  into = c("Best.Bowling.Figure.BOWLS", "Best.Bowling.Figure.RunsConceded"),
  sep = "/",
  remove = TRUE,
  extra = "merge",
  fill = "warn",
)
```

```
## Warning: Expected 2 pieces. Missing pieces filled with `NA` in 115 rows [2, 3,
## 8, 9, 11, 12, 14, 16, 18, 19, 21, 22, 23, 25, 26, 27, 28, 31, 32, 33, ...].
```

```r
IPLDATA$Best.Bowling.Figure.BOWLS <- as.numeric(IPLDATA$Best.Bowling.Figure.BOWLS)
IPLDATA$Best.Bowling.Figure.RunsConceded <- as.numeric(IPLDATA$Best.Bowling.Figure.RunsConceded)
#####################################################################################################
###                   Duplicate Columns

### Country
Country_Check1 <- data.frame(IPLDATA$National.Side, IPLDATA$Country)
Country_Check2 <- ifelse(as.character(IPLDATA$National.Side) == as.character(IPLDATA$Country), "Yes", "N

### Specialism vs Type - Type appears more complete
Specialism_Check1 <- data.frame(IPLDATA$Specialism, IPLDATA$Type)
#Specialism_Check2 <- ifelse(as.character(IPLDATA$Specialism) == as.character(IPLDATA$Type), "Yes", "No

### Batting vs Batting Style - BattingStyle from our auction dataframe is more complete
Batting_Check1 <- data.frame(IPLDATA$Batting, IPLDATA$Batting.Style)

### Bowling - Column from Player data is more complete of the two
Bowling_Check1 <- data.frame(IPLDATA$Bowling.x, IPLDATA$Bowling.y)

### IPL vs Match Played - we can see similar, but MatchPlayed has more data.
Matches_Check1 <- data.frame(IPLDATA$IPL, IPLDATA$MatchPlayed, IPLDATA$Mtc)

### SoldPrice vs ValueinCR - we can see they are the same. ValueinCR has no missing data.
SoldPrice_Check1 <- data.frame(IPLDATA$Sold.Price, IPLDATA$ValueinCR)

### NewFranchise vs Team - Team is more complete even though the team names are abbreviated.
Team_Check1 <- data.frame(IPLDATA$New.Franchise, IPLDATA$Team)
```

```
###########       Dropping Duplicate Columns and unnecessary              ###

IPLDATA <- IPLDATA[, !colnames(IPLDATA) %in% c("Url", "MONTH", "DAY", "YEAR", "Age.x", "Specialism", "Ba
```

Tidy & Manipulate Data II - Creating a Variable from Existing Ones

For our Player data set we created a new colum by splitting Date of Birth from Place of Birth. The Age column in the existing data was incomplete, but now that we don't have and missing values in our merged dataset, we can create a new age column using **difftime()** function as per the code below.

Final Touches on Tidying up our new Data Frame

Lasltly, column names are converted to uppercase, and the **select**function is used to rearrange the columns so that player information is on the left and statistics are on the right hand side of the data frame. Columns *R.O*, *S.R*, *NationalSide*, *X3s*, and *X5s* are respectively renamed to *Runouts*, *StrikeRate*, *Country*, *Wickets X3s* and *Wickets X5s* for better understandability.

Our data is now tidy according to Hadley Wickham's tidy data principles. Also, in examing our **IPLDATA2** we don't have any inconsistencies with values being presented. Forcing rows to be presented in uppercase resolved all inconsistencies in character data.

```
# Creating Age from Date of Birth (Data Science Made Simple, 2022)

IPLDATA$AGE = as.numeric(difftime(Sys.Date(),IPLDATA$DATEOFBIRTH, units = "weeks"))/52.25




#########            Change column names to upper case
names(IPLDATA)<-toupper(names(IPLDATA))




#########                Rearrange column order                  #############

IPLDATA2 = select(IPLDATA, PLAYER, DATEOFBIRTH, AGE, NATIONAL.SIDE, TEAM, TYPE,  BATTING.STYLE, BOWLING




################################################################################
#####         Rename Columns for Understandability                         ######


IPLDATA2 <- IPLDATA2 %>%
                rename(
                  BOWLING = BOWLING.X,
                  RUNOUTS = R.O,
                  STRIKERATE = S.R,
                  COUNTRY = NATIONAL.SIDE,
                  WICKETS.X3S = X3S,
```

```
                WICKETS.X5S = X5S
                )
```

Scan I - NA values: Scanning and Imputing

From our **IPLDATA2** data frame we can observe that there are a lot of NA values. We can use the following code to return the column names in our data set that has missing values.
NA_Col_Names <- colnames(IPLDATA2)[colSums(is.na(IPLDATA2)) > 0]

The following code will also return the number of NAs within each column.
colSums(is.na(IPLDATA2[NA_Col_Names]))

This in turn gives us the following results:

- **PLACEOFBIRTH** - 10 NA values
- **MATCHPLAYED** - 75 NA values
- **INNINGSBATTED** - 75 NA values
- **NOTOUTS** - 75 NA values
- **RUNSSCORED** - 84 NA Values
- **HIGHEST.RUNS.SCORED** - 84 NA Values
- **X100S** - 75 NA values
- **X50S** - 75 NA values
- **X4S** - 75 NA values
- **X6S** - 75 NA values
- **BATTINGAVG** - 92 NA values
- **BATTINGS.R** - 75 NA values
- **CATCHESTAKEN** - 92 NA Values
- **STUMPINGSMADE** - 92 missing values
- **DUCKS** - 76 NA values
- **R.O** - 76 NA Values
- **INNINGSBOWLED** - 115 NA values
- **OVERS** - 115 NA values
- **MAIDENS** - 115 NA values
- **RUNSCONCEDED* - 115 NA values
- **WICKETS** - 115 NA values
- **WICKETS.X3S** - 115 NA values
- **WICKETS.X5S** - 115 NA values
- **BOWLINGAVG** - 132 NA values
- **ECONOMYRATE** - 115 NA values
- **S.R** - 132 NA values
- **STATE.ASSOCIATION** - 58 NA values
- **AGE.Y** - 58 NA values
- **PREVIOUS.IPLTEAM.S** - 58 NA values
- **X2021.TEAM** - 58 NA values
- **C.U.A** - 58 NA values
- **BASE.PRICE** - 58 NA values
- **BID** - 58 NA values
- **DATEOFBIRTH** 6 NA Values

We can see that there are numerous NAs across multiple variables that cover player statistics. The Player data set, contains a list of URLs that takes users to web pages that cover these statistics for the respective players. If given the time, we could scrape these statistics and fill in the missing data.

Given the limited time we have, we can omit rows where multiple variables contain NAs and impute the rest of the mising values with the mean, median or mode. This still leaves us with plenty of data that enables us to determine the statistics and factors that may make up the bid price of a player at auction.

We will use the *filter()* function and use the "OR" Operator to exclude rows from the following variables unless a value is present in the other and assign to a new dataframe that we will call **IPLDATA3**:

- **MATCHPLAYED**
- **INNINGSBATTED**
- **NOTOUTS**
- **RUNSSCORED**
- **HIGHEST.RUNS.SCORED**
- **X100S**
- **X50S**
- **X4S**
- **X6S**
- **BATTINGAVG**
- **BATTINGS.R**
- **CATCHESTAKEN**
- **STUMPINGSMADE**
- **DUCKS**
- **R.O**
- **INNINGSBOWLED**
- **OVERS**
- **MAIDENS**
- **RUNSCONCEDED**
- **WICKETS**
- **WICKETS.X3S**
- **WICKETS.X5S**
- **BOWLINGAVG**
- **ECONOMYRATE**
- **S.R**

IPLDATA3 we have a dataframe with 162 observations, and we can oberve there are still NA values in the following variables: *BOWLING*, *SPORT*, *RUNSSCORED*, *HIGHEST.RUNS.SCORED*, *BATTINGAVG*, *CATCHESTAKEN*, *STUMPINGSMADE*, *DUCKS*, *R.O*, *INNINGSBOWLED*, *OVERS*, *MAIDENS*, *RUNSCONCEDED*, *WICKETS*, *BEST.BOWLING.FIGURE.BOWLS*, *BEST.BOWLING.FIGURE.RUNSCONCEDED*, *X3S*, *WICKETS.X5S*, *BOWLINGAVG*, *ECONOMYRATE*, *S.R*, *STATE.ASSOCIATION*, *AGE*, *PREVIOUS.IPLTEAM.S.*, *X2021.TEAM*, *C.U.A*, *BASE.PRICE*, and *BID*

We will impute the following variables as below:

- **BOWLING:** - There are a number of Bowling styles unique to players, so we replace NA values with "UNKNOWN" value.
- **SPORT:** - We replace with the mode which is IPL. This is also in keeping with the more recent **Player** data set with current IPL players.
- **RUNSSCORED:** - discrete number so we replace with median.
- **HIGHEST.RUNS.SCORED:** - discrete number so we replace with median.
- **BATTINGAVG:** - Batting Average is a percentage, so we can impute with mean.
- **CATCHESTAKEN:** - discrete number so we replace with median.
- **STUMPINGSMADE:** - discrete number so we replace with median.
- **DUCKS:** - discrete number so we replace with median.
- **R.O:** - discrete number so we replace with median.
- **INNINGSBOWLED:** - discrete number so we replace with median.

- **OVERS:** - discrete number so we replace with median.
- **MAIDENS:** - discrete number so we replace with median.
- **RUNSCONCEDED:** - discrete number so we replace with median.
- **WICKETS:** - discrete number so we replace with median.
- **BEST.BOWLING.FIGURE.BOWLS:** - discrete number so we replace with median.
- **BEST.BOWLING.FIGURE.RUNSCONCEDED:** - discrete number so we replace with median.
- **WICKETS.X3S:** - discrete number so we replace with median.
- **WICKETS.X5S:** - discrete number so we replace with median.
- **BOWLINGAVG:** - Bowling Average is a percentage, so we can impute with mean.
- **ECONOMYRATE:** - Economy Rate is a percentage, so we can impute with mean.
- **S.R:** - Bowling Strike Rate is a percentage, so we impute with mean
- **STATE.ASSOCIATION:** - There are numerous states that players can be associated with, we replace the NA values with "UNKNOWN".
- **PREVIOUS.IPLTEAM.S.:** - We replace the NA values with "UNKNOWN" value, as players may have been associated with one, or numerous teams previously, or none at all and we do not know this value.
- **X2021.TEAM:** - We replace NA values with "UNKNOWN" value. We don't know which team a player was associated with in the previous year, if they were associatied with one. For our analysis, we can leave "Unkown".
- **C.U.A:** - Categorical Data, so we can replace with mode value
- **BASE.PRICE:** - The other values in this column are even and somewhat "discrete" in nature, so we replace with median value.
- **BID:** - We can replace NA values with the mode which is "SOLD". This is also a logical choices as the **Player** data set is more recent than the **Auction** dataset.

This removes all NA values from our **IPLDATA3** Data frame.

```
# This is a chunk where you scan the data for missing values, inconsistencies and obvious errors
## NA Values


#### Columns with spaces/ blanks instead of NAs - We replace the blank spaces with NAs for consistency
#(zx8754, 2021)
levels(IPLDATA2$COUNTRY)
```

```
##  [1] ""              "AFGHANISTAN"  "AUSTRALIA"    "BANGLADESH"   "ENGLAND"
##  [6] "INDIA"         "NEW ZEALAND"  "SINGAPORE"    "SOUTH AFRICA" "SRI LANKA"
## [11] "WEST INDIES"
```

```
levels(IPLDATA2$COUNTRY) <- c(levels(IPLDATA2$COUNTRY), "NA")
IPLDATA2$COUNTRY[IPLDATA2$COUNTRY == ""] <- 'NA'


# Sport
levels(IPLDATA2$SPORT)
```

```
## [1] ""        "CRICKET" "IPL"
```

```
levels(IPLDATA2$SPORT) <- c(levels(IPLDATA2$SPORT), "NA")
IPLDATA2$SPORT[IPLDATA2$SPORT == ""] <- 'NA'


# Batting Style
levels(IPLDATA2$BATTING.STYLE)
```

```
## [1] ""              "LEFT HANDED"  "RIGHT HANDED"
```

```r
levels(IPLDATA2$BATTING.STYLE) <- c(levels(IPLDATA2$BATTING.STYLE), "NA")
IPLDATA2$BATTING.STYLE[IPLDATA2$BATTING.STYLE == ""] <- 'NA'


# Bowling
levels(IPLDATA2$BOWLING)
```

```
##  [1] ""                        "LEFT-ARM FAST"           "LEFT-ARM FAST MEDIUM"
##  [4] "LEFT-ARM MEDIUM"         "LEFT-ARM MEDIUM FAST"    "LEG BREAK"
##  [7] "LEG BREAK GOOGLY"        "OFF BREAK"               "RIGHT-ARM FAST"
## [10] "RIGHT-ARM FAST MEDIUM"   "RIGHT-ARM MEDIUM"        "RIGHT-ARM MEDIUM FAST"
## [13] "SLOW LEFT-ARM CHINAMAN" "SLOW LEFT-ARM ORTHODOX"
```

```r
levels(IPLDATA2$BOWLING) <- c(levels(IPLDATA2$BOWLING), "NA")
IPLDATA2$BOWLING[IPLDATA2$BOWLING == ""] <- 'NA'


# Previous IPL Team
levels(IPLDATA2$PREVIOUS.IPLTEAM.S.)
```

```
## NULL
```

```r
levels(IPLDATA2$PREVIOUS.IPLTEAM.S.) <- c(levels(IPLDATA2$PREVIOUS.IPLTEAM.S.), "NA")
IPLDATA2$PREVIOUS.IPLTEAM.S.[IPLDATA2$PREVIOUS.IPLTEAM.S. == ""] <- 'NA'


# x2021 Team
levels(IPLDATA2$X2021.TEAM)
```

```
## [1] ""     "CSK"  "DC"   "KKR"  "MI"   "PBKS" "RCB"  "RR"   "SRH"
```

```r
levels(IPLDATA2$X2021.TEAM) <- c(levels(IPLDATA2$X2021.TEAM), "NA")
IPLDATA2$X2021.TEAM[IPLDATA2$X2021.TEAM == ""] <- 'NA'


#Best Bowling figure
#data.class(IPLDATA2$Best)
#IPLDATA2$Best.Bowling.Figure.BOWLS = str_replace_all(IPLDATA$Best.Bowling.Figure.BOWLS, "  ", "NA")

#State Association
levels(IPLDATA2$STATE.ASSOCIATION)
```

```
##  [1] ""       "ACA"    "ASCA"   "BCA"    "BICA"   "CAB"    "CAP"    "CAU"
##  [9] "CSCSCA" "DDCA"   "GCA"    "GUCA"   "HCA"    "HPCA"   "HYCA"   "JKCA"
## [17] "JSCA"   "KCA"    "KSCA"   "MACA"   "MCA"    "MECA"   "MPCA"   "NCA"
## [25] "OCA"    "PCA"    "RCA"    "RSPB"   "SCA"    "SSCB"   "TCA"    "TNCA"
## [33] "UPCA"   "UTCA"   "VCA"
```

```r
levels(IPLDATA2$STATE.ASSOCIATION) <- c(levels(IPLDATA2$STATE.ASSOCIATION), "NA")
IPLDATA2$STATE.ASSOCIATION[IPLDATA2$STATE.ASSOCIATION == ""] <- 'NA'



##########################################################################################

#Finds the variable names contain missing values.
NA_Col_Names <- colnames(IPLDATA2)[colSums(is.na(IPLDATA2)) > 0]
NA_Col_Names
```

```
##  [1] "DATEOFBIRTH"                  "AGE"
##  [3] "MATCHPLAYED"                  "INNINGSBATTED"
##  [5] "NOTOUTS"                      "RUNSSCORED"
##  [7] "HIGHEST.RUNS.SCORED"          "X100S"
##  [9] "X50S"                         "X4S"
## [11] "X6S"                          "BATTINGAVG"
## [13] "BATTINGS.R"                   "CATCHESTAKEN"
## [15] "STUMPINGSMADE"                "DUCKS"
## [17] "RUNOUTS"                      "INNINGSBOWLED"
## [19] "OVERS"                        "MAIDENS"
## [21] "RUNSCONCEDED"                 "WICKETS"
## [23] "BEST.BOWLING.FIGURE.BOWLS"    "BEST.BOWLING.FIGURE.RUNSCONCEDED"
## [25] "WICKETS.X3S"                  "WICKETS.X5S"
## [27] "BOWLINGAVG"                   "ECONOMYRATE"
## [29] "STRIKERATE"                   "STATE.ASSOCIATION"
## [31] "PREVIOUS.IPLTEAM.S."          "X2021.TEAM"
## [33] "C.U.A"                        "BASE.PRICE"
## [35] "BID"
```

```r
#Finds how many NA values are within each variable in Col_names
colSums(is.na(IPLDATA2[NA_Col_Names]))
```

```
##                      DATEOFBIRTH                              AGE
##                                6                                6
##                      MATCHPLAYED                    INNINGSBATTED
##                               75                               75
##                          NOTOUTS                       RUNSSCORED
##                               75                               84
##              HIGHEST.RUNS.SCORED                            X100S
##                               84                               75
##                             X50S                              X4S
##                               75                               75
##                              X6S                       BATTINGAVG
##                               75                               92
##                       BATTINGS.R                     CATCHESTAKEN
##                               75                               92
##                    STUMPINGSMADE                            DUCKS
##                               92                               76
##                          RUNOUTS                    INNINGSBOWLED
##                               76                              115
##                            OVERS                          MAIDENS
##                              115                              115
```

```
##                       RUNSCONCEDED                          WICKETS
##                                115                              115
##          BEST.BOWLING.FIGURE.BOWLS BEST.BOWLING.FIGURE.RUNSCONCEDED
##                                115                              115
##                         WICKETS.X3S                      WICKETS.X5S
##                                115                              115
##                         BOWLINGAVG                      ECONOMYRATE
##                                132                              115
##                         STRIKERATE                STATE.ASSOCIATION
##                                132                               58
##                PREVIOUS.IPLTEAM.S.                       X2021.TEAM
##                                 58                               58
##                              C.U.A                       BASE.PRICE
##                                 58                               58
##                                BID
##                                 58
```

```r
# Exclude rows from multiple columns -
# We use the OR | function to exclude na values in the following rows, unless another value is present.
IPLDATA3 <- IPLDATA2 %>% filter(!is.na(MATCHPLAYED) | !is.na(INNINGSBATTED) | !is.na(NOTOUTS) |  !is.na

nrow(IPLDATA3)
```

```
## [1] 162
```

```r
# NA values in new data frame
NA_Col_Names2 <- colnames(IPLDATA3)[colSums(is.na(IPLDATA3)) > 0]
NA_Col_Names2
```

```
##  [1] "RUNSSCORED"                       "HIGHEST.RUNS.SCORED"
##  [3] "BATTINGAVG"                       "CATCHESTAKEN"
##  [5] "STUMPINGSMADE"                    "DUCKS"
##  [7] "RUNOUTS"                          "INNINGSBOWLED"
##  [9] "OVERS"                            "MAIDENS"
## [11] "RUNSCONCEDED"                     "WICKETS"
## [13] "BEST.BOWLING.FIGURE.BOWLS"        "BEST.BOWLING.FIGURE.RUNSCONCEDED"
## [15] "WICKETS.X3S"                      "WICKETS.X5S"
## [17] "BOWLINGAVG"                       "ECONOMYRATE"
## [19] "STRIKERATE"                       "STATE.ASSOCIATION"
## [21] "PREVIOUS.IPLTEAM.S."              "X2021.TEAM"
## [23] "C.U.A"                            "BASE.PRICE"
## [25] "BID"
```

```r
colSums(is.na(IPLDATA3[NA_Col_Names2]))
```

```
##                         RUNSSCORED              HIGHEST.RUNS.SCORED
##                                  9                                9
##                         BATTINGAVG                     CATCHESTAKEN
##                                 17                               17
##                      STUMPINGSMADE                            DUCKS
##                                 17                                1
##                            RUNOUTS                    INNINGSBOWLED
```

```
##                           1                              40
##                       OVERS                         MAIDENS
##                          40                              40
##                RUNSCONCEDED                         WICKETS
##                          40                              40
##        BEST.BOWLING.FIGURE.BOWLS BEST.BOWLING.FIGURE.RUNSCONCEDED
##                          40                              40
##                  WICKETS.X3S                     WICKETS.X5S
##                          40                              40
##                  BOWLINGAVG                    ECONOMYRATE
##                          57                              40
##                  STRIKERATE               STATE.ASSOCIATION
##                          57                              47
##           PREVIOUS.IPLTEAM.S.                     X2021.TEAM
##                          47                              47
##                       C.U.A                      BASE.PRICE
##                          47                              47
##                         BID
##                          47
```

```
####                  Unused              code###
#IPLDATA3 <- IPLDATA2 %>% filter(!is.na(MATCHPLAYED) & !is.na(INNINGSBATTED) & !is.na(NOTOUTS) & !is.na
#nrow(IPLDATA3)


################################################################################
#######                     IMPUTING NA VALUES                       ##########

#BOWLING:
levels(IPLDATA3$BOWLING)
```

```
##  [1] ""                        "LEFT-ARM FAST"          "LEFT-ARM FAST MEDIUM"
##  [4] "LEFT-ARM MEDIUM"         "LEFT-ARM MEDIUM FAST"   "LEG BREAK"
##  [7] "LEG BREAK GOOGLY"        "OFF BREAK"              "RIGHT-ARM FAST"
## [10] "RIGHT-ARM FAST MEDIUM"   "RIGHT-ARM MEDIUM"       "RIGHT-ARM MEDIUM FAST"
## [13] "SLOW LEFT-ARM CHINAMAN" "SLOW LEFT-ARM ORTHODOX" "NA"
```

```
levels(IPLDATA3$BOWLING) <- c(levels(IPLDATA2$BOWLING), "UNKNOWN")
IPLDATA3$BOWLING[IPLDATA3$BOWLING == "NA"] <- 'UNKNOWN'

#SPORT:
levels(IPLDATA3$SPORT)
```

```
## [1] ""        "CRICKET" "IPL"     "NA"
```

```
levels(IPLDATA3$SPORT) <- c(levels(IPLDATA2$SPORT), "IPL")
IPLDATA3$SPORT[IPLDATA3$SPORT == "NA"] <- 'IPL'

#RUNSSCORED:
IPLDATA3$RUNSSCORED %<>% impute(IPLDATA3$RUNSSCORED, fun = median)

#HIGHEST.RUNS.SCORED:
```

```r
IPLDATA3$HIGHEST.RUNS.SCORED %<>% impute(IPLDATA3$HIGHEST.RUNS.SCORED, fun = median)

#BATTINGAVG:
IPLDATA3$BATTINGAVG %<>% impute(IPLDATA3$BATTINGAVG, fun = mean)

#CATCHESTAKEN:
IPLDATA3$CATCHESTAKEN %<>% impute(IPLDATA3$CATCHESTAKEN, fun = median)

#STUMPINGSMADE:
IPLDATA3$STUMPINGSMADE %<>% impute(IPLDATA3$STUMPINGSMADE, fun = median)

#DUCKS:
IPLDATA3$DUCKS %<>% impute(IPLDATA3$DUCKS, fun = median)

#RUNOUTS:
IPLDATA3$RUNOUTS %<>% impute(IPLDATA3$RUNOUTS, fun = median)

#INNINGSBOWLED:
IPLDATA3$INNINGSBOWLED %<>% impute(IPLDATA3$INNINGSBOWLED, fun = median)

#OVERS:
IPLDATA3$OVERS %<>% impute(IPLDATA3$OVERS, fun = median)

#MAIDENS:
IPLDATA3$MAIDENS %<>% impute(IPLDATA3$MAIDENS, fun = median)

#RUNSCONCEDED:
IPLDATA3$RUNSCONCEDED %<>% impute(IPLDATA3$RUNSCONCEDED, fun = median)

#WICKETS:
IPLDATA3$WICKETS %<>% impute(IPLDATA3$WICKETS, fun = median)

#BEST.BOWLING.FIGURE.BOWLS:
IPLDATA3$BEST.BOWLING.FIGURE.BOWLS %<>% impute(IPLDATA3$BEST.BOWLING.FIGURE.BOWLS, fun = median)

#BEST.BOWLING.FIGURE.RUNSCONCEDED:
IPLDATA3$BEST.BOWLING.FIGURE.RUNSCONCEDED %<>% impute(IPLDATA3$BEST.BOWLING.FIGURE.RUNSCONCEDED, fun = m

#WICKETS.X3S:
IPLDATA3$WICKETS.X3S %<>% impute(IPLDATA3$WICKETS.X3S, fun = median)

#WICKETS.X5S:
IPLDATA3$WICKETS.X5S %<>% impute(IPLDATA3$WICKETS.X5S, fun = median)

#BOWLINGAVG:
IPLDATA3$BOWLINGAVG %<>% impute(IPLDATA3$BOWLINGAVG, fun = mean)

#ECONOMYRATE:
IPLDATA3$ECONOMYRATE %<>% impute(IPLDATA3$ECONOMYRATE, fun = mean)

#STRIKERATE:
IPLDATA3$STRIKERATE %<>% impute(IPLDATA3$STRIKERATE, fun = mean)
```

```r
#STATE.ASSOCIATION: - Numerous number of states that players can be associated. We replace State associ
IPLDATA3$STATE.ASSOCIATION <- as.character(IPLDATA3$STATE.ASSOCIATION)
IPLDATA3$STATE.ASSOCIATION[is.na(as.character(IPLDATA3$STATE.ASSOCIATION))] <- "UNKNOWN"
IPLDATA3$STATE.ASSOCIATION <- as.factor(IPLDATA3$STATE.ASSOCIATION)

#AGE: - Impute with unknown... we have player date of birth so we can mutate this column so that all pl



#PREVIOUS.IPLTEAM.S.: - Replace NA values with unknown
IPLDATA3$PREVIOUS.IPLTEAM.S.[is.na(IPLDATA3$PREVIOUS.IPLTEAM.S.)] <- "UNKNOWN"


#X2021.TEAM:  - Replace NA values with unknown
IPLDATA3$X2021.TEAM <- as.character(IPLDATA3$X2021.TEAM)
IPLDATA3$X2021.TEAM[is.na(as.character(IPLDATA3$X2021.TEAM))] <- "UNKNOWN"
IPLDATA3$X2021.TEAM <- as.factor(IPLDATA3$X2021.TEAM)

#C.U.A:
IPLDATA3$C.U.A %<>% impute(IPLDATA3$C.U.A, fun = mode)


#BASE.PRICE:
IPLDATA3$BASE.PRICE %<>% impute(IPLDATA3$BASE.PRICE, fun = median)

#BID:
IPLDATA3$BID %<>% impute(IPLDATA3$BID, fun = mode)


##############################################################################
#Check for NAs in new dataframe


NA_Col_Names3 <- colnames(IPLDATA3)[colSums(is.na(IPLDATA3)) > 0]
NA_Col_Names3
```

```
## character(0)
```

```r
#Finds how many NA values are within each variable in Col_names
colSums(is.na(IPLDATA3[NA_Col_Names3]))
```

```
## numeric(0)
```

Scan II - Outliers: Scanning and Imputing

Our Data set currently contains 42 variables, of which 29 contain numeric data. Due to time, we will select a handful of these variables, covering a small selection of statistics covering bowling, batting, fielding and values for players. We will select the following numeric data and assign it to **IPLDATA4**:

Univariate Outliers

For detecting outliers, we will use Tuckey's method to detect them. Tukey's method captures values that are more than 1.5 times Inter-Quartile range on the lower and upper quartile of a variable.

The code below returns: The number of outliers' the rows they appear in; and their values. Number of outliers detected for our chosen variables are:

- **MATCHPLAYED:** 9
- **NOTOUTS:** 10
- **RUNSSCORED:** 0
- **BATTINGAVG:** 0
- **BATTINGS.R:** 0
- **X4S:** 18
- **X6S:** 17
- **CATCHESTAKEN:** 22
- **STUMPINGSMADE:** 0
- **WICKETS:** 17
- **STRIKERATE:** 23
- **ECONOMYRATE:** 26
- **BASE.PRICE:** 0
- **VALUEINCR:** 0

We then use the cap function which allows us to replace the outliers detected with either the mean, median or mode.

- **MATCHPLAYED: -** Discrete values. Impute outliers with Median Value.
- **NOTOUTS: -** Discrete values. Impute outliers with Median value.
- **X4S: -** Discrete values. Impute Outliers with Median value.
- **X6S: -** Discrete values. Impute Outliers with Median value.
- **CATCHESTAKEN: -** Discrete values. Impute Outliers with Median value.
- **WICKETS: -** Discrete values. Impute Outliers with Median value.
- **STRIKERATE: -** Percentage. Impute Outliers with mean value.
- **ECONOMYRATE: -** Percentage. Impute outliers with mean value.

```
# This is a chunk where you scan the numeric data for outliers


IPLDATA4 = select(IPLDATA3, PLAYER, DATEOFBIRTH, AGE, COUNTRY, TEAM, TYPE, BATTING.STYLE, BOWLING, SPOR

IPLDATA4
```

```
##                  PLAYER DATEOFBIRTH      AGE      COUNTRY TEAM          TYPE
## 1           ABDUL SAMAD  2001-10-28 20.46206        INDIA  SRH       BATSMAN
## 2       ABHISHEK SHARMA  2000-09-04 21.60766        INDIA  SRH   ALL-ROUNDER
## 3            ADAM MILNE  1992-04-13 29.99043  NEW ZEALAND  CSK        BOWLER
## 4         AIDEN MARKRAM  1994-10-04 27.51880 SOUTH AFRICA  SRH       BATSMAN
## 5        AJINKYA RAHANE  1988-06-06 33.83732        INDIA  KKR       BATSMAN
## 6            ALEX HALES  1989-01-03 33.26042      ENGLAND  KKR       BATSMAN
## 7        ALZARRI JOSEPH  1996-11-20 25.39166  WEST INDIES   GT        BOWLER
## 8         AMBATI RAYUDU  1985-09-23 36.53589        INDIA  CSK WICKET-KEEPER
## 9         ANDRE RUSSELL  1988-04-29 33.94122  WEST INDIES  KKR   ALL-ROUNDER
## 10        ANKIT RAJPOOT  1993-12-04 28.34997        INDIA  LSG        BOWLER
```

```
## 11      ANMOLPREET SINGH  1998-03-28 24.04375         INDIA   MI       BATSMAN
## 12         ANRICH NORTJE  1993-11-16 28.39918 SOUTH AFRICA   DC        BOWLER
## 13            ANUJ RAWAT  1999-10-17 22.49077         INDIA  RCB WICKET-KEEPER
## 14           ANUKUL ROY  1998-11-30 23.36842         INDIA  KKR   ALL-ROUNDER
## 15        ARSHDEEP SINGH  1999-02-05 23.18524         INDIA PBKS        BOWLER
## 16            AVESH KHAN  1996-12-13 25.32878         INDIA  LSG        BOWLER
## 17            AXAR PATEL  1994-01-20 28.22146         INDIA   DC   ALL-ROUNDER
## 18          BASIL THAMPI  1993-09-11 28.57963         INDIA   MI        BOWLER
## 19      BHUVNESHWAR KUMAR  1990-02-05 32.17225         INDIA  SRH        BOWLER
## 20       CHETAN SAKARIYA  1998-02-28 24.12030         INDIA   DC        BOWLER
## 21          CHRIS JORDAN  1988-10-04 33.50923       ENGLAND  CSK   ALL-ROUNDER
## 22           DANIEL SAMS  1992-10-27 29.45181     AUSTRALIA   MI   ALL-ROUNDER
## 23          DAVID MILLER  1989-06-10 32.82843 SOUTH AFRICA   GT       BATSMAN
## 24          DAVID WARNER  1986-10-27 35.44498     AUSTRALIA   DC       BATSMAN
## 25          DAVID WILLEY  1990-02-28 32.10936       ENGLAND  RCB   ALL-ROUNDER
## 26          DEEPAK CHAHAR  1992-07-07 29.75803         INDIA  CSK        BOWLER
## 27           DEEPAK HOODA  1995-04-19 26.98018         INDIA  LSG   ALL-ROUNDER
## 28      DEVDUTT PADIKKAL  2000-07-07 21.76897         INDIA   RR       BATSMAN
## 29        DINESH KARTHIK  1985-06-01 36.84757         INDIA  RCB WICKET-KEEPER
## 30          DWAYNE BRAVO  1983-10-07 38.49624   WEST INDIES  CSK   ALL-ROUNDER
## 31          FABIAN ALLEN  1995-05-07 26.93096   WEST INDIES   MI   ALL-ROUNDER
## 32        FAF DU PLESSIS  1984-07-13 37.73069 SOUTH AFRICA  RCB       BATSMAN
## 33         GLENN MAXWELL  1988-10-14 33.48189     AUSTRALIA  RCB   ALL-ROUNDER
## 34        GLENN PHILLIPS  1996-12-06 25.34792   NEW ZEALAND  SRH WICKET-KEEPER
## 35  GURKEERAT SINGH MANN  1990-06-29 31.77854         INDIA   GT   ALL-ROUNDER
## 36         HARDIK PANDYA  1993-10-11 28.49761         INDIA   GT   ALL-ROUNDER
## 37         HARPREET BRAR  1995-09-16 26.57006         INDIA PBKS   ALL-ROUNDER
## 38         HARSHAL PATEL  1990-11-23 31.37662         INDIA  RCB   ALL-ROUNDER
## 39          ISHAN KISHAN  1998-07-18 23.73753         INDIA   MI WICKET-KEEPER
## 40           ISHAN POREL  1998-09-05 23.60355         INDIA PBKS        BOWLER
## 41     JAGADEESHA SUCHITH  1994-01-16 28.23240         INDIA  SRH        BOWLER
## 42         JAMES NEESHAM  1990-09-17 31.55981   NEW ZEALAND   RR   ALL-ROUNDER
## 43     JASON BEHRENDORFF  1990-04-20 31.96992     AUSTRALIA  RCB        BOWLER
## 44          JASON HOLDER  1991-11-05 30.42789   WEST INDIES  LSG   ALL-ROUNDER
## 45             JASON ROY  1990-07-21 31.71839       ENGLAND   GT       BATSMAN
## 46        JASPRIT BUMRAH  1993-12-06 28.34450         INDIA   MI        BOWLER
## 47          JAYANT YADAV  1990-01-22 32.21053         INDIA   GT   ALL-ROUNDER
## 48        JAYDEV UNADKAT  1991-10-18 30.47710         INDIA   MI        BOWLER
## 49          JOFRA ARCHER  1995-04-01 27.02939       ENGLAND   MI   ALL-ROUNDER
## 50        JONNY BAIRSTOW  1989-09-26 32.53315       ENGLAND PBKS WICKET-KEEPER
## 51           JOS BUTTLER  1990-09-08 31.58442       ENGLAND   RR WICKET-KEEPER
## 52        JOSH HAZLEWOOD  1991-01-08 31.25085     AUSTRALIA  RCB        BOWLER
## 53         KAGISO RABADA  1995-05-25 26.88175 SOUTH AFRICA PBKS        BOWLER
## 54      KAMLESH NAGARKOTI  1999-12-28 22.29392         INDIA   DC   ALL-ROUNDER
## 55       KANE WILLIAMSON  1990-07-08 31.75393   NEW ZEALAND  SRH       BATSMAN
## 56           KARN SHARMA  1987-10-23 34.45796         INDIA  RCB        BOWLER
## 57           KARTIK TYAGI  2000-11-08 21.42994         INDIA  SRH        BOWLER
## 58            KARUN NAIR  1991-12-06 30.34313         INDIA   RR       BATSMAN
## 59           KC CARIAPPA  1994-04-13 27.99453         INDIA   RR        BOWLER
## 60         KHALEEL AHMED  1997-12-05 24.35270         INDIA   DC        BOWLER
## 61        KIERON POLLARD  1987-05-12 34.90636   WEST INDIES   MI   ALL-ROUNDER
## 62              KL RAHUL  1992-04-18 29.97676         INDIA  LSG WICKET-KEEPER
## 63               KM ASIF  1993-07-24 28.71360         INDIA  CSK        BOWLER
## 64     KRISHNAPPA GOWTHAM  1988-10-20 33.46548         INDIA  LSG   ALL-ROUNDER
```

```
## 65         KRUNAL PANDYA  1991-03-24 31.04580        INDIA  LSG   ALL-ROUNDER
## 66         KULDEEP YADAV  1994-12-14 27.32468        INDIA   DC        BOWLER
## 67          KULDIP YADAV  1996-10-15 25.49009        INDIA   RR        BOWLER
## 68           LALIT YADAV  1997-01-03 25.27136        INDIA   DC   ALL-ROUNDER
## 69      LIAM LIVINGSTONE  1993-07-04 28.76828      ENGLAND PBKS   ALL-ROUNDER
## 70       LOCKIE FERGUSON  1991-06-13 30.82433  NEW ZEALAND   GT        BOWLER
## 71           LUNGI NGIDI  1996-03-29 26.03691 SOUTH AFRICA   DC        BOWLER
## 72        MAHIPAL LOMROR  1999-11-16 22.40875        INDIA  RCB   ALL-ROUNDER
## 73           MANAN VOHRA  1993-07-18 28.73001        INDIA  LSG       BATSMAN
## 74         MANDEEP SINGH  1991-12-18 30.31032        INDIA   DC       BATSMAN
## 75         MANISH PANDEY  1989-09-10 32.57690        INDIA  LSG       BATSMAN
## 76          MARCO JANSEN  2000-05-01 21.95215 SOUTH AFRICA  SRH   ALL-ROUNDER
## 77        MARCUS STOINIS  1989-07-16 32.73001    AUSTRALIA  LSG   ALL-ROUNDER
## 78             MARK WOOD  1990-01-11 32.24060      ENGLAND  LSG        BOWLER
## 79          MATTHEW WADE  1987-12-26 34.28298    AUSTRALIA   GT WICKET-KEEPER
## 80        MAYANK AGARWAL  1991-02-16 31.14422        INDIA PBKS       BATSMAN
## 81       MAYANK MARKANDE  1997-11-11 24.41832        INDIA   MI        BOWLER
## 82        MITCHELL MARSH  1991-10-20 30.47163    AUSTRALIA   DC   ALL-ROUNDER
## 83      MITCHELL SANTNER  1992-02-05 30.17635  NEW ZEALAND  CSK   ALL-ROUNDER
## 84             MOEEN ALI  1987-06-18 34.80519      ENGLAND  CSK   ALL-ROUNDER
## 85         MOHAMMAD NABI  1985-01-01 37.26042  AFGHANISTAN  KKR   ALL-ROUNDER
## 86        MOHAMMED SHAMI  1990-09-03 31.59809        INDIA   GT        BOWLER
## 87        MOHAMMED SIRAJ  1994-03-13 28.07929        INDIA  RCB        BOWLER
## 88              MS DHONI  1981-07-07 40.74368        INDIA  CSK WICKET-KEEPER
## 89        MURUGAN ASHWIN  1990-09-08 31.58442        INDIA   MI        BOWLER
## 90     MUSTAFIZUR RAHMAN  1995-09-06 26.59740   BANGLADESH   DC        BOWLER
## 91    NARAYAN JAGADEESAN  1995-12-24 26.29938        INDIA  CSK WICKET-KEEPER
## 92   NATHAN COULTER-NILE  1987-10-11 34.49077    AUSTRALIA   RR        BOWLER
## 93          NATHAN ELLIS  1994-09-22 27.55161    AUSTRALIA PBKS        BOWLER
## 94         NAVDEEP SAINI  1992-11-23 29.37799        INDIA   RR        BOWLER
## 95       NICHOLAS POORAN  1995-10-02 26.52632  WEST INDIES  SRH WICKET-KEEPER
## 96           NITISH RANA  1993-12-27 28.28708        INDIA  KKR   ALL-ROUNDER
## 97           PAT CUMMINS  1993-05-08 28.92413    AUSTRALIA  KKR   ALL-ROUNDER
## 98     PRABHSIMRAN SINGH  2000-07-10 21.76077        INDIA PBKS WICKET-KEEPER
## 99       PRADEEP SANGWAN  1990-11-05 31.42584        INDIA   GT   ALL-ROUNDER
## 100      PRASIDH KRISHNA  1996-02-19 26.14354        INDIA   RR        BOWLER
## 101         PRAVIN DUBEY  1993-07-01 28.77649        INDIA   DC   ALL-ROUNDER
## 102         PRITHVI SHAW  1999-11-09 22.42789        INDIA   DC       BATSMAN
## 103          PRIYAM GARG  2000-11-30 21.36979        INDIA  SRH       BATSMAN
## 104      QUINTON DE KOCK  1992-12-17 29.31237 SOUTH AFRICA  LSG WICKET-KEEPER
## 105         RAHUL CHAHAR  1999-07-04 22.77785        INDIA PBKS        BOWLER
## 106       RAHUL TRIPATHI  1991-03-02 31.10595        INDIA  SRH       BATSMAN
## 107          RASHID KHAN  1998-09-20 23.56254  AFGHANISTAN   GT        BOWLER
## 108           RASIKH DAR  2001-04-05 21.02529        INDIA  KKR        BOWLER
## 109         RAVI BISHNOI  2000-09-05 21.60492        INDIA  LSG        BOWLER
## 110  RAVICHANDRAN ASHWIN  1986-09-17 35.55434        INDIA   RR   ALL-ROUNDER
## 111      RAVINDRA JADEJA  1988-12-06 33.33698        INDIA  CSK   ALL-ROUNDER
## 112       RILEY MEREDITH  1996-06-21 25.80725    AUSTRALIA   MI        BOWLER
## 113          RINKU SINGH  1997-10-12 24.50034        INDIA  KKR       BATSMAN
## 114          RIPAL PATEL  1995-09-28 26.53725        INDIA   DC   ALL-ROUNDER
## 115         RISHABH PANT  1997-10-04 24.52221        INDIA   DC WICKET-KEEPER
## 116         RISHI DHAWAN  1990-02-19 32.13397        INDIA PBKS   ALL-ROUNDER
## 117          RIYAN PARAG  2001-11-10 20.42652        INDIA   RR   ALL-ROUNDER
## 118        ROBIN UTHAPPA  1985-11-11 36.40191        INDIA  CSK       BATSMAN
```

```
## 119          ROHIT SHARMA  1987-04-30 34.93917        INDIA   MI        BATSMAN
## 120      RUTURAJ GAIKWAD  1997-01-31 25.19481        INDIA  CSK        BATSMAN
## 121         SAM BILLINGS  1991-06-15 30.81887      ENGLAND  KKR WICKET-KEEPER
## 122       SANDEEP SHARMA  1993-05-18 28.89679        INDIA PBKS         BOWLER
## 123         SANJU SAMSON  1994-11-11 27.41490        INDIA   RR WICKET-KEEPER
## 124        SARFARAZ KHAN  1997-10-22 24.47300        INDIA   DC    ALL-ROUNDER
## 125          SEAN ABBOTT  1992-02-29 30.11073    AUSTRALIA  SRH         BOWLER
## 126        SHAHBAZ AHMED  1994-12-12 27.33014        INDIA  RCB    ALL-ROUNDER
## 127       SHAHBAZ NADEEM  1989-07-12 32.74094        INDIA  LSG         BOWLER
## 128        SHAHRUKH KHAN  1995-05-27 26.87628        INDIA PBKS    ALL-ROUNDER
## 129       SHARDUL THAKUR  1991-10-16 30.48257        INDIA   DC         BOWLER
## 130  SHERFANE RUTHERFORD  1998-07-15 23.74573  WEST INDIES  RCB    ALL-ROUNDER
## 131       SHIKHAR DHAWAN  1985-12-05 36.33630        INDIA PBKS        BATSMAN
## 132      SHIMRON HETMYER  1996-12-26 25.29323  WEST INDIES   RR        BATSMAN
## 133          SHIVAM DUBE  1993-06-26 28.79016        INDIA  CSK    ALL-ROUNDER
## 134          SHIVAM MAVI  1998-11-26 23.37936        INDIA  KKR    ALL-ROUNDER
## 135        SHREYAS GOPAL  1993-09-04 28.59877        INDIA  SRH         BOWLER
## 136         SHREYAS IYER  1994-12-06 27.34655        INDIA  KKR        BATSMAN
## 137         SHUBMAN GILL  1999-09-08 22.59740        INDIA   GT        BATSMAN
## 138        SIDDARTH KAUL  1990-05-19 31.89064        INDIA  RCB         BOWLER
## 139        SRIKAR BHARAT  1993-10-03 28.51948        INDIA   DC WICKET-KEEPER
## 140         SUNIL NARINE  1988-05-26 33.86740  WEST INDIES  KKR    ALL-ROUNDER
## 141     SURYAKUMAR YADAV  1990-09-14 31.56801        INDIA   MI        BATSMAN
## 142          T NATARAJAN  1991-05-27 30.87081        INDIA  SRH         BOWLER
## 143         TEJAS BAROKA  1996-02-01 26.19275        INDIA   RR         BOWLER
## 144            TIM DAVID  1996-03-16 26.07245    SINGAPORE   MI    ALL-ROUNDER
## 145          TIM SEIFERT  1994-12-14 27.32468  NEW ZEALAND   DC WICKET-KEEPER
## 146          TIM SOUTHEE  1988-12-11 33.32331  NEW ZEALAND  KKR         BOWLER
## 147          TRENT BOULT  1989-07-22 32.71360  NEW ZEALAND   RR         BOWLER
## 148     TUSHAR DESHPANDE  1995-05-15 26.90909        INDIA  CSK         BOWLER
## 149          TYMAL MILLS  1992-07-12 29.74436      ENGLAND   MI         BOWLER
## 150          UMESH YADAV  1987-10-25 34.45249        INDIA  KKR         BOWLER
## 151          UMRAN MALIK  1999-11-22 22.39234        INDIA  SRH         BOWLER
## 152          VARUN AARON  1989-10-29 32.44293        INDIA   GT         BOWLER
## 153   VARUN CHAKRAVARTHY  1991-07-29 30.69856        INDIA  KKR         BOWLER
## 154       VENKATESH IYER  1994-12-25 27.29460        INDIA  KKR    ALL-ROUNDER
## 155        VIJAY SHANKAR  1991-01-26 31.20164        INDIA   GT    ALL-ROUNDER
## 156          VIRAT KOHLI  1988-11-05 33.42174        INDIA  RCB        BATSMAN
## 157         VISHNU VINOD  1993-12-02 28.35543        INDIA  SRH WICKET-KEEPER
## 158    WANINDU HASARANGA  1997-07-29 24.70540    SRI LANKA  RCB    ALL-ROUNDER
## 159    WASHINGTON SUNDAR  1999-10-05 22.52358        INDIA  SRH    ALL-ROUNDER
## 160      WRIDDHIMAN SAHA  1984-10-24 37.44908        INDIA   GT WICKET-KEEPER
## 161     YASHASVI JAISWAL  2001-12-28 20.29528        INDIA   RR        BATSMAN
## 162     YUZVENDRA CHAHAL  1990-07-23 31.71292        INDIA   RR         BOWLER
##     BATTING.STYLE                BOWLING SPORT MATCHPLAYED NOTOUTS RUNSSCORED
## 1    RIGHT HANDED              LEG BREAK   IPL          23       4        222
## 2     LEFT HANDED SLOW LEFT-ARM ORTHODOX   IPL          22       6        241
## 3    RIGHT HANDED        RIGHT-ARM FAST   IPL           9       2         23
## 4    RIGHT HANDED              OFF BREAK   IPL           6       1        146
## 5    RIGHT HANDED      RIGHT-ARM MEDIUM   IPL         151      16       3941
## 6    RIGHT HANDED      RIGHT-ARM MEDIUM   IPL           6       0        148
## 7    RIGHT HANDED  RIGHT-ARM FAST MEDIUM   IPL           3       2         15
## 8    RIGHT HANDED              OFF BREAK   IPL         175      31       3916
## 9    RIGHT HANDED        RIGHT-ARM FAST   IPL          84      12       1700
```

```
## 10  RIGHT HANDED         RIGHT-ARM FAST   IPL          29       2         26
## 11  RIGHT HANDED              OFF BREAK   IPL           1       0         16
## 12  RIGHT HANDED         RIGHT-ARM FAST   IPL          24       4          7
## 13   LEFT HANDED                UNKNOWN   IPL           2       0          0
## 14   LEFT HANDED SLOW LEFT-ARM ORTHODOX   IPL           1       0        148
## 15   LEFT HANDED   LEFT-ARM MEDIUM FAST   IPL          23       2          2
## 16  RIGHT HANDED  RIGHT-ARM MEDIUM FAST   IPL          25       1          9
## 17   LEFT HANDED SLOW LEFT-ARM ORTHODOX   IPL         109      23        953
## 18  RIGHT HANDED  RIGHT-ARM FAST MEDIUM   IPL          20       7         32
## 19  RIGHT HANDED       RIGHT-ARM MEDIUM   IPL         132      25        217
## 20   LEFT HANDED   LEFT-ARM MEDIUM FAST   IPL          14       1         16
## 21  RIGHT HANDED  RIGHT-ARM FAST MEDIUM   IPL          24       2         64
## 22  RIGHT HANDED   LEFT-ARM FAST MEDIUM   IPL           5       1          6
## 23   LEFT HANDED              OFF BREAK   IPL          89      26       1974
## 24   LEFT HANDED              LEG BREAK   IPL         150      19       5449
## 25   LEFT HANDED   LEFT-ARM FAST MEDIUM   IPL           3       0        148
## 26  RIGHT HANDED       RIGHT-ARM MEDIUM   IPL          63       5         79
## 27  RIGHT HANDED              OFF BREAK   IPL          80      14        785
## 28   LEFT HANDED              OFF BREAK   IPL          29       1        884
## 29  RIGHT HANDED              OFF BREAK   IPL         213      35       4046
## 30  RIGHT HANDED       RIGHT-ARM MEDIUM   IPL         151      40       1537
## 31  RIGHT HANDED SLOW LEFT-ARM ORTHODOX   IPL           4       2          6
## 32  RIGHT HANDED              LEG BREAK   IPL         100       9       2935
## 33  RIGHT HANDED              OFF BREAK   IPL          97      13       2018
## 34  RIGHT HANDED              OFF BREAK   IPL           3       1         26
## 35  RIGHT HANDED              OFF BREAK   IPL          41       8        511
## 36  RIGHT HANDED  RIGHT-ARM MEDIUM FAST   IPL          92      31       1476
## 37   LEFT HANDED SLOW LEFT-ARM ORTHODOX   IPL          10       6         84
## 38  RIGHT HANDED       RIGHT-ARM MEDIUM   IPL          63      11        187
## 39   LEFT HANDED        LEFT-ARM MEDIUM   IPL          61       5       1452
## 40  RIGHT HANDED       RIGHT-ARM MEDIUM   IPL           1       0        148
## 41   LEFT HANDED SLOW LEFT-ARM ORTHODOX   IPL          17       6         68
## 42   LEFT HANDED  RIGHT-ARM MEDIUM FAST   IPL          12       1         61
## 43  RIGHT HANDED   LEFT-ARM FAST MEDIUM   IPL           5       0        148
## 44  RIGHT HANDED  RIGHT-ARM MEDIUM FAST   IPL          26       3        189
## 45  RIGHT HANDED                UNKNOWN   IPL          13       2        329
## 46  RIGHT HANDED         RIGHT-ARM FAST   IPL         106      15         56
## 47  RIGHT HANDED              OFF BREAK   IPL          19       1         40
## 48  RIGHT HANDED        LEFT-ARM MEDIUM   IPL          86      12        105
## 49  RIGHT HANDED         RIGHT-ARM FAST   IPL          35      10        195
## 50  RIGHT HANDED       RIGHT-ARM MEDIUM   IPL          28       3       1038
## 51  RIGHT HANDED                UNKNOWN   IPL          65       8       1968
## 52   LEFT HANDED  RIGHT-ARM FAST MEDIUM   IPL          12       0        148
## 53   LEFT HANDED         RIGHT-ARM FAST   IPL          50       8        138
## 54  RIGHT HANDED         RIGHT-ARM FAST   IPL          11       3         22
## 55  RIGHT HANDED              OFF BREAK   IPL          63      15       1885
## 56   LEFT HANDED       LEG BREAK GOOGLY   IPL          68      14        317
## 57  RIGHT HANDED  RIGHT-ARM MEDIUM FAST   IPL          14       3          6
## 58  RIGHT HANDED              OFF BREAK   IPL          73       5       1480
## 59  RIGHT HANDED              LEG BREAK   IPL          11       2         24
## 60  RIGHT HANDED        LEFT-ARM MEDIUM   IPL          24       0          1
## 61  RIGHT HANDED       RIGHT-ARM MEDIUM   IPL         178      51       3268
## 62  RIGHT HANDED                UNKNOWN   IPL          94      16       3273
## 63  RIGHT HANDED       RIGHT-ARM MEDIUM   IPL           3       0        148
```

```
## 64   RIGHT HANDED              OFF BREAK   IPL         24        6         186
## 65    LEFT HANDED SLOW LEFT-ARM ORTHODOX   IPL         84       22        1143
## 66    LEFT HANDED SLOW LEFT-ARM CHINAMAN   IPL         45        5          57
## 67    LEFT HANDED   LEFT-ARM MEDIUM FAST   IPL          1        1           0
## 68   RIGHT HANDED              OFF BREAK   IPL          7        3          68
## 69   RIGHT HANDED              LEG BREAK   IPL          9        1         112
## 70   RIGHT HANDED         RIGHT-ARM FAST   IPL         22        6          62
## 71   RIGHT HANDED  RIGHT-ARM FAST MEDIUM   IPL         14        0         148
## 72    LEFT HANDED SLOW LEFT-ARM ORTHODOX   IPL         11        2         181
## 73   RIGHT HANDED       RIGHT-ARM MEDIUM   IPL         53        2        1054
## 74   RIGHT HANDED       RIGHT-ARM MEDIUM   IPL        105       16        1674
## 75   RIGHT HANDED       RIGHT-ARM MEDIUM   IPL        154       27        3560
## 76   RIGHT HANDED          LEFT-ARM FAST   IPL          2        0           0
## 77   RIGHT HANDED       RIGHT-ARM MEDIUM   IPL         56       16         914
## 78   RIGHT HANDED         RIGHT-ARM FAST   IPL          1        0           1
## 79    LEFT HANDED       RIGHT-ARM MEDIUM   IPL          3        0          22
## 80   RIGHT HANDED              OFF BREAK   IPL        100        4        2131
## 81   RIGHT HANDED              LEG BREAK   IPL         18        5          27
## 82   RIGHT HANDED       RIGHT-ARM MEDIUM   IPL         21        2         225
## 83    LEFT HANDED SLOW LEFT-ARM ORTHODOX   IPL          6        1          32
## 84    LEFT HANDED              OFF BREAK   IPL         34        3         666
## 85   RIGHT HANDED              OFF BREAK   IPL         17        2         180
## 86   RIGHT HANDED         RIGHT-ARM FAST   IPL         77       12          69
## 87   RIGHT HANDED  RIGHT-ARM MEDIUM FAST   IPL         50       10          66
## 88   RIGHT HANDED       RIGHT-ARM MEDIUM   IPL        220       73        4746
## 89   RIGHT HANDED       LEG BREAK GOOGLY   IPL         34        2          23
## 90    LEFT HANDED   LEFT-ARM FAST MEDIUM   IPL         38        6           9
## 91   RIGHT HANDED                UNKNOWN   IPL          5        0          33
## 92   RIGHT HANDED         RIGHT-ARM FAST   IPL         38        5          81
## 93   RIGHT HANDED  RIGHT-ARM FAST MEDIUM   IPL          3        1          18
## 94   RIGHT HANDED         RIGHT-ARM FAST   IPL         28        3          31
## 95    LEFT HANDED                UNKNOWN   IPL         33        4         606
## 96    LEFT HANDED              OFF BREAK   IPL         77        7        1820
## 97   RIGHT HANDED         RIGHT-ARM FAST   IPL         37       10         316
## 98   RIGHT HANDED                UNKNOWN   IPL          5        0          50
## 99   RIGHT HANDED        LEFT-ARM MEDIUM   IPL         39        7          24
## 100  RIGHT HANDED  RIGHT-ARM MEDIUM FAST   IPL         34        5           3
## 101  RIGHT HANDED       LEG BREAK GOOGLY   IPL          3        1           7
## 102  RIGHT HANDED              OFF BREAK   IPL         53        0        1305
## 103  RIGHT HANDED       RIGHT-ARM MEDIUM   IPL         19        1         205
## 104   LEFT HANDED                UNKNOWN   IPL         77        5        2256
## 105  RIGHT HANDED       LEG BREAK GOOGLY   IPL         42        4          31
## 106  RIGHT HANDED       RIGHT-ARM MEDIUM   IPL         62        7        1385
## 107  RIGHT HANDED       LEG BREAK GOOGLY   IPL         76       11         222
## 108  RIGHT HANDED       RIGHT-ARM MEDIUM   IPL          1        1           5
## 109  RIGHT HANDED       LEG BREAK GOOGLY   IPL         23        2           8
## 110  RIGHT HANDED              OFF BREAK   IPL        167       22         456
## 111   LEFT HANDED SLOW LEFT-ARM ORTHODOX   IPL        200       63        2386
## 112  RIGHT HANDED         RIGHT-ARM FAST   IPL          5        1           0
## 113   LEFT HANDED              OFF BREAK   IPL         10        1          77
## 114  RIGHT HANDED  RIGHT-ARM MEDIUM FAST   IPL          2        1          25
## 115   LEFT HANDED                UNKNOWN   IPL         84       13        2498
## 116  RIGHT HANDED  RIGHT-ARM MEDIUM FAST   IPL         26       10         153
## 117  RIGHT HANDED              LEG BREAK   IPL         30        3         339
```

```
## 118  RIGHT HANDED       RIGHT-ARM MEDIUM   IPL         193       17        4722
## 119  RIGHT HANDED              OFF BREAK   IPL         213       28        5611
## 120  RIGHT HANDED              OFF BREAK   IPL          22        4         839
## 121  RIGHT HANDED                UNKNOWN   IPL          22        0         334
## 122  RIGHT HANDED       RIGHT-ARM MEDIUM   IPL          99       19          52
## 123  RIGHT HANDED                UNKNOWN   IPL         121       12        3068
## 124  RIGHT HANDED              LEG BREAK   IPL          40        9         441
## 125  RIGHT HANDED  RIGHT-ARM FAST MEDIUM   IPL           2        0          15
## 126   LEFT HANDED SLOW LEFT-ARM ORTHODOX   IPL          13        1          60
## 127  RIGHT HANDED SLOW LEFT-ARM ORTHODOX   IPL          72        8          39
## 128  RIGHT HANDED              OFF BREAK   IPL          11        3         153
## 129  RIGHT HANDED  RIGHT-ARM MEDIUM FAST   IPL          61        6          53
## 130   LEFT HANDED  RIGHT-ARM FAST MEDIUM   IPL           7        2          73
## 131   LEFT HANDED              OFF BREAK   IPL         192       25        5784
## 132   LEFT HANDED                UNKNOWN   IPL          31        9         517
## 133   LEFT HANDED       RIGHT-ARM MEDIUM   IPL          24        4         399
## 134  RIGHT HANDED  RIGHT-ARM FAST MEDIUM   IPL          26        2          48
## 135  RIGHT HANDED              LEG BREAK   IPL          48        7         171
## 136  RIGHT HANDED       LEG BREAK GOOGLY   IPL          87       12        2375
## 137  RIGHT HANDED              OFF BREAK   IPL          58       10        1417
## 138  RIGHT HANDED       RIGHT-ARM MEDIUM   IPL          54        8          20
## 139  RIGHT HANDED                UNKNOWN   IPL           8        2         191
## 140   LEFT HANDED              OFF BREAK   IPL         134       15         954
## 141  RIGHT HANDED       RIGHT-ARM MEDIUM   IPL         115       19        2341
## 142   LEFT HANDED        LEFT-ARM MEDIUM   IPL          24        4           3
## 143  RIGHT HANDED       LEG BREAK GOOGLY   IPL           1        0         148
## 144  RIGHT HANDED              OFF BREAK   IPL           1        0           1
## 145  RIGHT HANDED                UNKNOWN   IPL           1        0           2
## 146  RIGHT HANDED  RIGHT-ARM MEDIUM FAST   IPL          43        6         118
## 147  RIGHT HANDED   LEFT-ARM FAST MEDIUM   IPL          62        7          13
## 148   LEFT HANDED       RIGHT-ARM MEDIUM   IPL           5        1          21
## 149  RIGHT HANDED          LEFT-ARM FAST   IPL           5        0           8
## 150  RIGHT HANDED         RIGHT-ARM FAST   IPL         121       24         122
## 151  RIGHT HANDED  RIGHT-ARM MEDIUM FAST   IPL           3        0         148
## 152  RIGHT HANDED         RIGHT-ARM FAST   IPL          50        8          50
## 153  RIGHT HANDED       LEG BREAK GOOGLY   IPL          31        3          12
## 154   LEFT HANDED       RIGHT-ARM MEDIUM   IPL          10        1         370
## 155  RIGHT HANDED       RIGHT-ARM MEDIUM   IPL          47       12         712
## 156  RIGHT HANDED       RIGHT-ARM MEDIUM   IPL         207       31        6283
## 157  RIGHT HANDED                UNKNOWN   IPL           3        0          19
## 158  RIGHT HANDED              LEG BREAK   IPL           2        1           1
## 159   LEFT HANDED              OFF BREAK   IPL          42       10         217
## 160  RIGHT HANDED                UNKNOWN   IPL         133       22        2110
## 161   LEFT HANDED              LEG BREAK   IPL          13        0         289
## 162  RIGHT HANDED       LEG BREAK GOOGLY   IPL         114       12          32
##     BATTINGAVG BATTINGS.R X4S X6S CATCHESTAKEN STUMPINGSMADE WICKETS STRIKERATE
## 1     15.85000     146.05  12  14           13             0     2.0   24.00000
## 2     17.21000     139.30  17  12            5             0     7.0   18.85000
## 3      5.75000      79.31   0   1            7             0     7.0   27.42000
## 4     29.20000     122.68  12   4            3             0     0.0   24.84143
## 5     31.52000     121.33 417  76           58             0     1.0    6.00000
## 6     24.66000     125.42  13   6            2             0    14.5   24.84143
## 7     17.98821     115.38   2   0            1             0     6.0    8.66000
## 8     29.44000     127.47 324 149           58             2    14.5   24.84143
```

```
## 9     29.31000     178.57 119 143            25            0    72.0   17.51000
## 10     5.20000      63.41   2   1             4            0    24.0   22.04000
## 11    16.00000     114.28   2   1            11            0    14.5   24.84143
## 12     7.00000     116.66   0   0             5            0    34.0   16.11000
## 13     0.00000       0.00   0   0             3            0    14.5   24.84143
## 14    17.98821       0.00   0   0            11            0     1.0   12.00000
## 15     2.00000      33.33   0   0             6            0    30.0   15.23000
## 16     9.00000     150.00   2   0             5            0    29.0   18.82000
## 17    17.32000     125.23  55  44            44            0    95.0   24.15000
## 18    32.00000      91.42   1   1             4            0    17.0   25.00000
## 19     8.34000      96.87  20   3            27            0   142.0   20.76000
## 20     3.20000      64.00   2   0             4            0    14.0   22.28000
## 21     9.14000     112.28   3   3             9            0    25.0   18.36000
## 22     3.00000      75.00   0   0             3            0     1.0  108.00000
## 23    32.90000     136.51 137  90            53            0    14.5   24.84143
## 24    41.59000     139.96 525 201            68            0     0.0   24.84143
## 25    17.98821       0.00   0   0             2            0     2.0   30.00000
## 26    11.28000     138.59   2   6            12            0    59.0   22.44000
## 27    16.70000     129.53  41  38            32            0     9.0   36.11000
## 28    31.57000     125.03  95  22            15            0    14.5   24.84143
## 29    25.77000     129.72 399 112           123           32    14.5   24.84143
## 30    22.94000     130.25 119  65            77            0   167.0   17.44000
## 31     6.00000      50.00   0   0             2            0     1.0   66.00000
## 32    34.94000     131.08 265  96            66            0     0.0   24.84143
## 33    25.22000     151.84 166 112            35            0    22.0   29.18000
## 34    13.00000      78.78   1   2             1            0     1.0   12.00000
## 35    21.29000     121.09  55  11            19            1     5.0   15.60000
## 36    27.33000     153.91  97  98            53            0    42.0   20.69000
## 37    17.98821     120.00   5   3             2            0     5.0   38.40000
## 38    11.00000     134.53  11  11            15            0    78.0   16.20000
## 39    28.47000     136.33 121  74            19            2    14.5   24.84143
## 40    17.98821       0.00   0   0            11            0     1.0   24.00000
## 41    34.00000     128.30   6   3             9            0    12.0   26.00000
## 42     8.71000      92.42   3   2             1            0     8.0   24.75000
## 43    17.98821       0.00   0   0            11            0     5.0   22.80000
## 44    14.53000     121.15  11  11             7            0    35.0   16.42000
## 45    29.90000     129.02  39   9             8            0    14.5   24.84143
## 46    11.20000      96.55   4   1            13            0   130.0   18.63000
## 47    10.00000     111.11   2   1             6            0     8.0   45.75000
## 48    11.66000     109.37   9   3            24            0    85.0   20.89000
## 49    15.00000     157.25  11  14             9            0    46.0   17.93000
## 50    41.52000     142.19  99  46            18            4    14.5   24.84143
## 51    35.14000     150.00 194  90            34            1    14.5   24.84143
## 52    17.98821       0.00   0   0             1            0    12.0   22.50000
## 53    13.80000     102.98  11   4            23            0    76.0   15.00000
## 54     5.50000      66.66   1   0             4            0     5.0   33.60000
## 55    40.10000     131.26 165  56            29            0     0.0   24.84143
## 56    15.09000     115.69  18  14            14            0    59.0   20.71000
## 57     3.00000      60.00   0   0             3            0    13.0   24.07000
## 58    24.26000     128.36 160  39            22            0    14.5   24.84143
## 59     8.00000     114.28   0   2             3            0     8.0   27.00000
## 60     0.33000      20.00   0   0             1            0    32.0   16.96000
## 61    29.98000     149.77 212 214            96            0    65.0   21.60000
## 62    47.43000     136.37 282 134            50            5    14.5   24.84143
```

```
## 63    17.98821      0.00   0   0         11         0     4.0  12.25000
## 64    14.30000    169.09  13  12         10         0    13.0  31.38000
## 65    22.86000    138.54 105  46         28         0    51.0  28.31000
## 66     9.50000     76.00   5   0         11         0    40.0  22.40000
## 67    17.98821      0.00   0   0         11         0     0.0  24.84143
## 68    34.00000     93.15   7   0          5         0     4.0  21.00000
## 69    14.00000    125.84   9   6          7         0     0.0  24.84143
## 70    17.98821    151.22   5   2          3         0    24.0  19.95000
## 71    17.98821      0.00   0   0          2         0    25.0  12.96000
## 72    22.62000    119.86   8   9          1         0     1.0  60.00000
## 73    22.42000    130.60 102  41         13         0    14.5  24.84143
## 74    22.02000    124.09 172  37         37         0     0.0  24.84143
## 75    30.68000    121.83 309 103         75         0    14.5  24.84143
## 76     0.00000      0.00   0   0          1         0     2.0  18.00000
## 77    27.69000    135.81  76  35         12         0    30.0  20.40000
## 78     1.00000     33.33   0   0          1         0     0.0  24.84143
## 79     7.33000     66.66   0   0         11         0    14.5  24.84143
## 80    23.41000    135.47 203  85         40         0    14.5  24.84143
## 81     9.00000     93.10   3   0          3         0    16.0  19.87000
## 82    17.30000    114.21   9  14          7         0    20.0  15.95000
## 83    32.00000    139.13   0   3          2         0     6.0  21.00000
## 84    22.96000    146.37  52  42         11         0    16.0  25.68000
## 85    15.00000    151.26  16   9         11         0    13.0  26.38000
## 86     6.27000     94.52   5   2         11         0    79.0  21.13000
## 87    13.20000     86.84   5   2         16         0    50.0  20.56000
## 88    39.55000    135.83 325 219        126        39    14.5  24.84143
## 89     3.83000     63.88   1   0          7         0    26.0  25.84000
## 90     9.00000     52.94   0   1          5         0    38.0  22.57000
## 91    16.50000    113.79   4   0         11         0    14.5  24.84143
## 92     7.36000    115.71   7   4         12         0    48.0  17.47000
## 93    18.00000    112.50   0   1         11         0     1.0  66.00000
## 94    10.33000     88.57   3   0          6         0    17.0  34.47000
## 95    22.44000    154.98  35  44         11         0    14.5  24.84143
## 96    28.43000    132.46 161  89         15         0     7.0  16.42000
## 97    19.75000    140.44  19  20          7         0    38.0  21.94000
## 98    10.00000     90.90   4   2          2         0    14.5  24.84143
## 99     3.42000     61.53   1   0          6         0    35.0  22.91000
## 100    1.50000     25.00   0   0          7         0    30.0  24.86000
## 101   17.98821     53.84   0   0          1         0     0.0  24.84143
## 102   24.62000    146.30 155  45         11         0    14.5  24.84143
## 103   14.64000    110.81  11   7          6         0    14.5  24.84143
## 104   31.33000    130.93 230  83         53        14    14.5  24.84143
## 105    4.42000     96.87   3   0         11         0    43.0  20.93000
## 106   26.13000    136.31 136  48         23         0     0.0  24.84143
## 107    9.25000    137.03  17  13         20         0    93.0  19.48000
## 108   17.98821    125.00   0   0          1         0     0.0  24.84143
## 109    4.00000     50.00   1   0          4         0    24.0  21.75000
## 110   11.12000    109.88  37  12         37         0   145.0  24.12000
## 111   27.11000    128.14 176  85         81         0   127.0  23.67000
## 112   17.98821      0.00   0   0         11         0     4.0  25.50000
## 113   11.00000    101.31   6   2          6         0    14.5  24.84143
## 114   25.00000     92.59   3   0         11         0     0.0  24.84143
## 115   35.18000    147.46 225 113         56        14    14.5  24.84143
## 116   21.85000    113.33  13   4          6         0    18.0  27.11000
```

```
## 117   16.95000     118.53  29  12           12           0     3.0   44.33000
## 118   27.94000     130.15 462 168           87          32    14.5   24.84143
## 119   31.17000     130.39 491 227           90           0    15.0   22.60000
## 120   46.61000     132.12  80  29           10           0    14.5   24.84143
## 121   17.57000     133.60  31  10           13           0    14.5   24.84143
## 122    8.66000      81.25   4   0           12           0   112.0   19.62000
## 123   29.21000     134.20 236 132           59          10    14.5   24.84143
## 124   23.21000     138.24  50  11            6           0     0.0   24.84143
## 125    7.50000     115.38   1   1           11           0     0.0   24.84143
## 126    8.57000     111.11   4   2            5           0     9.0   13.33000
## 127    2.78000      44.82   2   0           16           0    48.0   29.47000
## 128   21.85000     134.21   9  10            4           0    14.5   24.84143
## 129    6.62000     112.76   5   1           17           0    67.0   18.80000
## 130   14.60000     135.18   2   7            5           0     1.0   41.00000
## 131   34.84000     126.64 654 124           82           0     4.0   12.00000
## 132   25.85000     151.17  34  31           14           0    14.5   24.84143
## 133   22.16000     120.54  24  22            5           0     4.0   23.50000
## 134    6.85000      97.95   4   2           12           0    25.0   20.68000
## 135   12.21000     105.55  17   2           11           0    48.0   19.39000
## 136   31.66000     123.95 196  88           34           0    14.5   24.84143
## 137   31.48000     123.00 137  36           23           0    14.5   24.84143
## 138    5.00000      55.55   1   0            8           0    58.0   20.43000
## 139   38.20000     122.43  10   8            4           1    14.5   24.84143
## 140   15.63000     161.69 106  57           21           0   143.0   21.82000
## 141   28.90000     135.71 261  68           55           0     0.0   24.84143
## 142   17.98821      60.00   0   0            3           0    20.0   25.05000
## 143   17.98821       0.00   0   0           11           0     0.0   24.84143
## 144    1.00000      33.33   0   0           11           0    14.5   24.84143
## 145    2.00000      50.00   0   0           11           0    14.5   24.84143
## 146   13.11000     124.21   8   4           19           0    31.0   30.96000
## 147    4.33000      68.42   0   1           23           0    76.0   18.64000
## 148   21.00000     175.00   2   1            1           0     3.0   34.00000
## 149    2.66000      72.72   0   1            1           0     5.0   21.40000
## 150    8.71000      95.31  11   4           29           0   119.0   21.19000
## 151   17.98821       0.00   0   0           11           0     2.0   36.00000
## 152   10.00000      69.44   2   2            3           0    42.0   22.95000
## 153    4.00000      63.15   0   0            4           0    36.0   20.50000
## 154   41.11000     128.47  37  14            7           0     3.0   17.00000
## 155   26.37000     126.24  45  28           21           0     9.0   25.44000
## 156   37.39000     129.94 546 210           84           0     4.0   62.75000
## 157    6.33000      73.07   1   1            0           2    14.5   24.84143
## 158    1.00000      50.00   0   0           11           0     0.0   24.84143
## 159   13.56000     111.28  17   6            9           0    27.0   27.77000
## 160   24.53000     128.73 191  69           69          20    14.5   24.84143
## 161   22.23000     136.32  34  12            4           0    14.5   24.84143
## 162    5.33000      41.02   0   0           24           0   139.0   17.61000
##     ECONOMYRATE BASE.PRICE VALUEINCR
## 1     13.120000        100      4.00
## 2      8.000000         20      6.50
## 3      9.620000        150      1.90
## 4      5.750000        100      2.60
## 5      5.000000        100      1.00
## 6      8.656393        150      1.50
## 7     10.030000         75      2.40
```

```
## 8     8.656393      200     6.75
## 9     9.040000      100    12.00
## 10    9.230000      100     0.50
## 11    8.656393       20     0.20
## 12    7.650000      100     6.50
## 13    8.656393       20     3.40
## 14    5.500000       20     0.20
## 15    8.780000      100     4.00
## 16    8.230000       20    10.00
## 17    7.220000      100     9.00
## 18    9.790000       30     0.30
## 19    7.300000      200     4.20
## 20    8.190000       50     4.20
## 21    9.120000      200     3.60
## 22    8.500000      100     2.60
## 23    8.656393      100     3.00
## 24   12.000000      200     6.25
## 25    9.500000      200     2.00
## 26    7.800000      200    14.00
## 27    8.450000       40     5.75
## 28    8.656393      200     7.75
## 29    8.656393      200     5.50
## 30    8.360000      200     4.40
## 31    8.180000       75     0.75
## 32   16.000000      200     7.00
## 33    8.550000      100    11.00
## 34   10.000000      150     1.50
## 35    7.460000      100     0.50
## 36    9.060000      100    15.00
## 37    7.120000       20     3.80
## 38    8.580000      200    10.75
## 39    8.656393      200    15.25
## 40    9.750000       20     0.25
## 41    8.880000       20     0.20
## 42    9.240000      150     1.50
## 43    8.680000       75     0.75
## 44    8.200000      150     8.75
## 45    8.656393      200     2.00
## 46    7.420000      100    12.00
## 47    6.860000      100     1.70
## 48    8.740000       75     1.30
## 49    7.130000      200     8.00
## 50    8.656393      150     6.75
## 51    8.656393      100    10.00
## 52    7.930000      200     7.75
## 53    8.210000      200     9.25
## 54    9.140000       40     1.10
## 55   10.330000      100    14.00
## 56    7.900000       50     0.50
## 57    9.410000       20     4.00
## 58    8.656393       50     1.40
## 59    9.660000      100     0.30
## 60    8.680000      100     5.25
## 61    8.780000      100     6.00
```

```
## 62     8.656393         100      17.00
## 63    11.380000         100       0.20
## 64     8.260000         100       0.90
## 65     7.360000         200       8.25
## 66     8.270000         100       2.00
## 67     8.000000          20       0.20
## 68     7.210000          20       0.65
## 69    13.000000         100      11.50
## 70     8.110000         200      10.00
## 71     8.290000         100       0.50
## 72     7.400000          40       0.95
## 73     8.656393          20       0.20
## 74    13.000000          50       1.10
## 75     8.656393         100       4.60
## 76     7.500000          50       4.20
## 77     9.500000         100       9.20
## 78    12.250000         200       7.50
## 79     8.656393         200       2.40
## 80     8.656393         100      12.00
## 81     8.540000          50       0.65
## 82     7.890000         200       6.50
## 83     7.000000         100       1.90
## 84     6.840000         100       8.00
## 85     7.130000         100       1.00
## 86     8.620000         100       6.25
## 87     8.380000         100       7.00
## 88     8.656393         100      12.00
## 89     7.860000          20       1.60
## 90     7.830000         200       2.00
## 91     8.656393         100       0.20
## 92     7.520000         200       2.00
## 93     8.180000          75       0.75
## 94     8.470000          75       2.60
## 95     8.656393         150      10.75
## 96     8.030000         100       8.00
## 97     8.230000         200       7.25
## 98     8.656393          20       0.60
## 99     8.790000          20       0.20
## 100    9.260000         100      10.00
## 101    9.000000          20       0.50
## 102    8.656393         100       7.50
## 103    8.656393          20       0.20
## 104    8.656393         200       6.75
## 105    7.440000          75       5.25
## 106   12.000000          40       8.50
## 107    6.330000         100      15.00
## 108   10.500000          20       0.20
## 109    6.960000         100       4.00
## 110    6.910000         100       5.00
## 111    7.610000         100      16.00
## 112    9.940000         100       1.00
## 113    8.656393          20       0.55
## 114    7.330000          20       0.20
## 115    8.656393         100      16.00
```

```
## 116    7.860000         50      0.55
## 117    9.960000         30      3.80
## 118    8.656393        200      2.00
## 119    8.010000        100     16.00
## 120    8.656393        100      6.00
## 121    8.656393        200      2.00
## 122    7.770000         50      0.50
## 123    8.656393        100     14.00
## 124   18.000000         20      0.20
## 125   11.400000         75      2.40
## 126    6.800000        100      2.40
## 127    7.560000         50      0.50
## 128    8.656393         40      9.00
## 129    8.890000        200     10.75
## 130    8.630000        100      1.00
## 131    8.250000        200      8.25
## 132    8.656393        150      8.50
## 133    8.290000         50      4.00
## 134    8.290000         40      7.25
## 135    8.030000         20      0.75
## 136    8.656393        200     12.25
## 137    8.656393        100      8.00
## 138    8.580000        100      0.75
## 139    8.656393        100      2.00
## 140    6.740000        100      6.00
## 141    8.000000        100      8.00
## 142    8.230000        100      4.00
## 143    9.420000         20      0.20
## 144    8.656393         40      8.25
## 145    8.656393         50      0.50
## 146    8.670000        150      1.50
## 147    8.390000        200      8.00
## 148   11.290000         20      0.20
## 149    8.570000        100      1.50
## 150    8.510000        200      2.00
## 151    8.000000        100      4.00
## 152    8.890000         50      0.50
## 153    6.820000        100      8.00
## 154    8.110000        100      8.00
## 155    8.620000         50      1.40
## 156    8.790000        100     15.00
## 157    8.656393         20      0.50
## 158   10.000000        100     10.75
## 159    6.930000        150      8.75
## 160    8.656393        100      1.90
## 161    8.656393        100      4.00
## 162    7.590000        200      6.50
```

```
###########################################################################################
####                         UNIVARIATE OUTLIER DETECTION                          ########


########################    Matches Played   #################################
```

```r
# We Calculate the quartiles using the quantiles() function:
q1_MATCHPLAYED <- quantile(IPLDATA4$MATCHPLAYED, probs = 0.25)
q3_MATCHPLAYED <- quantile(IPLDATA4$MATCHPLAYED, probs = 0.75)
iqr_MATCHPLAYED <- q3_MATCHPLAYED - q1_MATCHPLAYED

# Once we have the IQR, we can calculate the upper and lower fence:
# Use any of the above methods to determine Q3 and Q1, and then:
MATCHPLAYED_lower_fence <- q1_MATCHPLAYED - (1.5 * iqr_MATCHPLAYED) # Recall that the lower fence is Q1
MATCHPLAYED_upper_fence <- q3_MATCHPLAYED + (1.5 * iqr_MATCHPLAYED) # Recall that the upper fence is Q3
MATCHPLAYED_up_outliers <- which(IPLDATA4$MATCHPLAYED > MATCHPLAYED_upper_fence)
MATCHPLAYED_low_outliers <- which(IPLDATA4$MATCHPLAYED < MATCHPLAYED_lower_fence)
length(MATCHPLAYED_up_outliers)
```

```
## [1] 9
```

```r
length(MATCHPLAYED_low_outliers)
```

```
## [1] 0
```

```r
MATCHPLAYED_low_outliers
```

```
## integer(0)
```

```r
MATCHPLAYED_up_outliers
```

```
## [1]   8  29  61  88 111 118 119 131 156
```

```r
IPLDATA4$MATCHPLAYED[MATCHPLAYED_up_outliers]
```

```
## [1] 175 213 178 220 200 193 213 192 207
```

```r
# 9 Outliers found.
# Rows: 8   29   61   88 111 118 119 131 156
# Values: 175 213 178 220 200 193 213 192 207



########################     NOTOUTS      ######################


# We Calculate the quartiles using the quantiles() function:
q1_NOTOUTS <- quantile(IPLDATA4$NOTOUTS, probs = 0.25)
q3_NOTOUTS <- quantile(IPLDATA4$NOTOUTS, probs = 0.75)
iqr_NOTOUTS <- q3_NOTOUTS - q1_NOTOUTS

# Once we have the IQR, we can calculate the upper and lower fence:
# Use any of the above methods to determine Q3 and Q1, and then:
NOTOUTS_lower_fence <- q1_NOTOUTS - (1.5 * iqr_NOTOUTS) # Recall that the lower fence is Q1 minus the i
NOTOUTS_upper_fence <- q3_NOTOUTS + (1.5 * iqr_NOTOUTS) # Recall that the upper fence is Q3 plus the in
NOTOUTS_up_outliers <- which(IPLDATA4$NOTOUTS > NOTOUTS_upper_fence)
NOTOUTS_low_outliers <- which(IPLDATA4$NOTOUTS < NOTOUTS_lower_fence)
length(NOTOUTS_up_outliers)
```

```
## [1] 10
```

```r
length(NOTOUTS_low_outliers)
```

```
## [1] 0
```

```r
NOTOUTS_low_outliers
```

```
## integer(0)
```

```r
NOTOUTS_up_outliers
```

```
##  [1]   8  29  30  36  61  75  88 111 119 156
```

```r
IPLDATA4$NOTOUTS[NOTOUTS_up_outliers]
```

```
##  [1] 31 35 40 31 51 27 73 63 28 31
```

```r
# 10 Outliers found.
# Rows: 8   29   30   36   61   75   88 111 119 156
# Values: 31 35 40 31 51 27 73 63 28 31



########################     RUNSSCORED      #######################


# We Calculate the quartiles using the quantiles() function:
q1_RUNSSCORED <- quantile(IPLDATA4$RUNSSCORED, probs = 0.25)
q3_RUNSSCORED <- quantile(IPLDATA4$RUNSSCORED, probs = 0.75)
iqr_RUNSSCORED <- q3_RUNSSCORED - q1_RUNSSCORED

# Once we have the IQR, we can calculate the upper and lower fence:
# Use any of the above methods to determine Q3 and Q1, and then:
RUNSSCORED_lower_fence <- q1_RUNSSCORED - (1.5 * iqr_RUNSSCORED) # Recall that the lower fence is Q1 mi
RUNSSCORED_upper_fence <- q3_RUNSSCORED + (1.5 * iqr_RUNSSCORED) # Recall that the upper fence is Q3 pl
RUNSSCORED_up_outliers <- which(IPLDATA4$RUNSSCORED > RUNSSCORED_upper_fence)
RUNSSCORED_low_outliers <- which(IPLDATA4$RUNSSCORED < RUNSSCORED_lower_fence)
length(RUNSSCORED_up_outliers)
```

```
## [1] 19
```

```r
length(RUNSSCORED_low_outliers)
```

```
## [1] 0
```

```r
RUNSSCORED_low_outliers
```

```
## integer(0)
```

```
RUNSSCORED_up_outliers
```

```
##  [1]   5   8  24  29  32  61  62  75  88 104 111 115 118 119 123 131 136 141 156
```

```
IPLDATA4$RUNSSCORED[RUNSSCORED_up_outliers]
```

```
##  [1] 3941 3916 5449 4046 2935 3268 3273 3560 4746 2256 2386 2498 4722 5611 3068
## [16] 5784 2375 2341 6283
```

```
# no outliers




########################     BATTINGAVG      #######################


# We Calculate the quartiles using the quantiles() function:
q1_BATTINGAVG <- quantile(IPLDATA4$BATTINGAVG, probs = 0.25)
q3_BATTINGAVG <- quantile(IPLDATA4$BATTINGAVG, probs = 0.75)
iqr_BATTINGAVG <- q3_BATTINGAVG - q1_BATTINGAVG

# Once we have the IQR, we can calculate the upper and lower fence:
# Use any of the above methods to determine Q3 and Q1, and then:
BATTINGAVG_lower_fence <- q1_BATTINGAVG - (1.5 * iqr_BATTINGAVG) # Recall that the lower fence is Q1 mi
BATTINGAVG_upper_fence <- q3_BATTINGAVG + (1.5 * iqr_BATTINGAVG) # Recall that the upper fence is Q3 pl
BATTINGAVG_up_outliers <- which(IPLDATA4$BATTINGAVG > BATTINGAVG_upper_fence)
BATTINGAVG_low_outliers <- which(IPLDATA4$BATTINGAVG < BATTINGAVG_lower_fence)
length(BATTINGAVG_up_outliers)
```

```
## [1] 0
```

```
length(BATTINGAVG_low_outliers)
```

```
## [1] 0
```

```
BATTINGAVG_low_outliers
```

```
## integer(0)
```

```
BATTINGAVG_up_outliers
```

```
## integer(0)
```

```
IPLDATA4$BATTINGAVG[BATTINGAVG_up_outliers]
```

```
## numeric(0)
```

```
# No Outliers found

#######################     BATTINGS.R      ######################


# We Calculate the quartiles using the quantiles() function:
q1_BATTINGS.R <- quantile(IPLDATA4$BATTINGS.R, probs = 0.25)
q3_BATTINGS.R <- quantile(IPLDATA4$BATTINGS.R, probs = 0.75)
iqr_BATTINGS.R <- q3_BATTINGS.R - q1_BATTINGS.R


# Once we have the IQR, we can calculate the upper and lower fence:
# Use any of the above methods to determine Q3 and Q1, and then:
BATTINGS.R_lower_fence <- q1_BATTINGS.R - (1.5 * iqr_BATTINGS.R) # Recall that the lower fence is Q1 mi
BATTINGS.R_upper_fence <- q3_BATTINGS.R + (1.5 * iqr_BATTINGS.R) # Recall that the upper fence is Q3 pl
BATTINGS.R_up_outliers <- which(IPLDATA4$BATTINGS.R > BATTINGS.R_upper_fence)
BATTINGS.R_low_outliers <- which(IPLDATA4$BATTINGS.R < BATTINGS.R_lower_fence)
length(BATTINGS.R_up_outliers)
```

```
## [1] 0
```

```
length(BATTINGS.R_low_outliers)
```

```
## [1] 0
```

```
BATTINGS.R_low_outliers
```

```
## integer(0)
```

```
BATTINGS.R_up_outliers
```

```
## integer(0)
```

```
IPLDATA4$BATTINGS.R[BATTINGS.R_up_outliers]
```

```
## numeric(0)
```

```
##  No Outliers Found




#######################     X4S      ######################


# We Calculate the quartiles using the quantiles() function:
q1_X4S <- quantile(IPLDATA4$X4S, probs = 0.25)
q3_X4S <- quantile(IPLDATA4$X4S, probs = 0.75)
iqr_X4S <- q3_X4S - q1_X4S
```

```
# Once we have the IQR, we can calculate the upper and lower fence:
# Use any of the above methods to determine Q3 and Q1, and then:
X4S_lower_fence <- q1_X4S - (1.5 * iqr_X4S) # Recall that the lower fence is Q1 minus the inter-quartil
X4S_upper_fence <- q3_X4S + (1.5 * iqr_X4S) # Recall that the upper fence is Q3 plus the inter-quartile
X4S_up_outliers <- which(IPLDATA4$X4S > X4S_upper_fence)
X4S_low_outliers <- which(IPLDATA4$X4S < X4S_lower_fence)
length(X4S_up_outliers)
```

```
## [1] 18
```

```
length(X4S_low_outliers)
```

```
## [1] 0
```

```
X4S_low_outliers
```

```
## integer(0)
```

```
X4S_up_outliers
```

```
##  [1]   5   8  24  29  32  61  62  75  80  88 104 115 118 119 123 131 141 156
```

```
IPLDATA4$X4S[X4S_up_outliers]
```

```
##  [1] 417 324 525 399 265 212 282 309 203 325 230 225 462 491 236 654 261 546
```

```
# 18 Outliers found.
# Rows: 5   8  24  29  32  61  62  75  80  88 104 115 118 119 123 131 141 156
# Values: 417 324 525 399 265 212 282 309 203 325 230 225 462 491 236 654 261 546




########################     X6S      #######################


# We Calculate the quartiles using the quantiles() function:
q1_X6S <- quantile(IPLDATA4$X6S, probs = 0.25)
q3_X6S <- quantile(IPLDATA4$X6S, probs = 0.75)
iqr_X6S <- q3_X6S - q1_X6S


# Once we have the IQR, we can calculate the upper and lower fence:
# Use any of the above methods to determine Q3 and Q1, and then:
X6S_lower_fence <- q1_X6S - (1.5 * iqr_X6S) # Recall that the lower fence is Q1 minus the inter-quartil
X6S_upper_fence <- q3_X6S + (1.5 * iqr_X6S) # Recall that the upper fence is Q3 plus the inter-quartile
X6S_up_outliers <- which(IPLDATA4$X6S > X6S_upper_fence)
X6S_low_outliers <- which(IPLDATA4$X6S < X6S_lower_fence)
length(X6S_up_outliers)
```

```
## [1] 17
```

```
length(X6S_low_outliers)
```

```
## [1] 0
```

```
X6S_low_outliers
```

```
## integer(0)
```

```
X6S_up_outliers
```

```
##  [1]   8   9  24  29  32  33  36  61  62  75  88 115 118 119 123 131 156
```

```
IPLDATA4$X6S[X6S_up_outliers]
```

```
##  [1] 149 143 201 112  96 112  98 214 134 103 219 113 168 227 132 124 210
```

```
# 17 Outliers found.
# Rows: 8   9  24  29  32  33  36  61  62  75  88 115 118 119 123 131 156
# Values: 149 143 201 112  96 112  98 214 134 103 219 113 168 227 132 124 210





########################     CATCHESTAKEN      #######################


# We Calculate the quartiles using the quantiles() function:
q1_CATCHESTAKEN <- quantile(IPLDATA4$CATCHESTAKEN, probs = 0.25)
q3_CATCHESTAKEN <- quantile(IPLDATA4$CATCHESTAKEN, probs = 0.75)
iqr_CATCHESTAKEN <- q3_CATCHESTAKEN - q1_CATCHESTAKEN


# Once we have the IQR, we can calculate the upper and lower fence:
# Use any of the above methods to determine Q3 and Q1, and then:
CATCHESTAKEN_lower_fence <- q1_CATCHESTAKEN - (1.5 * iqr_CATCHESTAKEN) # Recall that the lower fence is
CATCHESTAKEN_upper_fence <- q3_CATCHESTAKEN + (1.5 * iqr_CATCHESTAKEN) # Recall that the upper fence is
CATCHESTAKEN_up_outliers <- which(IPLDATA4$CATCHESTAKEN > CATCHESTAKEN_upper_fence)
CATCHESTAKEN_low_outliers <- which(IPLDATA4$CATCHESTAKEN < CATCHESTAKEN_lower_fence)
length(CATCHESTAKEN_up_outliers)
```

```
## [1] 22
```

```
length(CATCHESTAKEN_low_outliers)
```

```
## [1] 0
```

```
CATCHESTAKEN_low_outliers
```

```
## integer(0)
```

```
CATCHESTAKEN_up_outliers
```

```
##  [1]   5   8  23  24  29  30  32  36  61  62  75  88 104 111 115 118 119 123 131
## [20] 141 156 160
```

```
IPLDATA4$CATCHESTAKEN[CATCHESTAKEN_up_outliers]
```

```
##  [1]  58  58  53  68 123  77  66  53  96  50  75 126  53  81  56  87  90  59  82
## [20]  55  84  69
```

```
# 22 Outliers found.
# Rows: 5   8  23  24  29  30  32  36  61  62  75  88 104 111 115 118 119 123 131 141 156 160
# Values: 58  58  53  68 123  77  66  53  96  50  75 126  53  81  56  87  90  59  82  55  84  69



########################     STUMPINGSMADE      #######################


# We Calculate the quartiles using the quantiles() function:
q1_STUMPINGSMADE <- quantile(IPLDATA4$STUMPINGSMADE, probs = 0.25)
q3_STUMPINGSMADE <- quantile(IPLDATA4$STUMPINGSMADE, probs = 0.75)
iqr_STUMPINGSMADE <- q3_STUMPINGSMADE - q1_STUMPINGSMADE


# Once we have the IQR, we can calculate the upper and lower fence:
# Use any of the above methods to determine Q3 and Q1, and then:
STUMPINGSMADE_lower_fence <- q1_STUMPINGSMADE - (1.5 * iqr_STUMPINGSMADE) # Recall that the lower fence
STUMPINGSMADE_upper_fence <- q3_STUMPINGSMADE + (1.5 * iqr_STUMPINGSMADE) # Recall that the upper fence
STUMPINGSMADE_up_outliers <- which(IPLDATA4$STUMPINGSMADE > STUMPINGSMADE_upper_fence)
STUMPINGSMADE_low_outliers <- which(IPLDATA4$STUMPINGSMADE < STUMPINGSMADE_lower_fence)
length(STUMPINGSMADE_up_outliers)
```

```
## [1] 15
```

```
length(STUMPINGSMADE_low_outliers)
```

```
## [1] 0
```

```
STUMPINGSMADE_low_outliers
```

```
## integer(0)
```

```r
STUMPINGSMADE_up_outliers
```

```
##  [1]   8  29  35  39  50  51  62  88 104 115 118 123 139 157 160
```

```r
IPLDATA4$STUMPINGSMADE[STUMPINGSMADE_up_outliers]
```

```
##  [1]  2 32  1  2  4  1  5 39 14 14 32 10  1  2 20
```

```r
#No Outliers Found





#######################     WICKETS      ######################


# We Calculate the quartiles using the quantiles() function:
q1_WICKETS <- quantile(IPLDATA4$WICKETS, probs = 0.25)
q3_WICKETS <- quantile(IPLDATA4$WICKETS, probs = 0.75)
iqr_WICKETS <- q3_WICKETS - q1_WICKETS


# Once we have the IQR, we can calculate the upper and lower fence:
# Use any of the above methods to determine Q3 and Q1, and then:
WICKETS_lower_fence <- q1_WICKETS - (1.5 * iqr_WICKETS) # Recall that the lower fence is Q1 minus the i
WICKETS_upper_fence <- q3_WICKETS + (1.5 * iqr_WICKETS) # Recall that the upper fence is Q3 plus the in
WICKETS_up_outliers <- which(IPLDATA4$WICKETS > WICKETS_upper_fence)
WICKETS_low_outliers <- which(IPLDATA4$WICKETS < WICKETS_lower_fence)
length(WICKETS_up_outliers)
```

```
## [1] 17
```

```r
length(WICKETS_low_outliers)
```

```
## [1] 0
```

```r
WICKETS_low_outliers
```

```
## integer(0)
```

```r
WICKETS_up_outliers
```

```
##  [1]   9  17  19  30  38  46  48  53  86 107 110 111 122 140 147 150 162
```

```r
IPLDATA4$WICKETS[WICKETS_up_outliers]
```

```
##  [1]  72  95 142 167  78 130  85  76  79  93 145 127 112 143  76 119 139
```

```
# 17 Outliers found.
# Rows: 9  17  19  30  38  46  48  53  86 107 110 111 122 140 147 150 162
# Values: 72  95 142 167  78 130  85  76  79  93 145 127 112 143  76 119 139




########################     STRIKERATE      ######################


# We Calculate the quartiles using the quantiles() function:
q1_STRIKERATE <- quantile(IPLDATA4$STRIKERATE, probs = 0.25)
q3_STRIKERATE <- quantile(IPLDATA4$STRIKERATE, probs = 0.75)
iqr_STRIKERATE <- q3_STRIKERATE - q1_STRIKERATE


# Once we have the IQR, we can calculate the upper and lower fence:
# Use any of the above methods to determine Q3 and Q1, and then:
STRIKERATE_lower_fence <- q1_STRIKERATE - (1.5 * iqr_STRIKERATE) # Recall that the lower fence is Q1 mi
STRIKERATE_upper_fence <- q3_STRIKERATE + (1.5 * iqr_STRIKERATE) # Recall that the upper fence is Q3 pl
STRIKERATE_up_outliers <- which(IPLDATA4$STRIKERATE > STRIKERATE_upper_fence)
STRIKERATE_low_outliers <- which(IPLDATA4$STRIKERATE < STRIKERATE_lower_fence)
length(STRIKERATE_up_outliers)
```

```
## [1] 15
```

```
length(STRIKERATE_low_outliers)
```

```
## [1] 8
```

```
STRIKERATE_low_outliers
```

```
## [1]   5   7  14  34  63  71 126 131
```

```
STRIKERATE_up_outliers
```

```
##  [1]  22  27  31  37  47  54  64  72  93  94 117 130 148 151 156
```

```
IPLDATA4$STRIKERATE[STRIKERATE_up_outliers]
```

```
##  [1] 108.00  36.11  66.00  38.40  45.75  33.60  31.38  60.00  66.00  34.47
## [11]  44.33  41.00  34.00  36.00  62.75
```

```
IPLDATA4$STRIKERATE[STRIKERATE_low_outliers]
```

```
## [1]  6.00  8.66 12.00 12.00 12.25 12.96 13.33 12.00
```

```r
# 23 Outliers found.
# Rows: 5   7  14  34  63  71 126 131 22  27  31  37  47  54  64  72  93  94 117 130 148 151 156
# Values: 6.00  8.66 12.00 12.00 12.25 12.96 13.33 12.00 108.00  36.11  66.00  38.40  45.75  33.60  31..



########################     ECONOMYRATE      ######################


# We Calculate the quartiles using the quantiles() function:
q1_ECONOMYRATE <- quantile(IPLDATA4$ECONOMYRATE, probs = 0.25)
q3_ECONOMYRATE <- quantile(IPLDATA4$ECONOMYRATE, probs = 0.75)
iqr_ECONOMYRATE <- q3_ECONOMYRATE - q1_ECONOMYRATE


# Once we have the IQR, we can calculate the upper and lower fence:
# Use any of the above methods to determine Q3 and Q1, and then:
ECONOMYRATE_lower_fence <- q1_ECONOMYRATE - (1.5 * iqr_ECONOMYRATE) # Recall that the lower fence is Q1
ECONOMYRATE_upper_fence <- q3_ECONOMYRATE + (1.5 * iqr_ECONOMYRATE) # Recall that the upper fence is Q3
ECONOMYRATE_up_outliers <- which(IPLDATA4$ECONOMYRATE > ECONOMYRATE_upper_fence)
ECONOMYRATE_low_outliers <- which(IPLDATA4$ECONOMYRATE < ECONOMYRATE_lower_fence)
length(ECONOMYRATE_up_outliers)
```

```
## [1] 18
```

```r
length(ECONOMYRATE_low_outliers)
```

```
## [1] 8
```

```r
ECONOMYRATE_low_outliers
```

```
## [1]   4   5  14  84 107 126 140 153
```

```r
ECONOMYRATE_up_outliers
```

```
##  [1]   1   7  24  32  34  55  63  69  74  78 106 108 112 117 124 125 148 158
```

```r
IPLDATA4$ECONOMYRATE[ECONOMYRATE_up_outliers]
```

```
##  [1] 13.12 10.03 12.00 16.00 10.00 10.33 11.38 13.00 13.00 12.25 12.00 10.50
## [13]  9.94  9.96 18.00 11.40 11.29 10.00
```

```r
IPLDATA4$ECONOMYRATE[ECONOMYRATE_low_outliers]
```

```
## [1] 5.75 5.00 5.50 6.84 6.33 6.80 6.74 6.82
```

```
# 26 Outliers found.
# Rows: 4   5  14  84 107 126 140 153 1   7  24  32  34  55  63  69  74  78 106 108 112 117 124 125 148
# Values: 5.75 5.00 5.50 6.84 6.33 6.80 6.74 6.82 13.12 10.03 12.00 16.00 10.00 10.33 11.38 13.00 13.00





########################     BASE.PRICE      #######################


# We Calculate the quartiles using the quantiles() function:
q1_BASE.PRICE <- quantile(IPLDATA4$BASE.PRICE, probs = 0.25)
q3_BASE.PRICE <- quantile(IPLDATA4$BASE.PRICE, probs = 0.75)
iqr_BASE.PRICE <- q3_BASE.PRICE - q1_BASE.PRICE


# Once we have the IQR, we can calculate the upper and lower fence:
# Use any of the above methods to determine Q3 and Q1, and then:
BASE.PRICE_lower_fence <- q1_BASE.PRICE - (1.5 * iqr_BASE.PRICE) # Recall that the lower fence is Q1 mi
BASE.PRICE_upper_fence <- q3_BASE.PRICE + (1.5 * iqr_BASE.PRICE) # Recall that the upper fence is Q3 pl
BASE.PRICE_up_outliers <- which(IPLDATA4$BASE.PRICE > BASE.PRICE_upper_fence)
BASE.PRICE_low_outliers <- which(IPLDATA4$BASE.PRICE < BASE.PRICE_lower_fence)
length(BASE.PRICE_up_outliers)
```

```
## [1] 0
```

```
length(BASE.PRICE_low_outliers)
```

```
## [1] 0
```

```
BASE.PRICE_low_outliers
```

```
## integer(0)
```

```
BASE.PRICE_up_outliers
```

```
## integer(0)
```

```
IPLDATA4$BASE.PRICE[BASE.PRICE_up_outliers]
```

```
## integer(0)
```

```
IPLDATA4$BASE.PRICE[BASE.PRICE_low_outliers]
```

```
## integer(0)
```

```r
# No Outliers




#######################     VALUEINCR      ######################


# We Calculate the quartiles using the quantiles() function:
q1_VALUEINCR <- quantile(IPLDATA4$VALUEINCR, probs = 0.25)
q3_VALUEINCR <- quantile(IPLDATA4$VALUEINCR, probs = 0.75)
iqr_VALUEINCR <- q3_VALUEINCR - q1_VALUEINCR


# Once we have the IQR, we can calculate the upper and lower fence:
# Use any of the above methods to determine Q3 and Q1, and then:
VALUEINCR_lower_fence <- q1_VALUEINCR - (1.5 * iqr_VALUEINCR) # Recall that the lower fence is Q1 minus
VALUEINCR_upper_fence <- q3_VALUEINCR + (1.5 * iqr_VALUEINCR) # Recall that the upper fence is Q3 plus
VALUEINCR_up_outliers <- which(IPLDATA4$VALUEINCR > VALUEINCR_upper_fence)
VALUEINCR_low_outliers <- which(IPLDATA4$VALUEINCR < VALUEINCR_lower_fence)
length(VALUEINCR_up_outliers)
```

```
## [1] 0
```

```r
length(VALUEINCR_low_outliers)
```

```
## [1] 0
```

```r
VALUEINCR_low_outliers
```

```
## integer(0)
```

```r
VALUEINCR_up_outliers
```

```
## integer(0)
```

```r
IPLDATA4$VALUEINCR[VALUEINCR_up_outliers]
```

```
## numeric(0)
```

```r
IPLDATA4$VALUEINCR[VALUEINCR_low_outliers]
```

```
## numeric(0)
```

```r
# No Outliers Found
```

```r
##################################################################################
###########               Impute outliers                    ################


# Function
# This is a user-defined function, that will appear in our environment for our use
# It will cap outliers.

cap <- function(x){
  quantiles <- quantile( x, c(0.05, 0.25, 0.75, 0.95 ) , na.rm = TRUE)
  x[ x < quantiles[2] - 1.5 * IQR(x, na.rm = TRUE) ] <- quantiles[1]
  x[ x > quantiles[3] + 1.5 * IQR(x, na.rm = TRUE) ] <- quantiles[4]
  x
}



################## MATCHPLAYED
MATCHPLAYED_cap <- as.data.frame(sapply(IPLDATA4$MATCHPLAYED, FUN = cap))

summary(IPLDATA4$MATCHPLAYED)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00   10.00   29.00   50.04   75.25  220.00
```

```r
summary(MATCHPLAYED_cap)
```

```
##  sapply(IPLDATA4$MATCHPLAYED, FUN = cap)
##  Min.   :  1.00
##  1st Qu.: 10.00
##  Median : 29.00
##  Mean   : 50.04
##  3rd Qu.: 75.25
##  Max.   :220.00
```

```r
IPLDATA4$MATCHPLAYED[MATCHPLAYED_up_outliers] <- median(IPLDATA4$MATCHPLAYED)
IPLDATA4$MATCHPLAYED
```

```
##   [1]  23  22   9   6 151   6   3  29  84  29   1  24   2   1  23  25 109  20
##  [19] 132  14  24   5  89 150   3  63  80  29  29 151   4 100  97   3  41  92
##  [37]  10  63  61   1  17  12   5  26  13 106  19  86  35  28  65  12  50  11
##  [55]  63  68  14  73  11  24  29  94   3  24  84  45   1   7   9  22  14  11
##  [73]  53 105 154   2  56   1   3 100  18  21   6  34  17  77  50  29  34  38
##  [91]   5  38   3  28  33  77  37   5  39  34   3  53  19  77  42  62  76   1
## [109]  23 167  29   5  10   2  84  26  30  29  29  22  22  99 121  40   2  13
## [127]  72  11  61   7  29  31  24  26  48  87  58  54   8 134 115  24   1   1
## [145]   1  43  62   5   5 121   3  50  31  10  47  29   3   2  42 133  13 114
```

```r
#Impute with median
```

```r
################## NOTOUTS
NOTOUTS_cap <- as.data.frame(sapply(IPLDATA4$NOTOUTS, FUN = cap))

summary(IPLDATA4$NOTOUTS)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   1.000   4.000   8.179  11.000  73.000
```

```r
summary(NOTOUTS_cap)
```

```
##  sapply(IPLDATA4$NOTOUTS, FUN = cap)
##  Min.   : 0.000
##  1st Qu.: 1.000
##  Median : 4.000
##  Mean   : 8.179
##  3rd Qu.:11.000
##  Max.   :73.000
```

```r
IPLDATA4$NOTOUTS[NOTOUTS_up_outliers] <- median(IPLDATA4$NOTOUTS)
IPLDATA4$NOTOUTS
```

```
##   [1]  4  6  2  1 16  0  2  4 12  2  0  4  0  0  2  1 23  7 25  1  2  1 26 19  0
##  [26]  5 14  1  4  4  2  9 13  1  8  4  6 11  5  0  6  1  0  3  2 15  1 12 10  3
##  [51]  8  0  8  3 15 14  3  5  2  0  4 16  0  6 22  5  1  3  1  6  0  2  2 16  4
##  [76]  0 16  0  0  4  5  2  1  3  2 12 10  4  2  6  0  5  1  3  4  7 10  0  7  5
## [101]  1  0  1  5  4  7 11  1  2 22  4  1  1  1 13 10  3 17  4  4  0 19 12  9  0
## [126]  1  8  3  6  2 25  9  4  2  7 12 10  8  2 15 19  4  0  0  0  6  7  1  0 24
## [151]  0  8  3  1 12  4  0  1 10 22  0 12
```

```r
#Impute with median




################## X4S
X4S_cap <- as.data.frame(sapply(IPLDATA4$X4S, FUN = cap))

summary(IPLDATA4$X4S)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    1.00    8.50   65.99   79.00  654.00
```

```r
summary(X4S_cap)
```

```
##  sapply(IPLDATA4$X4S, FUN = cap)
##  Min.   :  0.00
##  1st Qu.:  1.00
##  Median :  8.50
##  Mean   : 65.99
##  3rd Qu.: 79.00
##  Max.   :654.00
```

```r
IPLDATA4$X4S[X4S_up_outliers] <- median(IPLDATA4$X4S)
IPLDATA4$X4S
```

```
##   [1]  12.0  17.0   0.0  12.0   8.5  13.0   2.0   8.5 119.0   2.0   2.0   0.0
##  [13]   0.0   0.0   0.0   2.0  55.0   1.0  20.0   2.0   3.0   0.0 137.0   8.5
##  [25]   0.0   2.0  41.0  95.0   8.5 119.0   0.0   8.5 166.0   1.0  55.0  97.0
##  [37]   5.0  11.0 121.0   0.0   6.0   3.0   0.0  11.0  39.0   4.0   2.0   9.0
##  [49]  11.0  99.0 194.0   0.0  11.0   1.0 165.0  18.0   0.0 160.0   0.0   0.0
##  [61]   8.5   8.5   0.0  13.0 105.0   5.0   0.0   7.0   9.0   5.0   0.0   8.0
##  [73] 102.0 172.0   8.5   0.0  76.0   0.0   0.0   8.5   3.0   9.0   0.0  52.0
##  [85]  16.0   5.0   5.0   8.5   1.0   0.0   4.0   7.0   0.0   3.0  35.0 161.0
##  [97]  19.0   4.0   1.0   0.0   0.0 155.0  11.0   8.5   3.0 136.0  17.0   0.0
## [109]   1.0  37.0 176.0   0.0   6.0   3.0   8.5  13.0  29.0   8.5   8.5  80.0
## [121]  31.0   4.0   8.5  50.0   1.0   4.0   2.0   9.0   5.0   2.0   8.5  34.0
## [133]  24.0   4.0  17.0 196.0 137.0   1.0  10.0 106.0   8.5   0.0   0.0   0.0
## [145]   0.0   8.0   0.0   2.0   0.0  11.0   0.0   2.0   0.0  37.0  45.0   8.5
## [157]   1.0   0.0  17.0 191.0  34.0   0.0
```

```r
#Impute with median


################## X6S
X6S_cap <- as.data.frame(sapply(IPLDATA4$X6S, FUN = cap))

summary(IPLDATA4$X6S)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##    0.00    0.00    4.00   28.91   37.75  227.00
```

```r
summary(X6S_cap)
```

```
##  sapply(IPLDATA4$X6S, FUN = cap)
##  Min.   :  0.00                 
##  1st Qu.:  0.00                 
##  Median :  4.00                 
##  Mean   : 28.91                 
##  3rd Qu.: 37.75                 
##  Max.   :227.00
```

```r
IPLDATA4$X6S[X6S_up_outliers] <- median(IPLDATA4$X6S)
IPLDATA4$X6S
```

```
##   [1] 14 12  1  4 76  6  0  4  4  1  1  0  0  0  0  0 44  1  3  0  3  0 90  4  0
##  [26]  6 38 22  4 65  0  4  4  2 11  4  3 11 74  0  3  2  0 11  9  1  1  3 14 46
##  [51] 90  0  4  0 56 14  0 39  2  0  4  4  0 12 46  0  0  0  6  2  0  9 41 37  4
##  [76]  0 35  0  0 85  0 14  3 42  9  2  2  4  0  1  0  4  1  0 44 89 20  2  0  0
## [101]  0 45  7 83  0 48 13  0  0 12 85  0  2  0  4  4 12  4  4 29 10  0  4 11  1
## [126]  2  0 10  1  7  4 31 22  2  2 88 36  0  8 57 68  0  0  0  0  4  1  1  1  4
## [151]  0  2  0 14 28  4  1  0  6 69 12  0
```

```r
#Impute with median




################## CATCHESTAKEN
CATCHESTAKEN_cap <- as.data.frame(sapply(IPLDATA4$CATCHESTAKEN, FUN = cap))

summary(IPLDATA4$CATCHESTAKEN)
```

```
##
##  17 values imputed to 11
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    5.00   11.00   19.60   22.75  126.00
```

```r
summary(CATCHESTAKEN_cap)
```

```
##  sapply(IPLDATA4$CATCHESTAKEN, FUN = cap)
##  Min.   :  0.00
##  1st Qu.:  5.00
##  Median : 11.00
##  Mean   : 19.60
##  3rd Qu.: 22.75
##  Max.   :126.00
```

```r
IPLDATA4$CATCHESTAKEN[CATCHESTAKEN_up_outliers] <- median(IPLDATA4$CATCHESTAKEN)
IPLDATA4$CATCHESTAKEN
```

```
##   [1] 13   5   7   3  11   2   1  11  25   4  11*  5   3  11*  6   5  44   4
##  [19] 27   4   9   3  11  11   2  12  32  15  11  11   2  11  35   1  19  11
##  [37]  2  15  19  11*  9   1  11*  7   8  13   6  24   9  18  34   1  23   4
##  [55] 29  14   3  22   3   1  11  11  11* 10  28  11  11*  5   7   3   2   1
##  [73] 13  37  11   1  12   1  11* 40   3   7   2  11  11  11  16  11   7   5
##  [91] 11* 12  11*  6  11  15   7   2   6   7   1  11   6  11  11  23  20   1
## [109]  4  37  11  11*  6  11* 11   6  12  11  11  10  13  12  11   6  11*  5
## [127] 16   4  17   5  11  14   5  12  11  34  23   8   4  21  11   3  11* 11*
## [145] 11* 19  23   1   1  29  11*  3   4   7  21  11   0  11*  9  11   4  24
```

```r
#Impute with median




################## WICKETS
WICKETS_cap <- as.data.frame(sapply(IPLDATA4$WICKETS, FUN = cap))

summary(IPLDATA4$WICKETS)
```

```
##
##  40 values imputed to 14.5
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    5.00   14.50   26.04   30.00  167.00
```

```r
summary(WICKETS_cap)
```

```
##  sapply(IPLDATA4$WICKETS, FUN = cap)
##  Min.   :  0.00
##  1st Qu.:  5.00
##  Median : 14.50
##  Mean   : 26.04
##  3rd Qu.: 30.00
##  Max.   :167.00
```

```r
IPLDATA4$WICKETS[WICKETS_up_outliers] <- median(IPLDATA4$WICKETS)
IPLDATA4$WICKETS
```

```
##   [1]  2.0   7.0   7.0   0.0   1.0  14.5*  6.0  14.5* 14.5  24.0  14.5* 34.0
##  [13] 14.5*  1.0  30.0  29.0  14.5  17.0  14.5  14.0  25.0   1.0  14.5*  0.0
##  [25]  2.0  59.0   9.0  14.5* 14.5* 14.5   1.0   0.0  22.0   1.0   5.0  42.0
##  [37]  5.0  14.5  14.5*  1.0  12.0   8.0   5.0  35.0  14.5* 14.5   8.0  14.5
##  [49] 46.0  14.5* 14.5* 12.0  14.5   5.0   0.0  59.0  13.0  14.5*  8.0  32.0
##  [61] 65.0  14.5*  4.0  13.0  51.0  40.0   0.0   4.0   0.0  24.0  25.0   1.0
##  [73] 14.5*  0.0  14.5*  2.0  30.0   0.0  14.5* 14.5* 16.0  20.0   6.0  16.0
##  [85] 13.0  14.5  50.0  14.5* 26.0  38.0  14.5* 48.0   1.0  17.0  14.5*  7.0
##  [97] 38.0  14.5* 35.0  30.0   0.0  14.5* 14.5* 14.5* 43.0   0.0  14.5   0.0
## [109] 24.0  14.5  14.5   4.0  14.5*  0.0  14.5* 18.0   3.0  14.5* 15.0  14.5*
## [121] 14.5* 14.5  14.5*  0.0   0.0   9.0  48.0  14.5* 67.0   1.0   4.0  14.5*
## [133]  4.0  25.0  48.0  14.5* 14.5* 58.0  14.5* 14.5   0.0  20.0   0.0  14.5*
## [145] 14.5* 31.0  14.5   3.0   5.0  14.5   2.0  42.0  36.0   3.0   9.0   4.0
## [157] 14.5*  0.0  27.0  14.5* 14.5* 14.5
```

```r
#Impute with median



################## STRIKERATE
STRIKERATE_cap <- as.data.frame(sapply(IPLDATA4$STRIKERATE, FUN = cap))

summary(IPLDATA4$STRIKERATE)
```

```
##
##  57 values imputed to 24.84143
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    6.00   20.70   24.84   24.84   24.84  108.00
```

```r
summary(STRIKERATE_cap)
```

```
##  sapply(IPLDATA4$STRIKERATE, FUN = cap)
##  Min.   :  6.00
##  1st Qu.: 20.70
```

```
##   Median : 24.84
##   Mean   : 24.84
##   3rd Qu.: 24.84
##   Max.   :108.00
```

```r
IPLDATA4$STRIKERATE[STRIKERATE_up_outliers] <- mean(IPLDATA4$STRIKERATE)
IPLDATA4$STRIKERATE
```

```
##    [1] 24.00000  18.85000  27.42000  24.84143*  6.00000  24.84143*  8.66000
##    [8] 24.84143* 17.51000  22.04000  24.84143* 16.11000  24.84143* 12.00000
##   [15] 15.23000  18.82000  24.15000  25.00000  20.76000  22.28000  18.36000
##   [22] 24.84143  24.84143* 24.84143* 30.00000  22.44000  24.84143  24.84143*
##   [29] 24.84143* 17.44000  24.84143  24.84143* 29.18000  12.00000  15.60000
##   [36] 20.69000  24.84143  16.20000  24.84143* 24.00000  26.00000  24.75000
##   [43] 22.80000  16.42000  24.84143* 18.63000  24.84143  20.89000  17.93000
##   [50] 24.84143* 24.84143* 22.50000  15.00000  24.84143  24.84143* 20.71000
##   [57] 24.07000  24.84143* 27.00000  16.96000  21.60000  24.84143* 12.25000
##   [64] 24.84143  28.31000  22.40000  24.84143* 21.00000  24.84143* 19.95000
##   [71] 12.96000  24.84143  24.84143* 24.84143* 24.84143* 18.00000  20.40000
##   [78] 24.84143* 24.84143* 24.84143* 19.87000  15.95000  21.00000  25.68000
##   [85] 26.38000  21.13000  20.56000  24.84143* 25.84000  22.57000  24.84143*
##   [92] 17.47000  24.84143  24.84143  24.84143* 16.42000  21.94000  24.84143*
##   [99] 22.91000  24.86000  24.84143* 24.84143* 24.84143* 24.84143* 20.93000
##  [106] 24.84143* 19.48000  24.84143* 21.75000  24.12000  23.67000  25.50000
##  [113] 24.84143* 24.84143* 24.84143* 27.11000  24.84143  24.84143* 22.60000
##  [120] 24.84143* 24.84143* 19.62000  24.84143* 24.84143* 24.84143* 13.33000
##  [127] 29.47000  24.84143* 18.80000  24.84143  12.00000  24.84143* 23.50000
##  [134] 20.68000  19.39000  24.84143* 24.84143* 20.43000  24.84143* 21.82000
##  [141] 24.84143* 25.05000  24.84143* 24.84143* 24.84143* 30.96000  18.64000
##  [148] 24.84143  21.40000  21.19000  24.84143  22.95000  20.50000  17.00000
##  [155] 25.44000  24.84143  24.84143* 24.84143* 27.77000  24.84143* 24.84143*
##  [162] 17.61000
```

```r
#Impute with mean



################## ECONOMYRATE
ECONOMYRATE_cap <- as.data.frame(sapply(IPLDATA4$ECONOMYRATE, FUN = cap))

summary(IPLDATA4$ECONOMYRATE)
```

```
##
##   40 values imputed to 8.656393

##     Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    5.000   8.000   8.656   8.656   8.770  18.000
```

```r
summary(ECONOMYRATE_cap)
```

```
##   sapply(IPLDATA4$ECONOMYRATE, FUN = cap)
##   Min.   : 5.000
```

```
##  1st Qu.: 8.000
##  Median : 8.656
##  Mean   : 8.656
##  3rd Qu.: 8.770
##  Max.   :18.000
```

```
IPLDATA4$ECONOMYRATE[ECONOMYRATE_up_outliers] <- mean(IPLDATA4$ECONOMYRATE)
IPLDATA4$ECONOMYRATE
```

```
##   [1] 8.656393  8.000000  9.620000  5.750000  5.000000  8.656393* 8.656393
##   [8] 8.656393* 9.040000  9.230000  8.656393* 7.650000  8.656393* 5.500000
##  [15] 8.780000  8.230000  7.220000  9.790000  7.300000  8.190000  9.120000
##  [22] 8.500000  8.656393* 8.656393  9.500000  7.800000  8.450000  8.656393*
##  [29] 8.656393* 8.360000  8.180000  8.656393  8.550000  8.656393  7.460000
##  [36] 9.060000  7.120000  8.580000  8.656393* 9.750000  8.880000  9.240000
##  [43] 8.680000  8.200000  8.656393* 7.420000  6.860000  8.740000  7.130000
##  [50] 8.656393* 8.656393* 7.930000  8.210000  9.140000  8.656393  7.900000
##  [57] 9.410000  8.656393* 9.660000  8.680000  8.780000  8.656393* 8.656393
##  [64] 8.260000  7.360000  8.270000  8.000000  7.210000  8.656393  8.110000
##  [71] 8.290000  7.400000  8.656393* 8.656393  8.656393* 7.500000  9.500000
##  [78] 8.656393  8.656393* 8.656393* 8.540000  7.890000  7.000000  6.840000
##  [85] 7.130000  8.620000  8.380000  8.656393* 7.860000  7.830000  8.656393*
##  [92] 7.520000  8.180000  8.470000  8.656393* 8.030000  8.230000  8.656393*
##  [99] 8.790000  9.260000  9.000000  8.656393* 8.656393* 8.656393* 7.440000
## [106] 8.656393  6.330000  8.656393  6.960000  6.910000  7.610000  8.656393
## [113] 8.656393* 7.330000  8.656393* 7.860000  8.656393  8.656393* 8.010000
## [120] 8.656393* 8.656393* 7.770000  8.656393* 8.656393  8.656393  6.800000
## [127] 7.560000  8.656393* 8.890000  8.630000  8.250000  8.656393* 8.290000
## [134] 8.290000  8.030000  8.656393* 8.656393* 8.580000  8.656393* 6.740000
## [141] 8.000000  8.230000  9.420000  8.656393* 8.656393* 8.670000  8.390000
## [148] 8.656393  8.570000  8.510000  8.000000  8.890000  6.820000  8.110000
## [155] 8.620000  8.790000  8.656393* 8.656393  6.930000  8.656393* 8.656393*
## [162] 7.590000
```

```
#Impute with mean
```

Multivariate Outliers

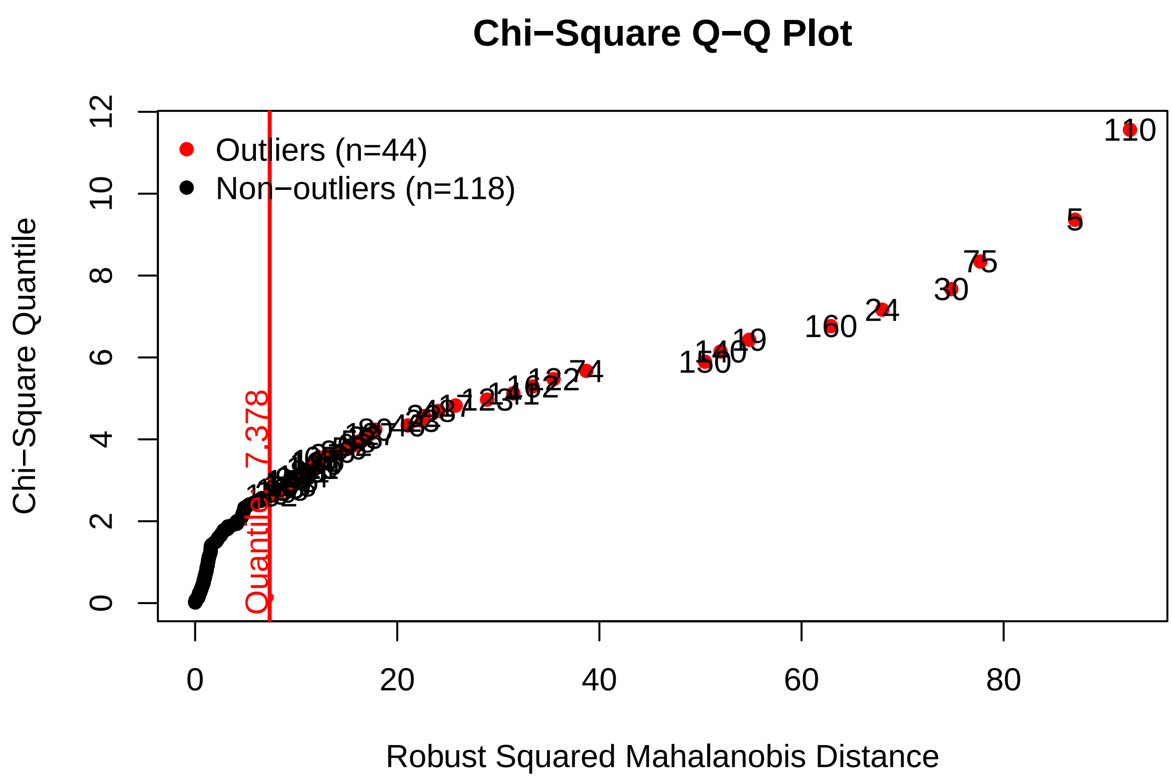
In searching for multivariate outliers we apply the MVN method accross the numeric variables in **IPLDATA4**.

We test the following combinations of variables and detect the following outliers:

- **Matches Played vs Sold Price** 44 outliers found.
- **Batting Statistics** - consisting of *NOTOUTS*, *BATTINGS.R*, *X4S*, *X6S*, we detect 70 outliers
- **ECONOMYRATE and STRIKERATE** - 77 outliers detected
- **NOTOUTS and RUNSSCORED** - 60 outliers detected
- **WICKETS and STUMPINGSMADE** - error as *STUMPINGSMADE has IQR less than 0. Computer says no hahaha.
- **MATCHPLAYED and NOTOUTS** - 56 outliers detected.
- **IPLDATA$ NUMERIC values** - Consists of *MATCHPLAYED*, *NOTOUTS*, *RUNSSCORED*, *BATTINGAVG*, *BATTINGS.R*, *X4S,X6S*, *CATCHESTAKEN*, *WICKETS*, *STRIKERATE*, *ECONOMYRATE*, *BASE.PRICE*, *VALUEINCR*. 78 Outliers found.

Given the high number of outliers present accross the variables versus the observations in our dataset, it does not make sense to exclude them.

```
######################## Multivariate Outliers  ################################

#########                    MVN method                            ###########

##  Matches Played vs Sold Price
IPL4_sub1 <- IPLDATA4 %>% select(MATCHPLAYED, VALUEINCR)

results1 <- IPL4_sub1 %>%
  MVN::mvn(multivariateOutlierMethod = "quan",
           showOutliers = TRUE)
```

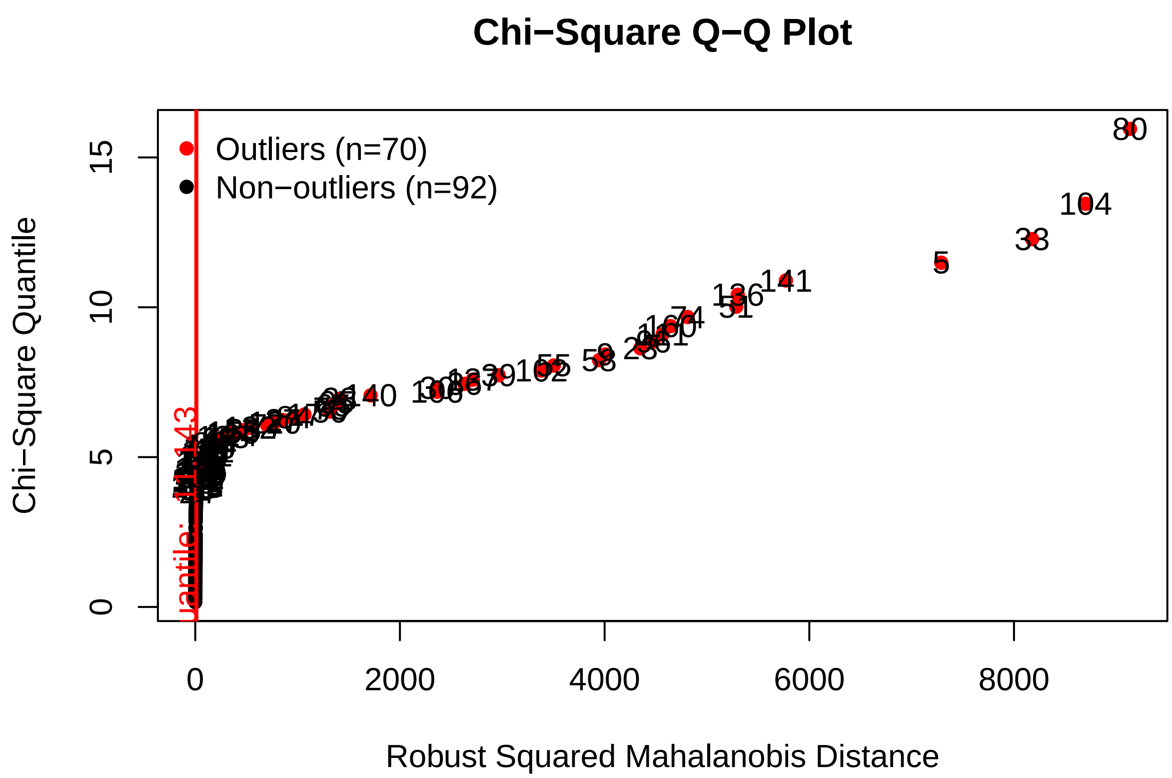



```
results1$multivariateOutliers
```

```
##     Observation Mahalanobis Distance Outlier
## 110         110               92.496    TRUE
## 5             5               87.065    TRUE
## 75           75               77.693    TRUE
## 30           30               74.814    TRUE
## 24           24               67.984    TRUE
## 160         160               62.904    TRUE
## 19           19               54.829    TRUE
## 140         140               51.990    TRUE
```

```
## 150          150              50.449    TRUE
## 74            74              38.710    TRUE
## 122          122              35.465    TRUE
## 162          162              33.392    TRUE
## 141          141              31.488    TRUE
## 123          123              28.887    TRUE
## 17            17              25.774    TRUE
## 48            48              24.083    TRUE
## 32            32              22.656    TRUE
## 23            23              22.474    TRUE
## 46            46              21.043    TRUE
## 80            80              17.827    TRUE
## 127          127              17.360    TRUE
## 33            33              16.839    TRUE
## 58            58              16.156    TRUE
## 62            62              15.780    TRUE
## 56            56              15.244    TRUE
## 36            36              14.119    TRUE
## 27            27              13.083    TRUE
## 65            65              12.483    TRUE
## 115          115              12.159    TRUE
## 136          136              11.955    TRUE
## 111          111              11.618    TRUE
## 119          119              11.618    TRUE
## 86            86              11.133    TRUE
## 9              9              10.855    TRUE
## 104          104              10.650    TRUE
## 96            96               9.657    TRUE
## 156          156               9.592    TRUE
## 107          107               9.542    TRUE
## 158          158               9.371    TRUE
## 73            73               9.072    TRUE
## 138          138               8.601    TRUE
## 69            69               8.384    TRUE
## 39            39               7.653    TRUE
## 152          152               7.523    TRUE
```

```
################################################################################

##  Batting Statistics - NOTOUTS, BATTINGS.R, X4S, X6S
IPL4_sub2 <- IPLDATA4 %>% select(NOTOUTS, BATTINGS.R, X4S, X6S)

results2 <- IPL4_sub2 %>%
  MVN::mvn(multivariateOutlierMethod = "quan",
           showOutliers = TRUE)
```



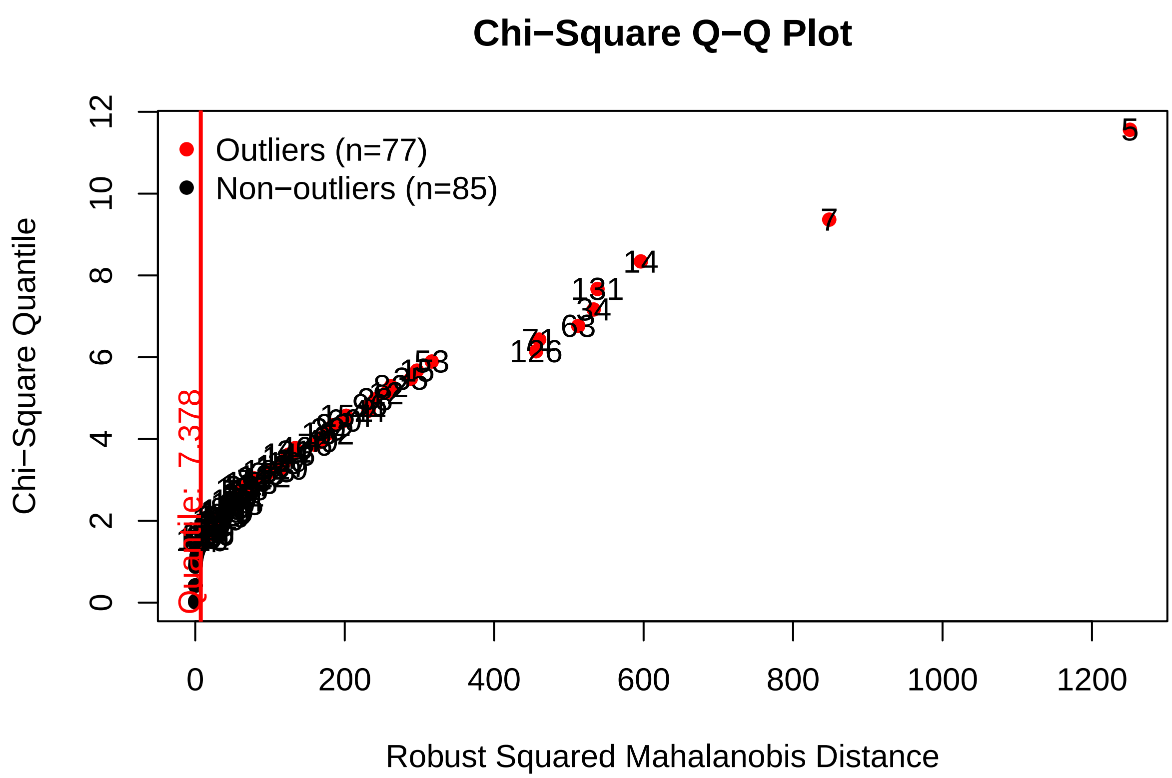

```
results2$multivariateOutliers
```

```
##      Observation Mahalanobis Distance Outlier
## 80            80             9133.280    TRUE
## 104          104             8699.199    TRUE
## 33            33             8176.823    TRUE
## 5              5             7289.928    TRUE
## 141          141             5771.175    TRUE
## 136          136             5301.261    TRUE
## 51            51             5286.040    TRUE
## 74            74             4810.736    TRUE
## 160          160             4644.283    TRUE
## 111          111             4566.737    TRUE
## 96            96             4477.835    TRUE
## 23            23             4349.186    TRUE
## 9              9             4006.212    TRUE
## 58            58             3946.512    TRUE
## 55            55             3504.007    TRUE
## 102          102             3381.388    TRUE
## 39            39             2964.762    TRUE
## 137          137             2712.333    TRUE
## 36            36             2631.364    TRUE
## 30            30             2363.580    TRUE
## 106          106             2362.200    TRUE
## 140          140             1713.913    TRUE
## 28            28             1411.360    TRUE
```

```
## 95           95             1382.365    TRUE
## 65           65             1348.078    TRUE
## 73           73             1322.351    TRUE
## 50           50             1314.909    TRUE
## 17           17             1066.765    TRUE
## 84           84              965.542    TRUE
## 27           27              854.168    TRUE
## 120         120              777.175    TRUE
## 77           77              703.890    TRUE
## 132         132              536.690    TRUE
## 35           35              472.430    TRUE
## 155         155              368.706    TRUE
## 124         124              351.830    TRUE
## 133         133              269.797    TRUE
## 97           97              235.646    TRUE
## 45           45              213.888    TRUE
## 82           82              147.117    TRUE
## 154         154              147.030    TRUE
## 110         110              137.972    TRUE
## 49           49              124.487    TRUE
## 161         161              123.380    TRUE
## 1             1              116.162    TRUE
## 121         121              105.180    TRUE
## 56           56               91.021    TRUE
## 117         117               83.504    TRUE
## 19           19               72.225    TRUE
## 107         107               69.676    TRUE
## 64           64               64.346    TRUE
## 38           38               62.957    TRUE
## 44           44               62.888    TRUE
## 128         128               54.292    TRUE
## 2             2               53.401    TRUE
## 135         135               50.620    TRUE
## 130         130               47.861    TRUE
## 72           72               44.827    TRUE
## 26           26               34.296    TRUE
## 131         131               26.020    TRUE
## 150         150               25.622    TRUE
## 139         139               24.658    TRUE
## 85           85               24.545    TRUE
## 159         159               19.768    TRUE
## 122         122               18.328    TRUE
## 103         103               16.254    TRUE
## 83           83               14.554    TRUE
## 68           68               13.603    TRUE
## 6             6               13.181    TRUE
## 24           24               12.902    TRUE
```

```
###############################################################################

##  ECONOMYRATE and STRIKERATE
IPL4_sub3 <- IPLDATA4 %>% select(ECONOMYRATE, STRIKERATE)

results3 <- IPL4_sub3 %>%
```

```
MVN::mvn(multivariateOutlierMethod = "quan",
         showOutliers = TRUE)
```



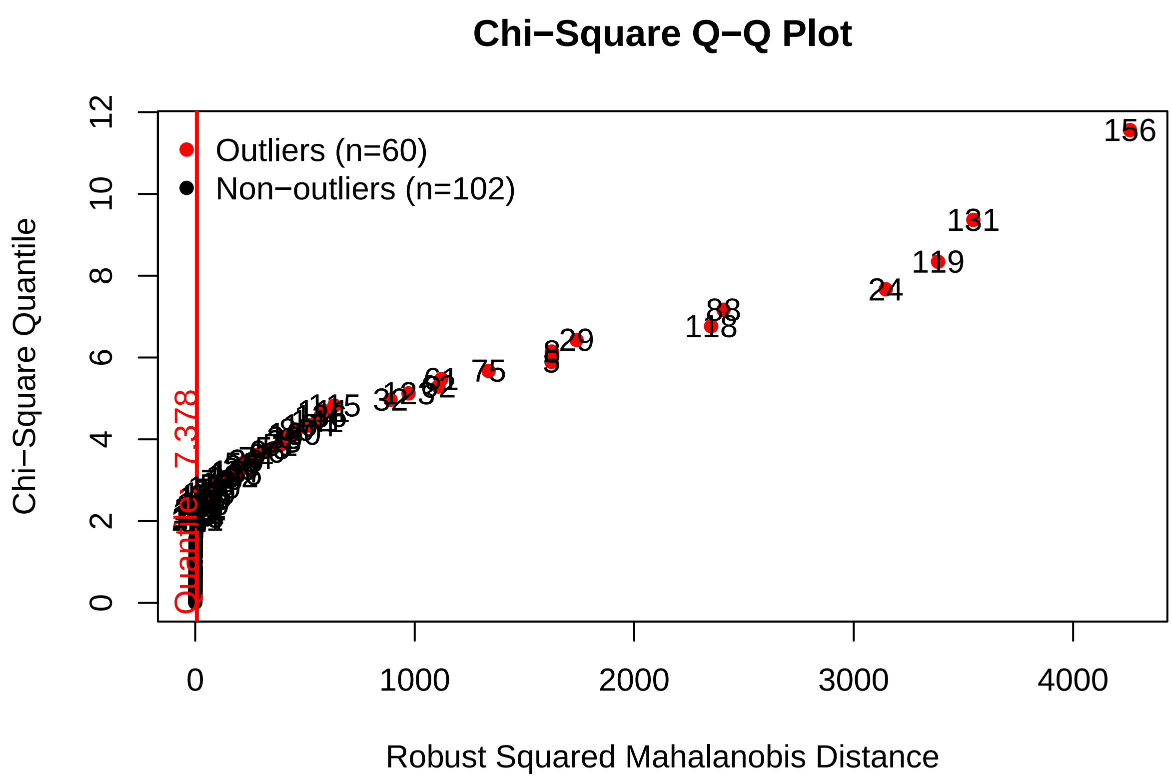

```
results3$multivariateOutliers
```

```
##     Observation Mahalanobis Distance Outlier
## 5             5             1250.824    TRUE
## 7             7              848.413    TRUE
## 14           14              596.259    TRUE
## 131         131              538.271    TRUE
## 34           34              533.184    TRUE
## 63           63              512.521    TRUE
## 71           71              460.273    TRUE
## 126         126              456.112    TRUE
## 53           53              316.527    TRUE
## 15           15              296.684    TRUE
## 35           35              288.424    TRUE
## 82           82              261.880    TRUE
## 12           12              255.562    TRUE
## 38           38              240.829    TRUE
## 96           96              233.625    TRUE
## 44           44              231.911    TRUE
## 154         154              201.932    TRUE
## 60           60              199.405    TRUE
## 92           92              184.844    TRUE
```

```
## 30            30                177.869    TRUE
## 162          162                177.235    TRUE
## 9              9                170.528    TRUE
## 49            49                168.115    TRUE
## 76            76                160.123    TRUE
## 46            46                133.752    TRUE
## 21            21                132.621    TRUE
## 146          146                125.203    TRUE
## 147          147                124.529    TRUE
## 2              2                119.051    TRUE
## 16            16                118.447    TRUE
## 107          107                116.083    TRUE
## 129          129                115.690    TRUE
## 135          135                 98.526    TRUE
## 25            25                 96.281    TRUE
## 122          122                 92.591    TRUE
## 81            81                 79.128    TRUE
## 70            70                 78.958    TRUE
## 153          153                 74.800    TRUE
## 127          127                 69.414    TRUE
## 33            33                 62.989    TRUE
## 138          138                 61.994    TRUE
## 77            77                 61.728    TRUE
## 19            19                 61.720    TRUE
## 87            87                 59.171    TRUE
## 83            83                 58.569    TRUE
## 56            56                 57.862    TRUE
## 134          134                 56.312    TRUE
## 68            68                 56.229    TRUE
## 105          105                 55.764    TRUE
## 36            36                 53.791    TRUE
## 48            48                 49.145    TRUE
## 86            86                 43.558    TRUE
## 150          150                 42.473    TRUE
## 140          140                 42.258    TRUE
## 109          109                 41.157    TRUE
## 65            65                 39.874    TRUE
## 149          149                 37.492    TRUE
## 61            61                 32.768    TRUE
## 159          159                 30.914    TRUE
## 3              3                 28.828    TRUE
## 97            97                 27.585    TRUE
## 10            10                 24.406    TRUE
## 59            59                 22.087    TRUE
## 20            20                 21.684    TRUE
## 26            26                 21.008    TRUE
## 4              4                 20.158    TRUE
## 66            66                 19.426    TRUE
## 52            52                 19.331    TRUE
## 90            90                 18.800    TRUE
## 119          119                 17.415    TRUE
## 116          116                 17.102    TRUE
## 43            43                 12.788    TRUE
## 99            99                 11.361    TRUE
```

```
## 85           85              11.059    TRUE
## 152         152              10.900    TRUE
## 110         110               9.250    TRUE
## 84           84               8.456    TRUE
```

```
################################################################################

##  NOTOUTS and RUNSSCORED
IPL4_sub4 <- IPLDATA4 %>% select(NOTOUTS, RUNSSCORED)

results4 <- IPL4_sub4 %>%
  MVN::mvn(multivariateOutlierMethod = "quan",
           showOutliers = TRUE)
```

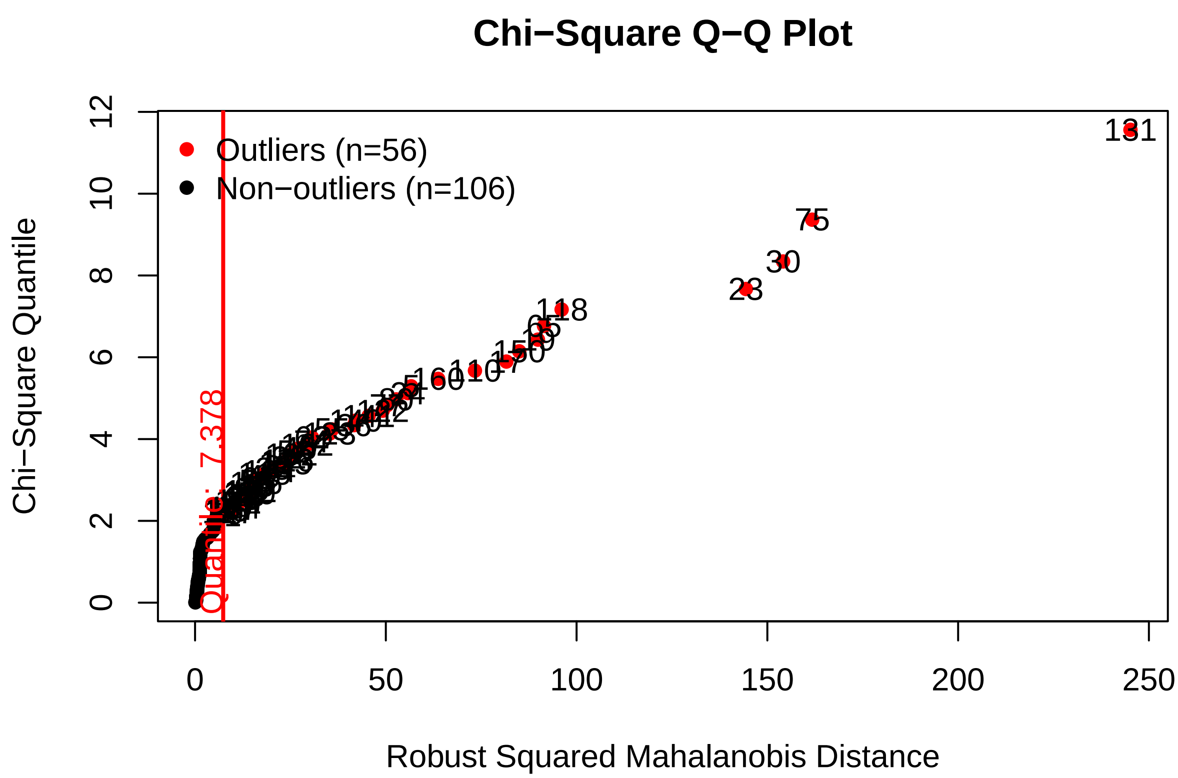



```
results4$multivariateOutliers
```

```
##     Observation Mahalanobis Distance Outlier
## 156         156             4259.024    TRUE
## 131         131             3545.210    TRUE
## 119         119             3384.624    TRUE
## 24           24             3146.253    TRUE
## 88           88             2406.826    TRUE
## 118         118             2350.398    TRUE
## 29           29             1737.274    TRUE
## 8             8             1624.918    TRUE
```

```
## 5             5           1623.311    TRUE
## 75           75           1336.461    TRUE
## 61           61           1120.887    TRUE
## 62           62           1107.722    TRUE
## 123         123            971.728    TRUE
## 32           32            889.702    TRUE
## 115         115            634.003    TRUE
## 111         111            584.793    TRUE
## 136         136            571.066    TRUE
## 141         141            555.694    TRUE
## 104         104            518.907    TRUE
## 80           80            462.017    TRUE
## 160         160            452.477    TRUE
## 33           33            406.303    TRUE
## 23           23            402.947    TRUE
## 51           51            387.015    TRUE
## 55           55            352.935    TRUE
## 96           96            328.871    TRUE
## 9             9            283.416    TRUE
## 74           74            276.345    TRUE
## 30           30            232.058    TRUE
## 58           58            213.229    TRUE
## 36           36            212.883    TRUE
## 39           39            204.700    TRUE
## 137         137            192.519    TRUE
## 106         106            183.860    TRUE
## 102         102            168.151    TRUE
## 65           65            136.228    TRUE
## 73           73            104.469    TRUE
## 17           17            100.944    TRUE
## 50           50            100.186    TRUE
## 140         140             86.226    TRUE
## 77           77             80.411    TRUE
## 28           28             71.978    TRUE
## 120         120             62.045    TRUE
## 27           27             57.201    TRUE
## 155         155             44.954    TRUE
## 110         110             37.813    TRUE
## 19           19             37.648    TRUE
## 84           84             37.066    TRUE
## 150         150             34.166    TRUE
## 95           95             29.397    TRUE
## 132         132             21.033    TRUE
## 122         122             20.502    TRUE
## 35           35             19.981    TRUE
## 124         124             14.791    TRUE
## 56           56             13.096    TRUE
## 46           46             11.475    TRUE
## 133         133             10.512    TRUE
## 154         154              9.667    TRUE
## 121         121              8.145    TRUE
## 97           97              8.142    TRUE
```

```
################################################################################

##  WICKETS and STUMPINGSMADE
#IPL4_sub5 <- IPLDATA4 %>% select(STUMPINGSMADE, WICKETS)

#results5 <- IPL4_sub5 %>%
#  MVN::mvn(multivariateOutlierMethod = "quan",
#           showOutliers = TRUE)

#results5$multivariateOutliers



################################################################################

##  MATCHPLAYED and NOTOUTS
IPL4_sub6 <- IPLDATA4 %>% select(MATCHPLAYED, NOTOUTS)

results6 <- IPL4_sub6 %>%
  MVN::mvn(multivariateOutlierMethod = "quan",
           showOutliers = TRUE)
```



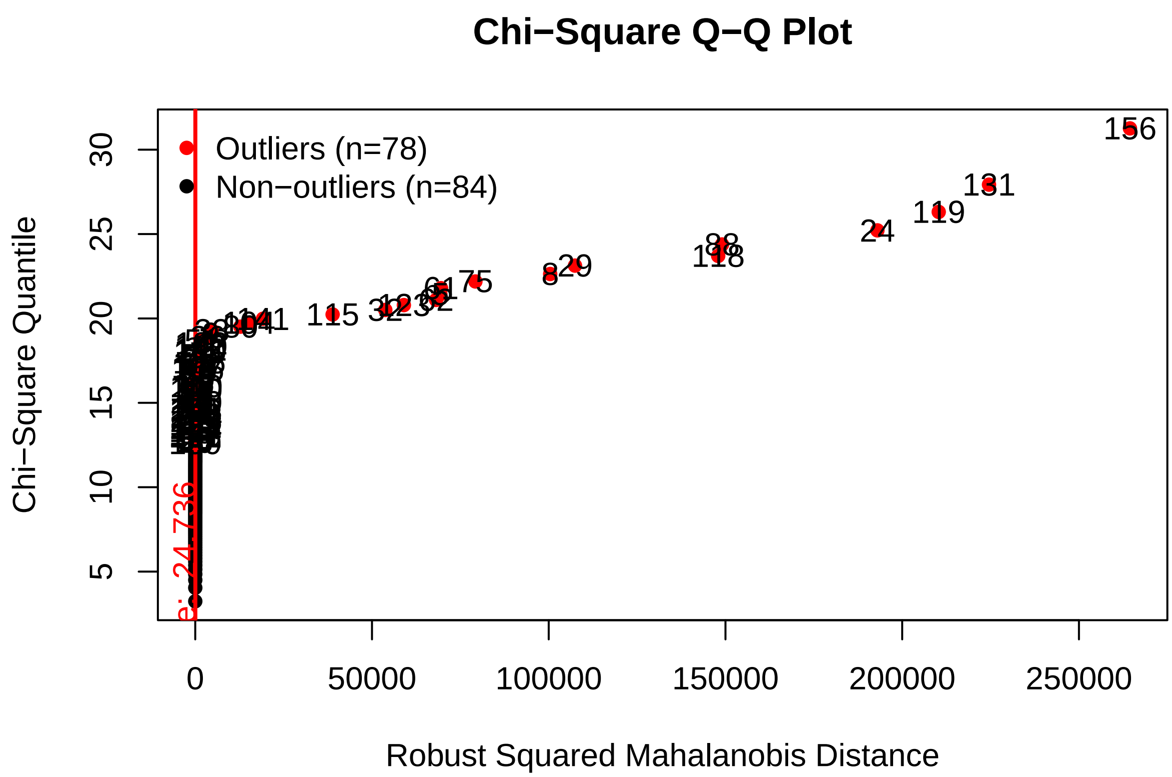

```
results6$multivariateOutliers
```

```
##     Observation Mahalanobis Distance Outlier
```

```
## 131          131              245.167    TRUE
## 75            75              161.755    TRUE
## 30            30              154.131    TRUE
## 23            23              144.382    TRUE
## 118          118               96.078    TRUE
## 65            65               91.461    TRUE
## 19            19               89.868    TRUE
## 150          150               84.970    TRUE
## 17            17               81.596    TRUE
## 110          110               73.363    TRUE
## 160          160               63.709    TRUE
## 5              5               56.652    TRUE
## 24            24               55.769    TRUE
## 80            80               52.733    TRUE
## 77            77               50.325    TRUE
## 122          122               49.182    TRUE
## 141          141               45.512    TRUE
## 140          140               42.073    TRUE
## 36            36               41.663    TRUE
## 55            55               35.915    TRUE
## 123          123               35.290    TRUE
## 62            62               30.666    TRUE
## 74            74               30.497    TRUE
## 162          162               29.384    TRUE
## 46            46               27.310    TRUE
## 56            56               26.088    TRUE
## 102          102               25.265    TRUE
## 32            32               24.590    TRUE
## 116          116               24.237    TRUE
## 155          155               23.646    TRUE
## 27            27               22.576    TRUE
## 33            33               20.399    TRUE
## 104          104               19.485    TRUE
## 115          115               18.215    TRUE
## 49            49               17.994    TRUE
## 97            97               16.832    TRUE
## 58            58               16.161    TRUE
## 136          136               16.008    TRUE
## 48            48               15.799    TRUE
## 9              9               15.443    TRUE
## 86            86               14.844    TRUE
## 132          132               14.515    TRUE
## 159          159               14.286    TRUE
## 73            73               13.723    TRUE
## 96            96               12.793    TRUE
## 38            38               12.384    TRUE
## 107          107               12.161    TRUE
## 37            37               11.384    TRUE
## 87            87               11.277    TRUE
## 18            18               10.351    TRUE
## 124          124               10.297    TRUE
## 137          137                9.579    TRUE
## 26            26                9.286    TRUE
## 127          127                8.791    TRUE
```

```
## 39            39                 8.157     TRUE
## 41            41                 7.522     TRUE
```

```
##############################################################################

##  IPLDATA4 - Numeric Values
IPL4_sub7 <- IPLDATA4 %>% select(MATCHPLAYED, NOTOUTS, RUNSSCORED, BATTINGAVG, BATTINGS.R, X4S,X6S, CATC

results7 <- IPL4_sub7 %>%
  MVN::mvn(multivariateOutlierMethod = "quan",
           showOutliers = TRUE)
```




```
results7$multivariateOutliers
```

```
##      Observation Mahalanobis Distance Outlier
## 156          156           264437.106    TRUE
## 131          131           224559.679    TRUE
## 119          119           210335.616    TRUE
## 24            24           192984.949    TRUE
## 88            88           149031.363    TRUE
## 118          118           147944.417    TRUE
## 29            29           107431.292    TRUE
## 8              8           100410.051    TRUE
## 75            75            79263.248    TRUE
## 61            61            69535.230    TRUE
```

```
## 5               5             69352.136    TRUE
## 62             62             68131.167    TRUE
## 123           123             59017.094    TRUE
## 32             32             53767.332    TRUE
## 115           115             38866.996    TRUE
## 141           141             19165.754    TRUE
## 104           104             15144.868    TRUE
## 80             80             12780.328    TRUE
## 33             33              4713.463    TRUE
## 9               9              4380.215    TRUE
## 36             36              3530.219    TRUE
## 51             51              1923.762    TRUE
## 102           102              1693.889    TRUE
## 140           140              1607.152    TRUE
## 111           111              1464.594    TRUE
## 160           160              1295.029    TRUE
## 74             74              1262.631    TRUE
## 58             58              1196.950    TRUE
## 96             96              1156.741    TRUE
## 136           136              1124.213    TRUE
## 55             55               977.259    TRUE
## 137           137               730.521    TRUE
## 106           106               700.409    TRUE
## 30             30               667.025    TRUE
## 39             39               557.768    TRUE
## 23             23               556.028    TRUE
## 73             73               444.430    TRUE
## 50             50               426.836    TRUE
## 28             28               420.608    TRUE
## 65             65               375.594    TRUE
## 110           110               258.150    TRUE
## 17             17               247.406    TRUE
## 120           120               235.740    TRUE
## 27             27               184.713    TRUE
## 14             14               179.744    TRUE
## 84             84               177.066    TRUE
## 19             19               176.909    TRUE
## 77             77               167.752    TRUE
## 63             63               167.168    TRUE
## 150           150               162.032    TRUE
## 40             40               153.913    TRUE
## 25             25               153.441    TRUE
## 52             52               151.565    TRUE
## 151           151               151.339    TRUE
## 143           143               150.699    TRUE
## 71             71               147.275    TRUE
## 43             43               146.700    TRUE
## 162           162               144.607    TRUE
## 124           124               141.195    TRUE
## 35             35               137.866    TRUE
## 46             46               116.532    TRUE
## 155           155               114.481    TRUE
## 95             95               104.963    TRUE
## 122           122               103.179    TRUE
```

```
## 45            45                  99.031    TRUE
## 161          161                  96.089    TRUE
## 154          154                  74.491    TRUE
## 48            48                  69.208    TRUE
## 107          107                  64.154    TRUE
## 86            86                  49.830    TRUE
## 147          147                  41.473    TRUE
## 121          121                  35.169    TRUE
## 132          132                  31.218    TRUE
## 38            38                  30.245    TRUE
## 53            53                  29.503    TRUE
## 56            56                  28.000    TRUE
## 26            26                  25.605    TRUE
## 159          159                  24.749    TRUE
```

```
################################################################################
#IPL4_sub1_outliers <- IPL4_sub1[c(as.numeric(results1$multivariateOutliers[["Observation"]])), ]

# Break it down:
#results1$multivariateOutliers[["Observation"]]
# This part returns the observation numbers of the outliers.

#as.numeric(results1$multivariateOutliers[["Observation"]])
# This part is taking those observation numbers and converting from character to numeric
# Then we subset the dataset by including rows that are in this vector of observation numbers


#IPL4_clean <- IPL4_sub1[-c(as.numeric(results1$multivariateOutliers[["Observation"]])), ]
# What is the difference between iris_outliers and iris_clean?
# We've used "-c()" to remove those observations in iris_clean, whereas in
# iris_outliers, we kept only those observations by using "c()"

#dim(IPL4_clean)
#dim(IPL4_sub1_outliers)
#dim(IPL4_sub1)
```

Transform

For data transformation we have chosen the **RUNSSCORED** for further analysis. Looking at the histogram below, we can see that the distribution for **RUNSSCORED** data is highly skewed to the right.

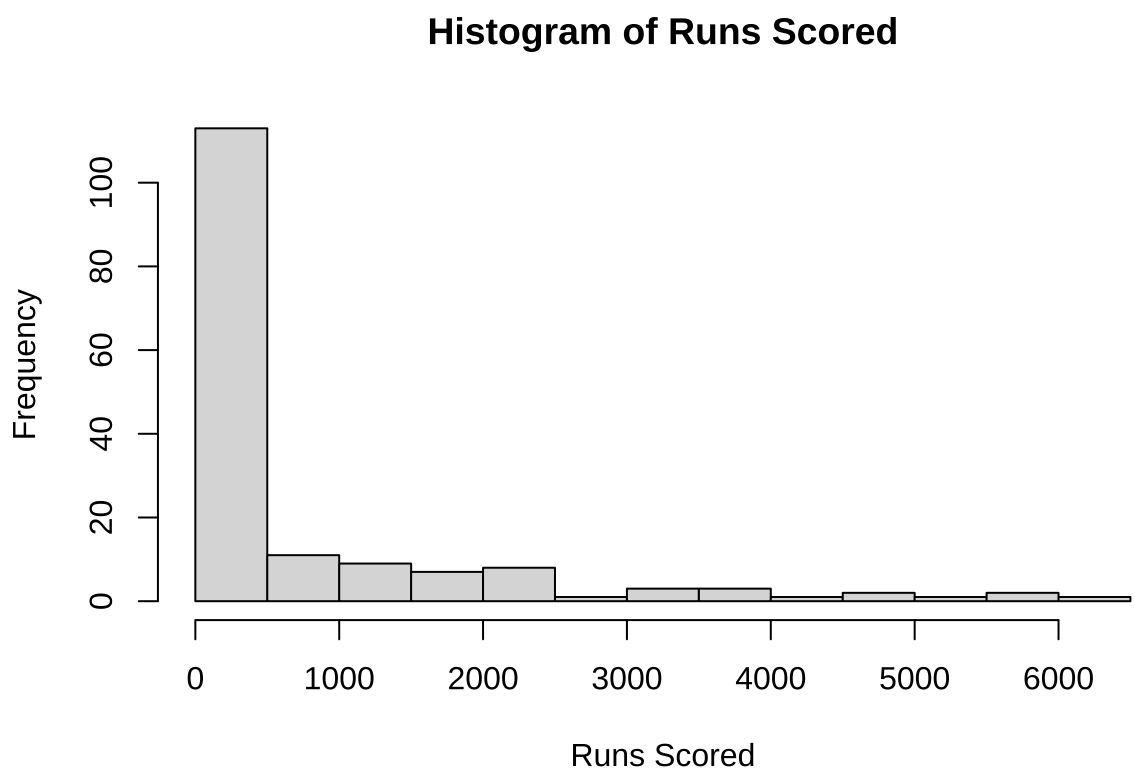
A more symmetrical distribution makes the data easier to work with for use with statistical analysis techniques such as parametric and linear regression, so in order to correct the skewedness, we will apply a number of transformation techniques.

Applying a **log10** transformation, we can see that the distribution now looks more symmetrical.

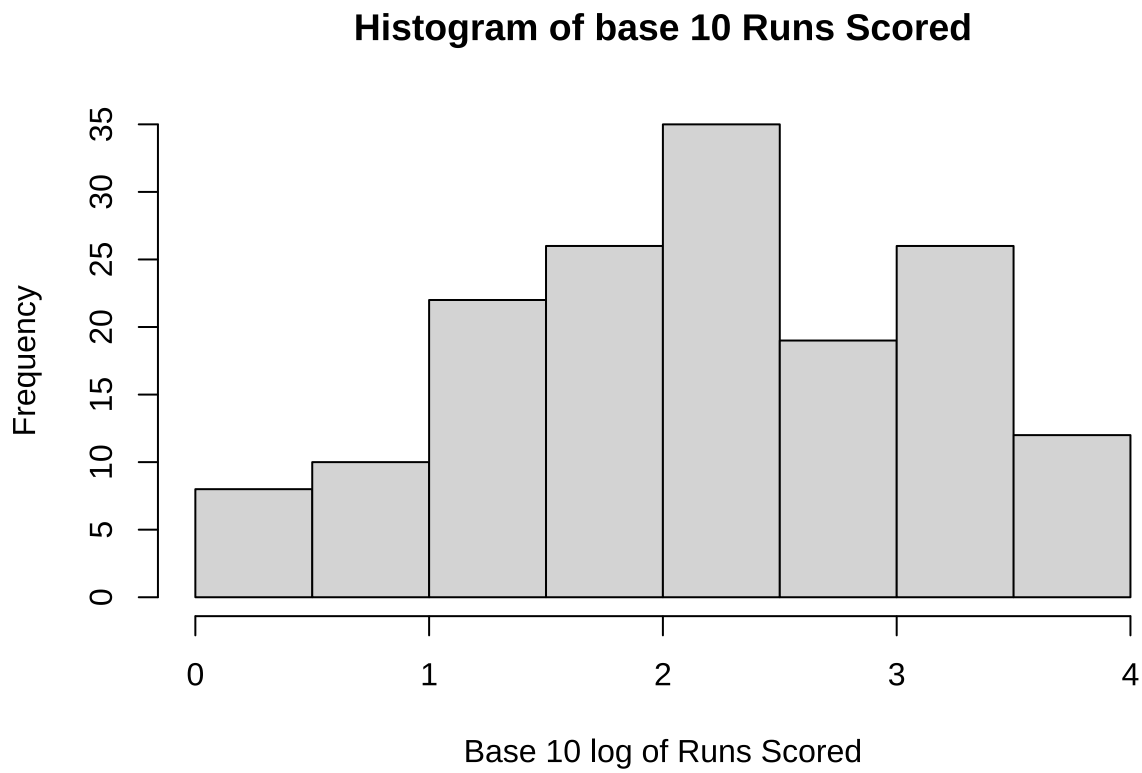
We also applied a Square Root transformation, however when we examin the histograph, we can see that the distribution is still skewed to the right.

log10 transformation works best in this instance as the range in the **RUNSSCORED** variable differs by several orders of magnitude. The transformation makes the data easier to understand, and by converting the data to a normal distribution, makes it easier to work with for linear regression and parametric analysis techniques.

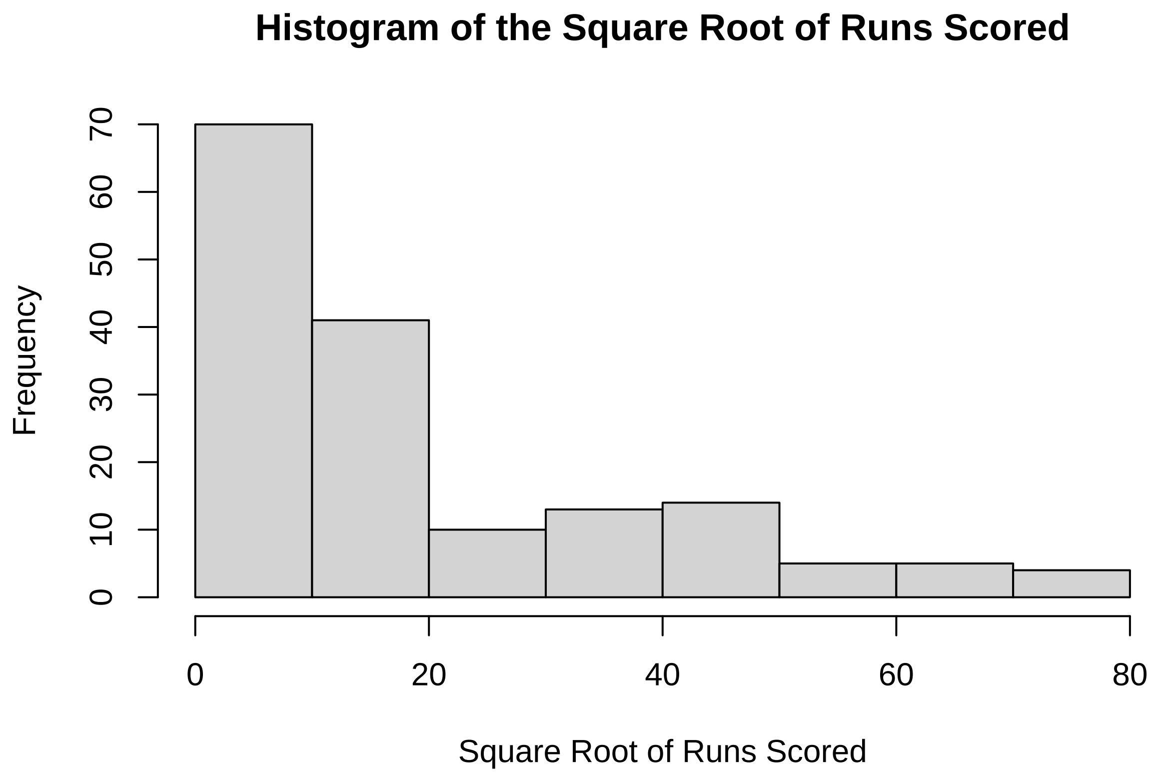
```r
# This is a chunk where you apply an appropriate transformation to at least one of the variables

## Histogram of Runs Scored:
hist(IPLDATA4$RUNSSCORED, breaks = 10,
main = "Histogram of Runs Scored",
xlab = "Runs Scored")
```




```r
# Applying log Transformation:
log_RunsScored <- log10(IPLDATA4$RUNSSCORED)

# Histogram of Log10 Transformed Trip Distance
hist(log_RunsScored, breaks = 10,
main = "Histogram of base 10 Runs Scored",
xlab = "Base 10 log of Runs Scored")
```




```
# Applying Square Root Transformation
sqrt_RunsScored <- sqrt(IPLDATA4$RUNSSCORED)
# Histogram of SQRT Transformed Trip Distance
hist(sqrt_RunsScored, breaks = 10,
main = "Histogram of the Square Root of Runs Scored",
xlab = "Square Root of Runs Scored")
```



Reflective journal

Initial Plan:- Starting the assignment, it was really a struggle to find a good data set that met the assignment requirements. I searched data.gov.au, figshare.com and sites such as kaggle for days until I found a couple of data sets I was happy with.

Once I found the datasets I was happy with, I felt that I was applying the skills from assignment 2, except that this time there was the challenge of working with more realistic and messy data.

Looking at the Player and Auction dataset, I was hoping to combine them in order to create a dataset where we might be able to see the relationship between a players stats and how this may reflect in their auction price.

Key Questions:- My plan was to combine the Player and Auction data sets as I thought it would be interesting to see how player stats could inform their bid price.

Difficulties Encountered:- My data set contained many variables, and I feel that I could have achieved more with them had I had more time, I would be able to do more with outliers in the variables I didn't cover. There was a lot difficulties around NA figures due to the amount and uncertainty of how I should treat them. Initially my data set had a lot of blank spaces that I initially had to replace with NAs before I can impute them with another value.

For example with the following code:
levels(IPLDATA2COUNTRY)∗∗levels(IPLDATA2COUNTRY) <- c(levels(IPLDATA2COUNTRY), "NA")∗
∗IPLDATA2COUNTRY[IPLDATA2$COUNTRY == " "] <- 'NA'

When I scanned for NAs, the NAs would not appear in columns like *Country* or *Sport* and when I later tried to replace the NAs in the the columns median mode or an "Unknown" value, I would get an error with factor levels, so I had to convert the variable back to a character datatype first to resolve it.

The data set was difficult because of the sheer amount of NAs. The data sets I had, particularly the Player dataset, had records full of NAs. I feel had I had more time, or maybe in a work environment we would be able to scrape the missing stats from online and incorporate it into our dataset.

Solutions Used to Resolve Problems:- As for solutions. When it came to imputing those categorical variables, I was able to resolve the issue by converting the variable back to a character to do the replace, then change back to a factor.

For the large amount of NA figures, even after the datasets were merged and the non-matching rows were dropped, I still had players with entire rows of NAs which defeated the intent of merging the datasets to begin with. I used the below code to drop variables present in the Player dataset unless a value was present in another row:
IPLDATA3 <- IPLDATA2 %>% filter(!is.na(MATCHPLAYED) | !is.na(INNINGSBATTED) | !is.na(NOTOUTS) | !is.na(RUNSSCORED) | !is.na(HIGHEST.RUNS.SCORED) |!is.na(X100S) | !is.na(X50S) | !is.na(X4S) | !is.na(X6S) | !is.na(BATTINGAVG) |!is.na(BATTINGS.R) | !is.na(CATCHESTAKEN) | !is.na(STUMPINGSMADE) | !is.na(DUCKS) | !is.na(RUNOUTS) | !is.na(INNINGSBOWLED | !is.na(OVERS) | !is.na(MAIDENS) | !is.na(RUNSCONCEDED) | !is.na(WICKETS) | !is.na(WICKETS.X3S) | !is.na(WICKETS.X5S) | !is.na(BOWLINGAVG) | !is.na(ECONOMYRATE) | !is.na(STRIKERATE)))

This helped drop some columns and allowed me to impute the rest of the remaining variables.

Insights Gained:- Even though there were a lot of challenges with this assignment. Mainly with finding a data set and dealing how untidy it is, and the sheer amount of missing values. I also have some doubts in regards to how I handled multivariate outliers. I feel that in the end I was able to gain a better understanding and appreciation of the whole data wrangling process, and can use the skills and experiences gained to improve my approach to data wrangling in future.

Presentation link

Include the link to your video walkthrough here.

https://www.loom.com/share/14a3c8924da6409dbdf2ed375c27018e

References

VINITSHAH0110, 2022, *IPL Auction 2022*, Kaggle, viewed 14 April 2022, https://www.kaggle.com/datasets/vinitshah0110/ipl-auction-2022

Vora, S, 2022, *IPL 2022 Player Statistics*, Kaggle, viewed 14 April 2022, https://www.kaggle.com/datasets/vora1011/ipl-2022-player-statistics

Cricket Mastery, 2022, *How Does the IPL Auction Work*, Cricket Mastery, viewed 14 April 2022, https://cricketmastery.com/how-does-the-ipl-auction-work/

Harris, M, 2022, *What is a Century in Cricket? – Records and the Most Centuries*, It's Only Cricket, viewed 14 April 2022, https://www.itsonlycricket.com/what-is-a-century-in-cricket

Luke, 2022, *What Is A Four In Cricket? – All You Need To Know!*, Cricketers Hub, viewed 14 April 2022, https://cricketershub.com/what-is-a-four-in-cricket-all-you-need-to-know/

Harris, M, 2022, *What is an Over in Cricket? – How Many Balls are in an Over?*, It's Only Cricket, viewed 14 April 2022, https://www.itsonlycricket.com/what-is-an-over-in-cricket

Harris, M, 2022, *What is a Duck in Cricket Lingo: from Golden Duck to Platinum*, It's Only Cricket, viewed 14 April 2022, https://www.itsonlycricket.com/duck-in-cricket

Harris, M, 2022, *Maiden Over: Meaning of the Term and Most Maiden Overs in Cricket*, It's Only Cricket, viewed 14 April 2022, https://www.itsonlycricket.com/maiden-over

Wikipedia, 2022, *Economy rate*, Wikipedia, viewed 14 April 2022, https://en.wikipedia.org/wiki/Economy_rate#:~:text=In%20cricket%2C%20a%20player's%20economy,better%20the%20bowler%20is%20performing.

Wikipedia, 2022, *Crore*, Wikipedia, viewed 14 April 2022, https://en.wikipedia.org/wiki/Crore

Data Science Made Simple, 2022, *GET AGE FROM DATE OF BIRTH IN R*, Data Science Made Simple, viewed 22 April 2022, https://www.datasciencemadesimple.com/get-age-from-date-of-birth-in-r-2/#:~:text=Age%20is%20extracted%20from%20Date_of_birth,by%2052.25%2C%20as%20shown%20below.

Stack Overflow zx8754, 2021, *Replace contents of factor column in R dataframe* Stack Overflow, viewed 22 April 2022, https://stackoverflow.com/questions/11810605/replace-contents-of-factor-column-in-r-dataframe

R for Data Science Wickham, H and Grolemund, G, 2016, *R for Data Science*, viewed 15 April 2022

Garrett Grolemund, Hadley Wickham (2011). Dates and Times Made Easy with lubridate. Journal of Statistical Software, 40(3), 1-25. URL https://www.jstatsoft.org/v40/i03/.

Stefan Milton Bache and Hadley Wickham (2022). magrittr: A Forward-Pipe Operator for R. R package version 2.0.2. https://CRAN.R-project.org/package=magrittr

Hadley Wickham, Romain François, Lionel Henry and Kirill Müller (2022). dplyr: A Grammar of Data Manipulation. R package version 1.0.8. https://CRAN.R-project.org/package=dplyr

Hadley Wickham and Maximilian Girlich (2022). tidyr: Tidy Messy Data. R package version 1.2.0. https://CRAN.R-project.org/package=tidyr

Lukasz Komsta (2022). outliers: Tests for Outliers. R package version 0.15. https://CRAN.R-project.org/package=outliers

Wickham et al., (2019). Welcome to the tidyverse. Journal of Open Source Software, 4(43), 1686, https://doi.org/10.21105/joss.01686

Mark van der Loo, Edwin de Jonge and Sander Scholtus (2015). deducorrect: Deductive Correction, Deductive Imputation, and Deterministic Correction. R package version 1.3.7. https://CRAN.R-project.org/package=deducorrect

Mark van der Loo and Edwin de Jonge (2021). deductive: Data Correction and Imputation Using Deductive Methods. R package version 1.0.0. https://CRAN.R-project.org/package=deductive

Mark P. J. van der Loo, Edwin de Jonge (2021). Data Validation Infrastructure for R. Journal of Statistical Software, 97(10), 1-31. doi:10.18637/jss.v097.i10

Frank E Harrell Jr (2021). Hmisc: Harrell Miscellaneous. R package version 4.6-0. https://CRAN.Rproject.org/package=Hmisc

Korkmaz S, Goksuluk D, Zararsiz G. MVN: An R Package for Assessing Multivariate Normality. The R Journal. 2014 6(2):151-162.

Hadley Wickham, Jim Hester and Jennifer Bryan (2022). readr: Read Rectangular Text Data. R package version 2.1.2. https://CRAN.R-project.org/package=readr

Philipp Schauberger and Alexander Walker (2021). openxlsx: Read, Write and Edit xlsx Files. R package version 4.2.5. https://CRAN.R-project.org/package=openxlsx

Adrian Dragulescu and Cole Arendt (2020). xlsx: Read, Write, Format Excel 2007 and Excel 97/2000/XP/2003 Files. R package version 0.6.5. https://CRAN.R-project.org/package=xlsx

Yihui Xie (2022). tinytex: Helper Functions to Install and Maintain TeX Live, and Compile LaTeX Documents. R package version 0.38.

Yihui Xie (2019) TinyTeX: A lightweight, cross-platform, and easy-to-maintain LaTeX distribution based on TeX Live. TUGboat 40 (1): 30–32. https://tug.org/TUGboat/Contents/contents40-1.html