MATH2349 Semester 2, 2018

Assignment 3

Amal Joy (S3644794) & Nupura Sanjay Sawle (S3639703)

14 October 2018

Contents

1 Required packages

The packages that are required for this project are loaded here.

```
library(readr)
library(lubridate)
library(dplyr)
library(tidyr)
library(knitr)
library(outliers)
library(forecast)
```

2 Executive Summary

This project is performed to illustrate the data preprocessing tasks carried out for preparing a data ready for statistical tests and predictions. The data set included in the project is a cricket (IPL) dataset obtained from a open source data website called Kaggle. The purpose of this analysis is to make the data ready for an analysis that includes Match details and the man of the match player details. The dataset is obtained from different sources which is then merged using data joining techniques in the dplyr package in R program. First summary and structure of the dataset is studies using relevant tools avvailable in R. Then there is Data type coversion which includes character to date, character to factor and character to integer. There was a factor which required labelling of the factor levels. Data idying includes identifying the variables that doesnot conform to the tidy data princle and using the spread function to make it tidy. A new variable called Player age was created from the date of birth of the player. The data was then scanned for any null values and inconsistencies. Although there were no inconsistencies in the dataset there were few null values which has been treated in a proper way and finnaly made the data set without any null values. Then the data was scanned for outliers using two different techniques, BoxPlot and Z-score methods. There were few outlier found in the dataset which was then identified as not outliers. Data transformation techniques like log transformation and BoxCox transformation was applied on the numeric variables to change the scale of variables and to make the distribution symmetric. Finnaly the data was normally distrinuted and had no outliers or NA values and thus was ready for the further analysis.

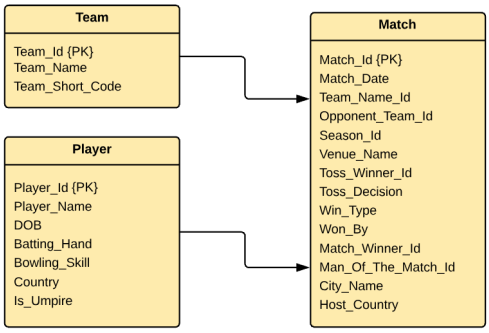
3 Data

The original source of the data set is Cricsheet.org. The data is available in YAML format with different matches recorded in different files. This data was combined by Kaggle users and were provided in the Kaggle website for downloads. The data was provided in 3 different files, including data for team, data for the player information and match data. Click to download the Team data, Player Data and Match Data from Kaggle.
The Team data contains team informations given below:
1. **Team_Id** - *Primary key for the team table*
2. **Team_Name** - *Name of the Team*
3. **Team_Short_Code** - *Short code for each team*

The Player dataset contains information given below:
1. **Player_Id** - *Primary key for the player table*
2. **Player_Name** - *Name of the player*
3. **DOB** - *Date of Birth of the player*
4. **Batting_Hand** - *Batting Hand of the Player*
5. **Bowling_Skill** - *Type of bowling practised by the bowler*
5. **Country** - *Country of the Player*
6. **Is_Umpire** - *Person is Umpire or not*

The Match dataset contains the information given below:
1. **Match_Id** - *Primary key for Match table*
2. **Match_Date** - *Date of match*
3. **Season** - *Season of the match*
4. **Venue_Name** - *Venue of the match*
5. **Toss_Decision** - *Outcome of the Toss*
6. **Win_Type** - *Winning Type of the Match*
7. **Won_By** - *Number of runs or wickets team won the match*
8. **City_Name** - *Name of the city where is match is happening*
9. **Host_Country** - *Name of the Country Hosting the match*
10. **Team_1** - *Name of one Team*
11. **Team_2** - *Name of team two*
12. **Toss_Winner** - *Name of the team winning the Toss*
13. **Match_Winner** - *Name of the winning Team*
14. **Man_Of_The_Matc** - *Name of the Man of the Match player*
15. **DOB** - *Date of Birth of the man of the match player*
16. **Player_Country** - *Man of The Match player country name*
17. **Age** - *Age of the Man of the Match player*

The Match dataset is the main datset where we are merging other two tables into. This dataset contains Team ID information for both the teams in the match, winning team ID, Man of the Match player ID etc which are used as the key for merging. The ER Diagram of the datasets are given below.



3.1 Importing datasets

```
Team <- read_csv("Cricket/Team.csv") %>% as.data.frame()

Player <- read_csv("Cricket/Player.csv",
                   col_types = cols(X8 = col_skip())) %>% as.data.frame()

Match <- read_csv("Cricket/Match.csv") %>% as.data.frame()
```

The datasets are imported using the read_csv function as data frames. The read files are saves in to files named `Team`, `Player`, and `Match`. The player dataset contained an eighth column which was a null column. We removed that column using col_skip function. The head of the datsets are given below.

```
head(Team) # Head of Team dataset
```

```
##   Team_Id                  Team_Name Team_Short_Code
## 1       1       Kolkata Knight Riders             KKR
## 2       2 Royal Challengers Bangalore             RCB
## 3       3         Chennai Super Kings             CSK
## 4       4             Kings XI Punjab            KXIP
```

```
## 5       5            Rajasthan Royals              RR
## 6       6            Delhi Daredevils              DD
```

```
head(Player) # Head of Player dataset
```

```
##   Player_Id     Player_Name       DOB Batting_Hand      Bowling_Skill
## 1         1      SC Ganguly  8-Jul-72    Left_Hand   Right-arm medium
## 2         2     BB McCullum 27-Sep-81   Right_Hand   Right-arm medium
## 3         3      RT Ponting 19-Dec-74   Right_Hand   Right-arm medium
## 4         4       DJ Hussey 15-Jul-77   Right_Hand Right-arm offbreak
## 5         5 Mohammad Hafeez 17-Oct-80   Right_Hand Right-arm offbreak
## 6         6        R Dravid 11-Jan-73   Right_Hand Right-arm offbreak
##       Country Is_Umpire
## 1       India         0
## 2 New Zealand         0
## 3   Australia         0
## 4   Australia         0
## 5    Pakistan         0
## 6       India         0
```

```
head(Match) # Head of Match dataset
```

```
##   Match_Id Match_Date Team_Name_Id Opponent_Team_Id Season_Id
## 1   335987  18-Apr-08            2                1         1
## 2   335988  19-Apr-08            4                3         1
## 3   335989  19-Apr-08            6                5         1
## 4   335990  20-Apr-08            7                2         1
## 5   335991  20-Apr-08            1                8         1
## 6   335992  21-Apr-08            5                4         1
##                                 Venue_Name Toss_Winner_Id Toss_Decision
## 1                     M Chinnaswamy Stadium              2         field
## 2 Punjab Cricket Association Stadium, Mohali              3           bat
## 3                          Feroz Shah Kotla              5           bat
## 4                          Wankhede Stadium              7           bat
## 5                              Eden Gardens              8           bat
## 6                     Sawai Mansingh Stadium              4           bat
##     Win_Type Won_By Match_Winner_Id Man_Of_The_Match_Id  City_Name
## 1    by runs    140               1                   2  Bangalore
## 2    by runs     33               3                  19 Chandigarh
## 3 by wickets      9               6                  90      Delhi
## 4 by wickets      5               2                  11     Mumbai
## 5 by wickets      5               1                   4    Kolkata
## 6 by wickets      6               5                  32     Jaipur
##   Host_Country
## 1        India
## 2        India
## 3        India
## 4        India
## 5        India
## 6        India
```

3.2 Dataset Merging

The Match data set is taking data from other two data sets for filling in the informations like team names, player names etc. Left_join is used to merge the datasets. All the variables in Match dataset is kept and only

few necessary variables from other datsets are merged in to Match dataset. The below code shows Match dataset and the Team dataset where `Team_Id` is used as the common key. This is used as the `Team1` details with the team names. The column `Team_Name_Id` is then removed from the Match dataset.

```
# Taking team name for Team1 from Team table
Match <- Match %>%
  left_join(select(Team, Team_Id, Team_Name),
            c("Team_Name_Id" = "Team_Id")) %>%
  rename(Team_1 = "Team_Name") %>%
  select(-Team_Name_Id)
```

The code below shoes the merging of Match dataset with the Team dataset for substituting the opponent team IDs with the team names. This column is then given the name as `Team_2`.

```
# Taking team name for Team2 from Team table
Match <- Match %>%
  left_join(select(Team, Team_Id, Team_Name),
            c("Opponent_Team_Id" = "Team_Id")) %>%
  rename(Team_2 = "Team_Name") %>%
  select(-Opponent_Team_Id)
```

The `Toss_Winner_Id` is replaced with the corrosponding team names by joining Match dataset with Team dataset on `Team_Id` and `Toss_Winner_Id`. Toss_Winner_Id is then removed from the datset.

```
# Taking team name for Toss Winner from Team table
Match <- Match %>%
  left_join(select(Team, Team_Id, Team_Name),
            c("Toss_Winner_Id" = "Team_Id")) %>%
  rename(Toss_Winner = "Team_Name") %>%
  select(-Toss_Winner_Id)
```

The variable `Match_Winner_Id` is replaced with the corrosponding team names by joining Match dataset with Team dataset on `Team_Id` and `Match_Winner_Id`. Match_Winner_Id is then removed from the datset.

```
# Taking team name for Match winner from Team table
Match <- Match %>%
  left_join(select(Team, Team_Id, Team_Name),
            c("Match_Winner_Id" = "Team_Id")) %>%
  rename(Match_Winner = "Team_Name") %>%
  select(-Match_Winner_Id)
```

Additional Data is added to the Match dataset from the Player dataset by using left_join. The datasets are joined using the column `Man_Of_The_Match_Id` from the Match datset and `Player_Id` from the player dataset. `Man_Of_The_Match_Id` is replaced with the corrosposnding player names. Additional variables like player date of birth and player country are added to the dataset.

```
# Taking player name from player table
Match <- Match %>%
  left_join(select(Player, Player_Id, Player_Name, DOB, Country),
            c("Man_Of_The_Match_Id"="Player_Id")) %>%
  rename(Player_Country = "Country", Man_Of_The_Match="Player_Name") %>%
  select(-Man_Of_The_Match_Id)
```

The head of the dataset after all the merging is given below.

```
head(Match)
```

```
##   Match_Id Match_Date Season_Id                                Venue_Name
## 1   335987  18-Apr-08         1                    M Chinnaswamy Stadium
```

```
## 2   335988  19-Apr-08         1 Punjab Cricket Association Stadium, Mohali
## 3   335989  19-Apr-08         1                           Feroz Shah Kotla
## 4   335990  20-Apr-08         1                           Wankhede Stadium
## 5   335991  20-Apr-08         1                               Eden Gardens
## 6   335992  21-Apr-08         1                       Sawai Mansingh Stadium
##   Toss_Decision   Win_Type Won_By  City_Name Host_Country
## 1         field    by runs    140  Bangalore        India
## 2           bat    by runs     33 Chandigarh        India
## 3           bat by wickets      9      Delhi        India
## 4           bat by wickets      5     Mumbai        India
## 5           bat by wickets      5    Kolkata        India
## 6           bat by wickets      6     Jaipur        India
##                        Team_1                       Team_2
## 1 Royal Challengers Bangalore       Kolkata Knight Riders
## 2             Kings XI Punjab         Chennai Super Kings
## 3            Delhi Daredevils            Rajasthan Royals
## 4              Mumbai Indians Royal Challengers Bangalore
## 5       Kolkata Knight Riders             Deccan Chargers
## 6            Rajasthan Royals             Kings XI Punjab
##                   Toss_Winner                Match_Winner Man_Of_The_Match
## 1 Royal Challengers Bangalore       Kolkata Knight Riders      BB McCullum
## 2         Chennai Super Kings         Chennai Super Kings       MEK Hussey
## 3            Rajasthan Royals            Delhi Daredevils      MF Maharoof
## 4              Mumbai Indians Royal Challengers Bangalore       MV Boucher
## 5             Deccan Chargers       Kolkata Knight Riders        DJ Hussey
## 6             Kings XI Punjab            Rajasthan Royals        SR Watson
##         DOB Player_Country
## 1 27-Sep-81    New Zealand
## 2 27-May-75      Australia
## 3  7-Sep-84          India
## 4  3-Dec-76   South Africa
## 5 15-Jul-77      Australia
## 6 17-Jun-81      Australia
```

4 Understand the data

The structure of the Match dataset is given below.

```
str(Match,  vec.len=2) # Structure of Match data
```

```
## 'data.frame':    577 obs. of  16 variables:
##  $ Match_Id        : int  335987 335988 335989 335990 335991 ...
##  $ Match_Date      : chr  "18-Apr-08" "19-Apr-08" ...
##  $ Season_Id       : int  1 1 1 1 1 ...
##  $ Venue_Name      : chr  "M Chinnaswamy Stadium" "Punjab Cricket Association Stadium, Mohali" ...
##  $ Toss_Decision   : chr  "field" "bat" ...
##  $ Win_Type        : chr  "by runs" "by runs" ...
##  $ Won_By          : chr  "140" "33" ...
##  $ City_Name       : chr  "Bangalore" "Chandigarh" ...
##  $ Host_Country    : chr  "India" "India" ...
##  $ Team_1          : chr  "Royal Challengers Bangalore" "Kings XI Punjab" ...
##  $ Team_2          : chr  "Kolkata Knight Riders" "Chennai Super Kings" ...
##  $ Toss_Winner     : chr  "Royal Challengers Bangalore" "Chennai Super Kings" ...
```

```
##  $ Match_Winner    : chr  "Kolkata Knight Riders" "Chennai Super Kings" ...
##  $ Man_Of_The_Match: chr  "BB McCullum" "MEK Hussey" ...
##  $ DOB             : chr  "27-Sep-81" "27-May-75" ...
##  $ Player_Country  : chr  "New Zealand" "Australia" ...
```

There are 577 observations with 17 variables. Except `Match_ID` and `Season_ID`, all other variables are character variables. `Match_Date` and `DOB` are date variables and are represented as character here. Similarly, the variable `Won_by` is showing the run by which the team has won. It is an integer variable and is represent as character. Also there are some variable which should hav been factor by default.

4.1 Data Type Conversions

Now we will convert all these variables into their respective formats.

4.1.1 Date Conversions

The variables to be converted as date type are `Match_Date` and `DOB`. The date format given here is `DD-MM-YY`. So we will use the Lubridate function `dmy()` to covert in to date type.

```
colm1 <- c("Match_Date", "DOB") #Columns to be converted
Match[colm1] <- lapply(Match[colm1], dmy) # List apply the dmy() function
str(Match[colm1]) # Structure after aconversion
```

```
## 'data.frame':    577 obs. of  2 variables:
##  $ Match_Date: Date, format: "2008-04-18" "2008-04-19" ...
##  $ DOB       : Date, format: "1981-09-27" "1975-05-27" ...
```

The columns has been successfully converted to date format.

4.1.2 Character to factor coversion

The variables to be converted as factor type are `Season_Id`, `Toss_Decision`, `Win_Type`, `City_Name`, `Host_Country`, `Team_1`, `Team_2`, `Toss_Winner`, `Match_Winner`, and `Player_Country`. These variable names are first saved as a list and then fed this in to a list apply function with factor function in it.

```
colm2 <- c("Season_Id", "Toss_Decision", "Win_Type", "City_Name", "Host_Country",
           "Team_1", "Team_2", "Toss_Winner", "Match_Winner", "Player_Country")
Match[colm2] <- lapply(Match[colm2], factor) # Factor conversion
summary(Match[colm2]) # summary of the factor variables
```

```
##    Season_Id   Toss_Decision       Win_Type        City_Name
##  6      : 76   bat  :262     by runs   :261   Mumbai    : 77
##  5      : 74   field:315     by wickets:307   Bangalore : 58
##  4      : 73                 No Result :  3   Kolkata   : 54
##  3      : 60                 Tie       :  6   Delhi     : 53
##  7      : 60                                  Chennai   : 48
##  9      : 60                                  Chandigarh: 42
##  (Other):174                                  (Other)   :245
##        Host_Country                          Team_1
##  India       :500   Royal Challengers Bangalore: 78
##  South Africa: 57   Chennai Super Kings        : 74
##  U.A.E       : 20   Delhi Daredevils           : 69
##                     Kings XI Punjab            : 65
##                     Kolkata Knight Riders      : 62
```

```
##                     Mumbai Indians             : 62
##                     (Other)                    :167
##                          Team_2                   Toss_Winner
##  Mumbai Indians             : 78   Mumbai Indians       : 74
##  Kolkata Knight Riders      : 70   Kolkata Knight Riders: 69
##  Kings XI Punjab            : 69   Chennai Super Kings  : 66
##  Rajasthan Royals           : 66   Delhi Daredevils     : 64
##  Delhi Daredevils           : 64   Kings XI Punjab      : 64
##  Royal Challengers Bangalore: 61   Rajasthan Royals     : 63
##  (Other)                    :169   (Other)              :177
##                       Match_Winner      Player_Country
##  Mumbai Indians             : 80   India       :293
##  Chennai Super Kings        : 79   Australia   :110
##  Royal Challengers Bangalore: 70   South Africa: 63
##  Kolkata Knight Riders      : 68   West Indies : 53
##  Kings XI Punjab            : 63   Sri Lanka   : 24
##  (Other)                    :214   (Other)     : 31
##  NA's                       :  3   NA's        :  3
```

4.1.3 Charater to Integer coversion

The variable `Won_By` should be an interger variables as it represents runs. So we are converting them to integer by using the the following function.

```
Match$Won_By <- as.integer(Match$Won_By)
summary(Match$Won_By)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    1.00    6.00    8.00   17.35   21.00  144.00       9
```

The minimum run by which any team had won is 1 run and maximum run by which any team had won is 144 runs. There are 9 NA values in this column.

4.2 Factor level labelling

From the structure of the dataset it has been seen that the season is given as integers while it actually represents different game seasons. Afterwards we converted this factor to factor but the factor levels are 1 to 6. So here we are renaming the factor levels to proper season names. Also we are renaming the variable from Season_ID to `Season`.

```
summary(Match$Season_Id) #Summary of the Season_ID
```

```
##  1  2  3  4  5  6  7  8  9
## 58 57 60 73 74 76 60 59 60
```

```
colnames(Match)[3] <- "Season" # Renaming the variable Season_Id
levels(Match$Season) <- c("Season 1", "Season 2", "Season 3",
                          "Season 4", "Season 5", "Season 6",
                          "Season 7", "Season 8", "Season 9")
levels(Match$Season) # New levels of season
```

```
## [1] "Season 1" "Season 2" "Season 3" "Season 4" "Season 5" "Season 6"
## [7] "Season 7" "Season 8" "Season 9"
```

Labelling of the Seasons are successfully done here.

5 Tidy & Manipulate Data I

From the summary of the dataset and the head of the dataset we have seen that each variable in our dataset is having its own column except the columns `Won_by` and `Win_Type`. No other column headers are values and variable names.

```
Match[6:10,c(12,5,13,6,7)]
```

```
##         Toss_Winner Toss_Decision        Match_Winner   Win_Type Won_By
## 6   Kings XI Punjab           bat    Rajasthan Royals by wickets      6
## 7   Deccan Chargers           bat    Delhi Daredevils by wickets      9
## 8    Mumbai Indians         field Chennai Super Kings    by runs      6
## 9  Rajasthan Royals         field    Rajasthan Royals by wickets      3
## 10   Mumbai Indians         field     Kings XI Punjab    by runs     66
```

As seen from the dataset above, if Team wins after fielding first they win by wickets. If the team wins after bating first, they win by runs. There is no match were a team win by bith runs and wickets. It is mutually exclusive. So these variabeles has to be seperated, otherwise this variable will effect in transforming and normalising procedure towards the end.

```
a <- max(Match$Won_By[Match$Win_Type=="by runs"]) # max value of win by runs
b <- min(Match$Won_By[Match$Win_Type=="by runs"]) # max value of win by runs
c <- max(Match$Won_By[Match$Win_Type=="by wickets"]) # max value of win by runs
d <- min(Match$Won_By[Match$Win_Type=="by wickets"]) # max value of win by runs
data.frame("Win_Type"=c("By Runs","By wickets"),
           "Minimum" = c(b,d),
           "Maximum" = c(a,c))
```

```
##     Win_Type Minimum Maximum
## 1    By Runs       1     144
## 2 By wickets       1      10
```

```
table(Match$Win_Type) # number of observations under each variable
```

```
##
##    by runs by wickets  No Result        Tie
##        261        307          3          6
```

The maximum value of win by wickets is 10 and maximum value of win by runs is 144. There is a big difference in the scale if these values. Also there is almost equal representation of win by runs and win by wickets in the dataset. In this case most of the values of win by runs will be classified as outliers while cheking for outliers. Also this will posses problem while standardisation of the variables. So we are going to spread the variable Won by on key = Win_type and value as Won_By. We are filling the null values produced as a result of this spread with 999. This is an impossibe value for the variable and can be used as a reference for further analysis while transforming the variables. We are not keeping it as NA as it will introduce a NA value to all the observations and analysing the other null values will become diffficult. We are also saving a copy of the variable Win_Type for future reference as this variable will be removed from the dataset after the spread operation.

```
Win_Type <- Match$Win_Type # saving the original variable for further reference
Match <- spread(Match, key = Win_Type, value = Won_By, fill = 999)
table(Match$Tie) # table of Tie column
```

```
##
## 999
## 577
```

```
table(Match$`No Result`) # Table of No Reuslt Column
```

```
##
## 999
## 577
```

The variables formed after the spread oprations are `by_runs`, `by_wickets`, `Tie` and `No result`. The last two variable are not having any values as they were factors indicating that there was no result generated in the match. So we will remove this variables from the further analysis.

```
Match <- Match %>% select(-c("Tie","No Result"))
```

After spread operation on `Won_by` and `Win_Type` there is no multiple variables which are stored in one column. Also multiple columns are not forming a single variable. Each observation in the dataset represent a unique cricket match and thus each observation is having its own row. No multiple variables are stored in any rows. Each row is unique in its representation. Also each value is having its own cell. This proves that our dataset is now conforming to the tidy data principles.

6 Tidy & Manipulate Data II

Man of the Match Player's age is calculated from the Date of birth of the Player. Mutate function is used for this purpose. The base r function `as.period()` is used to find the current age of the player from today's date using today() function from Lubridate package.

6.1 Creation of calculated variable - Age

```
Match <- mutate(Match, Age = year(as.period(interval(Match$DOB, today()),
                        unit = "year")))
summary(Match$Age)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   21.00   31.00   34.00   34.84   38.00   49.00       3
```

The age of the player is successfully mutated in to the dataset. The minimum age of any player who became man of the match is 21 years and the maximum age any player who became man of the match is 49. But there are 3 NA's in the dataset. We will look in to the missing dataset in the next section.

7 Scan I

The dataset is scanned for any missing values. The columns with missing values are found by combining `colSums()` function and `is.na()` function. The columns names of those columns containing the null values and other interested variables are made in to a list . This list is used to create a dataset containing the rows of the those observation which has null values. This is done by using the function complete.cases for scanning the null value containing rows.

7.1 Scanning for NULL values

```
Match$Win_Type <- Win_Type
colSums(is.na(Match))
```

```
##         Match_Id       Match_Date           Season       Venue_Name
##                0                0                0                0
##    Toss_Decision        City_Name     Host_Country           Team_1
##                0                0                0                0
##           Team_2      Toss_Winner     Match_Winner Man_Of_The_Match
##                0                0                3                3
##              DOB   Player_Country          by runs       by wickets
##                3                3                0                0
##              Age         Win_Type
##                3                0
```

```
colm3 <- c( "Match_Id", "Match_Winner", "Win_Type", "by runs", "by wickets",
            "Man_Of_The_Match","Player_Country", "DOB"  )
Match[!complete.cases(Match), colm3] # Rows with NA values
```

```
##     Match_Id Match_Winner  Win_Type by runs by wickets Man_Of_The_Match
## 242   501270         <NA> No Result     999        999             <NA>
## 487   829768         <NA> No Result     999        999             <NA>
## 512   829818         <NA> No Result     999        999             <NA>
##     Player_Country  DOB
## 242           <NA> <NA>
## 487           <NA> <NA>
## 512           <NA> <NA>
```

It is found that the columns `DOB`, `Player_Country`, `Age`, `Man_Of_The_Match`, and `Match_Winner` contains 3 NULL values each and the variable `Won_By` is shown as `no result` for thses three variables. In total there are 15 NULL values in the datset. Using the function complete.cases(), we were able to see the actual issue with the dataset. The `DOB`, `Player_Country`, `Age`, `Man_Of_The_Match`, and `Match_Winner` which are having 3 missing values each registered as No result in the `Win_Type` variable. This means that the match was forfeited either due to bad weather or some other situations. So no result was produced fo that matches. This makes sense for missing values in those datasets. These 3 rows can be excluded as they are very less in number.

```
Match[Match$Win_Type=="Tie",c(1,18,11)]
```

```
##     Match_Id Win_Type               Match_Winner
## 67    392195      Tie            Rajasthan Royals
## 131   419126      Tie             Kings XI Punjab
## 329   598009      Tie         Sunrisers Hyderabad
## 342   598022      Tie Royal Challengers Bangalore
## 417   729320      Tie            Rajasthan Royals
## 477   829746      Tie             Kings XI Punjab
```

It can also be seen that for the matches which was a Tie, there was a Match winner recorded. This can be an error and can be corrected.

```
Match$Match_Winner <- as.character(Match$Match_Winner)
Match[Match$Win_Type=="Tie", "Match_Winner"] <- "Tie"
Match$Match_Winner <- as.factor(Match$Match_Winner)
```

So we added a new level to the variable `Match_Winner`. We converted it to a character first and then added the new level since an already declared factor will not allow addition of new factor levels in this manner. Then we converted that back to a factor using as.factor function.

7.2 Handling Null Values

Now we can impute zero values to the variables that have NA values for matches that are Tie while spreading the data. After this step we are be left with only 3 columns that have NA values. These are those matches that was forfeited due to some reasons and there was no outcome of the match. So we remove these columns as it constitute only 0.5199307% of the total dataset. The function `na.omit` is used to omit this variable from the analysis.

```
Match[!complete.cases(Match), colm3] # Rows with NA values
```

```
##     Match_Id Match_Winner  Win_Type by runs by wickets Man_Of_The_Match
## 242   501270         <NA> No Result     999        999             <NA>
## 487   829768         <NA> No Result     999        999             <NA>
## 512   829818         <NA> No Result     999        999             <NA>
##     Player_Country  DOB
## 242           <NA> <NA>
## 487           <NA> <NA>
## 512           <NA> <NA>
```

```
Match <- na.omit(Match) # Excluding remaining NA values from the dataset
sum(is.na(Match)) # Total number of NA values remaining in the dataset
```

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

After this operations the total sum of the na values in the datset is found to be zero and now we can proceed with the further steps without null values.

7.3 Checking inconsistencies

Inconsistencies in the dataframe like presence of `-Inf`, `Inf` and `NaN` are scanned using the following function called is.special. This fuction checks if the current value is finite ot not. A subset of numeric dataset is created from the Match dataset using the is.numeric function. Then the is.special function is applied to the entire numeric dataframe using sapply as follows.

```
# function for searhing the infinite, and NaN values
is.special <- function(x){
 !is.finite(x)
}
# checking for the inconsistencies in the numeric values
numData <- Match[,sapply(Match, is.numeric)] # subset of numeric dataset
colSums(sapply(numData, is.special))
```

```
##   Match_Id    by runs by wickets        Age
##          0          0          0          0
```

It is found that there are no inconsistend or special values in the dataset.

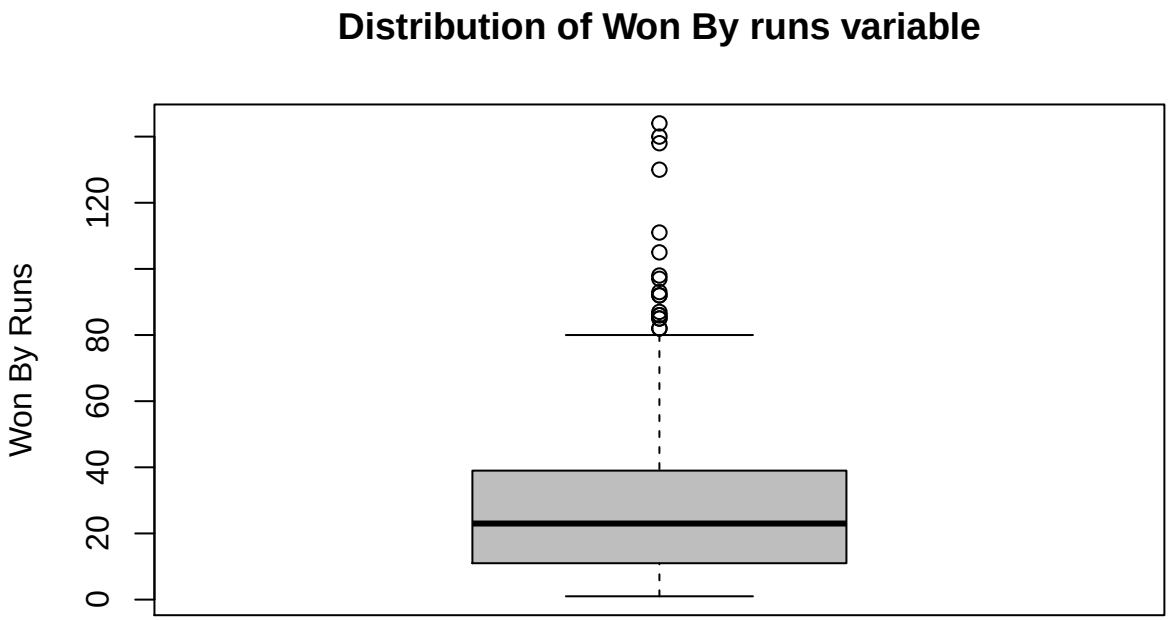
8 Scan II - Outliers

In this step data is scanned for outliers. In Match dataset there are three numeric variables `Age`,`by run` and `by wickets`. Now data set have three numeric variables `by runs`, `by wickets` and `age`. We have to convert the 999 values in the datset back to null values for better clarity of the ouliers.

```
Match$`by runs`[Match$`by runs`==999] <- NA
Match$`by wickets`[Match$`by wickets`==999] <- NA
```

Now let's check outliers for each of it.

```
By_runs <- Match$`by runs` %>% na.omit()
# by runs
By_runs %>%  boxplot(main = "Distribution of Won By runs variable", ylab =
"Won By Runs", col = "grey")
```



The `by run` is having many outliers as shown by the BoxPlot. The boxplot is influenced by the low level values in the dataet. In fact these are not actual outliers and this appers to be outliers because of the low run values present in the data. Lets look the zscore values.

```
# Z score
z.scores <- By_runs %>%  scores(type = "z") %>% na.omit()
z.scores %>% summary()
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -1.0849 -0.7149 -0.2709  0.0000  0.3211  4.2062
```

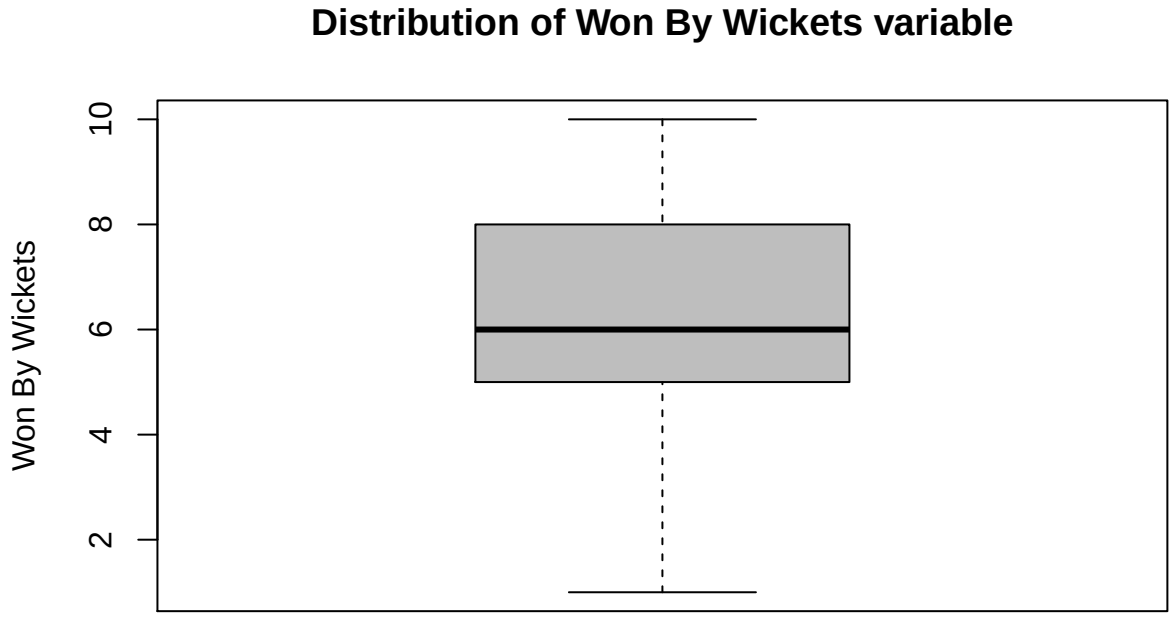
```
length (which(abs(z.scores) > 3))
```

```
## [1] 4
```

Checking outliers using z score method. Applying z-score method it is found that there are 4 outliers in `by runs` method.

Next we check the `by wickets` variable for outliers, First we plot the BoxPlot.

```
# by wickets
By_wickets <- Match$`by wickets` %>% na.omit()
By_wickets %>%  boxplot(main = "Distribution of Won By Wickets variable", ylab =
"Won By Wickets", col = "grey")
```



There seems to be no outlier at all. So we will check the z-scores.

```
# Z score
z.scores <- By_wickets %>%  scores(type = "z")
z.scores %>% summary()
```

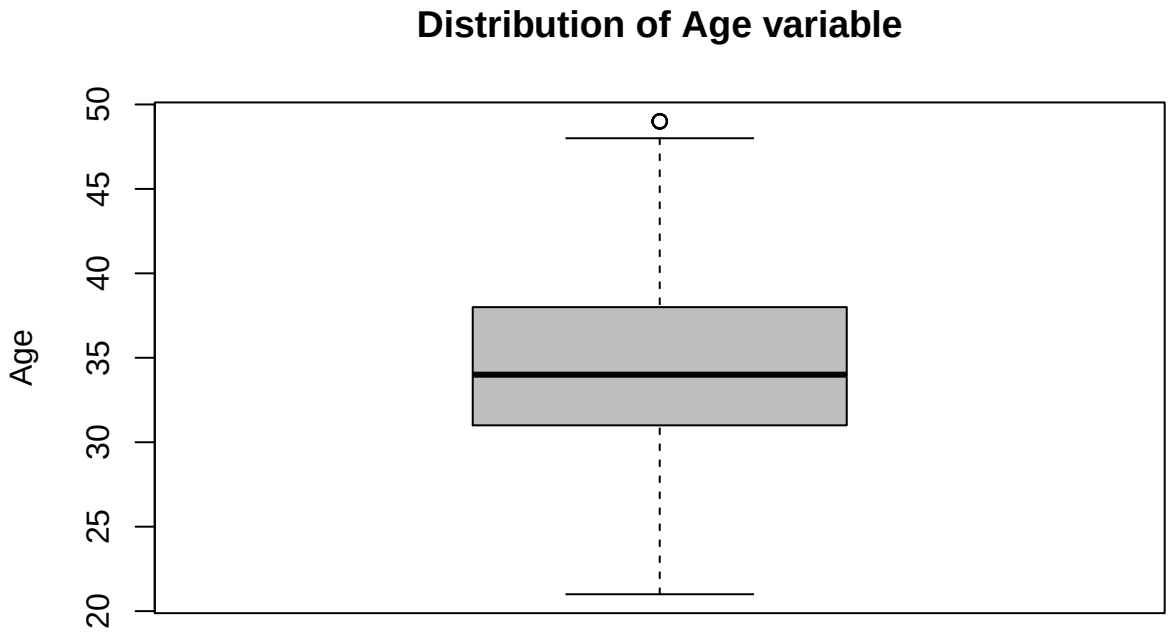
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -2.9755 -0.7393 -0.1803  0.0000  0.9378  2.0559
```

```
length (which(abs(z.scores) > 3))
```

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

Since the maximum z-score is 2.05 which is less than 3, there is no outlier with the Z-score as well. Next we will check for the age parameter.

```
# Age
Match$Age %>%  boxplot(main = "Distribution of Age variable", ylab =
"Age", col = "grey")
```



There seems to be one variable which is shown as an outlier by the Boxplot method. We found that this came from the maximum age of 49.

```
max(Match$Age) # Maximum age in the dataset
```

```
## [1] 49
```

```
unique(Match$Man_Of_The_Match[Match$Age>48]) # players with maximum age
```

```
## [1] "ST Jayasuriya" "SK Warne"
```

We found that "ST Jayasuriya" and "SK Warne" are the players with maximum age and confirmed from Internet Source, CrickBuzz, that Sanath Jayasuriya and Shane Warne are having 49 years of age. So this is not an outlier.

```
# Z score
z.scores_Age <- Match$Age %>%  scores(type = "z")
z.scores_Age %>% summary()
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -2.4663 -0.6849 -0.1505  0.0000  0.5620  2.5215
```

```
length (which(abs(z.scores_Age) > 3))
```

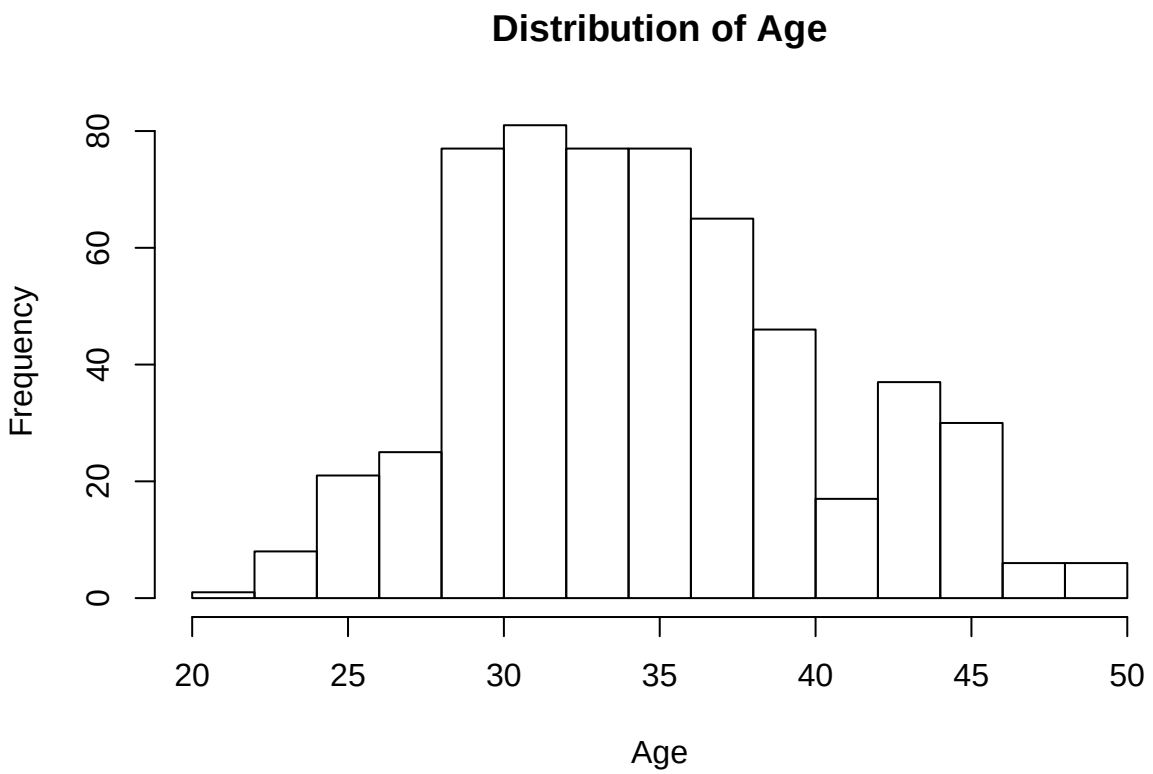
```
## [1] 0
```

Proving that the z-score shows there is no outliers. As the above box plots indicates `by run` variable is showing outliers. Where as `age` and `by wickets` is not showing any outliers in box plot.

9 Transform

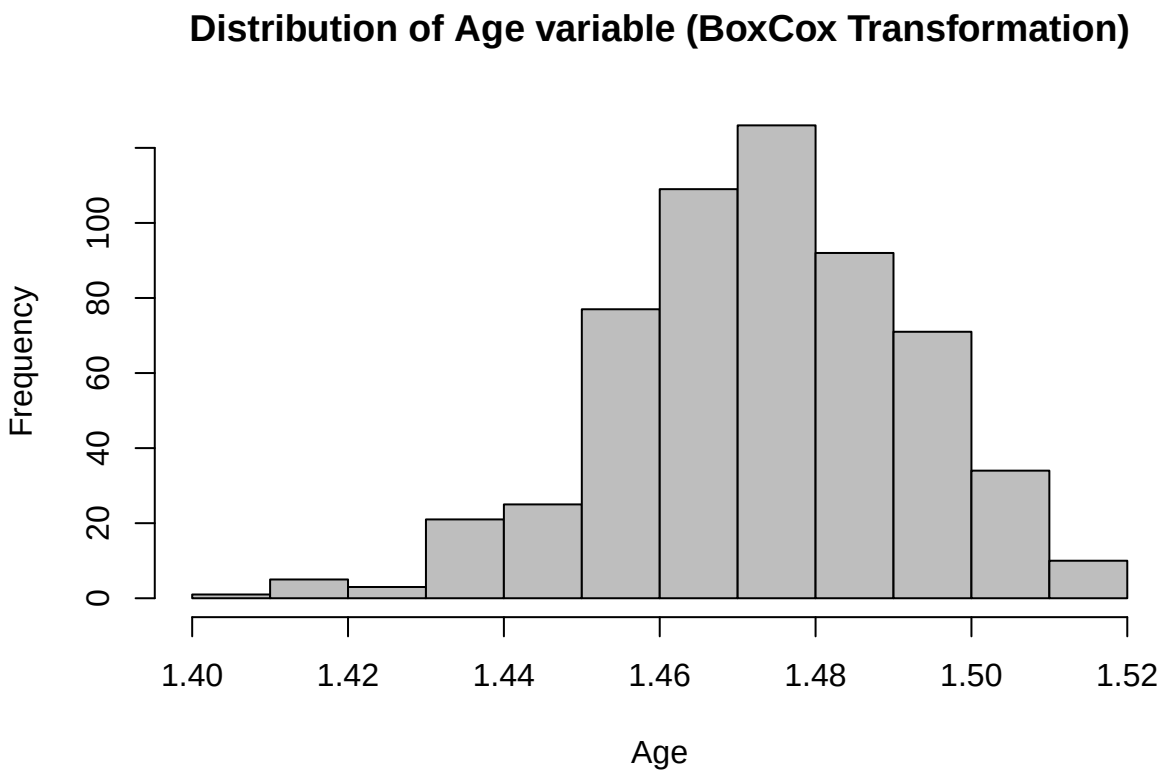
The the numerical varianles in the datasert is checked for further proceeding with statistical analysis. As we have seen in the Outliers of all the numerical variables, there are mutually exclusive sets which are having different scaling in the datset. Also due to introduction of zero in case of null values we have to seperate this values while analysis of First we will analysis the Age variable.

```r
# Age is almost normally distributed
hist(Match$Age,main = "Distribution of Age",xlab="Age")
```



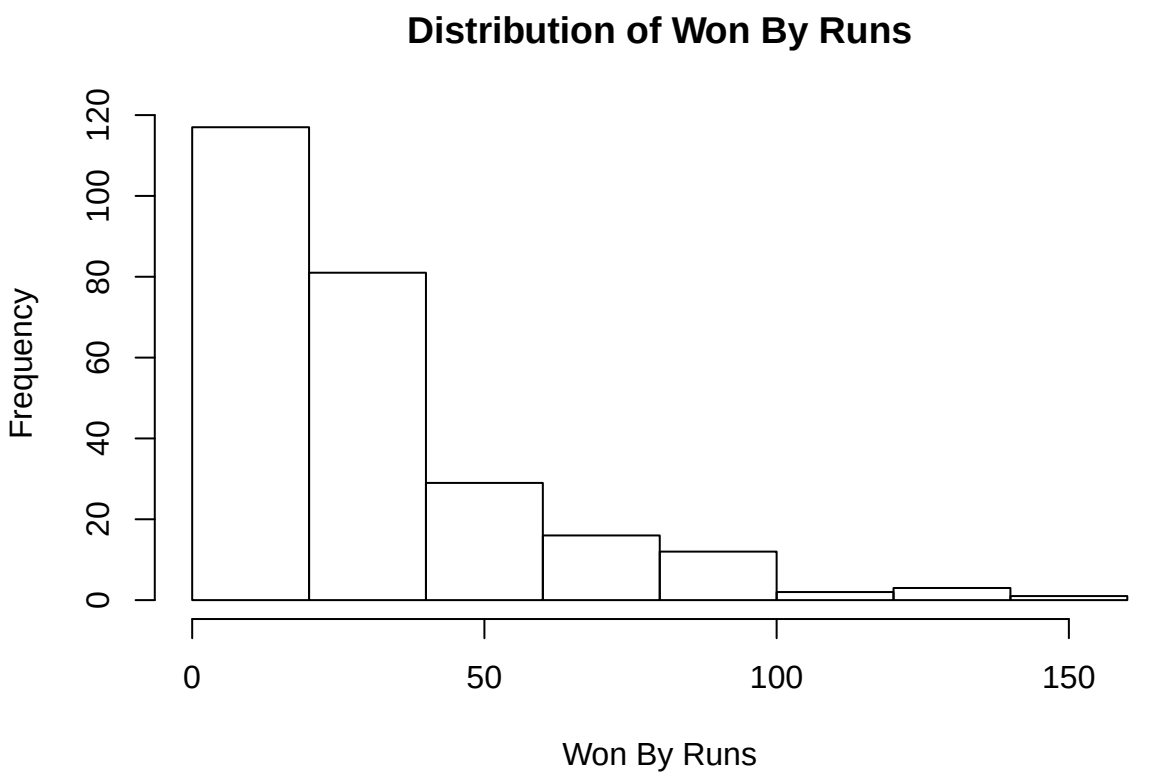
The histogram is slightly skewed to the right. After numerous trial and error methods BoxCox transformation was making the data better. It is shown below.

```r
# Box-Cox transformation
Age_box<- BoxCox(Match$Age,lambda = "auto")
hist(Age_box,
     main = "Distribution of Age variable (BoxCox Transformation)",
     xlab = "Age",
     col = "grey")
```



Even though it is slightly skewed to the left the dataset has improved in its symmetry. Now lets look the variable `By_runs`.

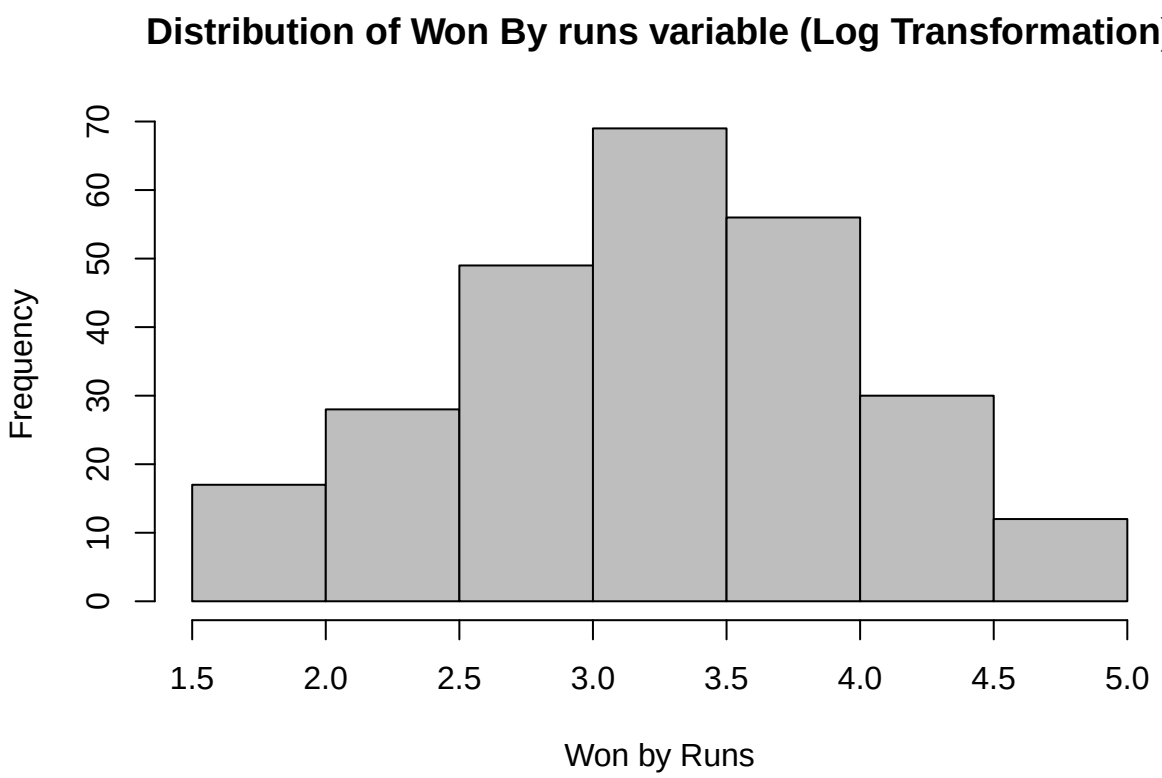
```
hist(By_runs, main = "Distribution of Won By Runs", xlab = "Won By Runs")
```



`by runs` is hghly skewed to the right. Since log transformation is better for improving the disrtibutions with

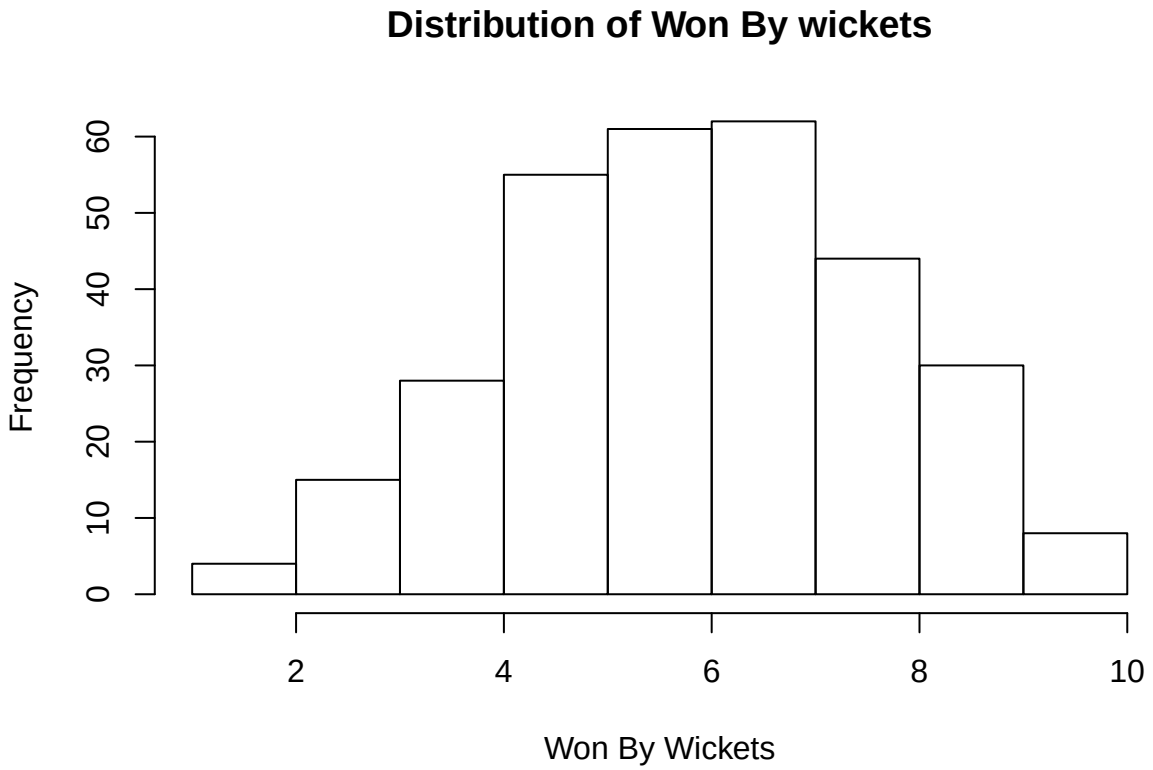
right skewness we apply log transformation to this dataset. We also add a constant 4 to the data for batter results.

```
By_runs <- By_runs+4
# Natural Log transformation
runs_ln <- log(By_runs)
hist(runs_ln,
     main = "Distribution of Won By runs variable (Log Transformation)",
     xlab = "Won by Runs",
     col = "grey")
```



It is seen that the dataset has improved a lot in this case. Log transformation has removed the high skewness in the data. Now lets check for the wicktes data.

```
hist(By_wickets, main = "Distribution of Won By wickets",
     xlab = "Won By Wickets")
```



Here the data is almost symmetrically distribted and doesnt need any more transformation.

10 Conclusion

The data set was in untidy format which was converted in to a tidy format according to tidy data principles. Many NA values in the data set was also handled properly by imputing and excluding some observations. Many data type conversion were necessary. Outliers in the datset were proved to be not an outlier and they were identified not influencing the distribution of the dataet as a whole. The data was transformed using log and bocox transformation methods to make the numreical variables normal. Now the dataset is ready for further data analysis techniques like regression, classification and clustering..