## The Problem:

- **Topic**: Rain in Australia.
- **Description**: This dataset contains about 10 years of daily weather observations from many locations across Australia.
  RainTomorrow is the target variable to predict. It means -- did it rain the next day, Yes or No? This column is Yes if the rain for that day was 1mm or more.
- Predict next-day rain by training classification models on the target variable Rain Tomorrow.

  Note: We will accept the model if accuracy greater than 75%, we will delete missing values and outliers if percentage less than 3%,

## The Tool Used:

- R

## The Link For The Dataset:

https://www.kaggle.com/datasets/jsphyg/weather-dataset-rattle-package

## Number Of Attributes:

- 23 attributes.

## Number Of Observations:

- 145460 observations.

Importing dataset to Rstudio and show the dimension of data frame

```
> df<-read.csv("C:/dataMining/weatherAUS.csv")
> dim(df)
[1] 145460     23
```

## Description (Definition And Type Of Attributes):

| Attribute | Definition | Type |
|---|---|---|
| Date | The date of observation | Nominal |
| Location | The common name of the location of the weather station | Nominal |
| Min Temp | The minimum temperature in degrees celsius | Numeric |
| Max Temp | The maximum temperature in degrees celsius | Numeric |
| Rainfall | The amount of rainfall recorded for the day in mm | Numeric |
| Evaporation | The so-called Class A pan evaporation (mm) in the 24 hours to 9am | Numeric |
| Sunshine | The number of hours of bright sunshine in the day | Numeric |
| WindGustDir | The direction of the strongest wind gust in the 24 hours to midnight | Nominal |
| WindGustSpeed | The speed (km/h) of the strongest wind gust in the 24 hours to midnight | Numeric |
| WindDir9am | Direction of the wind at 9am | Nominal |
| WindDir3pm | Direction of the wind at 3pm | Nominal |
| WindSpeed9am | Wind speed (km/hr) averaged over 10 minutes prior to 9am | Numeric |
| WindSpeed3pm | Wind speed (km/hr) averaged over 10 minutes prior to 3pm | Numeric |
| Humidity9am | Humidity (percent) at 9am | Numeric |
| Humidity3pm | Humidity (percent) at 3pm | Numeric |
| Pressure9am | Atmospheric pressure (hpa) reduced to mean sea level at 9am | Numeric |
| Pressure3pm | Atmospheric pressure (hpa) reduced to mean sea level at 3pm | Numeric |
| Cloud9am | Fraction of sky obscured by cloud at 9am. This is measured in "oktas", which are a unit of eigths. It records how many | Numeric |
| Cloud3pm | Fraction of sky obscured by cloud (in "oktas": eighths) at 3pm. See Cload9am for a description of the values | Numeric |
| Temp9am | Temperature (degrees C) at 9am | Numeric |
| Temp3pm | Temperature (degrees C) at 3pm | Numeric |
| RainToday | Boolean: 1 if precipitation (mm) in the 24 hours to 9am exceeds 1mm, otherwise 0 | Binary |
| RainTomorrow | The amount of next day rain in mm. Used to create response variable RainTomorrow. A kind of measure of the "risk" | Binary |

# Structure Of Dataset

```
> str(df)
'data.frame':   145460 obs. of  23 variables:
 $ Date         : chr  "2008-12-01" "2008-12-02" "2008-12-03" "2008-12-04" ...
 $ Location     : chr  "Albury" "Albury" "Albury" "Albury" ...
 $ MinTemp      : num  13.4 7.4 12.9 9.2 17.5 14.6 14.3 7.7 9.7 13.1 ...
 $ MaxTemp      : num  22.9 25.1 25.7 28 32.3 29.7 25 26.7 31.9 30.1 ...
 $ Rainfall     : num  0.6 0 0 0 1 0.2 0 0 0 1.4 ...
 $ Evaporation  : num  NA NA NA NA NA NA NA NA NA NA ...
 $ Sunshine     : num  NA NA NA NA NA NA NA NA NA NA ...
 $ WindGustDir  : chr  "W" "WNW" "WSW" "NE" ...
 $ WindGustSpeed: int  44 44 46 24 41 56 50 35 80 28 ...
 $ WindDir9am   : chr  "W" "NNW" "W" "SE" ...
 $ WindDir3pm   : chr  "WNW" "WSW" "WSW" "E" ...
 $ WindSpeed9am : int  20 4 19 11 7 19 20 6 7 15 ...
 $ WindSpeed3pm : int  24 22 26 9 20 24 24 17 28 11 ...
 $ Humidity9am  : int  71 44 38 45 82 55 49 48 42 58 ...
 $ Humidity3pm  : int  22 25 30 16 33 23 19 19 9 27 ...
 $ Pressure9am  : num  1008 1011 1008 1018 1011 ...
 $ Pressure3pm  : num  1007 1008 1009 1013 1006 ...
 $ Cloud9am     : int  8 NA NA NA 7 NA 1 NA NA NA ...
 $ Cloud3pm     : int  NA NA 2 NA 8 NA NA NA NA NA ...
 $ Temp9am      : num  16.9 17.2 21 18.1 17.8 20.6 18.1 16.3 18.3 20.1 ...
 $ Temp3pm      : num  21.8 24.3 23.2 26.5 29.7 28.9 24.6 25.5 30.2 28.2 ...
 $ RainToday    : chr  "No" "No" "No" "No" ...
 $ RainTomorrow : chr  "No" "No" "No" "No" ...
```

## First we will check if we have biased column for (nominal attribute)

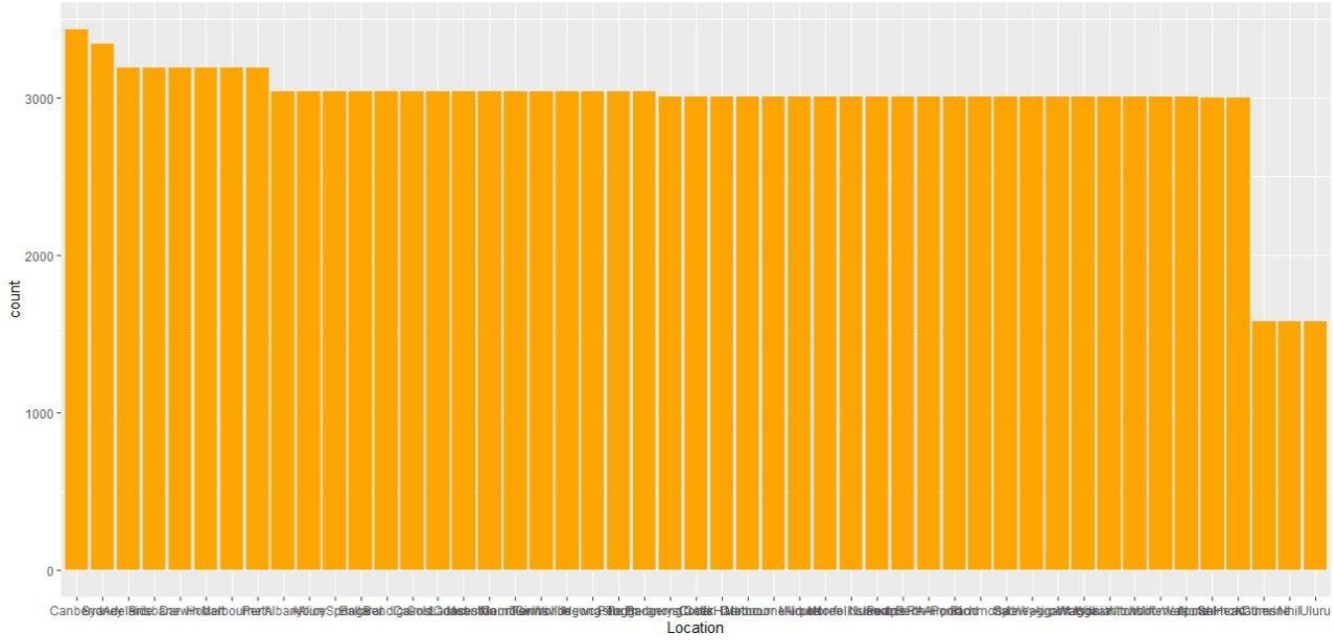
**Note: numeric attribute will checked in Outliers section**

**ggplot** function used to data visualization we need to install the package and use the library that contain **ggplot** function

```
> install.packages("ggplot2")
> library(ggplot2)
```

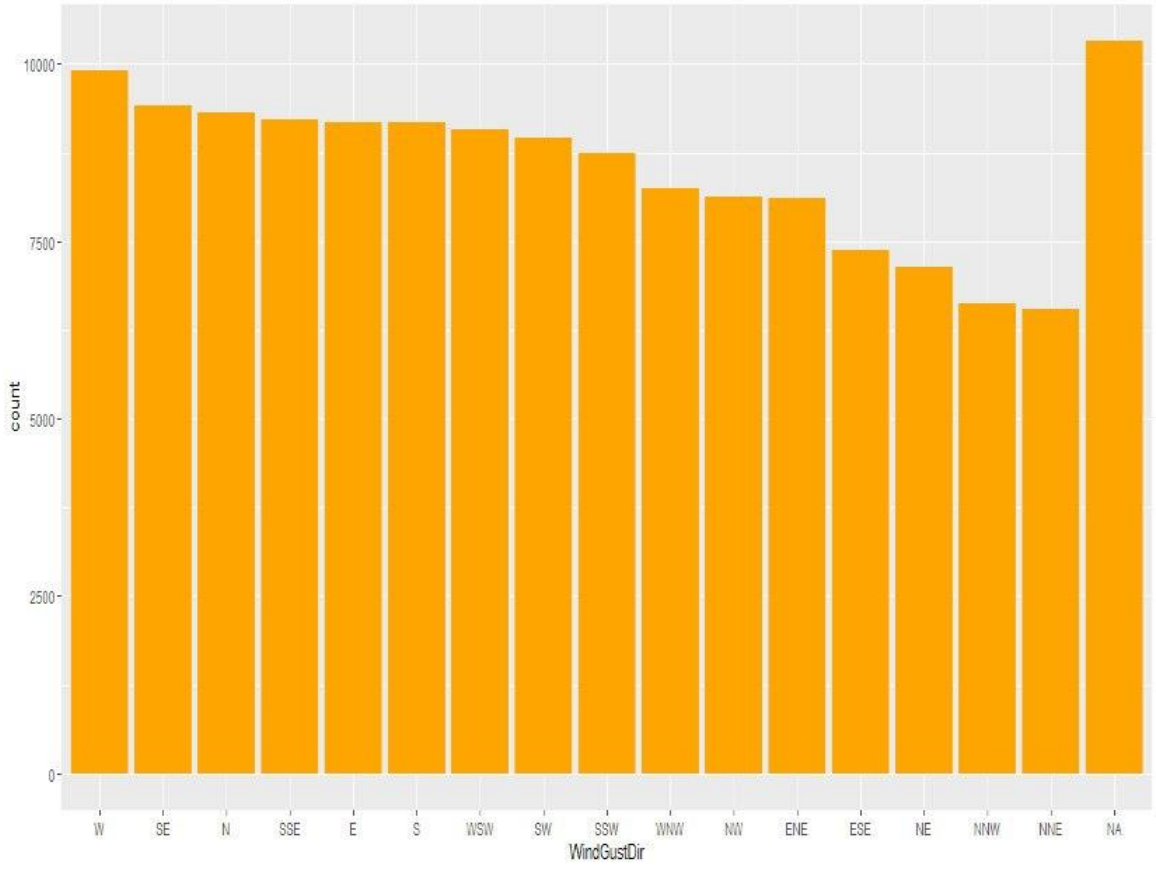
- **Location**

```
> ggplot(df, aes(x=reorder(Location, Location, function(x)-length(x)))) +
+   geom_bar(fill='orange') +  labs(x='Location')
```



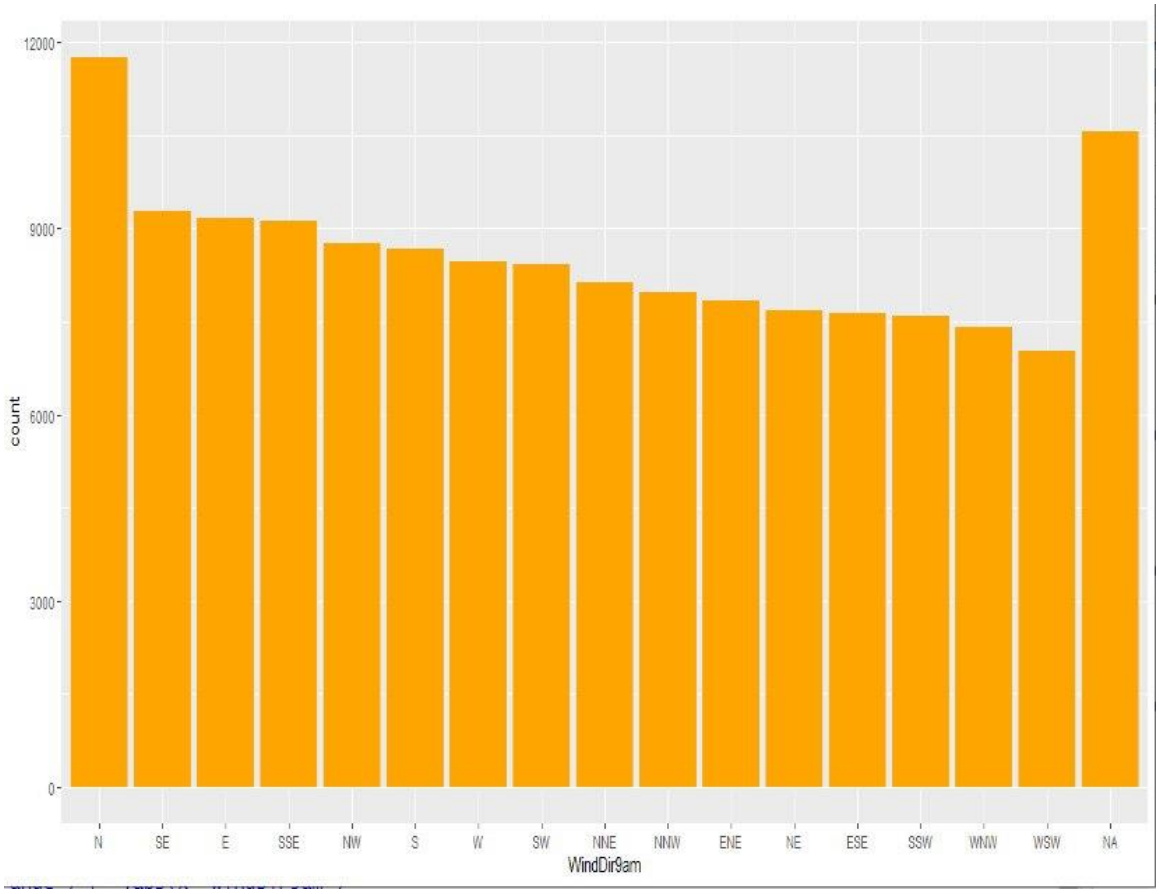
- **WindGustDir**

```
> ggplot(df, aes(x=reorder(WindGustDir, WindGustDir, function(x)-length(x)))) +
+   geom_bar(fill='orange') +  labs(x='WindGustDir')
```



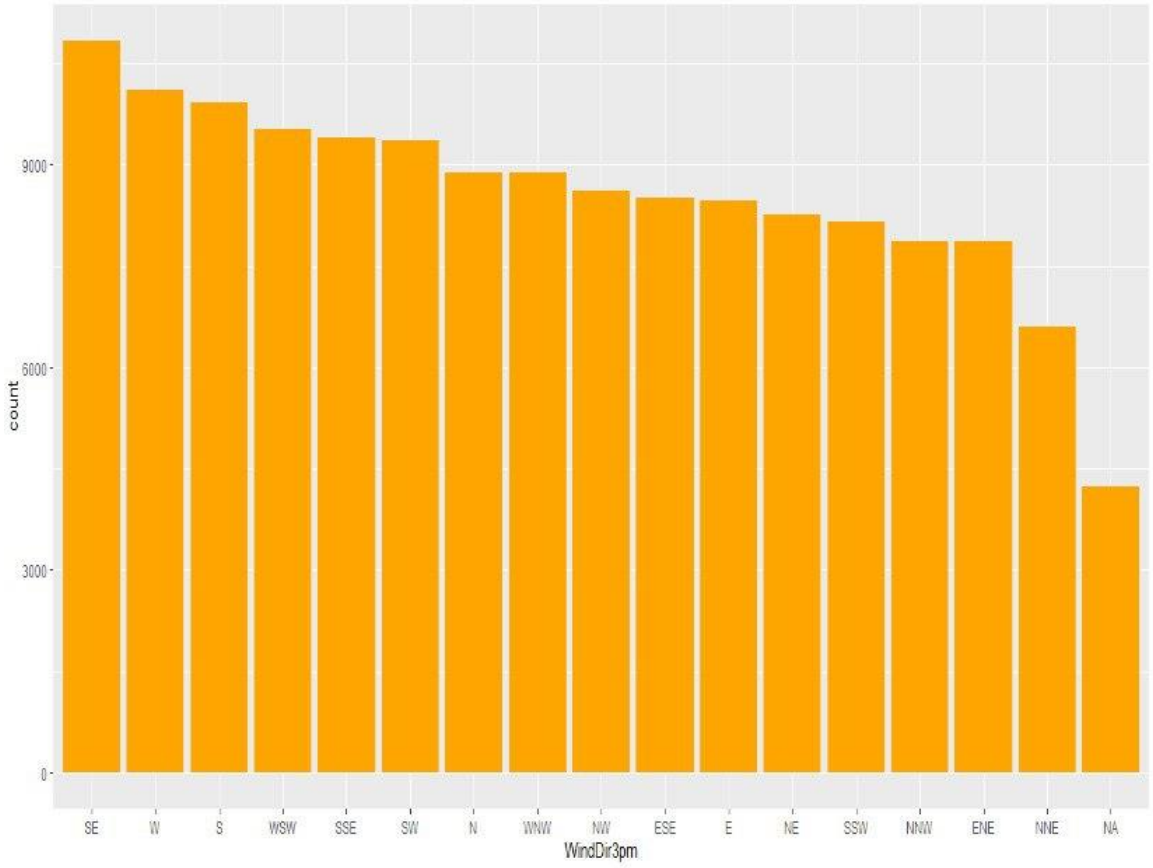
- **WindDir9am**

```
> ggplot(df, aes(x=reorder(WindDir9am, WindDir9am, function(x)-length(x)))) +
+   geom_bar(fill='orange') +  labs(x='WindDir9am')
```



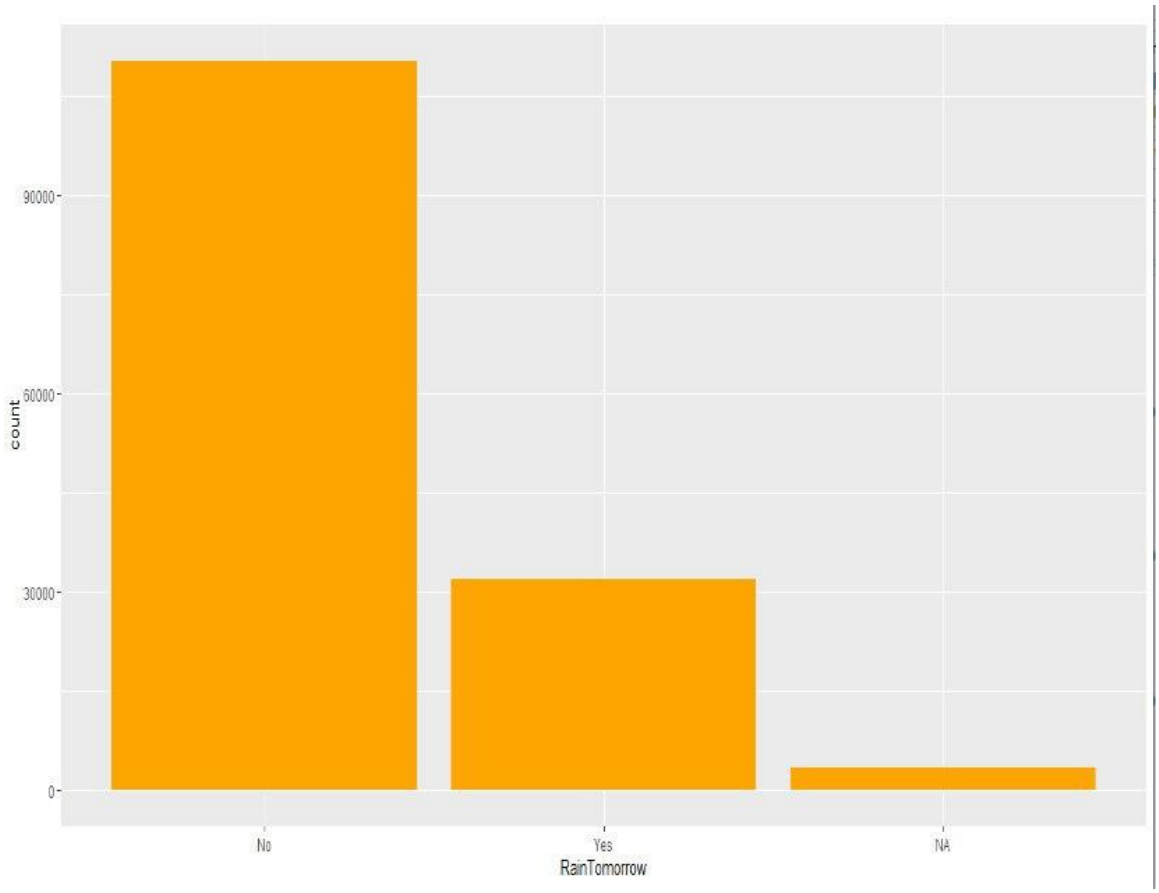
- **WindDir3pm**

```
> ggplot(df, aes(x=reorder(WindDir3pm, WindDir3pm, function(x)-length(x)))) +
+   geom_bar(fill='orange') +  labs(x='WindDir3pm')
>
```



- **RainTomorrow**

```
> ggplot(df, aes(x=reorder(RainTomorrow,RainTomorrow, function(x)-length(x)))) +
+   geom_bar(fill='orange') +  labs(x='RainTomorrow')
```




**From the above we see that all nominal attributes except RainTomorrow are not biased and well distributed**

**So we just have one biased attribute (RainTomorrow)**

## Data Preparing

### 1) Dealing with missing values

- **First we will calculate the number of records that have percentage of missing values greater than 25% .**

```
> sum(apply(X=is.na(df),MARGIN=1,FUN=mean)>.25)
[1] 17187
```

  - **From this result we notice that 17187 observations from the dataset is many, in the rate of 11.8% from records which have percentage of missing values greater than 25%, therefore we can't fill these records, because the data becomes unreliable, so we will delete these records.**

```
> (17187/145460)*100
[1] 11.81562
```

```
> df<-subset(df,!apply(X=is.na(df),MARGIN=1,FUN=mean)>.25)
```

The Number Of Observations And Attributes After Deletion

- Attributes: 23
- Observations:128273

```
> dim(df)
[1] 128273     23
```

- **Second we will calculate the percentage of missing values in columns(Attributes)**

There are 21 columns (attributes) with missing data within dataset which are:

1.Min Temp: 103
2.Max Temp: 77
3.Rainfall: 1034
4. Evaporation: 47070
5. Sunshine: 53699
6. WindGustDir: 4467
7. WindGustSpeed: 4444
11. WindDir3pm: 616
12. Humidity9am: 533
13. Humidity3pm: 1169
14. Pressure9am: 1030
15. Pressure3pm: 974
16. Cloud9am: 42583
17. Cloud3pm: 44679

8. WindDir9am: 6648
9. WindSpeed9am: 80
10.WindSpeed3pm:50
18. RainTomorrow: 1262
19. RainToday: 1034
20.Temp9am: 98
21. Temp3pm: 702

**Now We Will Explain Each Attribute Which Has Missing Values**

1. **Min Temp**

   The percentage of missing values in the Min Temp attribute is low and is approximately equal to 0.08% , so we will delete records which contain these missing values.

   Code for calculating the percentage of missing data in Min Temp attribute

```
> sum(is.na(df$MinTemp))
[1] 103
> 103/length(df$MinTemp)
[1] 0.0008029749
```

2. **Max Temp**

   The percentage of missing values in the Max Temp attribute is low and is approximately equal to 0.06% , so we will delete records which contain these missing values

   Code for calculating the percentage of missing data in Max Temp attribute

```
> sum(is.na(df$MaxTemp))
[1] 77
> 77/length(df$MaxTemp)
[1] 0.0006002822
```

### 3. Rainfall

The percentage of missing values in the Rainfall attribute is low and is approximately equal to 0.806% , so we will delete records which contain these missing values

Code for calculating the percentage of missing data in Rainfall attribute

```
> sum(is.na(df$Rainfall))
[1] 1034
> 1034/length(df$Rainfall)
[1] 0.008060933
```

### 4. WindGustDir

The percentage of missing values in the WindGustDir attribute is high and is approximately equal to 3.48%, So we don't want to delete these records containing them, we will fill these missing values in the records containing them using mode because it's nominal attribute.

Code for calculating the percentage of missing data in WindGustDir attribute

```
> sum(is.na(df$WindGustDir))
[1] 4467
> 4467/length(df$WindGustDir)
[1] 0.03482416
```

### 5. WindGustSpeed

The percentage of missing values in the WindGustSpeed attribute is high and is approximately equal to 3.46%, So we don't want to delete these records containing them, we will fill these missing values in the records containing them using median because it's numeric attribute.

Code for calculating the percentage of missing data in WindGustSpeed attribute

```
> sum(is.na(df$WindGustSpeed))
[1] 4444
> 4444/length(df$WindGustSpeed)
[1] 0.03464486
```

## 6. WindDir9am

The percentage of missing values in the WindDir9am attribute is high and is approximately equal to 5.18%, So we don't want to delete these records containing them, we will fill these missing values in the records containing them using mode because it's nominal attribute.

Code for calculating the percentage of missing data in WindDir9am attribute

```
> sum(is.na(df$WindDir9am))
[1] 6648
> 6648/length(df$WindDir9am)
[1] 0.05182696
```

## 7. WindDir3pm

The percentage of missing values in the WindDir3pm attribute is low and is approximately equal to 0.48% , so we will delete records which contain these missing values.

Code for calculating the percentage of missing data in WindDir3pm attribute

```
> sum(is.na(df$WindDir3pm))
[1] 616
> 616/length(df$WindDir3pm)
[1] 0.004802258
```

## 8. Humidity9am

The percentage of missing values in the Humidity9am attribute is low and is approximately equal to 0.415%,so we will delete records which contain these missing values.

Code for calculating the percentage of missing data in Humidity9am attribute

```
> sum(is.na(df$Humidity9am))
[1] 533
> 533/length(df$Humidity9am)
[1] 0.0041552
```

## 9. Humidity3pm

The percentage of missing values in the Humidity3pm attribute is low and is approximately equal to 0.911%,so we will delete records which contain these missing values.

Code for calculating the percentage of missing data in Humidity3pm attribute

```
> sum(is.na(df$Humidity3pm))
[1] 1169
> 1169/length(df$Humidity3pm)
[1] 0.009113375
```

## 10. Pressure9am

The percentage of missing values in the Pressure9am attribute is low and is approximately equal to 0.8%,so we will delete records which contain these missing values.

Code for calculating the percentage of missing data in Pressure9am

```
> sum(is.na(df$Pressure9am))
[1] 1030
> 1030/length(df$Pressure9am)
[1] 0.008029749
```

## 11. Pressure3pm

The percentage of missing values in the Pressure3pm attribute is low and is approximately equal to 0.759%,so we will delete records which contain these missing values.

Code for calculating the percentage of missing data in Pressure3pm

```
> sum(is.na(df$Pressure3pm))
[1] 974
> 974/length(df$Pressure3pm)
[1] 0.00759318
```

## 12. RainTomorrow

The percentage of missing values in the RainTomorrow attribute is low and is approximately equal to 0.98% , so we will delete records which contain these missing values.

Code for calculating the percentage of missing data in RainTomorrow attribute

```
> sum(is.na(df$RainTomorrow))
[1] 1262
> 1262/length(df$RainTomorrow)
[1] 0.009838392
```

## 13. WindSpeed9am

The percentage of missing values in the WindSpeed9am attribute is low and is approximately equal to 0.06% , so we will delete records which contain these missing values.

Code for calculating the percentage of missing data in WindSpeed9am attribute

```
> sum(is.na(df$WindSpeed9am))
[1] 80
> 80/length(df$WindSpeed9am)
[1] 0.0006236698
```

## 14. RainToday

The percentage of missing values in the RainToday attribute is low and is approximately equal to 0.8% , so we will delete records which contain these missing values.

Code for calculating the percentage of missing data in RainToday attribute

```
> sum(is.na(df$RainToday))
[1] 1034
> 1034/length(df$RainToday)
[1] 0.008060933
```

## 15. WindSpeed3pm

The percentage of missing values in the WindSpeed3pm attribute is low and is approximately equal to 0.0389% , so we will delete records which contain these missing values.

Code for calculating the percentage of missing data in WindSpeed3pm attribute

```
> sum(is.na(df$WindSpeed3pm))
[1] 50
> 50/length(df$WindSpeed3pm)
[1] 0.0003897936
```

## 16. Temp9am

The percentage of missing values in the Temp9am attribute is low and is approximately equal to 0.076% , so we will delete records which contain these missing values.

Code for calculating the percentage of missing data in Temp9am attribute

```
> sum(is.na(df$Temp9am))
[1] 98
> 98/length(df$Temp9am)
[1] 0.0007639955
```

## 17. Temp3pm

The percentage of missing values in the Temp3pm attribute is low and is approximately equal to 0.547% , so we will delete records which contain these missing values.

Code for calculating the percentage of missing data in Temp3pm attribute

```
> sum(is.na(df$Temp3pm))
[1] 702
> 702/length(df$Temp3pm)
[1] 0.005472703
```

**Now, These following attributes contain very high percentage of missing value, so we will delete these attributes from the dataset :**

- Evaporation: 36.69%
- Sunshine: 41.86%
- Cloud9am: 33.19%
- Cloud3pm: 34.83%

Code for calculating the percentage of missing data in these attributes

- Evaporation

```
> sum(is.na(df$Evaporation))
[1] 47070
> 47070/length(df$Evaporation)
[1] 0.3669517
```

- Sunshine

```
> sum(is.na(df$Sunshine))
[1] 53699
> 53699/length(df$Sunshine)
[1] 0.4186306
```

- Cloud9am

```
> sum(is.na(df$Cloud9am))
[1] 42583
> 42583/length(df$Cloud9am)
[1] 0.3319717
```

- Cloud3pm

```
> sum(is.na(df$Cloud3pm))
[1] 44679
> 44679/length(df$Cloud3pm)
[1] 0.3483118
```

**This following attribute doesn't contain any missing values:**

- Location.

```
> sum(is.na(df$Location))
[1] 0
```

**Now, we will delete/fill records which contain missing values and delete attributes that contain missing values as we explained previous for each attribute:**

1. **Min Temp**

**Delete records contain missing values**

```
> #delete the record that have missing values
> df<-subset(df,!is.na(df$MinTemp))
> sum(is.na(df$MinTemp))
[1] 0
```

2. **Max Temp**

**Delete records contain missing values**

```
> #delete the record that has missing values
> df<-subset(df,!is.na(df$MaxTemp))
> sum(is.na(df$MaxTemp))
[1] 0
```

3. **Rainfall**

**Delete records contain missing values**

```
> #delete the record that has missing values
> df<-subset(df,!is.na(df$Rainfall))
> sum(is.na(df$Rainfall))
[1] 0
```

## 4. WindGustDir

**Fill records contain missing values by mode**

**mode is W because is the most frequent value in the column**

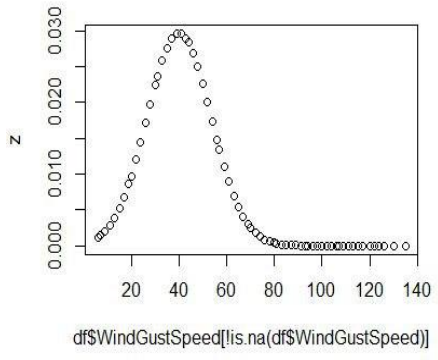
```
> #filling with mode
> table(df$WindGustDir)

   E  ENE  ESE    N   NE  NNE  NNW   NW    S   SE  SSE  SSW   SW    W  WNW
8503 7350 6797 8493 6586 5944 6062 7182 8251 8679 8308 7891 8135 8960 7346
 WSW
8199
> #mode is "W"
> df$WindGustDir[is.na(df$WindGustDir)]<-"W"
> sum(is.na(df$WindGustDir))
[1] 0
```

## 5. WindGustSpeed

**Fill records contain missing values by median**

```
> #filling with median
> z<-dnorm(df$WindGustSpeed[!is.na(df$WindGustSpeed)],mean(df$WindGustSpeed,na.rm = TRUE),sd(df$WindGustSpeed,na.rm = TRUE))
> plot(df$WindGustSpeed[!is.na(df$WindGustSpeed)],z)
```




**Because we have outliers we will fill these missing values in records using median not mean, because mean is affected by extreme values .**

```
> median1<-median(df$WindGustSpeed,na.rm=TRUE)
> median1
[1] 39
> df$WindGustSpeed[is.na(df$WindGustSpeed)]<-median1
> sum(is.na(df$WindGustSpeed))
[1] 0
```

6. **WindDir9am**

**Fill records contain missing values by mode**

**mode is N because is the most frequent value in the column**

```
> #filling with mode
> table(df$WindDir9am)

    E   ENE   ESE     N    NE   NNE   NNW    NW     S    SE   SSE   SSW    SW     W   WNW   WSW
 8497  7219  7082 10406  6717  7328  7245  7274  7876  8292  8496  6584  7322  7408  6483  6212
> df$WindDir9am[is.na(df$WindDir9am)]<-"N"
> sum(is.na(df$WindDir9am))
[1] 0
```

7. **WindDir3pm**

**Delete records contain missing values**

```
> #delete the record that has missing values
> df<-subset(df,!is.na(df$WindDir3pm))
> sum(is.na(df$WindDir3pm))
[1] 0
```

8. **Humidity9am**

**Delete records contain missing values**

```
> #delete the record that has missing values
> df<-subset(df,!is.na(df$Humidity9am))
> sum(is.na(df$Humidity9am))
[1] 0
```

9. **Humidity3pm**

**Delete records contain missing values**

```
> #delete the record that has missing values
> df<-subset(df,!is.na(df$Humidity3pm))
> sum(is.na(df$Humidity3pm))
[1] 0
```

**10. Pressure9am**

**Delete records contain missing values**

```
> #delete the record that have missing values
> df<-subset(df,!is.na(df$Pressure9am))
> sum(is.na(df$Pressure9am))
[1] 0
```

**11. Pressure3pm**

**Delete records contain missing values**

```
> #delete the record that have missing values
> df<-subset(df,!is.na(df$Pressure3pm))
> sum(is.na(df$Pressure3pm))
[1] 0
```

**12. RainTomorrow**

**Delete records contain missing values**

```
> #delete the record that has missing values
> df<-subset(df,!is.na(df$RainTomorrow))
> sum(is.na(df$RainTomorrow))
[1] 0
```

**13. WindSpeed9am**

**Delete records contain missing values**

```
> #delete the record that has missing values
> df<-subset(df,!is.na(df$WindSpeed9am))
> sum(is.na(df$WindSpeed9am))
[1] 0
```

**14. RainToday**

**Delete records contain missing values**

```
> #delete the record that has missing values
> df<-subset(df,!is.na(df$RainToday))
> sum(is.na(df$RainToday))
[1] 0
```

**15. WindSpeed3pm**

**Delete records contain missing values**

```
> #delete the record that has missing values
> df<-subset(df,!is.na(df$WindSpeed3pm))
> sum(is.na(df$WindSpeed3pm))
[1] 0
```

**16. Temp9am**

**Delete records contain missing values**

```
> #delete the record that has missing values
> df<-subset(df,!is.na(df$Temp9am))
> sum(is.na(df$Temp9am))
[1] 0
```

**17. Temp3pm**

**Delete records contain missing values**

```
> #delete the record that has missing values
> df<-subset(df,!is.na(df$Temp3pm))
> sum(is.na(df$Temp3pm))
[1] 0
```

**18. Evaporation, Sunshine, Cloud9am, Cloud3pm**

**Delete columns (attributes) contain missing values**

```
> df<-subset(df,select=-c(Cloud9am,Cloud3pm,Evaporation,Sunshine))
```

**Number of attributes & observations after delete records (rows) and attributes (columns) containing missing values from the dataset:**

```
> dim(df)
[1] 123151     19
```

- Number of attributes : 19
- Number of observations :123151

- **Now, our data set is cleaned from missing values.**

```
> sum(is.na(df))
[1] 0
```

## 2) Dealing with Outliers

- Now we want to see if we have outliers in numeric attributes or not.
- There are 12 numeric attributes and they are :

1. Min Temp.
2. Max Temp.
3. Rainfall.
4. WindGustSpeed.
5. WindSpeed9am.
6. WindSpeed3pm.
7. Humidity9am.
8. Humidity3pm.
9. Pressure9am .
10. Pressure3pm .
11. Temp9am.
12. Temp3pm.

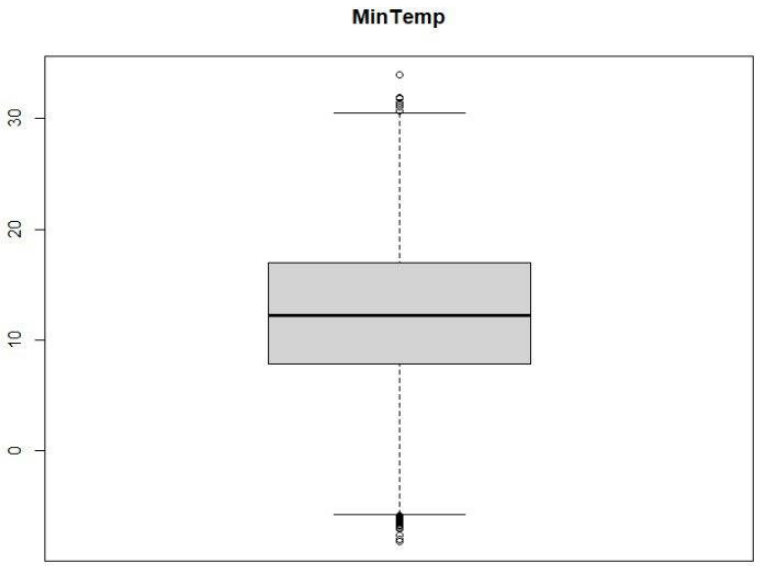
- Then we want to draw the Boxplot of each of these attributes to get know if we have outliers in these columns (attributes) or not.

  **Now, We Will Explain Each Numeric Attribute with Boxplot for it**

### 1. Min Temp

- First draw the Boxplot for the attribute and find if it has outliers or not.

```
> boxplot(df$MinTemp,main="MinTemp")
```




From this Boxplot we notice that there is outliers in the column of Min Temp.

- We notice that the data in Min Temp is **Relatively Symmetric attribute**.

- Now we will calculate the percentage of these outliers.
  quart: it is an array which include two values q1 &q3 :
  q1: there exist 25% from the data under it.
  q3: there exist 75% from the data under it.

```
> quart<-quantile(df$MinTemp,probs=c(.25,.75))
```

- iqr:it is equal to (q3-q1).

```
> iqr<-IQR(df$MinTemp)
```

- The outliers mean that these values is:

  greater than upperBound OR smaller than lowerBound

  - upperBound is equal to : q3 + [1.5*IQR] // q3=quart[2]
  - loweBound is equal to : q1 - [1.5*IQR] // q1=quart[1]

```
> upperBound<-quart[2]+(1.5*iqr)
> lowerBound<-quart[1]-(1.5*iqr)
```

- The number of outliers in the column of Min Temp is:

```
> length(df$MinTemp[df$MinTemp>upperBound|df$MinTemp<lowerBound])
[1] 41
```

- The percentage of outliers in the column of Min Temp is:

```
> (41/ length(df$MinTemp))*100
[1] 0.03329246
```

**Because the percentage is equal to 0.033%, it's very small, so we will delete these records that contain outliers.**

- Then we delete.

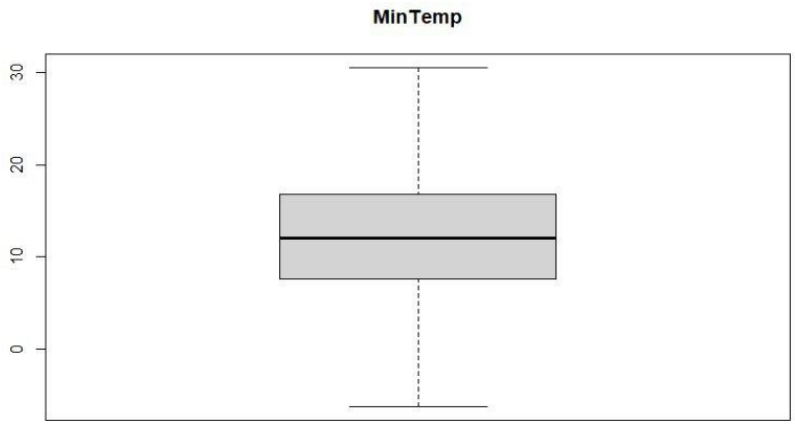
```
> df<-subset(df,df$MinTemp<=upperBound&df$MinTemp>=lowerBound)
```

- Now number of outliers is equal to 0.

```
> length(df$MinTemp[df$MinTemp>upperBound|df$MinTemp<lowerBound])
[1] 0
```

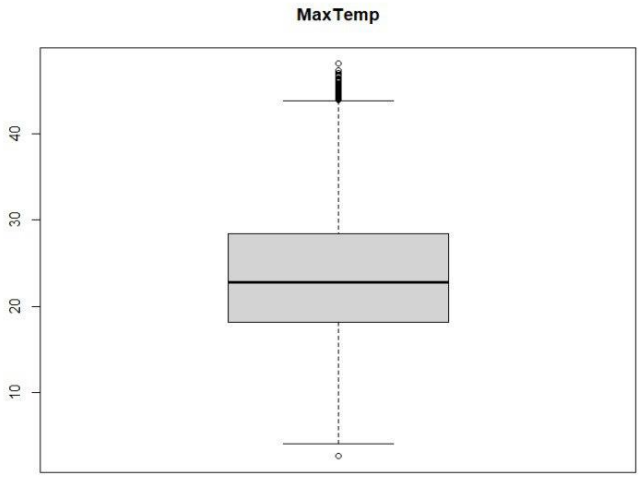
- Then draw the Boxplot of the attribute, and find that it's free of outliers.

```
> boxplot(df$MinTemp,main="MinTemp")
```



## 2. Max Temp

```
> boxplot(df$MaxTemp,main="MaxTemp")
```

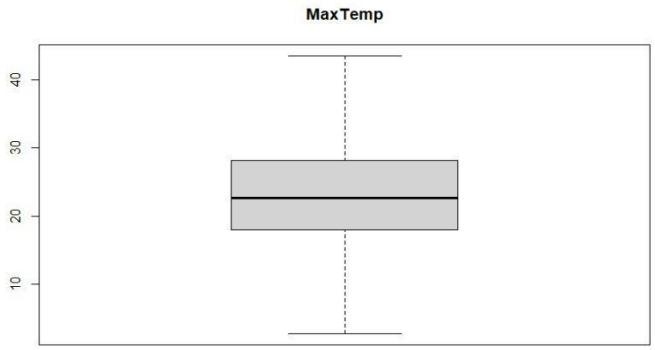



- **From Boxplot, we notice that there is outliers, so we calculate percentage of them.**
  - We notice that the **Max Temp is Relatively Symmetric attribute**

```
> quart<-quantile(df$MaxTemp,probs=c(.25,.75))
> iqr<-IQR(df$MaxTemp)
> upperBound<-quart[2]+(1.5*iqr)
> lowerBound<-quart[1]-(1.5*iqr)
> length(df$ MaxTemp[df$MaxTemp>upperBound|df$MaxTemp<lowerBound])
[1] 123
> (123/ length(df$MaxTemp))*100
[1] 0.09991065
> df<-subset(df,df$MaxTemp<=upperBound&df$MaxTemp>=lowerBound)
> length(df$MinTemp[df$MaxTemp>upperBound|df$MaxTemp<lowerBound])
[1] 0
```

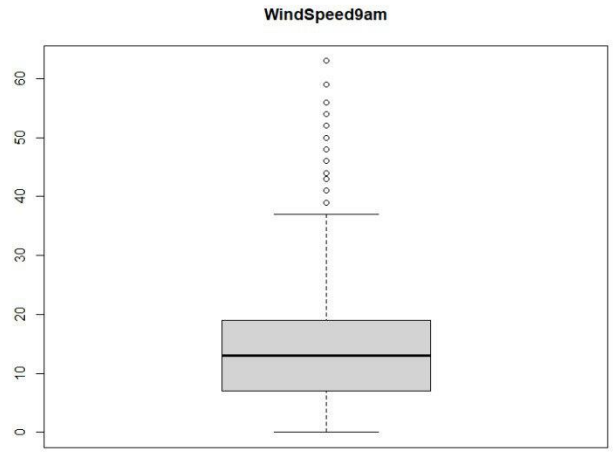
**The percentage of outliers in Max Temp equal to 0.099%, it's very small, so we delete records which contain these outliers , then we get Boxplot of the attribute without outliers**.

```
> boxplot(df$MaxTemp,main="MaxTemp")
```



### 3. WindSpeed9am

```
> boxplot(df$WindSpeed9am,main="WindSpeed9am")
```

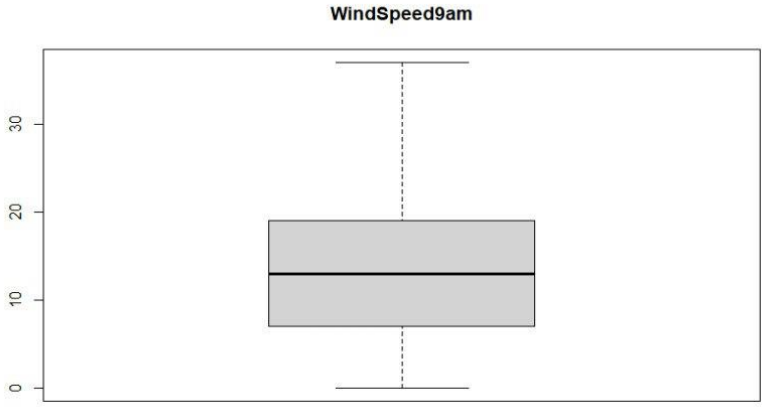



- **From Boxplot, we notice that there is outliers, so we calculate percentage of them.**
  - We notice that the **WindSpeed9am is Symmetric attribute.**

```
> quart<-quantile(df$WindSpeed9am,probs=c(.25,.75))
> iqr<-IQR(df$WindSpeed9am)
> upperBound<-quart[2]+(1.5*iqr)
> lowerBound<-quart[1]-(1.5*iqr)
> length(df$WindSpeed9am[df$WindSpeed9am>upperBound|df$WindSpeed9am<lowerBound])
[1] 981
> (981/ length(df$WindSpeed9am))*100
[1] 0.8150684
> df<-subset(df,df$WindSpeed9am<=upperBound&df$WindSpeed9am>=lowerBound)
> length(df$WindSpeed9am[df$WindSpeed9am>upperBound|df$WindSpeed9am<lowerBound])
[1] 0
```

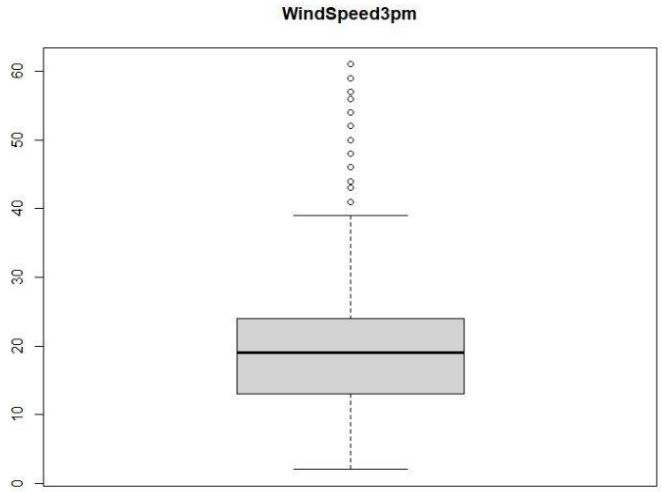
**The percentage of outliers in WindSpeed9am equal to 0.815%, it's very small, so we want to delete records which contain these outliers , then we get Boxplot of the attribute without outliers.**

```
> boxplot(df$WindSpeed9am,main="WindSpeed9am")
```



## 4. WindSpeed3pm

```
> boxplot(df$WindSpeed3pm,main="WindSpeed3pm")
```

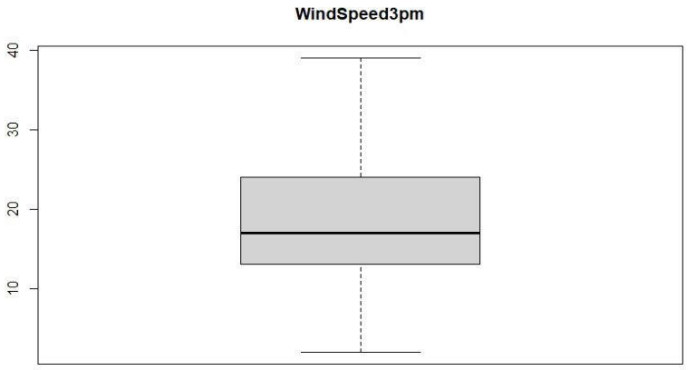



- **From Boxplot, we notice that there is outliers, so we calculate percentage of them.**
  - We notice that the**WindSpeed3pm is Relatively Symmetric attribute.**

```
> quart<-quantile(df$WindSpeed3pm,probs=c(.25,.75))
> iqr<-IQR(df$WindSpeed3pm)
> upperBound<-quart[2]+(1.5*iqr)
> lowerBound<-quart[1]-(1.5*iqr)
> length(df$WindSpeed3pm[df$WindSpeed3pm>upperBound|df$WindSpeed3pm<lowerBound])
[1] 1187
> (1187/ length(df$WindSpeed3pm))*100
[1] 0.9943289
> df<-subset(df,df$WindSpeed3pm<=upperBound&df$WindSpeed3pm>=lowerBound)
> length(df$WindSpeed3pm[df$WindSpeed3pm>upperBound|df$WindSpeed3pm<lowerBound])
[1] 0
```

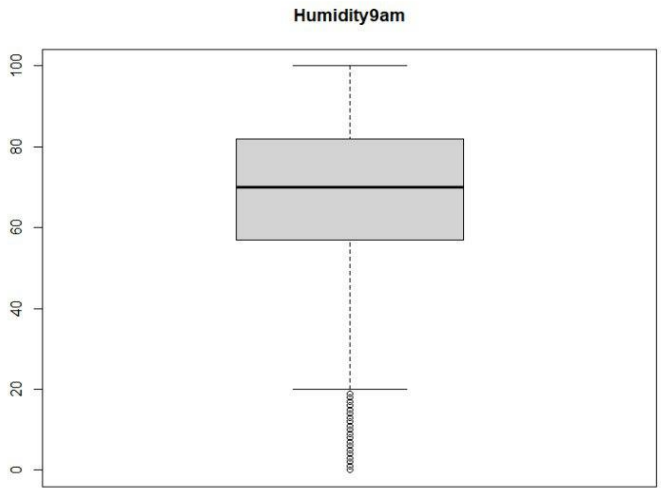
**The percentage of outliers in WindSpeed3pm equal to 0.994%, it's very small, so we want to delete records which contain these outliers , then we get Boxplot of the attribute without outliers**.

```
> boxplot(df$WindSpeed3pm,main="WindSpeed3pm")
```



## 5. Humidity9am

```
> boxplot(df$Humidity9am,main="Humidity9am")
```

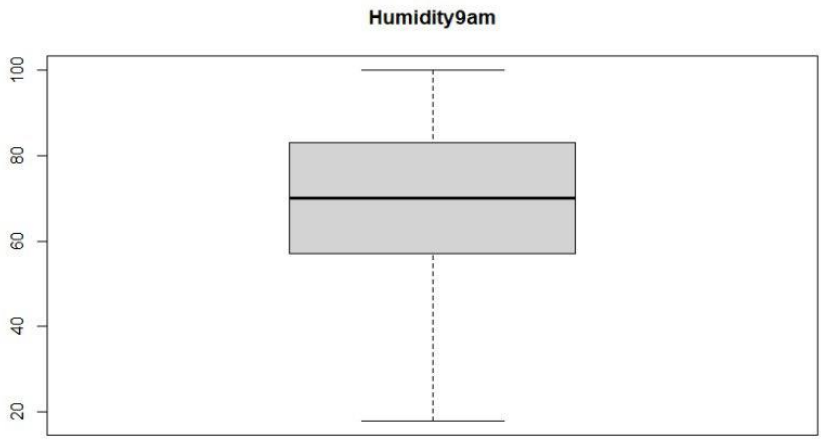



- **From Boxplot, we notice that there is outliers, so we calculate percentage of them.**
  - We notice that the **Humidity9am is Relatively Symmetric attribute.**

```
> quart<-quantile(df$Humidity9am,probs=c(.25,.75))
> iqr<-IQR(df$Humidity9am)
> upperBound<-quart[2]+(1.5*iqr)
> lowerBound<-quart[1]-(1.5*iqr)
> length(df$Humidity9am[df$Humidity9am>upperBound|df$Humidity9am<lowerBound])
[1] 1555
> (1555/ length(df$Humidity9am))*100
[1] 1.315678
> df<-subset(df,df$Humidity9am<=upperBound&df$Humidity9am>=lowerBound)
> length(df$Humidity9am[df$Humidity9am>upperBound|df$Humidity9am<lowerBound])
[1] 0
```

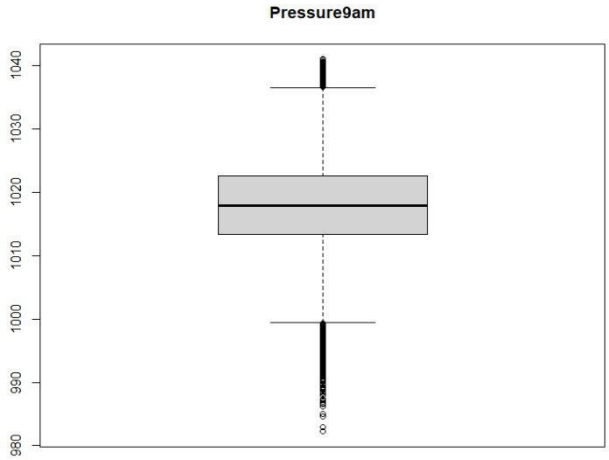
**The percentage of outliers in Humidity9am equal to 1.315%, it's very small, so we want to delete records which contain these outliers , then we get Boxplot of the attribute without outliers**

```
> boxplot(df$Humidity9am,main="Humidity9am")
```



## 6. Pressure9am

```
> boxplot(df$Pressure9am,main="Pressure9am")
```

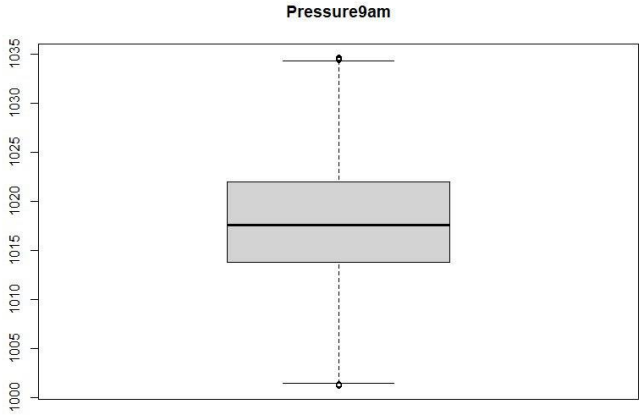



- **From Boxplot, we notice that there is outliers, so we calculate percentage of them.**
  - We notice that the **Pressure9am is Symmetric attribute.**

```
> quart<-quantile(df$Pressure9am,probs=c(.25,.75))
> iqr<-IQR(df$Pressure9am)
> upperBound<-quart[2]+(1.5*iqr)
> lowerBound<-quart[1]-(1.5*iqr)
> length(df$Pressure9am[df$Pressure9am>upperBound|df$Pressure9am<lowerBound])
[1] 947
> (947/ length(df$Pressure9am))*100
[1] 0.8119347
> df<-subset(df,df$Pressure9am<=upperBound&df$Pressure9am>=lowerBound)
> length(df$Pressure9am[df$Pressure9am>upperBound|df$Pressure9am<lowerBound])
[1] 0
```

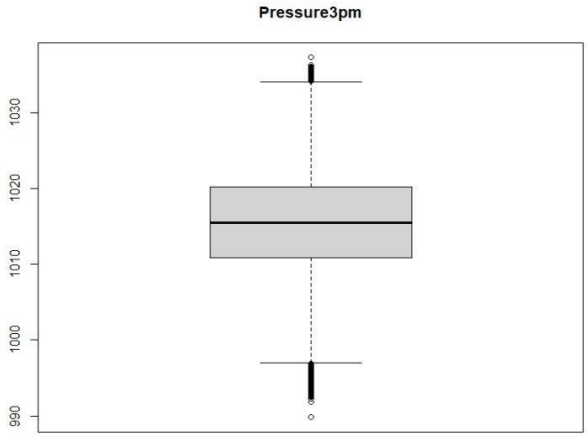
**The percentage of outliers in Pressure9am equal to 0.812%, it's very small, so we want to delete records which contain these outliers , then we get Boxplot of the attribute without outliers**

```
> boxplot(df$Pressure9am,main="Pressure9am")
```



## 7. Pressure3pm

```
> boxplot(df$Pressure3pm,main="Pressure3pm")
```

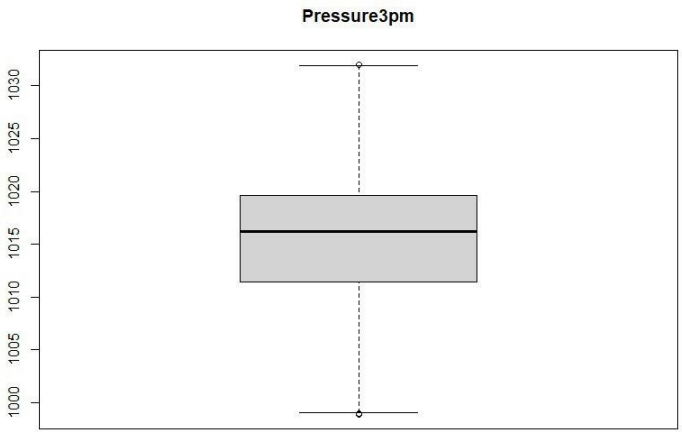



- **From Boxplot, we notice that there is outliers, so we calculate percentage of them.**
  - We notice that the **Pressure3pm is Symmetric attribute.**

```
> quart<-quantile(df$Pressure3pm,probs=c(.25,.75))
> iqr<-IQR(df$Pressure3pm)
> upperBound<-quart[2]+(1.5*iqr)
> lowerBound<-quart[1]-(1.5*iqr)
> length(df$Pressure3pm[df$Pressure3pm>upperBound|df$Pressure3pm<lowerBound])
[1] 243
> (243/ length(df$Pressure3pm))*100
[1] 0.2100477
> df<-subset(df,df$Pressure3pm<=upperBound&df$Pressure3pm>=lowerBound)
> length(df$Pressure3pm[df$Pressure3pm>upperBound|df$Pressure3pm<lowerBound])
[1] 0
```

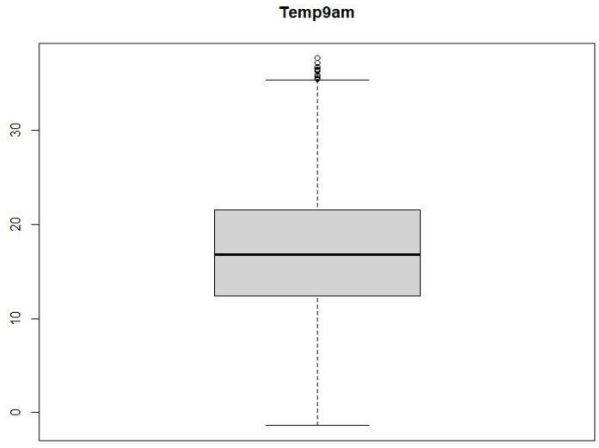
**The percentage of outliers in Pressure3pm equal to 0.21%, it's very small, so we want to delete records which contain these outliers , then we get Boxplot of the attribute without outliers**

```
> boxplot(df$Pressure3pm,main="Pressure3pm")
```



## 8. Temp9am

```
> boxplot(df$Temp9am,main="Temp9am")
```

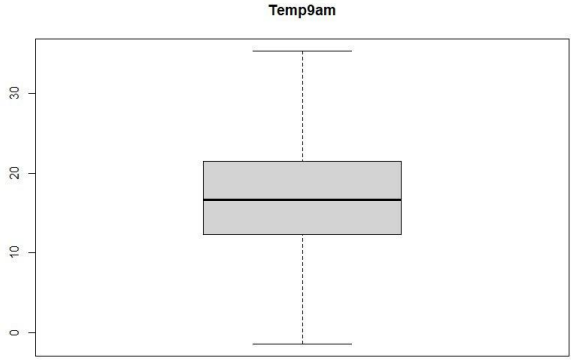



- **From Boxplot, we notice that there is outliers, so we calculate percentage of them.**
  - We notice that the **Temp9am is Symmetric attribute**

```
> quart<-quantile(df$Temp9am,probs=c(.25,.75))
> iqr<-IQR(df$Temp9am)
> upperBound<-quart[2]+(1.5*iqr)
> lowerBound<-quart[1]-(1.5*iqr)
> length(df$Temp9am[df$Temp9am>upperBound|df$Temp9am<lowerBound])
[1] 15
> (15/ length(df$Temp9am))*100
[1] 0.0129932
> df<-subset(df,df$Temp9am<=upperBound&df$Temp9am>=lowerBound)
> length(df$Temp9am[df$Temp9am>upperBound|df$Temp9am<lowerBound])
[1] 0
```

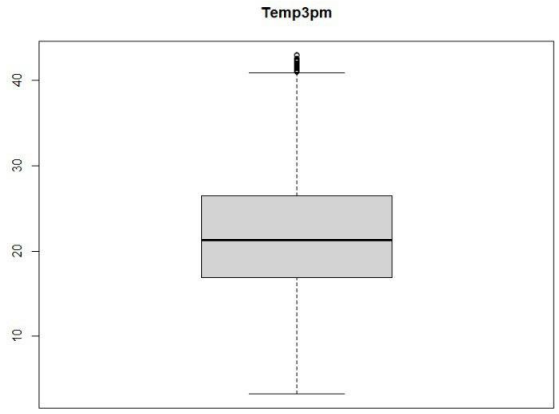
**The percentage of outliers in Temp9am equal to 0.0129%, it's very small, so we want to delete records which contain these outliers , then we get Boxplot of the attribute without outliers**

```
> boxplot(df$Temp9am,main="Temp9am")
```



## 9. Temp3pm

```
> boxplot(df$Temp3pm,main="Temp3pm")
```

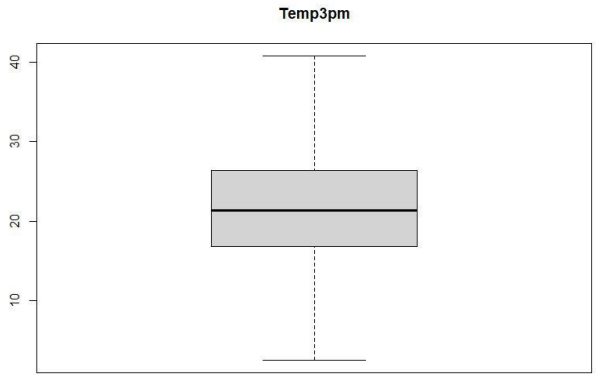



- **From Boxplot, we notice that there is outliers, so we calculate percentage of them.**
  - We notice that the **Temp3pam is Relatively Symmetric attribute**

```
> quart<-quantile(df$Temp3pm,probs=c(.25,.75))
> iqr<-IQR(df$Temp3pm)
> upperBound<-quart[2]+(1.5*iqr)
> lowerBound<-quart[1]-(1.5*iqr)
> length(df$Temp3pm[df$Temp3pm>upperBound|df$Temp3pm<lowerBound])
[1] 150
> (150/ length(df$Temp3pm))*100
[1] 0.1299489
> df<-subset(df,df$Temp3pm<=upperBound&df$Temp3pm>=lowerBound)
> length(df$Temp3pm[df$Temp3pm>upperBound|df$Temp3pm<lowerBound])
[1] 0
```

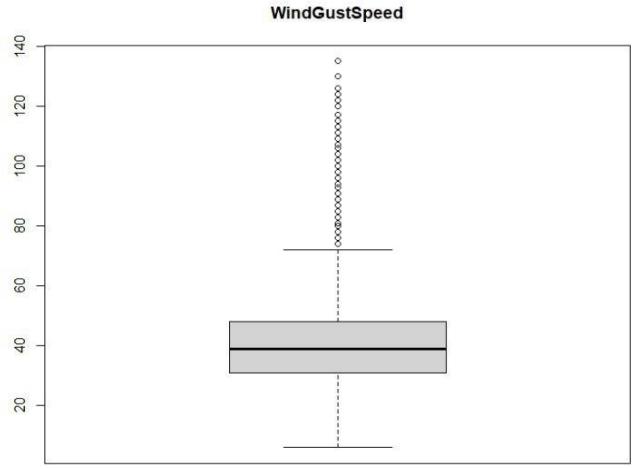
**The percentage of outliers in Temp3pam equal to 0.129%, it's very small, so we want to delete records which contain these outliers , then we get Boxplot of the attribute without outliers**

```
> boxplot(df$Temp3pm,main="Temp3pm")
```



## 10. WindGustSpeed

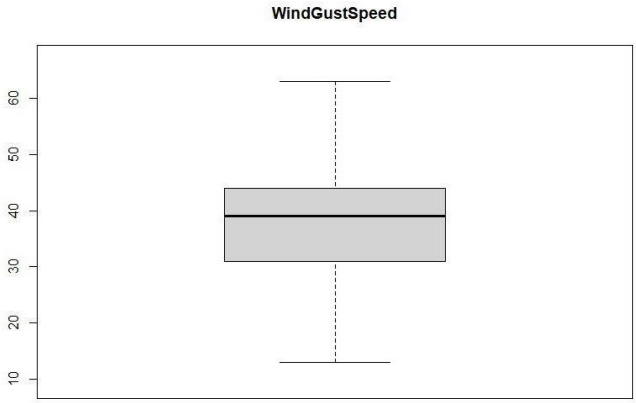
```
> boxplot(df$WindGustSpeed,main="WindGustSpeed")
```




- **From Boxplot, we notice that there is outliers, so we calculate percentage of them.**
  - We notice that the **WindGustSpeed is Relatively Symmetric attribute.**

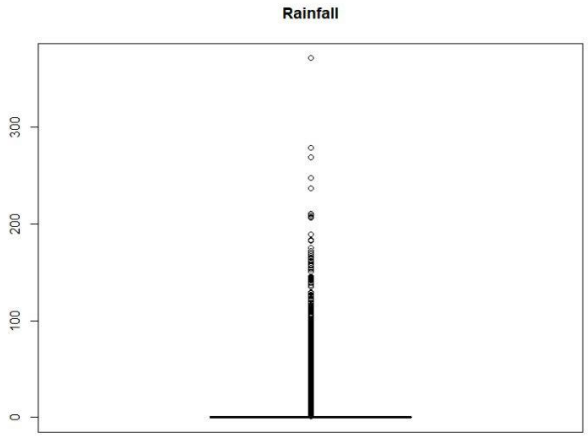
```
> quart<-quantile(df$WindGustSpeed,probs=c(.25,.75))
> iqr<-IQR(df$WindGustSpeed)
> upperBound<-quart[2]+(1.5*iqr)
> lowerBound<-quart[1]-(1.5*iqr)
> length(df$WindGustSpeed[df$WindGustSpeed>upperBound|df$WindGustSpeed<lowerBound])
[1] 2629
> (2629/ length(df$WindGustSpeed))*100
[1] 2.137624
> df<-subset(df,df$WindGustSpeed<=upperBound&df$WindGustSpeed>=lowerBound)
> length(df$WindGustSpeed[df$WindGustSpeed>upperBound|df$WindGustSpeed<lowerBound])
[1] 0
```

**The percentage of outliers in WindGustSpeed equal to 2.137%, it's very small, so we want to delete records which contain these outliers , then we get Boxplot of the attribute without outliers**



## 11.Rainfall

```
> boxplot(df$Rainfall,main="Rainfall")
```




```
> quart<-quantile(df$Rainfall,probs=c(.25,.75))
> iqr<-IQR(df$Rainfall)
> upperBound<-quart[2]+(1.5*iqr)
> lowerBound<-quart[1]-(1.5*iqr)
> length(df$Rainfall[df$Rainfall>upperBound|df$Rainfall<lowerBound])
[1] 22710
> (22710/ length(df$Rainfall))*100
[1] 19.69986
```

**The percentage of outliers in Rainfall equal to 19.699%, this percentage considered high, so we will smooth these outliers instead of delete it .**

## Using binning method to smooth the outliers in Rainfall attribute:

**ntile()** function to break up a vector first we need to install the package and use the library that contain ntile() function

```
> install.packages("dplyr")
```

install the package

```
> library(dplyr)
```

include the library

First divide data into equal frequency bins (number of bins=10):

```
> x<-ntile(df$Rainfall,n=10)
```

ntile() Function to break up a vector int bins

```
> head(x)
[1] 8 1 1 1 8 7
```

For example: X[1] represent the bin for the first value in Rainfall column

**Smoothing by mean:**

```
> mean<-replicate(10,0)
> for(i in 1 : 10){
+   mean[i]<-mean(df$Rainfall[x==i])
+ }
> mean
 [1]  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000  0.1053522  0.7086138  3.0810201
[10] 18.0699948
```

Mean vector to find the mean values for each bin

For example: Mean [1] represent the mean for values in bin number 1

Rainfall column before smoothing

```
> head(df)
        Date Location MinTemp MaxTemp Rainfall WindGustDir WindGustSpeed WindDir9am WindDir3pm
1 2008-12-01   Albury    13.4    22.9      0.6          W            44          W        WNW
2 2008-12-02   Albury     7.4    25.1      0.0        WNW            44        NNW        WSW
3 2008-12-03   Albury    12.9    25.7      0.0        WSW            46          W        WSW
4 2008-12-04   Albury     9.2    28.0      0.0         NE            24         SE          E
5 2008-12-05   Albury    17.5    32.3      1.0          W            41        ENE         NW
6 2008-12-06   Albury    14.6    29.7      0.2        WNW            56          W          W
  WindSpeed9am WindSpeed3pm Humidity9am Humidity3pm Pressure9am Pressure3pm Temp9am Temp3pm RainToday
1           20           24          71          22      1007.7      1007.1    16.9    21.8        No
2            4           22          44          25      1010.6      1007.8    17.2    24.3        No
3           19           26          38          30      1007.6      1008.7    21.0    23.2        No
4           11            9          45          16      1017.6      1012.8    18.1    26.5        No
5            7           20          82          33      1010.8      1006.0    17.8    29.7        No
6           19           24          55          23      1009.2      1005.4    20.6    28.9        No
  RainTomorrow
1           No
2           No
3           No
4           No
5           No
6           No
```

Now replace each value in rainfall column based on bin value

For example make all values in bin number one equal to the mean of bin number one

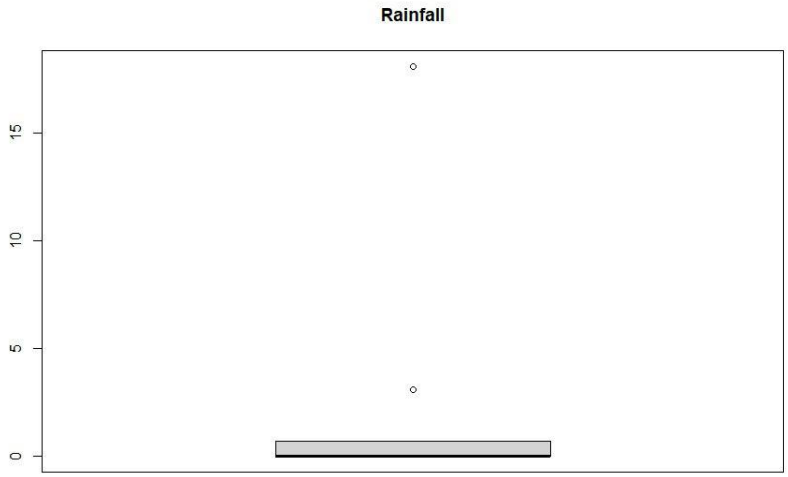
## Rainfall column after smoothing

```
> for(i in 1 : 10){
+  df$Rainfall[x==i]=mean[i]
+ }
> head(df)
        Date Location MinTemp MaxTemp  Rainfall WindGustDir WindGustSpeed WindDir9am WindDir3pm
1 2008-12-01   Albury    13.4    22.9 0.7086138          W            44          W        WNW
2 2008-12-02   Albury     7.4    25.1 0.0000000        WNW            44        NNW        WSW
3 2008-12-03   Albury    12.9    25.7 0.0000000        WSW            46          W        WSW
4 2008-12-04   Albury     9.2    28.0 0.0000000         NE            24         SE          E
5 2008-12-05   Albury    17.5    32.3 0.7086138          W            41        ENE         NW
6 2008-12-06   Albury    14.6    29.7 0.1053522        WNW            56          W          W
  WindSpeed9am WindSpeed3pm Humidity9am Humidity3pm Pressure9am Pressure3pm Temp9am Temp3pm RainToday
1           20           24          71          22      1007.7      1007.1    16.9    21.8        No
2            4           22          44          25      1010.6      1007.8    17.2    24.3        No
3           19           26          38          30      1007.6      1008.7    21.0    23.2        No
4           11            9          45          16      1017.6      1012.8    18.1    26.5        No
5            7           20          82          33      1010.8      1006.0    17.8    29.7        No
6           19           24          55          23      1009.2      1005.4    20.6    28.9        No
  RainTomorrow
1           No
2           No
3           No
4           No
5           No
6           No
```

Now after smoothing outliers using binning method , we get this boxplot, we still have outliers but in this way we reduce the variance for this data and mean for data does not change.

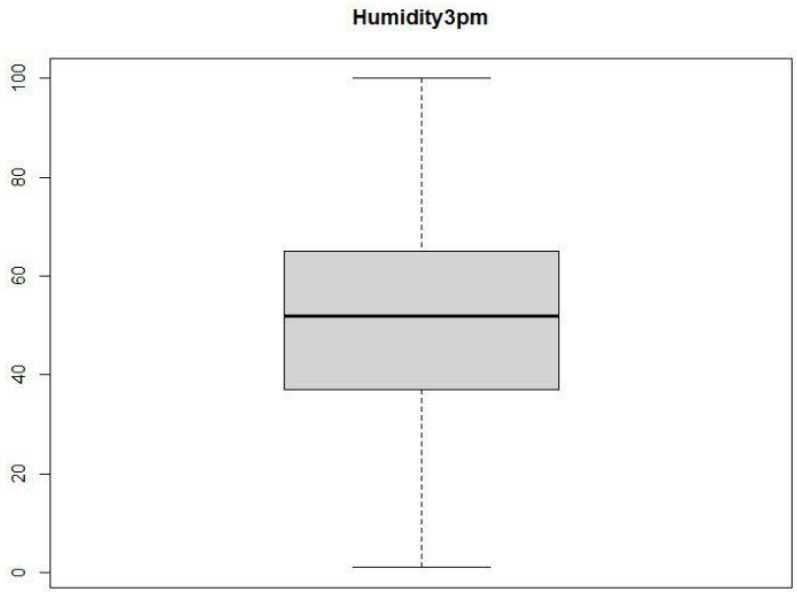
binning method tends to improve the accuracy of the model

```
> boxplot(df$Rainfall,main="Rainfall")
```



## 12.Humidity3pm

```
> boxplot(df$Humidity3pm,main="Humidity3pm")
```




- **From Boxplot, we notice that there isn't any outlier in the column of Humidity3pm, so it's free from the outliers.**

- **Now, our data set is cleaned from Outliers.**

**Normalization (Transformation)**

we transform all numeric column in dataset and make it in the same range from zero to one.

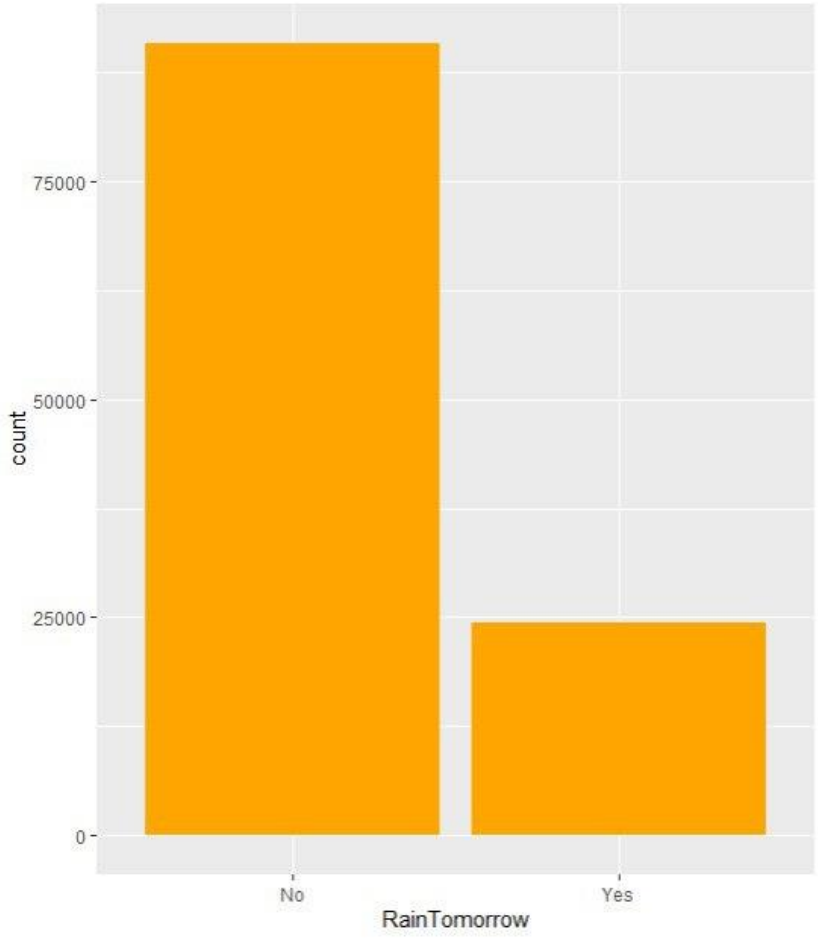
```
> #normalization
> normalize <- function(x,newMin,newMax) {
+   return ((x-min(x))/(max(x)-min(x)) * (newMax-newMin) +newMin )}
> df$MinTemp<-normalize(df$MinTemp,0,1)
> df$MaxTemp<-normalize(df$MaxTemp,0,1)
> df$Rainfall<-normalize(df$Rainfall,0,1)
> df$WindGustSpeed<-normalize(df$WindGustSpeed,0,1)
> df$WindSpeed9am<-normalize(df$WindSpeed9am,0,1)
> df$WindSpeed3pm<-normalize(df$WindSpeed3pm,0,1)
> df$Humidity9am<-normalize(df$Humidity9am,0,1)
> df$Humidity3pm<-normalize(df$Humidity3pm,0,1)
> df$Pressure9am<-normalize(df$Pressure9am,0,1)
> df$Pressure3pm<-normalize(df$Pressure3pm,0,1)
> df$Temp9am<-normalize(df$Temp9am,0,1)
> df$Temp3pm<-normalize(df$Temp3pm,0,1)
```

**Now after these steps the data is ready to apply the algorithm on it (data is clean)**

# Modeling

**Now, we will check the class attribute (RainTomorrow) if it's biased or not**

```
> ggplot(df, aes(x=reorder(RainTomorrow, RainTomorrow, function(x)-length(x)))) +
+   geom_bar(fill='orange') +  labs(x='RainTomorrow')
```




- **We notice that the class attribute (RainTomorrow) is biased, so Now to avoid biasing in train data, We will divide our dataset into two datasets, first dataset contain observations that its class is YES, and the other dataset contain observations that its class is NO, Then we will take random samples in rate of 70% from each of these datasets to get train data of 70%from original data, and then take random samples in rate of 30% from each of these datasets to get test data of 30% from original dataset.**
  **In this way there is guarantee that train data is not biased so model will be constructed correctly.**

## Apply the Classification using Naïve Bayes classifier

First we need library e1071

```
> install.packages("e1071")

> library(e1071)
```

**Split data into a training set and a test set by percentage 70 and 30 respectively**

```
> #modeling

> set.seed(5000)
> dfNo<-df[df$RainTomorrow=="No",]
> dfYes<-df[df$RainTomorrow=="Yes",]
> yesSample<-sample(2, nrow(dfYes), replace=TRUE, prob=c(.7,.3))
> NoSample<-sample(2, nrow(dfNo), replace=TRUE, prob=c(.7,.3))
> trainYes<-dfYes[yesSample==1,]
> trainNo<-dfNo[NoSample==1,]
> train<-union(trainYes,trainNo)
> testYes<-dfYes[yesSample==2,]
> testNo<-dfNo[NoSample==2,]
> test<-union(testYes,testNo)
```

- **Train a Naïve Bayes classifier using the training set**

**We will use all attribute to construct the model except date attribute because it just represent id for record.**

```
> model <- naiveBayes(RainTomorrow ~ Location + MinTemp +MaxTemp + Rainfall + WindGustDir +
+                       WindGustSpeed + WindDir9am + WindDir3pm + WindSpeed9am +
+                       WindSpeed3pm + Humidity9am + Humidity3pm + Pressure9am +
+                       Pressure3pm + Temp9am + Temp3pm + RainToday ,data=train)
```

## Evaluate the accuracy

**Use the model to predict the class labels of the train**

```
> pred<-predict(model,train)
> counter<-sum(pred==train$RainTomorrow)
> (counter/length(train$RainTomorrow))*100
[1] 81.24644
```

- We get accuracy equal 81.25% on train data and that sound good that's mean we don't have over fitting

**Use the model to predict the class labels of the train**

```
> predtest<-predict(model,test)
> countertest<-sum(predtest==test$RainTomorrow)
> (countertest/length(test$RainTomorrow))*100
[1] 80.87955
```

- We get accuracy equal 80,88% on test data so we will accept this accuracy