Data pre-processing

Manoj Kumar Nagabandi

16 07 2022

```
knitr::opts_chunk$set(warning = FALSE)
```

This file is used to preprocess the dataset and save the cleaned data for model training and comparison.

#1. Obtaining data The dataset represents a metadata of the articles, published in a website of State of Massachusetts,USA https://tnc.sites.digital.mass.gov/. A csv file was downloaded from the Kaggle https://www.kaggle.com/datasets/brllrb/uber-and-lyft-dataset-boston-ma

The data has been collected from different sources, including real-time data collection using Uber and Lyft API (Application Programming Interface) queries. The data set covers Boston's selected locations and covers approximately 2 month's data from November 2018.

```
file_input = 'rideshare_kaggle.csv'
save_output = 'rideshare_kaggle_modified.csv'
data = read.csv(file_input)
```

2. Clean and filter data

The size of the data is what you'll first notice. The articles were further divided into train, validation, and test sets to handle about 693071 data instances

```
# Shape
print('Dataset shape:')
```

```
## [1] "Dataset shape:"
```

```
dim(data)
```

```
## [1] 693071     57
```

2.1. Data types

The dataset consists of numeric and character datatypes.

```
print('Data types that are unique:')
```

```
## [1] "Data types that are unique:"
```

```
print(unique(sapply(data, class)))
```

```
## [1] "character" "numeric"   "integer"
```

To check first values of each column in data set.

```
head(data)
```

```
##                                      id  timestamp hour day month
## 1 424553bb-7174-41ea-aeb4-fe06d4f4b9d7 1544952608    9  16    12
```

```
## 2 4bd23055-6827-41c6-b23b-3c491f24e74d 1543284024    2  27    11
## 3 981a3613-77af-4620-a42a-0c0866077d1e 1543366822    1  28    11
## 4 c2d88af2-d278-4bfd-a8d0-29ca77cc5512 1543553583    4  30    11
## 5 e0126e1f-8ca9-4f2e-82b3-50505a09db9a 1543463360    3  29    11
## 6 f6f6d7e4-3e18-4922-a5f5-181cdd3fa6f2 1545071112   18  17    12
##              datetime         timezone           source   destination cab_type
## 1 2018-12-16 09:30:07 America/New_York Haymarket Square North Station     Lyft
## 2 2018-11-27 02:00:23 America/New_York Haymarket Square North Station     Lyft
## 3 2018-11-28 01:00:22 America/New_York Haymarket Square North Station     Lyft
## 4 2018-11-30 04:53:02 America/New_York Haymarket Square North Station     Lyft
## 5 2018-11-29 03:49:20 America/New_York Haymarket Square North Station     Lyft
## 6 2018-12-17 18:25:12 America/New_York Haymarket Square North Station     Lyft
##     product_id         name price distance surge_multiplier latitude longitude
## 1    lyft_line       Shared   5.0     0.44                1  42.2148   -71.033
## 2 lyft_premier          Lux  11.0     0.44                1  42.2148   -71.033
## 3         lyft         Lyft   7.0     0.44                1  42.2148   -71.033
## 4  lyft_luxsuv Lux Black XL  26.0     0.44                1  42.2148   -71.033
## 5    lyft_plus      Lyft XL   9.0     0.44                1  42.2148   -71.033
## 6     lyft_lux    Lux Black  16.5     0.44                1  42.2148   -71.033
##   temperature apparentTemperature   short_summary
## 1       42.34               37.12  Mostly Cloudy
## 2       43.58               37.35           Rain
## 3       38.33               32.93          Clear
## 4       34.38               29.63          Clear
## 5       37.44               30.88  Partly Cloudy
## 6       38.75               33.51       Overcast
##                                         long_summary precipIntensity
## 1                             Rain throughout the day.          0.0000
## 2  Rain until morning, starting again in the evening.          0.1299
## 3                           Light rain in the morning.          0.0000
## 4                    Partly cloudy throughout the day.          0.0000
## 5                    Mostly cloudy throughout the day.          0.0000
## 6             Light rain in the morning and overnight.          0.0000
##   precipProbability humidity windSpeed windGust windGustTime visibility
## 1                 0     0.68      8.66     9.17   1545015600     10.000
## 2                 1     0.94     11.98    11.98   1543291200      4.786
## 3                 0     0.75      7.33     7.33   1543334400     10.000
## 4                 0     0.73      5.28     5.28   1543514400     10.000
## 5                 0     0.70      9.14     9.14   1543446000     10.000
## 6                 0     0.84      7.19     8.88   1545022800      8.325
##   temperatureHigh temperatureHighTime temperatureLow temperatureLowTime
## 1           43.68          1544968800          34.19         1545048000
## 2           47.30          1543251600          42.10         1543298400
## 3           47.55          1543320000          33.10         1543402800
## 4           45.03          1543510800          28.90         1543579200
## 5           42.18          1543420800          36.71         1543478400
## 6           40.61          1545076800          24.07         1545130800
##   apparentTemperatureHigh apparentTemperatureHighTime apparentTemperatureLow
## 1                   37.95                  1544968800                  27.39
## 2                   43.92                  1543251600                  36.20
## 3                   44.12                  1543320000                  29.11
## 4                   38.53                  1543510800                  26.20
## 5                   35.75                  1543420800                  30.29
## 6                   34.97                  1545080400                  12.04
```

```
##   apparentTemperatureLowTime                 icon dewPoint pressure
## 1                 1545044400  partly-cloudy-night     32.70  1021.98
## 2                 1543291200                 rain     41.83  1003.97
## 3                 1543392000          clear-night     31.10   992.28
## 4                 1543575600          clear-night     26.64  1013.73
## 5                 1543460400  partly-cloudy-night     28.61   998.36
## 6                 1545134400               cloudy     34.41  1000.46
##   windBearing cloudCover uvIndex visibility.1 ozone sunriseTime sunsetTime
## 1          57       0.72       0       10.000 303.8  1544962084 1544994864
## 2          90       1.00       0        4.786 291.1  1543232969 1543266992
## 3         240       0.03       0       10.000 315.7  1543319437 1543353364
## 4         310       0.00       0       10.000 291.1  1543492370 1543526114
## 5         303       0.44       0       10.000 347.7  1543405904 1543439738
## 6         294       1.00       1        8.325 335.8  1545048523 1545081282
##   moonPhase precipIntensityMax uvIndexTime temperatureMin temperatureMinTime
## 1      0.30             0.1276  1544979600          39.89         1545012000
## 2      0.64             0.1300  1543251600          40.49         1543233600
## 3      0.68             0.1064  1543338000          35.36         1543377600
## 4      0.75             0.0000  1543507200          34.67         1543550400
## 5      0.72             0.0001  1543420800          33.10         1543402800
## 6      0.33             0.0221  1545066000          34.19         1545048000
##   temperatureMax temperatureMaxTime apparentTemperatureMin
## 1          43.68         1544968800                  33.73
## 2          47.30         1543251600                  36.20
## 3          47.55         1543320000                  31.04
## 4          45.03         1543510800                  30.30
## 5          42.18         1543420800                  29.11
## 6          40.66         1545022800                  27.39
##   apparentTemperatureMinTime apparentTemperatureMax apparentTemperatureMaxTime
## 1                 1545012000                  38.07                 1544958000
## 2                 1543291200                  43.92                 1543251600
## 3                 1543377600                  44.12                 1543320000
## 4                 1543550400                  38.53                 1543510800
## 5                 1543392000                  35.75                 1543420800
## 6                 1545044400                  34.97                 1545080400
```

2.2. Summary of Data

Based on class of variable the skimr package is used to calculate Column frequency type, Number of missing values, minimum, maximun, mean, median and percentile check

The main information we got to know is our data has 55095 missing values.

```
library(skimr)
skim(data)
```

Table 1: Data summary

Name	data
Number of rows	693071
Number of columns	57
Column type frequency:	
character	11
numeric	46

Table 1: Data summary

Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
id	0	1	36	36	0	693071	0
datetime	0	1	19	19	0	31350	0
timezone	0	1	16	16	0	1	0
source	0	1	6	23	0	12	0
destination	0	1	6	23	0	12	0
cab_type	0	1	4	4	0	2	0
product_id	0	1	4	36	0	13	0
name	0	1	3	12	0	13	0
short_summary	0	1	6	18	0	9	0
long_summary	0	1	23	52	0	11	0
icon	0	1	5	21	0	7	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
timestamp	0	1.00	1544045706.87	689192.49	1543203646.00	1543443968.00	1543737478.00	1544827509.00	1545160511.00	
hour	0	1.00	11.62	6.95	0.00	6.00	12.00	18.00	23.00	
day	0	1.00	17.79	9.98	1.00	13.00	17.00	28.00	30.00	
month	0	1.00	11.59	0.49	11.00	11.00	12.00	12.00	12.00	
price	55095	0.92	16.55	9.32	2.50	9.00	13.50	22.50	97.50	
distance	0	1.00	2.19	1.14	0.02	1.28	2.16	2.92	7.86	
surge_multiplier	0	1.00	1.01	0.09	1.00	1.00	1.00	1.00	3.00	
latitude	0	1.00	42.34	0.05	42.21	42.35	42.35	42.36	42.37	
longitude	0	1.00	-71.07	0.02	-71.11	-71.08	-71.06	-71.05	-71.03	
temperature	0	1.00	39.58	6.73	18.91	36.45	40.49	43.58	57.22	
apparentTemperature	0	1.00	35.88	7.92	12.13	31.91	35.90	40.08	57.22	
precipIntensity	0	1.00	0.01	0.03	0.00	0.00	0.00	0.00	0.14	
precipProbability	0	1.00	0.15	0.33	0.00	0.00	0.00	0.00	1.00	
humidity	0	1.00	0.74	0.14	0.38	0.64	0.71	0.88	0.96	
windSpeed	0	1.00	6.19	3.15	0.45	3.41	5.91	8.41	15.00	
windGust	0	1.00	8.47	5.29	0.80	4.06	7.55	11.74	27.25	
windGustTime	0	1.00	1544048886.95	692824.40	1543150800.00	1543431600.00	1543755600.00	1544846400.00	1545127200.00	
visibility	0	1.00	8.47	2.60	0.72	8.43	9.88	10.00	10.00	
temperatureHigh	0	1.00	45.04	6.00	32.68	42.57	44.68	46.91	57.87	
temperatureHighTime	0	1.00	1544049896.96	693792.11	1543154400.00	1543438800.00	1543788000.00	1544814000.00	1545159600.00	
temperatureLow	0	1.00	34.15	6.38	17.85	30.17	34.18	38.73	46.60	
temperatureLowTime	0	1.00	1544102170.98	692292.33	1543233600.00	1543489200.00	1543816800.00	1544835600.00	1545220800.00	
apparentTemperatureHigh	0	1.00	41.61	7.67	22.62	36.57	40.95	44.12	57.20	
apparentTemperatureHighTime	0	1.00	1544050235.96	694169.87	1543186800.00	1543438800.00	1543788000.00	1544817600.00	1545159600.00	
apparentTemperatureLow	0	1.00	30.14	8.06	11.81	27.70	30.03	35.32	47.25	
apparentTemperatureLowTime	0	1.00	1544098726.92	691737.83	1543233600.00	1543478400.00	1543816800.00	1544835600.00	1545199200.00	
dewPoint	0	1.00	31.66	9.14	4.39	27.49	30.69	38.12	50.67	
pressure	0	1.00	1010.09	13.47	988.09	999.82	1009.25	1021.86	1035.55	

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
windBearing	0	1.00	220.06	99.10	2.00	124.00	258.00	303.00	356.00	
cloudCover	0	1.00	0.69	0.36	0.00	0.37	0.82	1.00	1.00	
uvIndex	0	1.00	0.25	0.47	0.00	0.00	0.00	0.00	2.00	
visibility.1	0	1.00	8.47	2.60	0.72	8.43	9.88	10.00	10.00	
ozone	0	1.00	313.51	27.95	269.40	290.90	307.40	331.80	378.90	
sunriseTime	0	1.00	15440270986921139.2715431465315434059385437517615447892395455135001.00							
sunsetTime	0	1.00	15440604386904663.3915431806115434397215437852335448220195455167693.00							
moonPhase	0	1.00	0.58	0.24	0.09	0.30	0.68	0.79	0.93	
precipIntensityMax	0	1.00	0.04	0.06	0.00	0.00	0.00	0.09	0.15	
uvIndexTime	0	1.00	15440439666924202.7715431616015434208015437700015448068015455152400.00							
temperatureMin	0	1.00	33.46	6.47	15.63	30.17	34.24	38.88	43.10	
temperatureMinTime	0	1.00	15440416006957195.4415431220015433992015437268015447888015455192000.00							
temperatureMax	0	1.00	45.26	5.65	33.51	42.57	44.68	46.91	57.87	
temperatureMaxTime	0	1.00	15440473066900135.3415431544015434388015437880015448140015455109200.00							
apparentTemperatureMin	0	1.00	29.73	7.11	11.81	27.76	30.13	35.71	40.05	
apparentTemperatureMinTime	0	1.00	15440480364874186.1915431364015433992015437448015447888015455134400.00							
apparentTemperatureMax	0	1.00	42.00	6.94	28.95	36.57	40.95	44.12	57.20	
apparentTemperatureMaxTime	0	1.00	15440479963915077.6515431868015434388015437880015448176015455109200.00							

Check infinite values in each column of data set

```
#
sprintf("Total number of infinite and nan values present in each column of dataset is:")
```

```
## [1] "Total number of infinite and nan values present in each column of dataset is:"
```

```
for (d in colnames(data))
{
 null_sum = sum(is.infinite(data$d))
 nan_sum = sum(is.nan(data$d))
}
cat((null_sum),(nan_sum),sep="\n")
```

```
## 0
## 0
```

As we already saw earlier there are 55095 NA values so we can remove them.

```
data <- na.omit(data)
sum(is.na(data)==TRUE)
```

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

We have 2 columns having the same data so we drop 1 of them.

```
head(data$visibility)
```

```
## [1] 10.000  4.786 10.000 10.000 10.000  8.325
```

```
head(data$visibility.1)
```

```
## [1] 10.000  4.786 10.000 10.000 10.000  8.325
```

```
library(dplyr)
```

```
##
## 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
```

```
data <- select(data, -c("visibility.1"))
```

```
dim(data)
```

```
## [1] 637976     56
```

Since there are 17 integer datatypes and 28 numeric datatypes we need to check how many datatypes are numeric and character and store them in seperate lists.

```
int_lst = c()
name_int_lst =c()
name_char_lst = c()
for (i in colnames(data))
  {
     if (class(data[,i]) == "integer" | (class(data[,i]) == "numeric"))
   {
     int_lst <- c(int_lst,as.integer("1"))
     name_int_lst <- c(name_int_lst,i)
     }
  else{
    name_char_lst <- c(name_char_lst,i)
  }
}

print('Number of integer and numeric datatypes')
```

```
## [1] "Number of integer and numeric datatypes"
```

```
cat(sum(int_lst),sep="\n")
```

```
## 45
```

```
print('Number of charcter datatypes')
```

```
## [1] "Number of charcter datatypes"
```

```
cat(length(name_char_lst),sep="\n")
```

```
## 11
```

2.3. Skewness & Distribution Check

We believe it is critical to examine the distribution of the variables before cleaning the data. As a result, the cells below contain all of the functions required to plot histograms using all of the numerical variables in the data set. .

```
library(ggplot2)
hist_plot <- function(name_int_lst)
  {
    i = 1
    while(i <= length(name_int_lst))
```

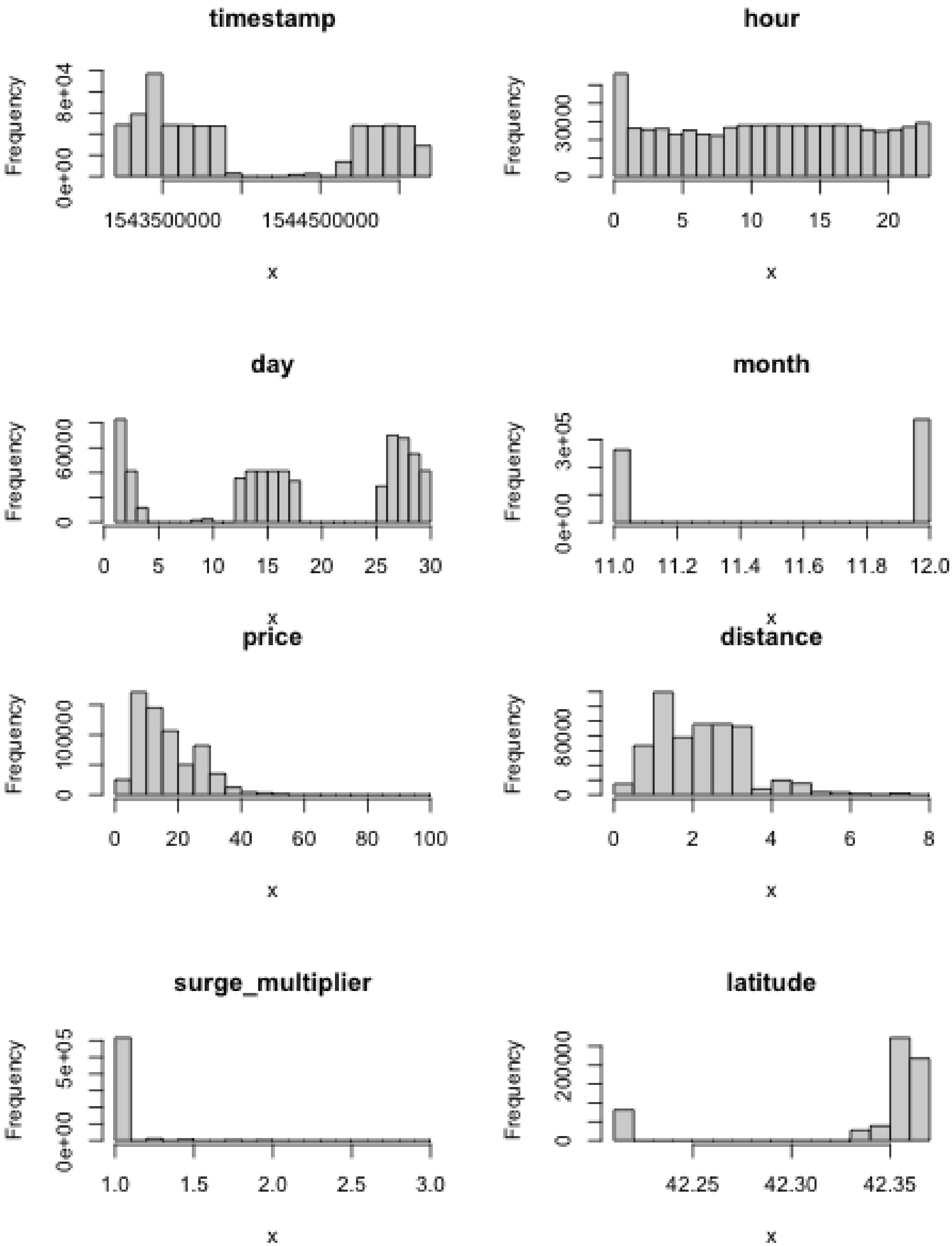
```r
        {
          set.seed(5)
          x <- unlist(data[name_int_lst[i]])

          # Histogram
          print(ggplot(data, aes(x = x)) +
          geom_histogram(colour = 1, fill = "lightblue",bins=30) +
          geom_density(lwd = 1, colour = 4,
                       fill = 4, alpha = 0.25)+labs(x = name_int_lst[i],
           y = "Frequency",
           title  = name_int_lst[i]))
          i = i + 1
    }
}
```

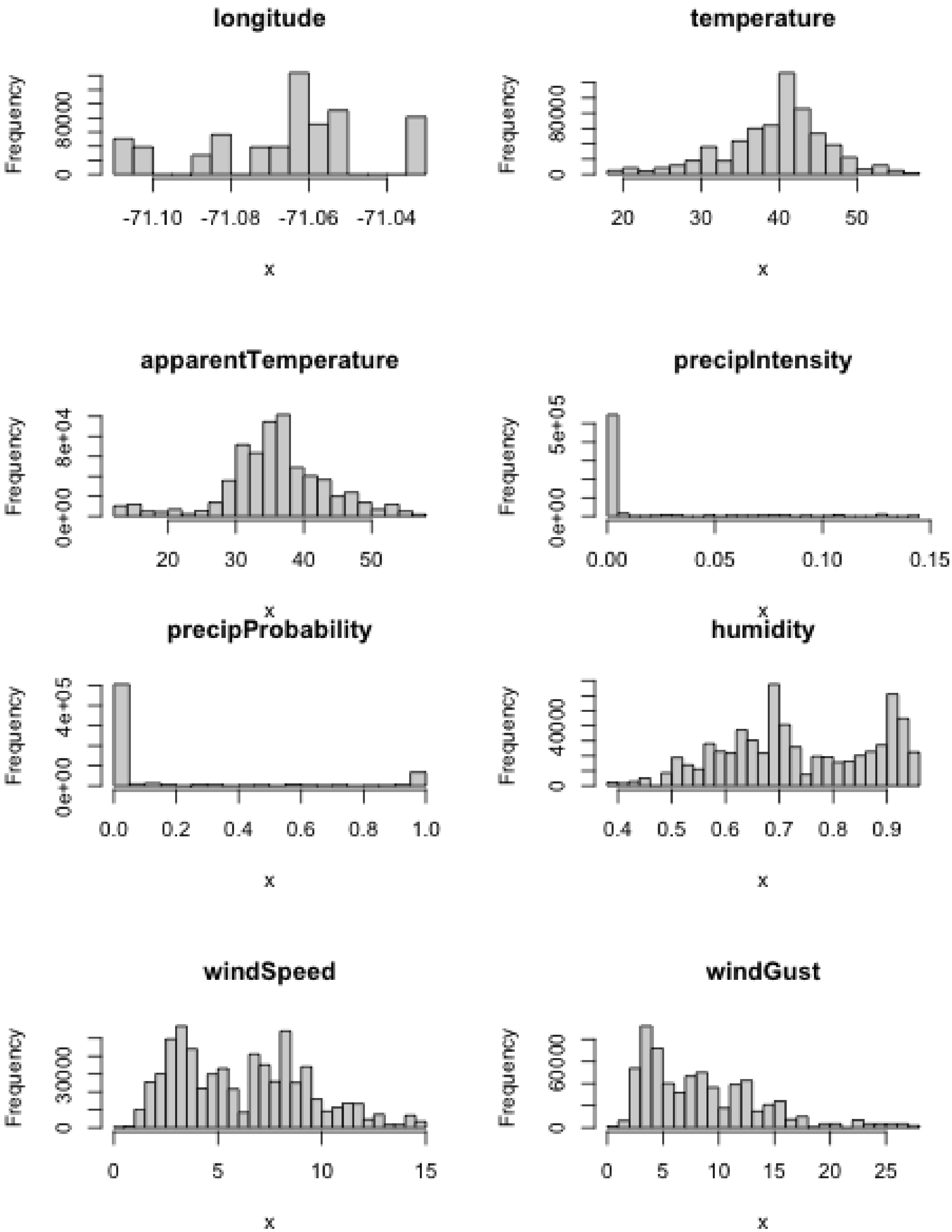
There is Bernouli distribution for some columns names like month

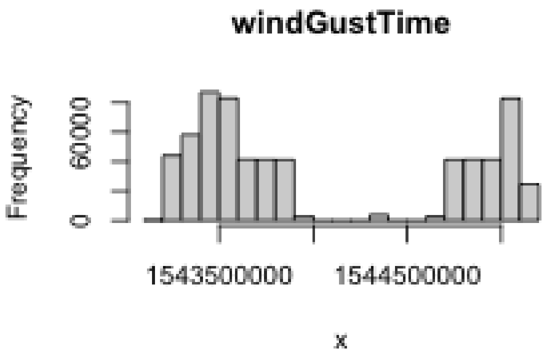
Since it is visually not possible to tell exact skewness for all features except some like pecepIntesitymax,visibility,uvIndex,precipProbability,precipintensity,surge multiplier ,we calculate below.

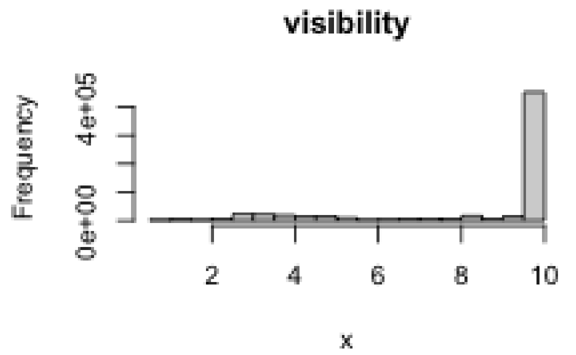
There also may be Potential outliers and therfore there is need of IQR filtering

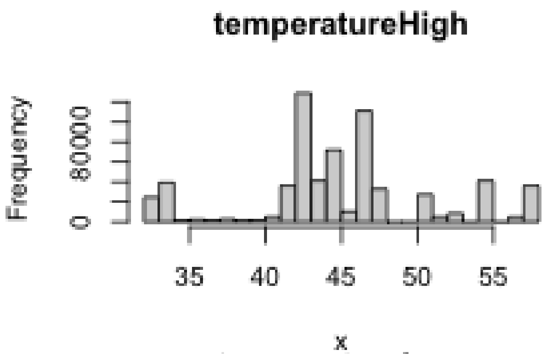
```r
i = 1
par(mfrow = c(2, 2))
  while(i <= length(name_int_lst))
    {


      # Sample data
      set.seed(2)
      x <- unlist(data[name_int_lst[i]])

      # Histogram
      hist(x,
           main = name_int_lst[i])

      i = i + 1
     }
```

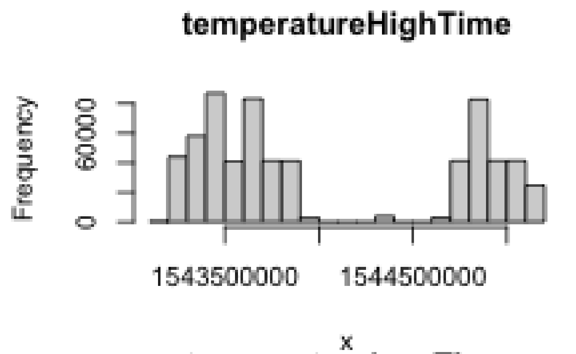

timestamp
Frequency
0e+00
8e+04
1543500000
1544500000
x
hour
Frequency
0
30000
0
5
10
15
20
x
day
Frequency
0
600000
0
5
10
15
20
25
30
x
month
Frequency
0e+00
3e+05
11.0
11.2
11.4
11.6
11.8
12.0
x
price
Frequency
0
100000
0
20
40
60
80
100
x
distance
Frequency
0
80000
0
2
4
6
8
x
surge_multiplier
Frequency
0e+00
5e+05
1.0
1.5
2.0
2.5
3.0
x
latitude
Frequency
0
200000
42.25
42.30
42.35
x

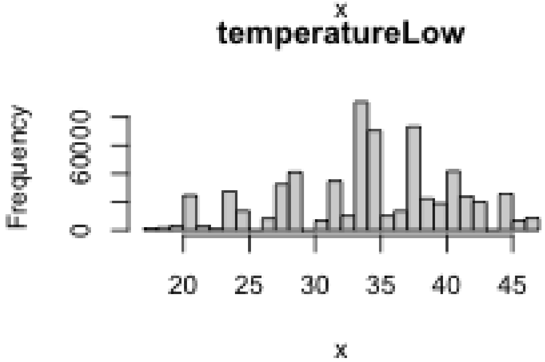

longitude
Frequency
0
80000
-71.10
-71.08
-71.06
-71.04
x
temperature
Frequency
0
80000
20
30
40
50
x
apparentTemperature
Frequency
0e+00
8e+04
20
30
40
50
x
precipIntensity
Frequency
0e+00
5e+05
0.00
0.05
0.10
0.15
x
precipProbability
Frequency
0e+00
4e+05
0.0
0.2
0.4
0.6
0.8
1.0
x
humidity
Frequency
0
40000
0.4
0.5
0.6
0.7
0.8
0.9
x
windSpeed
Frequency
0
30000
0
5
10
15
x
windGust
Frequency
0
60000
0
5
10
15
20
25
x

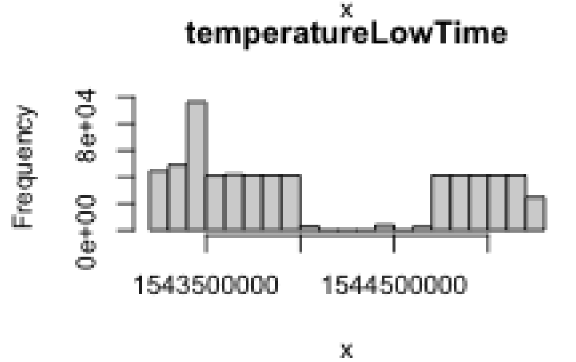

windGustTime
Frequency
0
60000
1543500000
1544500000
x

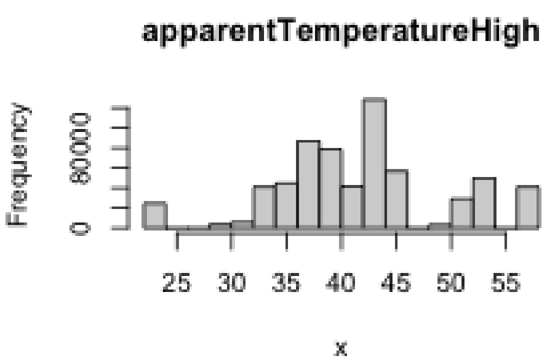

visibility
Frequency
0e+00
4e+05
2
4
6
8
10
x

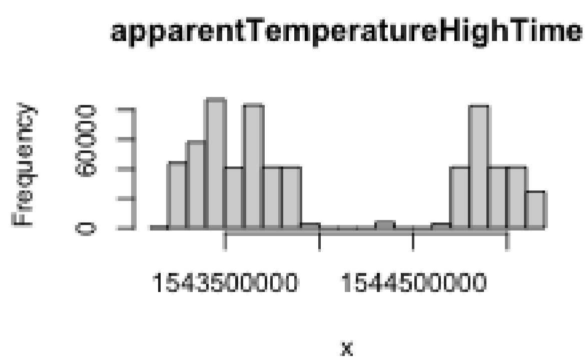

temperatureHigh
Frequency
0
80000
35
40
45
50
55
x

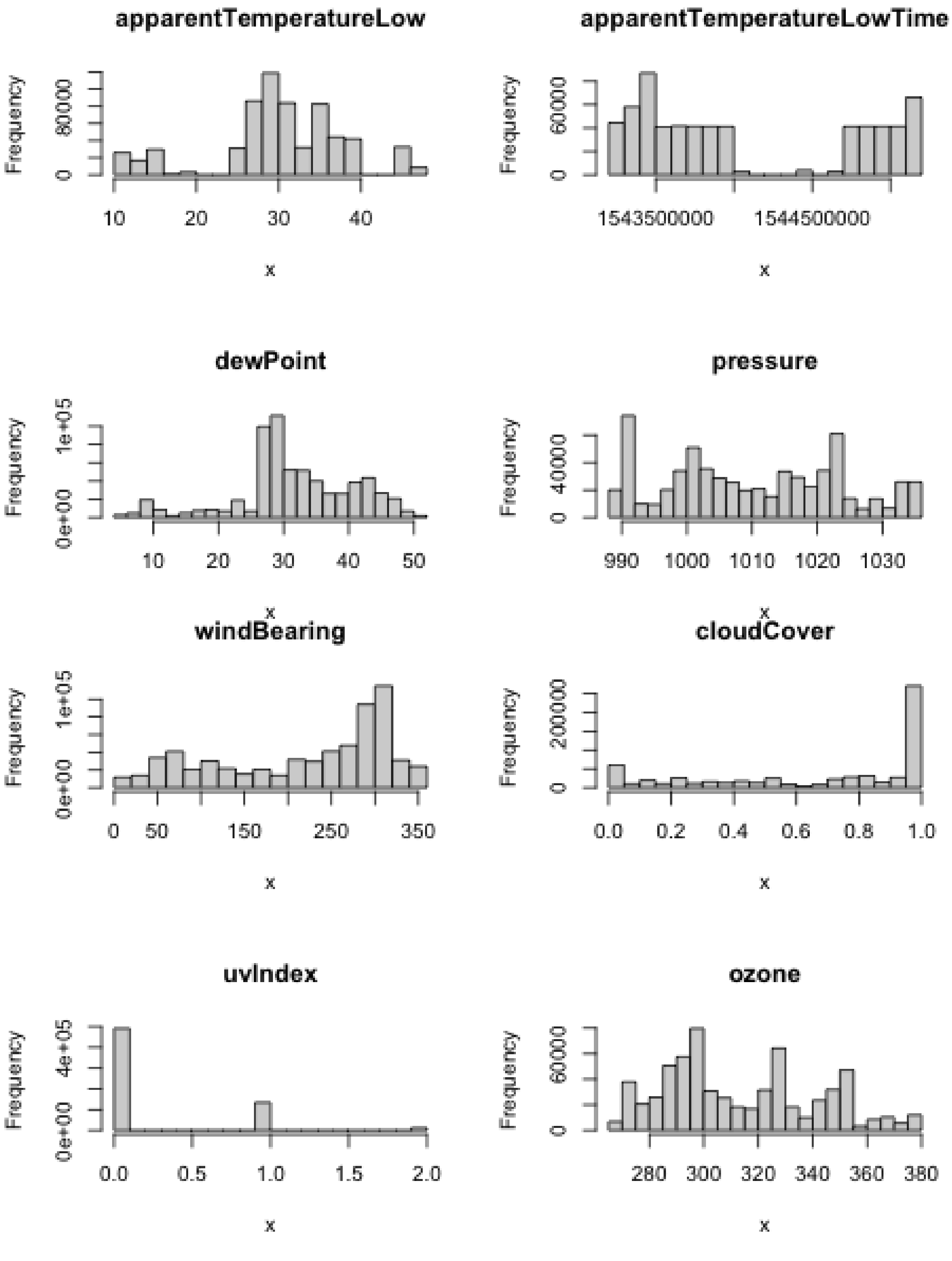

temperatureHighTime
Frequency
0
60000
1543500000
1544500000
x

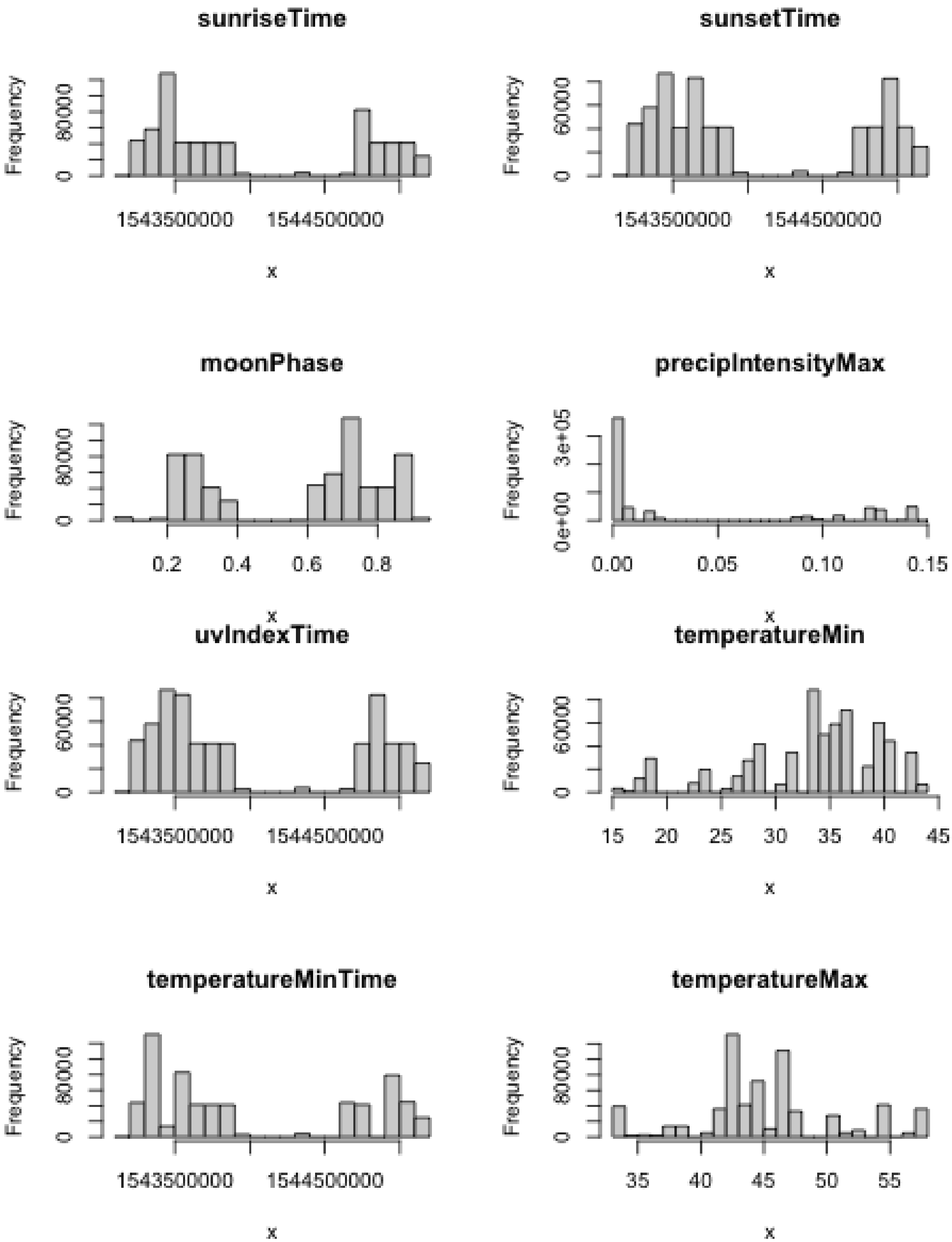

temperatureLow
Frequency
0
60000
20
25
30
35
40
45
x

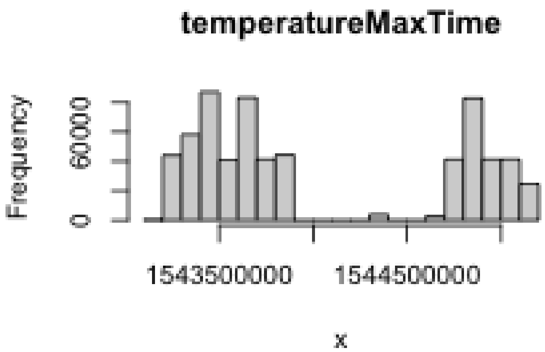

temperatureLowTime
Frequency
0e+00
8e+04
1543500000
1544500000
x

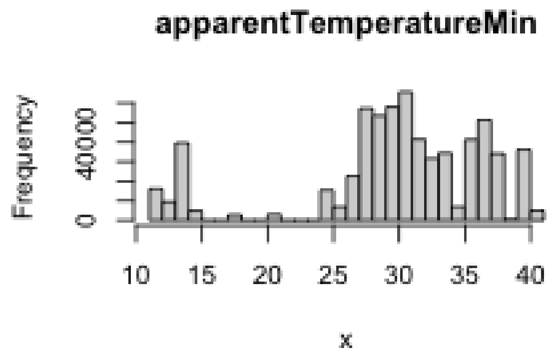

apparentTemperatureHigh
Frequency
0
80000
25
30
35
40
45
50
55
x

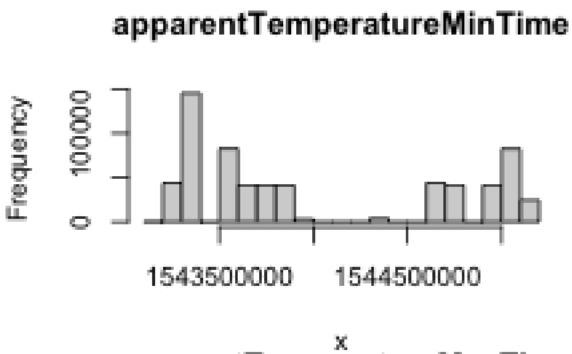

apparentTemperatureHighTime
Frequency
0
60000
1543500000
1544500000
x

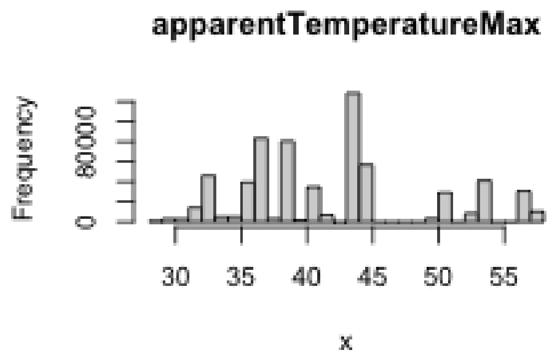

apparentTemperatureLow
Frequency
0
80000
10
20
30
40
x
apparentTemperatureLowTime
Frequency
0
60000
1543500000
1544500000
x
dewPoint
Frequency
0e+00
1e+05
10
20
30
40
50
x
pressure
Frequency
0
40000
990
1000
1010
1020
1030
x
windBearing
Frequency
0e+00
1e+05
0
50
150
250
350
x
cloudCover
Frequency
0
200000
0.0
0.2
0.4
0.6
0.8
1.0
x
uvIndex
Frequency
0e+00
4e+05
0.0
0.5
1.0
1.5
2.0
x
ozone
Frequency
0
60000
280
300
320
340
360
380
x

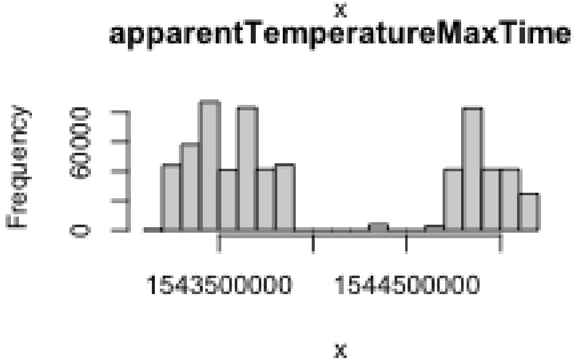

sunriseTime
sunsetTime
moonPhase
precipIntensityMax
uvIndexTime
temperatureMin
temperatureMinTime
temperatureMax
Frequency
x
0
60000
80000
0e+00
3e+05
1543500000
1544500000
0.2
0.4
0.6
0.8
0.00
0.05
0.10
0.15
15
20
25
30
35
40
45
50
55











Skewness function defined that is used to calculate skewness.

```
## The function is currently defined as
skew <- function (x, na.rm = TRUE) {
    if(length(dim(x))==0) {
       if (na.rm)   { x <- x[!is.na(x)] }        #remove missing values
    sum((x - mean(x))^3)/(length(x) * sd(x)^3)  #calculate skew  for a vector
 } else { apply(x,2,function(x) sum((x - mean(x,na.rm=na.rm))^3,na.rm=na.rm)/( (length(x)-sum(is.na(x))) 
```

To find columns that are positive or negatively skewed

```
pve_skw = c()
nve_skw = c()
sym = c()
i = 1
while(i <= length(name_int_lst))
  {

    # Sample data
    set.seed(2)
```

```
    if (skew(data[name_int_lst[i]])>0.10){
      pve_skw <- c(pve_skw,name_int_lst[i])
    }
    else if (skew(data[name_int_lst[i]])<(-0.10)){
      nve_skw <- c(nve_skw,name_int_lst[i])
    }
    else{
      sym <- c(sym,name_int_lst[i])
    }

    i = i + 1
  }
```

```
cat("Features that are negatively skewed:" , nve_skw,'\n' )
```

```
## Features that are negatively skewed: day month latitude longitude temperature apparentTemperature vi
```

```
cat("Features that are positively skewed:" , pve_skw )
```

```
## Features that are positively skewed: timestamp price distance surge_multiplier precipIntensity preci
```

2.4. Apply cube root transformation to skewed columns

In this part, we normalize the most highly skewed columns with cubical root transformation

"surge_multiplier" "precipIntensity" has skewness greater than 3.

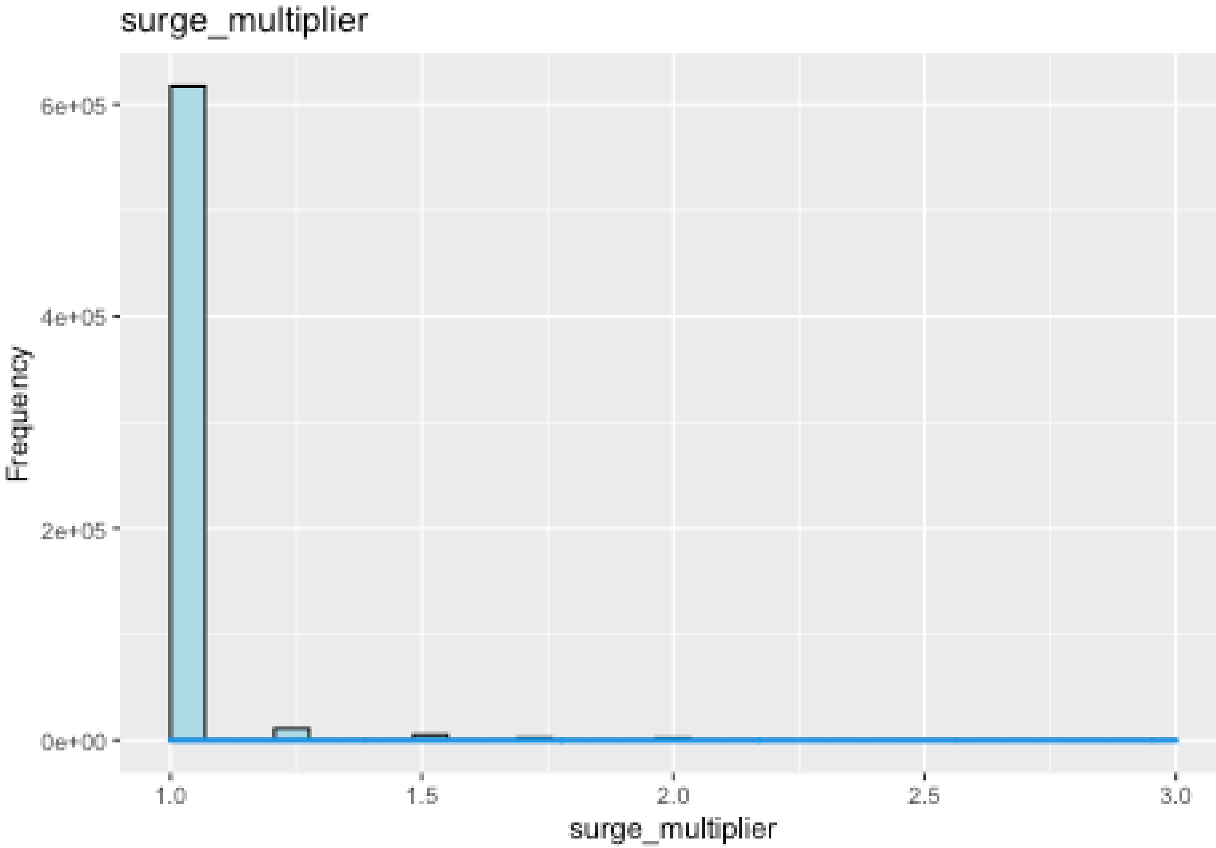
```
high_skw = c()
sk_thresh = 3
sym = c()
i = 1
while(i <= length(name_int_lst))
  {
    # Sample data
    set.seed(2)
    x <- name_int_lst[i]
    if (abs(skew(data[x]))>sk_thresh){
      high_skw <- c(high_skw,x)
    }
    i = i + 1
  }
```

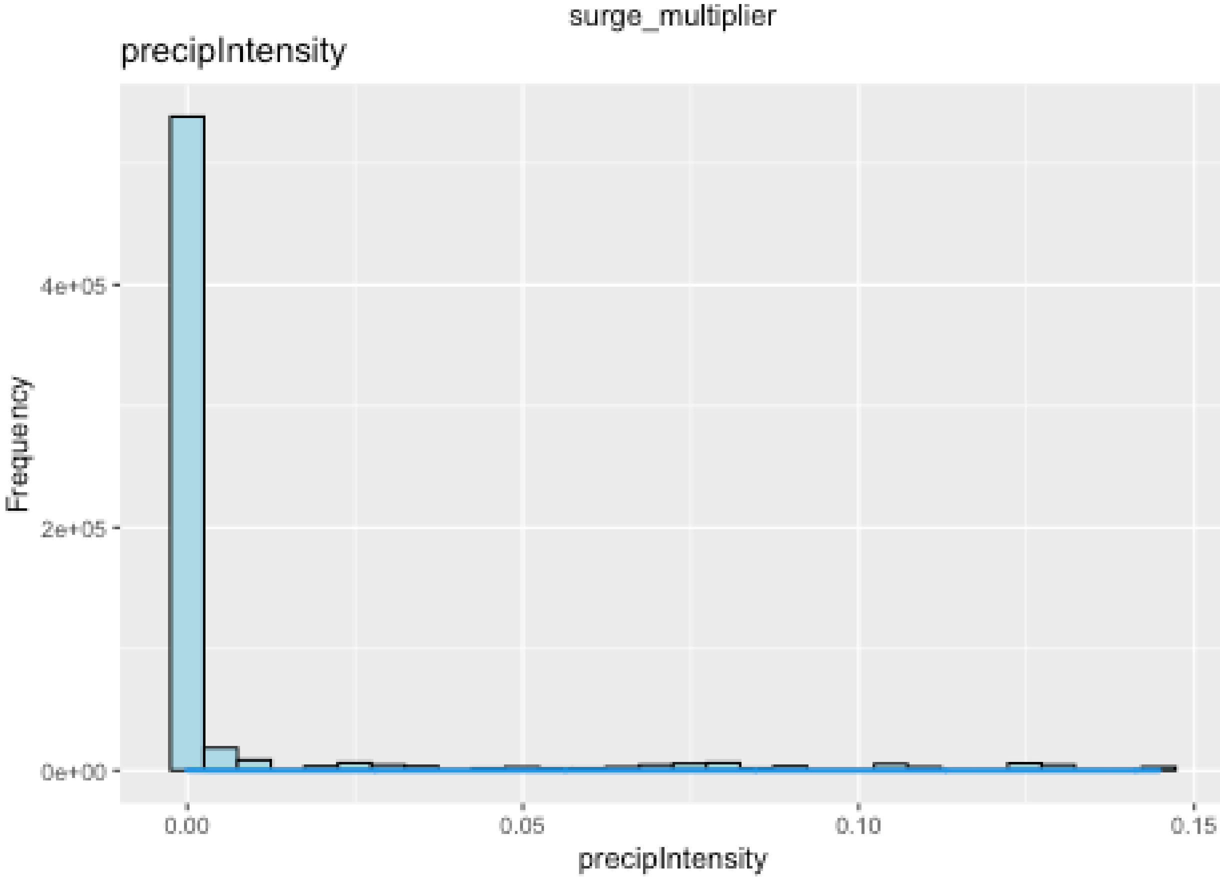
Now we need to apply cubical root transformation

```
cat("Before applying cubical root transformation")
```

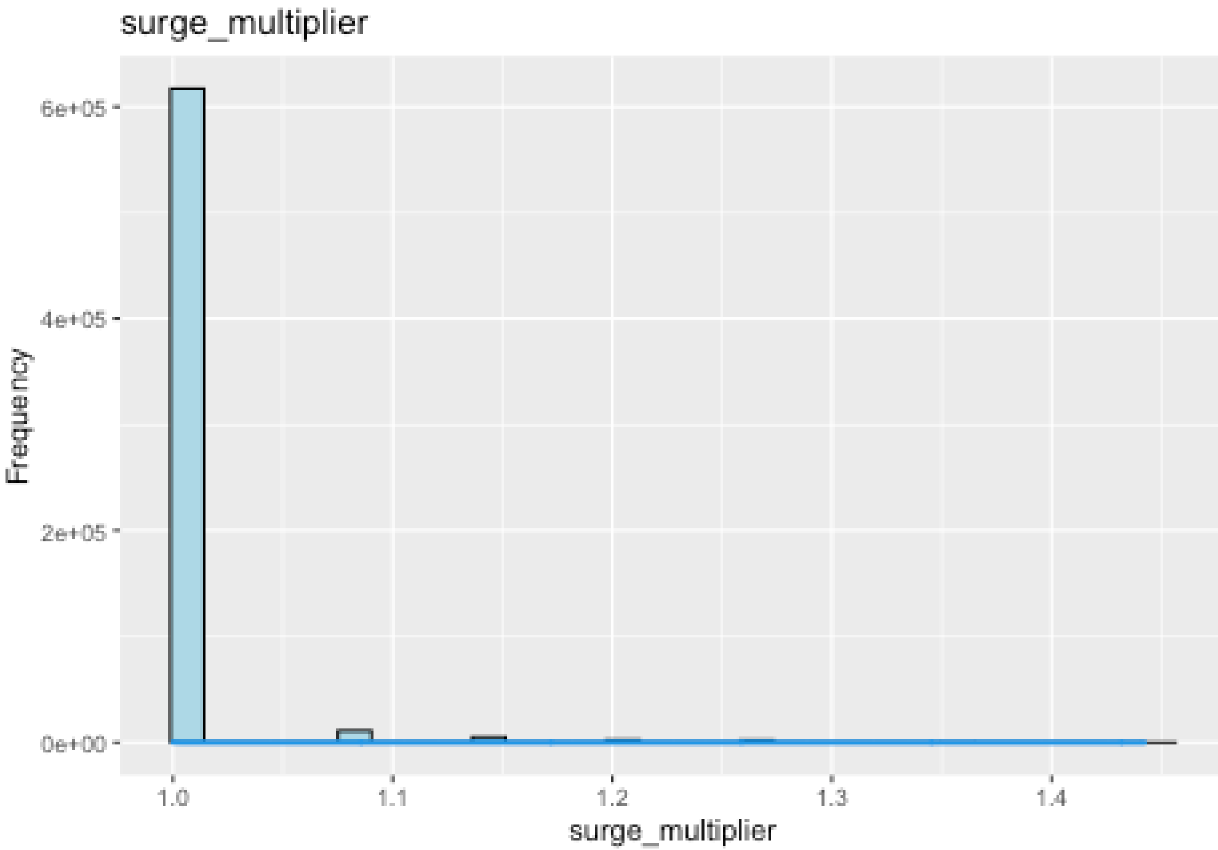
```
## Before applying cubical root transformation
```

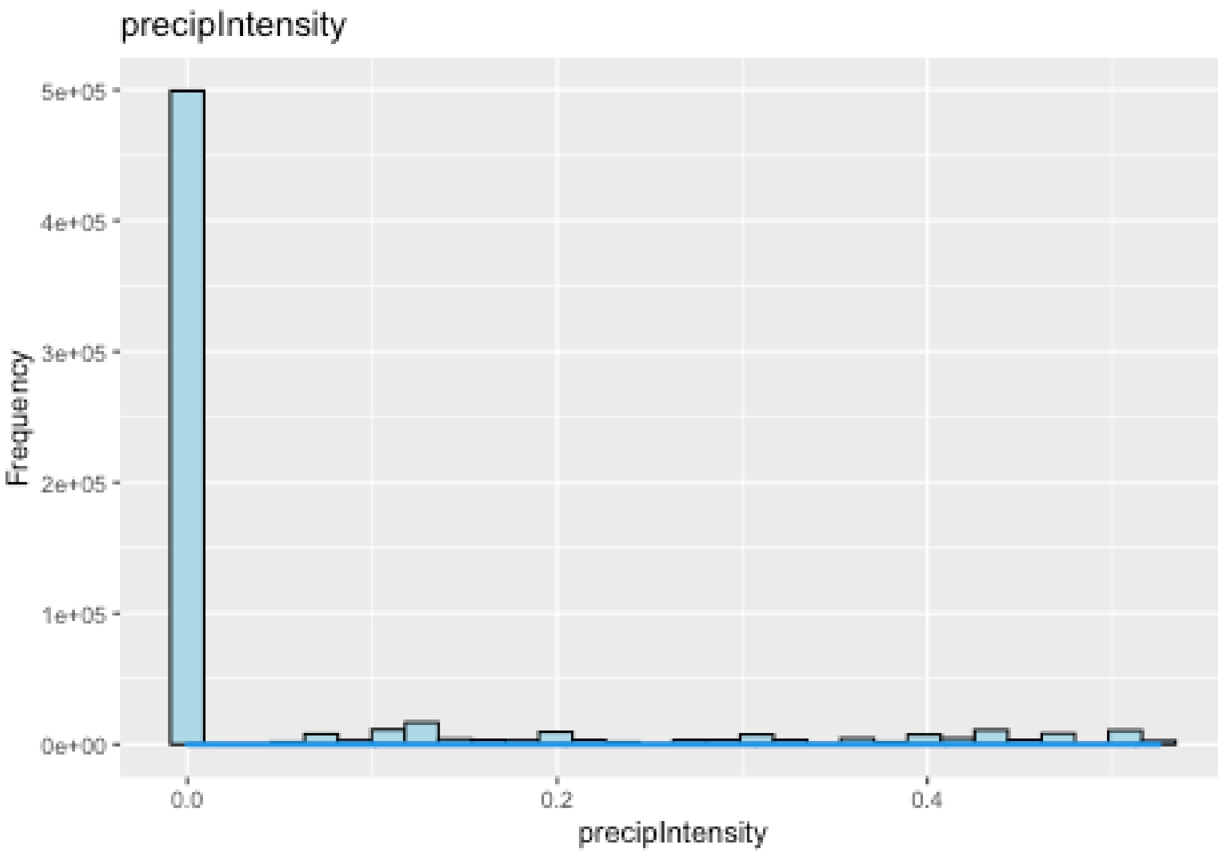
```
hist_plot(high_skw)
```


surge_multiplier
Frequency
6e+05
4e+05
2e+05
0e+00
1.0
1.5
2.0
2.5
3.0
surge_multiplier


precipIntensity
Frequency
4e+05
2e+05
0e+00
0.00
0.05
0.10
0.15
precipIntensity

```r
for(col in high_skw){
  data[, col] = (data[, col])^(1 / 3)
}
hist_plot(high_skw)
```






```r
outlier_cols = c()
q1c = c()
q3c = c()
col_percent = c()

for ( col in name_int_lst){
  Q1 = quantile(data[,col],0.25,names = FALSE)
  Q3 = quantile(data[,col],0.75,names = FALSE)
  IQR = Q3-Q1

  no_outliers <- subset(data, (data[, col] > (Q1 - 1.5*IQR)) & (data[, col] < (Q3 + 1.5*IQR)))

  outliers_per = (nrow(data) - nrow(no_outliers)) / nrow(data)
  if (outliers_per > 0.10){
     cat( col,'has', outliers_per,'percent outliers:', '\n')
  q1c = append(q1c,Q1)
  q3c = append(q3c,Q3)
  col_percent = append(col_percent,outliers_per)
  outlier_cols = append(outlier_cols, col)
  }
}
```

```
## surge_multiplier has 1 percent outliers:
## latitude has 0.1278575 percent outliers:
## precipIntensity has 1 percent outliers:
## precipProbability has 1 percent outliers:
## visibility has 0.1972754 percent outliers:
```

```
## temperatureHigh has 0.2364101 percent outliers:
## apparentTemperatureHigh has 0.1033236 percent outliers:
## apparentTemperatureLow has 0.1265063 percent outliers:
## uvIndex has 1 percent outliers:
## temperatureMax has 0.1976673 percent outliers:
## apparentTemperatureMin has 0.1097471 percent outliers:
```

Let's take a closer look at the histograms for columns with outliers. We see a lot of Binomially distributed variables, and keep them for statistical tests. Several columns may be candidates for filtering based on IQR values.

- latitude
- visibility
- temperatureHigh
- apparentTemperatureHigh
- apparentTemperatureLow
- temperatureMax
- apparentTemperatureMin

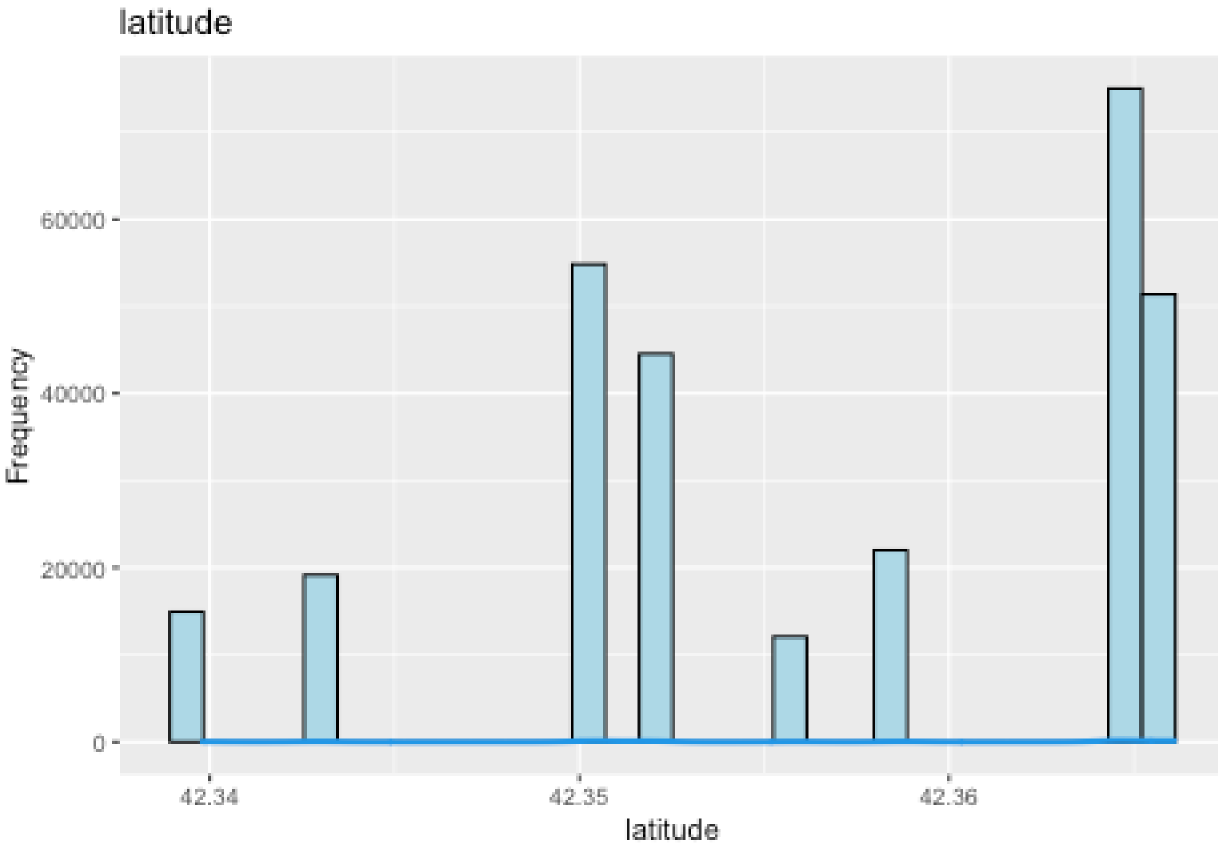
```
Q1c = c()
Q3c = c()
per_col = c()
iqr_filter_col = c()

for (i in 1:length(col_percent))
  {
  if (col_percent[i] != 1.0){
    Q1c = append(Q1c,q1c[i])
    Q3c = append(Q3c,q3c[i])
    per_col = append(per_col,col_percent[i])
    iqr_filter_col = c(iqr_filter_col,outlier_cols[i])
  }
}
```

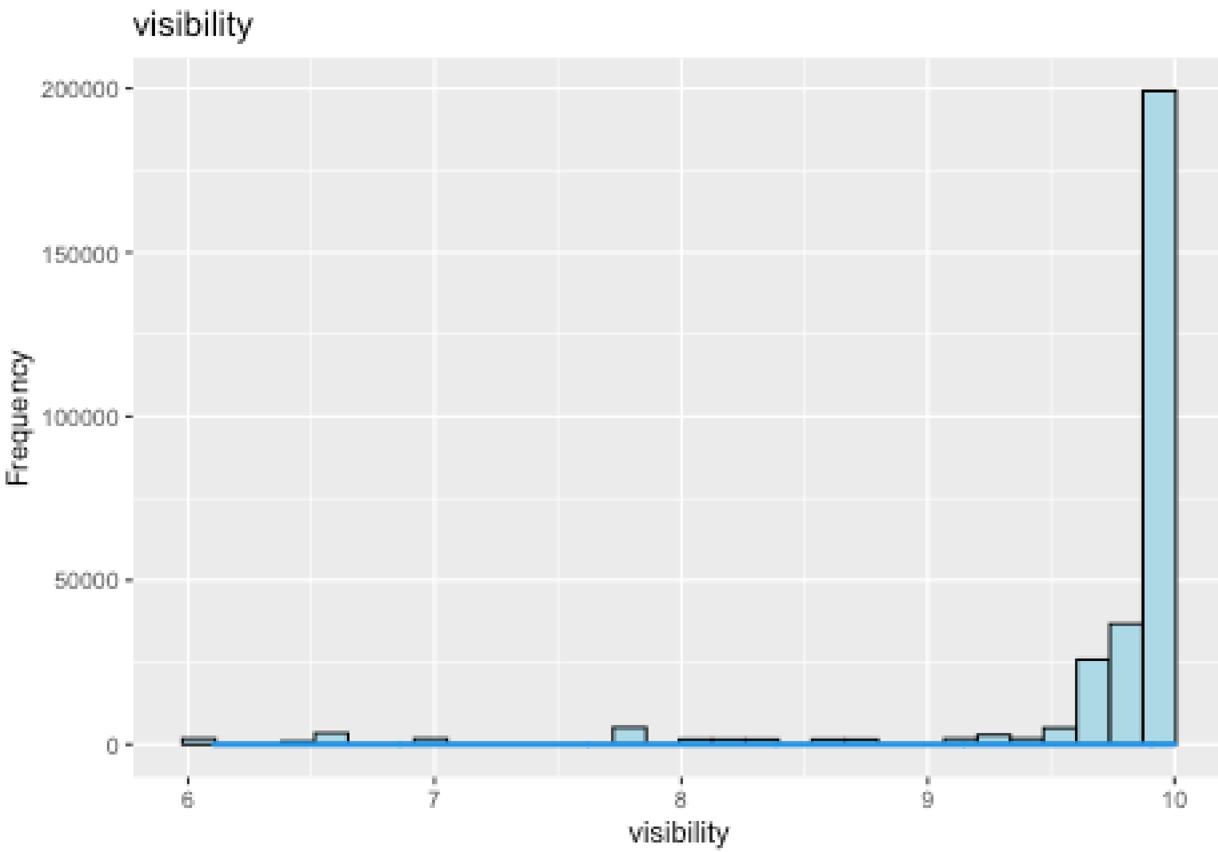
```
for (i in 1:length(iqr_filter_col)){
    iqr = Q3c[i] - Q1c[i]
    cat('Since the column', iqr_filter_col[i], 'has', per_col[i],'percent of outliers', '\n')
    col_data = data[, iqr_filter_col[i]]
    data = subset(
        data,
        (col_data > (Q1c[i] - 1.5 * iqr)) & (col_data < (Q3c[i] + 1.5 * iqr))
    )
    cat('New data size:', dim(data), '\n')
}
```

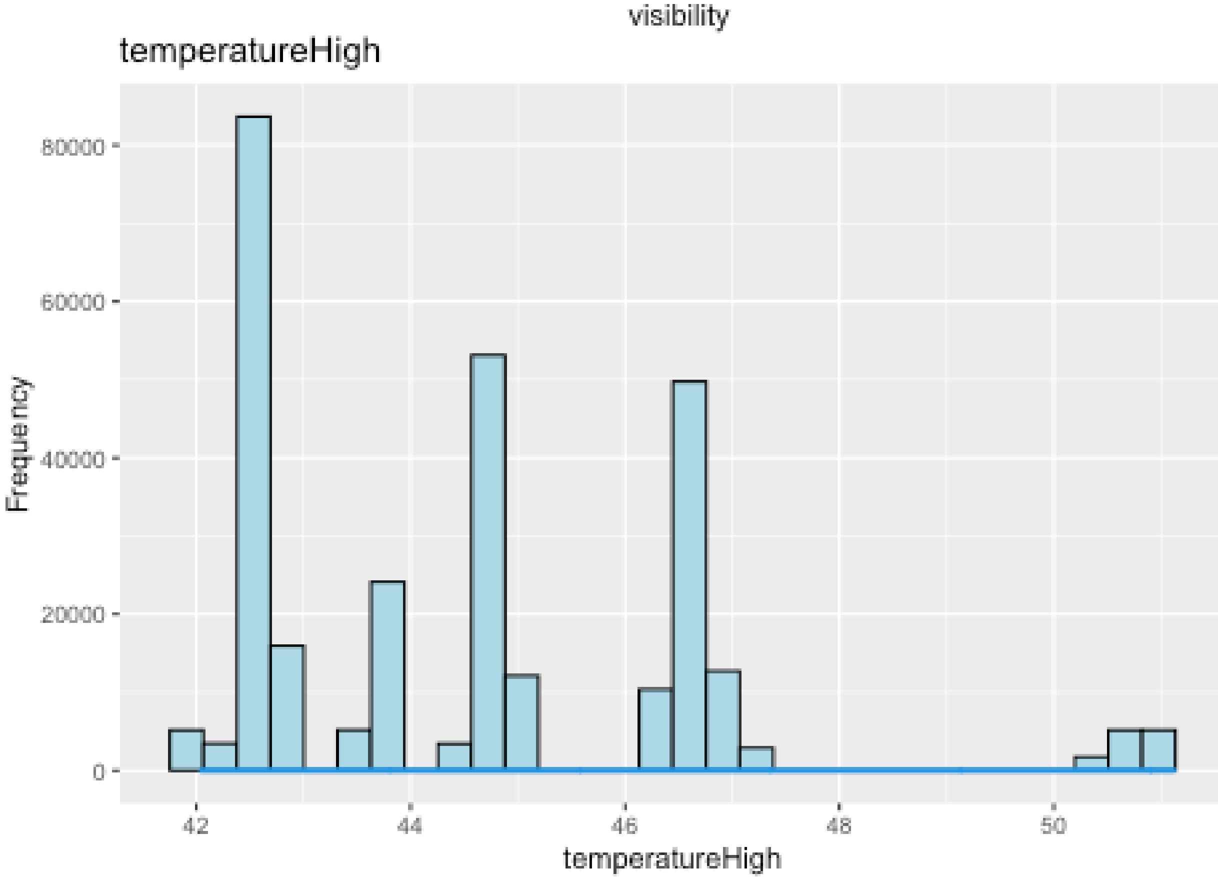
```
## Since the column latitude has 0.1278575 percent of outliers
## New data size: 556406 56
## Since the column visibility has 0.1972754 percent of outliers
## New data size: 458462 56
## Since the column temperatureHigh has 0.2364101 percent of outliers
## New data size: 329159 56
## Since the column apparentTemperatureHigh has 0.1033236 percent of outliers
## New data size: 329159 56
## Since the column apparentTemperatureLow has 0.1265063 percent of outliers
## New data size: 295589 56
## Since the column temperatureMax has 0.1976673 percent of outliers
```

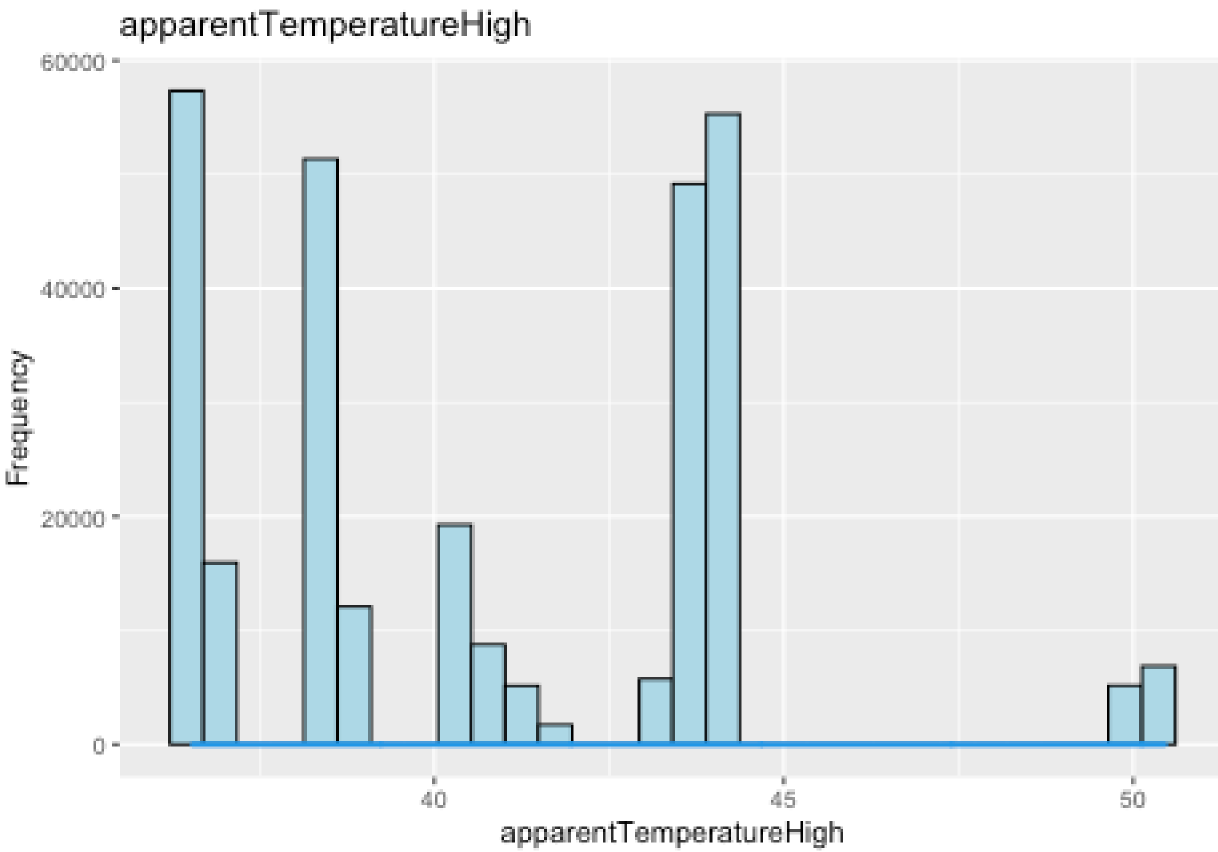
```
## New data size: 295589 56
## Since the column apparentTemperatureMin has 0.1097471 percent of outliers
## New data size: 293877 56
```

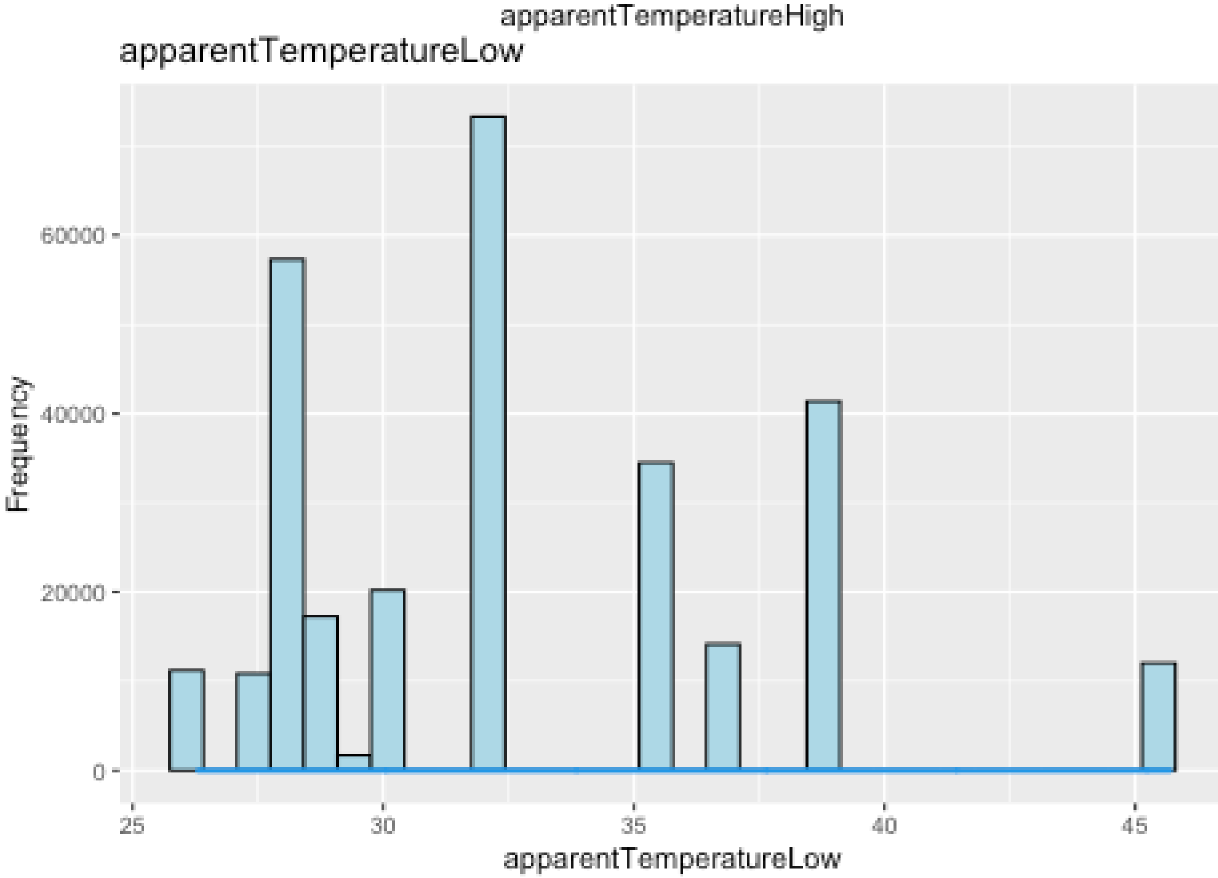
```
hist_plot(iqr_filter_col)
```

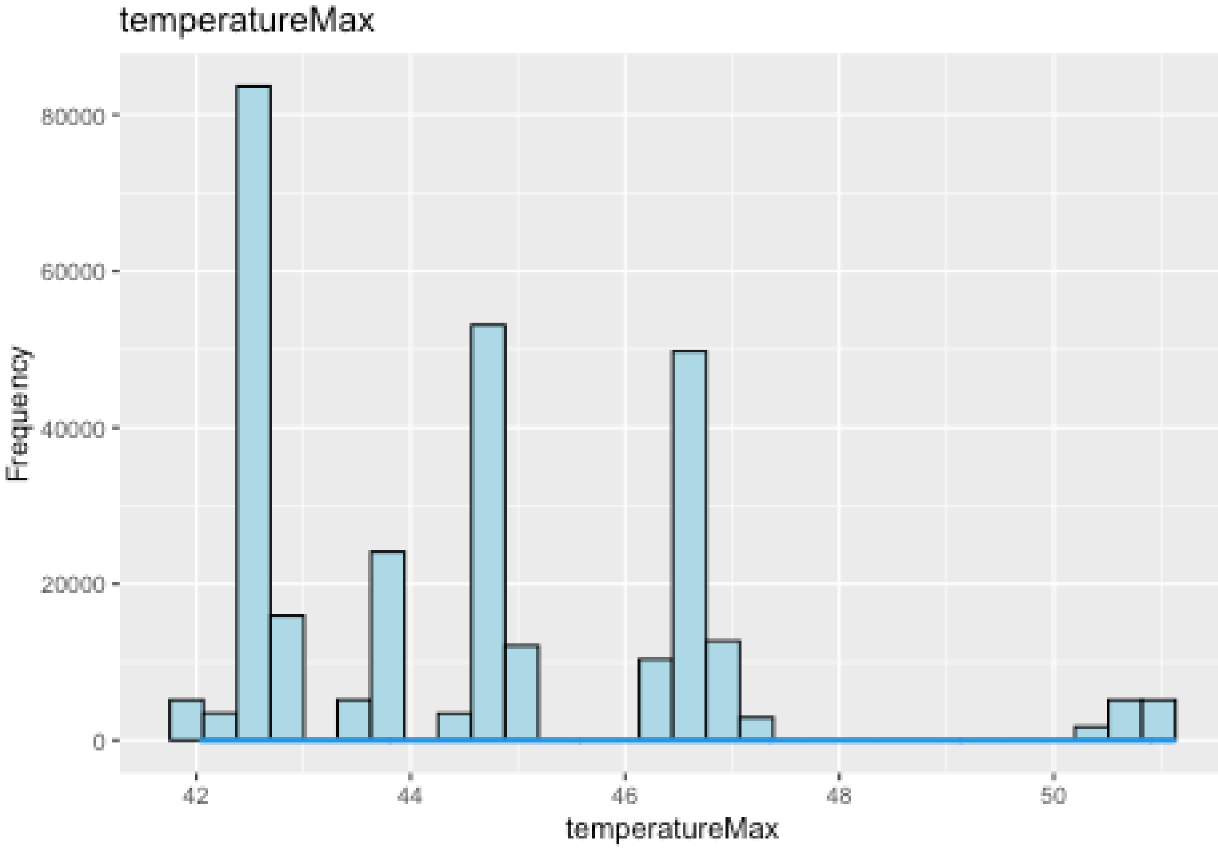


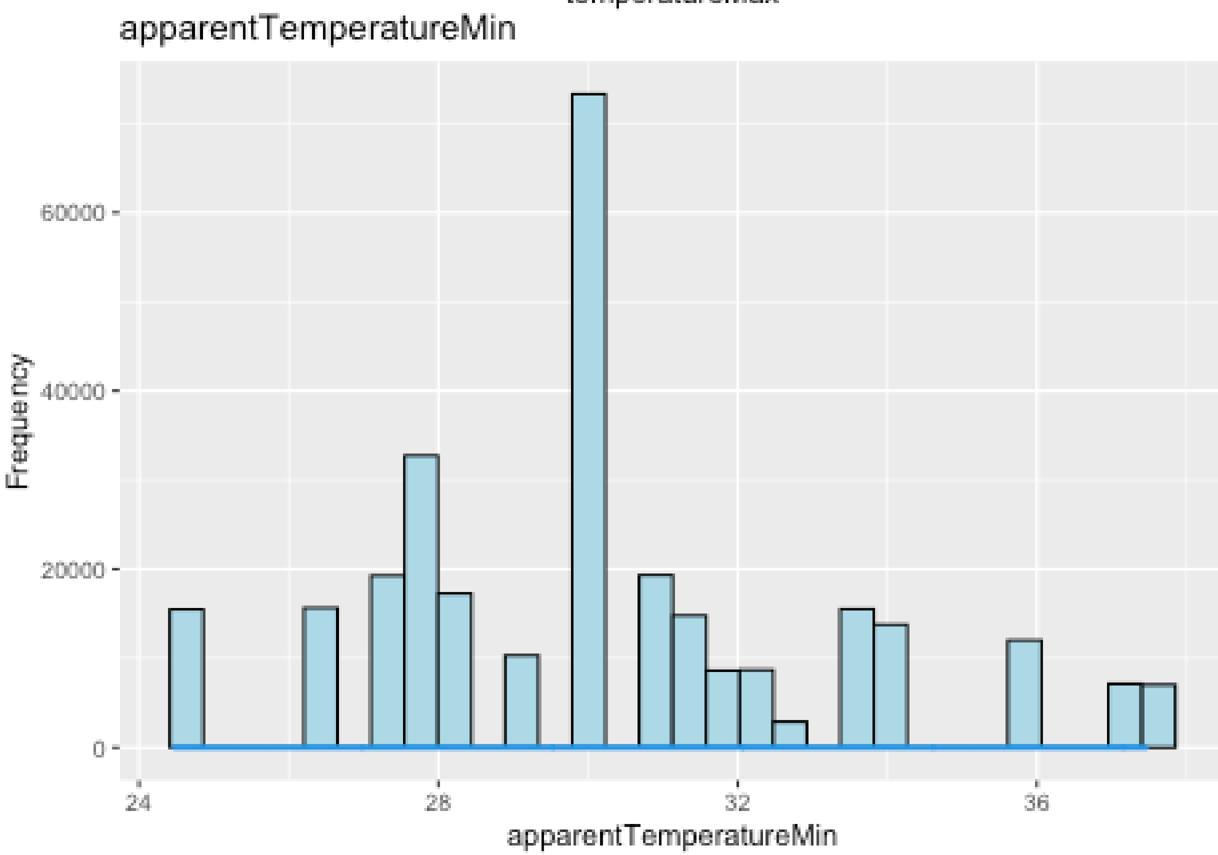

visibility
Frequency
200000
150000
100000
50000
0
6
7
8
9
10
visibility


temperatureHigh
Frequency
80000
60000
40000
20000
0
42
44
46
48
50
temperatureHigh


apparentTemperatureHigh
60000
40000
20000
0
Frequency
40
45
50
apparentTemperatureHigh


apparentTemperatureLow
60000
40000
20000
0
Frequency
25
30
35
40
45
apparentTemperatureLow





2.5. Converting of categorical variables into multiple variables using One-hot encoding.

One-hot encoding is the process of converting a categorical variable with multiple categories into multiple variables, each with a value of 1 or 0.

Let's see all the features that has character data type and check how many categorical values each features have.

```
print('The 11 features that has character as datatype are:')
```

```
## [1] "The 11 features that has character as datatype are:"
```

```
cat( name_char_lst)
```

```
## id datetime timezone source destination cab_type product_id name short_summary long_summary icon
```

1)Since in every row the id feature is unique or the whole feature contains the same value like timezone we can discard the feature from our data set since model does not learn anything for them.

2)The feature datetime contain 13795 types of unique values present so there are too many classes to perform label encoding so they can be removed

```
for (i in name_char_lst){
    print(i)
    print(length(unique(data[,i])))
}
```

```
## [1] "id"
## [1] 293877
## [1] "datetime"
## [1] 13795
## [1] "timezone"
## [1] 1
## [1] "source"
## [1] 12
## [1] "destination"
## [1] 12
## [1] "cab_type"
## [1] 2
## [1] "product_id"
## [1] 12
## [1] "name"
## [1] 12
## [1] "short_summary"
## [1] 6
## [1] "long_summary"
## [1] 7
## [1] "icon"
## [1] 6
```

Id and timezone features are removed Price feature is removed and added at end of data frame

```
print('Dimension of data before deleting id,datetime and timezone features')
```

```
## [1] "Dimension of data before deleting id,datetime and timezone features"
```

```
cat(dim(data))
```

```
## 293877 56
```

```
price = data$price
data <- select(data, -c("id","timezone","datetime","price"))
```

```r
cat('Dimension of data after deleting id, price and timezone features', '\n')
```

```
## Dimension of data after deleting id, price and timezone features
```

```r
cat(dim(data))
```

```
## 293877 52
```

We need to label encode 12 unique variables of source feature and add it to our data frame

```r
table(data$source)
```

```
##
##                Back Bay            Beacon Hill       Boston University
##                   24507                  24289                   24327
##                  Fenway      Financial District        Haymarket Square
##                   24657                  24862                   24403
##               North End           North Station Northeastern University
##                   24321                  24620                   24555
##           South Station        Theatre District                West End
##                   24716                  24176                   24444
```

```r
library(mltools)
library(data.table)
```

```
##
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

```r
lab_source = one_hot(as.data.table(as.factor(data$source)))

print(head(lab_source))
```

```
##    V1_Back Bay V1_Beacon Hill V1_Boston University V1_Fenway
## 1:           1              0                    0         0
## 2:           1              0                    0         0
## 3:           0              0                    0         0
## 4:           0              0                    0         0
## 5:           0              0                    0         0
## 6:           0              0                    0         0
##    V1_Financial District V1_Haymarket Square V1_North End V1_North Station
## 1:                     0                   0            0                0
## 2:                     0                   0            0                0
## 3:                     0                   0            1                0
## 4:                     0                   0            1                0
## 5:                     0                   0            0                1
## 6:                     0                   0            0                1
##    V1_Northeastern University V1_South Station V1_Theatre District V1_West End
## 1:                          0                0                   0           0
## 2:                          0                0                   0           0
## 3:                          0                0                   0           0
## 4:                          0                0                   0           0
## 5:                          0                0                   0           0
## 6:                          0                0                   0           0
```

```r
print(dim(lab_source))
```

```
## [1] 293877     12
```

We need to destination encode 12 unique variables of destination feature and add it to our data frame

```r
table(data$destination)
```

```
##
##                Back Bay             Beacon Hill       Boston University
##                   24197                   24239                   24646
##                  Fenway      Financial District        Haymarket Square
##                   24368                   24896                   24487
##               North End           North Station Northeastern University
##                   24421                   24224                   24584
##           South Station        Theatre District                West End
##                   24392                   24578                   24845
```

```r
library(mltools)
library(data.table)

lab_destination = one_hot(as.data.table(as.factor(data$destination)))

print(head(lab_destination))
```

```
##    V1_Back Bay V1_Beacon Hill V1_Boston University V1_Fenway
## 1:           0              0                    0         0
## 2:           0              0                    0         0
## 3:           0              0                    0         0
## 4:           0              0                    0         0
## 5:           0              0                    0         0
## 6:           0              0                    0         0
##    V1_Financial District V1_Haymarket Square V1_North End V1_North Station
## 1:                     0                   0            0                0
## 2:                     0                   0            0                0
## 3:                     0                   0            0                0
## 4:                     0                   0            0                0
## 5:                     0                   1            0                0
## 6:                     0                   1            0                0
##    V1_Northeastern University V1_South Station V1_Theatre District V1_West End
## 1:                          1                0                   0           0
## 2:                          1                0                   0           0
## 3:                          0                0                   0           1
## 4:                          0                0                   0           1
## 5:                          0                0                   0           0
## 6:                          0                0                   0           0
```

```r
print(dim(lab_destination))
```

```
## [1] 293877     12
```

We need to label encode 2 unique variables of cab_type feature and add it to our data frame

```r
table(data$cab_type)
```

```
##
##   Lyft   Uber
## 142317 151560
```

```r
library(mltools)
library(data.table)

lab_cab_type = one_hot(as.data.table(as.factor(data$cab_type)))

print(head(lab_cab_type))
```

```
##    V1_Lyft V1_Uber
## 1:       1       0
## 2:       1       0
## 3:       0       1
## 4:       0       1
## 5:       1       0
## 6:       1       0
```

```r
print(dim(lab_cab_type))
```

```
## [1] 293877      2
```

We need to destination encode 12 unique variables of product_id feature and add it to our data frame

Since we have many unidentified information about the categories present in the data frame we can drop this feature.

```r
table(data$product_id)
```

```
##
## 55c66225-fbe7-4fd5-9072-eab1ece5e23e 6c84fd89-3f11-4782-9b50-97c468b19529
##                                25156                                25305
## 6d318bcc-22a3-4af6-bddd-b409bfce1546 6f72dfc5-27f1-42e8-84db-ccc7a75f6969
##                                25214                                25347
## 997acbb5-e102-41e1-b155-9df7de0a73f2 9a0e7b09-b92b-4c41-9779-2ad22b4d779d
##                                25270                                25268
##                                 lyft                            lyft_line
##                                23937                                23638
##                             lyft_lux                          lyft_luxsuv
##                                23707                                23691
##                            lyft_plus                         lyft_premier
##                                23596                                23748
```

```r
print('Dimension of data before deleting product_id feature')
```

```
## [1] "Dimension of data before deleting product_id feature"
```

```r
cat(dim(data),'\n')
```

```
## 293877 52
```

```r
data <- select(data, -c("product_id"))
cat('New data size:', dim(data), '\n')
```

```
## New data size: 293877 51
```

We need to label encode 12 unique variables of name feature and add it to our data frame

```r
table(data$name)
```

```
##
##        Black    Black SUV          Lux    Lux Black Lux Black XL         Lyft
##        25305        25214        23748        23707        23691        23937
```

```
##      Lyft XL       Shared     UberPool        UberX       UberXL          WAV
##        23596        23638        25270        25156        25347        25268
```

```
library(mltools)
library(data.table)

lab_name = one_hot(as.data.table(as.factor(data$name)))

print(head(lab_name))
```

```
##    V1_Black V1_Black SUV V1_Lux V1_Lux Black V1_Lux Black XL V1_Lyft V1_Lyft XL
## 1:        0            0      1            0               0       0          0
## 2:        0            0      0            0               0       1          0
## 3:        0            0      0            0               0       0          0
## 4:        0            0      0            0               0       0          0
## 5:        0            0      0            0               0       0          1
## 6:        0            0      0            1               0       0          0
##    V1_Shared V1_UberPool V1_UberX V1_UberXL V1_WAV
## 1:         0           0        0         0      0
## 2:         0           0        0         0      0
## 3:         0           0        0         1      0
## 4:         0           1        0         0      0
## 5:         0           0        0         0      0
## 6:         0           0        0         0      0
```

```
print(dim(lab_name))
```

```
## [1] 293877     12
```

We need to label encode 6 unique variables of short_summary feature and add it to our data frame

```
table(data$short_summary)
```

```
## 
##            Clear           Drizzle      Mostly Cloudy          Overcast 
##             46288              1542              76732             98235 
##     Partly Cloudy   Possible Drizzle 
##             65256              5824
```

```
table(data$long_summary)
```

```
## 
##            Light rain in the morning and overnight. 
##                                                34497 
##                           Light rain in the morning. 
##                                                20208 
##                            Light rain until evening. 
##                                                12019 
##                   Mostly cloudy throughout the day. 
##                                               108191 
##                   Partly cloudy throughout the day. 
##                                                75523 
##                            Rain throughout the day. 
##                                                29294 
##  Rain until morning, starting again in the evening. 
##                                                14145
```

```r
table(data$icon)
```

```
##
##          clear-day          clear-night               cloudy
##              17233                29055                98235
##  partly-cloudy-day  partly-cloudy-night                 rain
##              56886                85102                 7366
```

```r
lab_shortsummary = one_hot(as.data.table(as.factor(data$short_summary)))
lab_longsummary = one_hot(as.data.table(as.factor(data$long_summary)))
lab_icon = one_hot(as.data.table(as.factor(data$icon)))
print(head(lab_shortsummary))
```

```
##    V1_ Clear  V1_ Drizzle  V1_ Mostly Cloudy  V1_ Overcast  V1_ Partly Cloudy
## 1:         1            0                 0             0                  0
## 2:         0            0                 0             1                  0
## 3:         0            0                 0             1                  0
## 4:         0            0                 1             0                  0
## 5:         1            0                 0             0                  0
## 6:         0            0                 1             0                  0
##    V1_ Possible Drizzle
## 1:                    0
## 2:                    0
## 3:                    0
## 4:                    0
## 5:                    0
## 6:                    0
```

```r
print(dim(lab_shortsummary))
```

```
## [1] 293877      6
```

```r
print(head(lab_longsummary))
```

```
##    V1_ Light rain in the morning and overnight.
## 1:                                            0
## 2:                                            0
## 3:                                            0
## 4:                                            0
## 5:                                            0
## 6:                                            0
##    V1_ Light rain in the morning.  V1_ Light rain until evening.
## 1:                              0                              0
## 2:                              0                              0
## 3:                              0                              0
## 4:                              0                              0
## 5:                              0                              0
## 6:                              1                              0
##    V1_ Mostly cloudy throughout the day.
## 1:                                     1
## 2:                                     1
## 3:                                     1
## 4:                                     0
## 5:                                     0
## 6:                                     0
##    V1_ Partly cloudy throughout the day.  V1_ Rain throughout the day.
```

```
## 1:                                      0                             0
## 2:                                      0                             0
## 3:                                      0                             0
## 4:                                      1                             0
## 5:                                      0                             1
## 6:                                      0                             0
##    V1_ Rain until morning, starting again in the evening.
## 1:                                                      0
## 2:                                                      0
## 3:                                                      0
## 4:                                                      0
## 5:                                                      0
## 6:                                                      0
```

```
print(dim(lab_longsummary))
```

```
## [1] 293877      7
```

```
print(head(lab_icon))
```

```
##    V1_ clear-day  V1_ clear-night  V1_ cloudy  V1_ partly-cloudy-day
## 1:             1                0           0                      0
## 2:             0                0           1                      0
## 3:             0                0           1                      0
## 4:             0                0           0                      1
## 5:             0                1           0                      0
## 6:             0                0           0                      1
##    V1_ partly-cloudy-night  V1_ rain
## 1:                       0         0
## 2:                       0         0
## 3:                       0         0
## 4:                       0         0
## 5:                       0         0
## 6:                       0         0
```

```
print(dim(lab_icon))
```

```
## [1] 293877      6
```

Now lets add all the labeled data features to the original data and remove the original feature from which label features are produced.

```
print('Dimension of data before deleting cab_type,destination,icon,long_summary,name,short_summary,sour
```

```
## [1] "Dimension of data before deleting cab_type,destination,icon,long_summary,name,short_summary,sou
```

```
cat(dim(data), '\n')
```

```
## 293877 51
```

```
data <- select(data, -c("cab_type","destination","icon","long_summary","name","short_summary","source")
cat('New data size:', dim(data))
```

```
## New data size: 293877 44
```

```
print("Old data size:")
```

```
## [1] "Old data size:"
```

```
cat(dim(data),'\n')
```

```
## 293877 44
```

```
data <- cbind(data,lab_cab_type,lab_destination,lab_icon,lab_longsummary,lab_name,lab_shortsummary,lab_s
print("New data size")
```

```
## [1] "New data size"
```

```
cat(dim(data),'\n')
```

```
## 293877 102
```

#3.Save data

```
write.csv(data, save_output, row.names = F)
```