AIRLINE ARRIVAL PROJECT

Code

Hide

```
# remove some junk
rm(list = ls())
```

A. Exploratory Data Analysis (EDA)

1. import libraries and some options

Hide

```
# import library
library(skimr) # for more detail summary
library(jsonlite) # for JSON purposes
library(jsonify) # for JSON purposes
library(glue) # general purposes
library(ggplot2) # for plotting
library(caret) # general purposes
library(bestNormalize) # for normalization
library(parallel) # for parallel computing
library(dplyr) # for general purposes
library(corrplot) # plot correlation
library(foreach) # for parallel computing
library(doParallel) # for parallel computing
library(mltools) # for machine learning purpose
library(data.table) # for create data.table
library(caTools) # for train/test split
library(Rtsne) # for tsne plotting
library(DMwR) # for smote implementation
library(ROSE)# for ROSE sampling
library(xgboost) # for xgboost model


# max print to 777 rows
options(max.print=777)

# define function to garbage collect
cl <- function(){
  rm()
  gc()
}
```

2. Prepare data

Hide

```
# get directory of all file
foo <- system("ls /home/apolong72/ds/r/data/airline/", intern = T)
boo <- paste("/home/apolong72/ds/r/data/airline/", foo, sep = "")

# read all csv file
## use mclapply to trigger multi core progress
system.time(
  df <- mclapply(boo, read.csv, mc.cores = 6)
)
```

```
   user  system elapsed 
 70.420   4.687  46.516 
```

Hide

```
# concat to 1 df
df <- bind_rows(df)
```

Quick look at data

Hide

```
df
```

	M... <int>	DAY_OF_M... <int>	DAY_OF_... <int>	OP_UNIQUE_CAR... <fctr>	OP_CARRIER_FL_... <int>	ORI... <fctr>	D... <fctr>	CR
1	10	8	2	OH	5120	PHL	CAK	
2	10	9	3	OH	5120	PHL	CAK	
3	10	10	4	OH	5120	PHL	CAK	
4	10	11	5	OH	5120	PHL	CAK	
5	10	12	6	OH	5120	PHL	CAK	
6	10	13	7	OH	5120	PHL	CAK	
7	10	14	1	OH	5120	PHL	CAK	
8	10	15	2	OH	5120	PHL	CAK	
9	10	16	3	OH	5120	PHL	CAK	
10	10	17	4	OH	5120	PHL	CAK	

1-10 of 7,422,037 rows | 1-9 of 22 columns Previous **1** 2 3 4 5 6 ... 78 Next

3.Remove some unused features (after manually analyse data)

Hide

```
# remove some feature
remove.abc = c('DEP_DEL15', 'DEP_DELAY_GROUP', 'FLIGHTS', 'CANCELLED', 'DIVERTED','OP_CA
RRIER_AIRLINE_ID','ORIGIN_AIRPORT_ID','DEST_AIRPORT_ID','ARR_DELAY_NEW','DEP_DELAY_NEW')
```

Hide

```
#
df <- subset(df, select = -c(DEP_DEL15, DEP_DELAY_GROUP, FLIGHTS, CANCELLED, DIVERTED, O
P_CARRIER_AIRLINE_ID, ORIGIN_AIRPORT_ID, DEST_AIRPORT_ID, ARR_DELAY_NEW, DEP_DELAY_NEW))

# remove index columns
df$X <- NULL

# create copy of df
## create sample for testing purpose
df2 <- df
df2_sample <- sample_n(df2, 100)
```

Hide

```
# get summary of df2 (use so much ram! cannot use `skim` but `summary`, but it retrieve
 immediately)
system.time(
  q_summary <- mclapply(df2, summary, mc.cores = 6)
)
```

```
   user  system elapsed
 12.288   5.064   4.137
```

Hide

```
q_summary
```

```
$MONTH
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  1.000   4.000   7.000   6.579  10.000  12.000 

$DAY_OF_MONTH
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
   1.00    8.00   16.00   15.73   23.00   31.00 

$DAY_OF_WEEK
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  1.000   2.000   4.000   3.937   6.000   7.000 

$OP_UNIQUE_CARRIER
     9E      AA      AS      B6      DL      EV      F9      G4      HA      MQ      NK 
OH      OO      UA      WN      YV      YX 
 257132  946776  264816  297411  991986  134683  135543  105305   83891  327007  204845 
289304  836445  625910 1363946  227888  329149 

$OP_CARRIER_FL_NUM
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
      1    1025    2158    2557    3917    7933 

$ORIGIN
    ATL     ORD     DFW     DEN     CLT     LAX     IAH     PHX     LGA     SFO     LAS 
DTW     MSP     BOS     MCO     SEA     DCA 
 395009  339606  304344  252026  235496  219952  179688  175328  171665  170918  164020 
161768  160960  150564  143094  142857  139388 
    EWR     JFK     PHL     SLC     BWI     FLL     SAN     MIA     MDW     BNA     TPA 
DAL     IAD     STL     AUS     RDU     SJC 
 136081  127145  119151  115130  104652   98409   93470   89214   83763   82654   76599 
73907   69223   68401   66811   64156   62946 
    PDX     HOU     MSY     MCI     OAK     HNL     SMF     PIT     CLE     IND     CVG 
CMH     SNA     SAT     RSW     JAX     BUR 
  62441   60415   58164   54942   53069   52139   51482   50401   49615   49135   48247 
47897   40754   40717   34479   33932   31905 
    MKE     BDL     OGG     SJU     PBI     OMA     MEM     CHS     BUF     RIC     ORF 
SDF     OKC     ABQ     ONT     RNO     BHM 
  30895   28700   27326   26828   26064   25960   25808   25715   24854   24808   24748 
24339   24127   23731   22608   20022   19475 
    TUS     GRR     BOI     ANC     PVD     TYS     SAV     TUL     ELP     DSM     GSP 
LGB     KOA     GSO     ROC     LIT     SYR 
  19308   19197   19075   18988   18159   18033   17640   17085   17050   16951   16020 
15818   15479   14634   14588   14265   14154 
    XNA     LIH     MSN     DAY     PNS     ALB     FAT     PSP     GEG     MYR     HPN 
PWM     SFB     ICT (Other) 
  14134   14063   14050   13834   12610   12568   12361   12232   12034   11790   11604 
11151   10755   10653  677617 

$DEST
    ATL     ORD     DFW     DEN     CLT     LAX     IAH     PHX     LGA     SFO     LAS 
DTW     MSP     BOS     MCO     SEA     DCA 
 395026  339569  304346  252064  235490  219996  179682  175343  171665  170966  164043 
161741  160955  150554  143095  142841  139393 
```

```
    EWR     JFK     PHL     SLC     BWI     FLL     SAN     MIA     MDW     BNA     TPA 
DAL     IAD     STL     AUS     RDU     SJC 
 136124  127130  119154  115252  104659   98426   93484   89195   83761   82656   76601 
73936   69162   68402   66819   64157   62777 
    PDX     HOU     MSY     MCI     OAK     HNL     SMF     PIT     CLE     IND     CVG 
CMH     SNA     SAT     RSW     JAX     BUR 
  62444   60414   58167   54964   53023   52141   51484   50399   49613   49132   48247 
47900   40802   40725   34472   33939   31907 
    MKE     BDL     OGG     SJU     PBI     OMA     MEM     CHS     BUF     RIC     ORF 
SDF     OKC     ABQ     ONT     RNO     BHM 
  30893   28701   27323   26831   26066   25958   25807   25713   24854   24806   24745 
24332   24130   23736   22613   20022   19478 
    TUS     GRR     BOI     ANC     PVD     TYS     SAV     TUL     ELP     DSM     GSP 
LGB     KOA     GSO     ROC     LIT     SYR 
  19307   19196   19074   18990   18156   18034   17640   17083   17052   16947   16021 
15817   15484   14630   14587   14261   14148 
    XNA     LIH     MSN     DAY     PNS     ALB     FAT     PSP     GEG     MYR     HPN 
PWM     SFB     ICT (Other) 
  14134   14064   14051   13835   12609   12567   12360   12284   12033   11790   11617 
11144   10761   10652  677464 

$CRS_DEP_TIME
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
      1     913    1321    1330    1736    2359 

$DEP_TIME
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
      1     914    1327    1335    1746    2400  130086 

$DEP_DELAY
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
 -82.00   -5.00   -2.00   10.92    7.00 2710.00  130110 

$TAXI_OUT
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
   1.00   11.00   15.00   17.39   20.00  227.00  133977 

$WHEELS_OFF
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
      1     930    1340    1358    1801    2400  133977 

$WHEELS_ON
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
      1    1042    1500    1459    1912    2400  137647 

$TAXI_IN
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
   1.00    4.00    6.00    7.74    9.00  316.00  137647 

$CRS_ARR_TIME
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
      1    1100    1515    1486    1921    2400 

$ARR_TIME
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
      1    1046    1504    1463    1917    2400  137646 

$ARR_DELAY
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
 -99.00  -15.00   -6.00    5.41    7.00 2695.00  153805 

$CRS_ELAPSED_TIME
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
    1.0    90.0   124.0   141.9   171.0   948.0     135 

$ACTUAL_ELAPSED_TIME
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
   15.0    84.0   119.0   136.7   167.0  1604.0  153805 

$AIR_TIME
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
    4.0    60.0    93.0   111.6   141.0  1557.0  153805 

$DISTANCE
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
   31.0   369.0   640.0   800.5  1034.0  5095.0 

$DISTANCE_GROUP
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  1.000   2.000   3.000   3.676   5.000  11.000 
```

Around 150,000 rows have NA values, its not too much compare to more than 7,000,000 rows in total. So that we just remove all NA rows

Hide

```
df3 <- df2[complete.cases(df2),]
```

4. Detail summary with `skim`

Hide

```
# system.time(
#   skim_sum <- mclapply(df3, skim, mc.cores = 6)
# )
skim_sum
```

```
$MONTH
── Data Summary ────────────────────────
                           Values
Name                       X[[i]]
Number of rows             7268232
Number of columns          1
_______________________
Column type frequency:
  numeric                  1
________________________
Group variables            None

── Variable type: numeric ──────────────────────────────────────────────────────────────
─────────────────────────────────────────────────────
  skim_variable n_missing complete_rate  mean    sd    p0   p25   p50   p75  p100 hist
1 data                  0             1  6.60  3.40     1     4     7    10    12 ▇▅▅▅▇

$DAY_OF_MONTH
── Data Summary ────────────────────────
                           Values
Name                       X[[i]]
Number of rows             7268232
Number of columns          1
_______________________
Column type frequency:
  numeric                  1
________________________
Group variables            None

── Variable type: numeric ──────────────────────────────────────────────────────────────
─────────────────────────────────────────────────────
  skim_variable n_missing complete_rate  mean    sd    p0   p25   p50   p75  p100 hist
1 data                  0             1  15.7  8.76     1     8    16    23    31 ▇▇▇▇▆

$DAY_OF_WEEK
── Data Summary ────────────────────────
                           Values
Name                       X[[i]]
Number of rows             7268232
Number of columns          1
_______________________
Column type frequency:
  numeric                  1
________________________
Group variables            None

── Variable type: numeric ──────────────────────────────────────────────────────────────
─────────────────────────────────────────────────────
  skim_variable n_missing complete_rate  mean    sd    p0   p25   p50   p75  p100 hist
1 data                  0             1  3.94  2.00     1     2     4     6     7 ▇▃▅▅▇

$OP_UNIQUE_CARRIER
── Data Summary ────────────────────────
```

```
                           Values
Name                       X[[i]]
Number of rows             7268232
Number of columns          1
_______________________
Column type frequency:
  factor                   1
________________________
Group variables            None

── Variable type: factor ──────────────────────────────────────────────────
────────────────────────────────────────────────────

  skim_variable n_missing complete_rate ordered n_unique top_counts
1 data                  0             1 FALSE         17 WN: 1327562, DL: 988025, AA: 92
4242, OO: 815839

$OP_CARRIER_FL_NUM
── Data Summary ────────────────────────
                           Values
Name                       X[[i]]
Number of rows             7268232
Number of columns          1
_______________________
Column type frequency:
  numeric                  1
________________________
Group variables            None

── Variable type: numeric ─────────────────────────────────────────────────
────────────────────────────────────────────────────

  skim_variable n_missing complete_rate  mean    sd    p0   p25   p50   p75  p100 hist
1 data                  0             1 2549. 1797.     1  1018  2149  3900  7933

$ORIGIN
── Data Summary ────────────────────────
                           Values
Name                       X[[i]]
Number of rows             7268232
Number of columns          1
_______________________
Column type frequency:
  factor                   1
________________________
Group variables            None

── Variable type: factor ──────────────────────────────────────────────────
────────────────────────────────────────────────────

  skim_variable n_missing complete_rate ordered n_unique top_counts
1 data                  0             1 FALSE        360 ATL: 391701, ORD: 328129, DFW:
295645, DEN: 246477

$DEST
── Data Summary ────────────────────────
                           Values
```

```
Name                       X[[i]]
Number of rows             7268232
Number of columns          1
_______________________
Column type frequency:
  factor                   1
________________________
Group variables            None

── Variable type: factor ──────────────────────────────────────────────────────────────
──────────────────────────────────────────────────────

  skim_variable n_missing complete_rate ordered n_unique top_counts
1 data                  0             1 FALSE        360 ATL: 391731, ORD: 327659, DFW:
294798, DEN: 246132

$CRS_DEP_TIME
── Data Summary ────────────────────────
                           Values
Name                       X[[i]]
Number of rows             7268232
Number of columns          1
_______________________
Column type frequency:
  numeric                  1
________________________
Group variables            None

── Variable type: numeric ─────────────────────────────────────────────────────────────
──────────────────────────────────────────────────────

  skim_variable n_missing complete_rate  mean    sd    p0   p25   p50   p75  p100 hist
1 data                  0             1 1329.  493.     1   912  1320  1735  2359 ▁▇▇▇▆

$DEP_TIME
── Data Summary ────────────────────────
                           Values
Name                       X[[i]]
Number of rows             7268232
Number of columns          1
_______________________
Column type frequency:
  numeric                  1
________________________
Group variables            None

── Variable type: numeric ─────────────────────────────────────────────────────────────
──────────────────────────────────────────────────────

  skim_variable n_missing complete_rate  mean    sd    p0   p25   p50   p75  p100 hist
1 data                  0             1 1334.  507.     1   914  1327  1746  2400 ▁▇▇▇▆

$DEP_DELAY
── Data Summary ────────────────────────
                           Values
Name                       X[[i]]
Number of rows             7268232
```

```
Number of columns          1
_______________________
Column type frequency:
  numeric                  1
________________________
Group variables            None

── Variable type: numeric ──────────────────────────────────────────
  skim_variable n_missing complete_rate  mean    sd    p0   p25   p50   p75  p100 hist
1 data                  0             1  10.8  48.8   -82    -5    -2     7  2710 ▇▁▁▁▁

$TAXI_OUT
── Data Summary ────────────────────────
                           Values
Name                       X[[i]]
Number of rows             7268232
Number of columns          1
_______________________
Column type frequency:
  numeric                  1
________________________
Group variables            None

── Variable type: numeric ──────────────────────────────────────────
  skim_variable n_missing complete_rate  mean    sd    p0   p25   p50   p75  p100 hist
1 data                  0             1  17.4  9.99     1    11    15    20   227 ▇▁▁▁▁

$WHEELS_OFF
── Data Summary ────────────────────────
                           Values
Name                       X[[i]]
Number of rows             7268232
Number of columns          1
_______________________
Column type frequency:
  numeric                  1
________________________
Group variables            None

── Variable type: numeric ──────────────────────────────────────────
  skim_variable n_missing complete_rate  mean    sd    p0   p25   p50   p75  p100 hist
1 data                  0             1 1358.  509.     1   930  1340  1801  2400 ▁▇▇▇▆

$WHEELS_ON
── Data Summary ────────────────────────
                           Values
Name                       X[[i]]
Number of rows             7268232
Number of columns          1
_______________________
Column type frequency:
```

```
  numeric                  1
────────────────────────
Group variables            None

── Variable type: numeric ──────────────────────────────────────────────────────
─────────────────────────────────────────────────────
  skim_variable n_missing complete_rate  mean    sd    p0   p25   p50   p75  p100 hist
1 data                  0             1 1459.  538.     1  1042  1500  1912  2400 ▁▃▇▇▇

$TAXI_IN
── Data Summary ────────────────────────
                           Values
Name                       X[[i]]
Number of rows             7268232
Number of columns          1
───────────────────────
Column type frequency:
  numeric                  1
────────────────────────
Group variables            None

── Variable type: numeric ──────────────────────────────────────────────────────
─────────────────────────────────────────────────────
  skim_variable n_missing complete_rate  mean    sd    p0   p25   p50   p75  p100 hist
1 data                  0             1  7.74  6.18     1     4     6     9   316 ▇▁▁▁▁

$CRS_ARR_TIME
── Data Summary ────────────────────────
                           Values
Name                       X[[i]]
Number of rows             7268232
Number of columns          1
───────────────────────
Column type frequency:
  numeric                  1
────────────────────────
Group variables            None

── Variable type: numeric ──────────────────────────────────────────────────────
─────────────────────────────────────────────────────
  skim_variable n_missing complete_rate  mean    sd    p0   p25   p50   p75  p100 hist
1 data                  0             1 1485.  521.     1  1100  1514  1920  2400 ▁▃▇▇▇

$ARR_TIME
── Data Summary ────────────────────────
                           Values
Name                       X[[i]]
Number of rows             7268232
Number of columns          1
───────────────────────
Column type frequency:
  numeric                  1
────────────────────────
Group variables            None
```

```
── Variable type: numeric ──────────────────────────────────────────────────────────
────────────────────────────────────────────────────
  skim_variable n_missing complete_rate  mean    sd    p0   p25   p50   p75  p100 hist
1 data                  0             1 1463.  542.     1  1046  1503  1917  2400

$ARR_DELAY
── Data Summary ────────────────────────
                           Values
Name                       X[[i]]
Number of rows             7268232
Number of columns          1
_______________________
Column type frequency:
  numeric                  1
________________________
Group variables            None

── Variable type: numeric ──────────────────────────────────────────────────────────
────────────────────────────────────────────────────
  skim_variable n_missing complete_rate  mean    sd    p0   p25   p50   p75  p100 hist
1 data                  0             1  5.41  51.1   -99   -15    -6     7  2695

$CRS_ELAPSED_TIME
── Data Summary ────────────────────────
                           Values
Name                       X[[i]]
Number of rows             7268232
Number of columns          1
_______________________
Column type frequency:
  numeric                  1
________________________
Group variables            None

── Variable type: numeric ──────────────────────────────────────────────────────────
────────────────────────────────────────────────────
  skim_variable n_missing complete_rate  mean    sd    p0   p25   p50   p75  p100 hist
1 data                  0             1  142.  72.5    17    90   124   171   813

$ACTUAL_ELAPSED_TIME
── Data Summary ────────────────────────
                           Values
Name                       X[[i]]
Number of rows             7268232
Number of columns          1
_______________________
Column type frequency:
  numeric                  1
________________________
Group variables            None

── Variable type: numeric ──────────────────────────────────────────────────────────
────────────────────────────────────────────────────
```

```
  skim_variable n_missing complete_rate  mean    sd    p0   p25   p50   p75  p100 hist
1 data                  0             1  137.  72.6    15    84   119   167  1604

$AIR_TIME
── Data Summary ────────────────────────
                           Values
Name                       X[[i]]
Number of rows             7268232
Number of columns          1
_______________________
Column type frequency:
  numeric                  1
________________________
Group variables            None

── Variable type: numeric ──────────────────────────────────────────────────
────────────────────────────────────────────────────
  skim_variable n_missing complete_rate  mean    sd    p0   p25   p50   p75  p100 hist
1 data                  0             1  112.  70.6     4    60    93   141  1557

$DISTANCE
── Data Summary ────────────────────────
                           Values
Name                       X[[i]]
Number of rows             7268232
Number of columns          1
_______________________
Column type frequency:
  numeric                  1
________________________
Group variables            None

── Variable type: numeric ──────────────────────────────────────────────────
────────────────────────────────────────────────────
  skim_variable n_missing complete_rate  mean    sd    p0   p25   p50   p75  p100 hist
1 data                  0             1  803.  594.    31   369   641  1035  5095

$DISTANCE_GROUP
── Data Summary ────────────────────────
                           Values
Name                       X[[i]]
Number of rows             7268232
Number of columns          1
_______________________
Column type frequency:
  numeric                  1
________________________
Group variables            None

── Variable type: numeric ──────────────────────────────────────────────────
────────────────────────────────────────────────────
  skim_variable n_missing complete_rate  mean    sd    p0   p25   p50   p75  p100 hist
1 data                  0             1  3.68  2.34     1     2     3     5    11
```

We got some useful information. as we can see: `variable type` is messy, `Standard Deviation` is not clean, `quick hist plot` show some tail plot or head plot, and `mean` variable is not balance in specific range in each features.

5. Next we will figure out correlation of all features

Hide

```
# pick only numeric columns for corr plot

foo <- unlist(lapply(df3,is.factor))
df3_numeric <- df3[!foo]

df3_numeric
```

	M... <int>	DAY_OF_M... <int>	DAY_OF_... <int>	OP_CARRIER_FL_... <int>	CRS_DEP_T... <int>	DEP_T... <int>	DEP_DE... <dbl>
1	10	8	2	5120	1835	1830	-5
2	10	9	3	5120	1835	1828	-7
3	10	10	4	5120	1835	1838	3
4	10	11	5	5120	1835	1833	-2
5	10	12	6	5120	1835	1827	-8
6	10	13	7	5120	1835	1834	-1
7	10	14	1	5120	1835	1831	-4
8	10	15	2	5120	1835	2058	143
9	10	16	3	5120	1835	1925	50
10	10	17	4	5120	1835	1857	22

1-10 of 7,268,232 rows | 1-9 of 19 columns Previous **1** 2 3 4 5 6 ... 78 Next

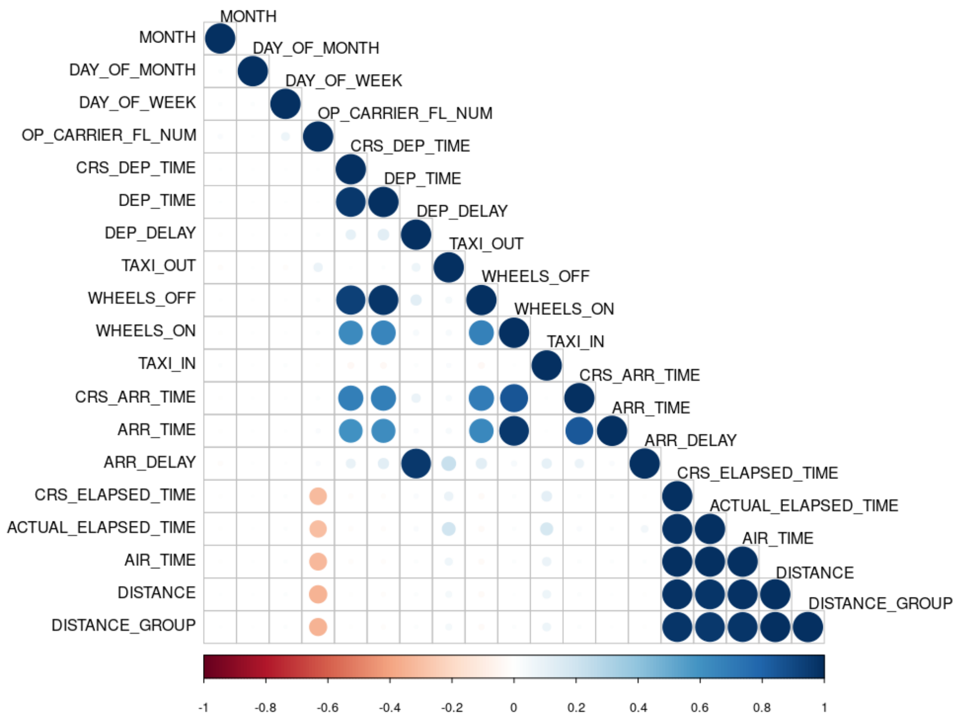
Hide

```
system.time(
cormat <- round(cor(df3_numeric), 2)
)
corr_df <- data.frame(cormat)

corr_df
```

Plot Corr

Hide

```
# plot correlation
## `tl.srt` is to rotate text
corrplot(cormat, type = "lower", tl.srt = 360, tl.col = "black")
```



show all correlation > 0.8

Hide

```
list_corr_above_0.8 <- list()
for (i in names(corr_df)) {
  a <- filter(corr_df[i], corr_df[i] > 0.8)
  list_corr_above_0.8[[i]] <- a
}

list_corr_above_0.8
```

```
$MONTH
```

	MONTH <dbl>
MONTH	1

1 row

```
$DAY_OF_MONTH
```

	DAY_OF_MONTH <dbl>
DAY_OF_MONTH	1

1 row

```
$DAY_OF_WEEK
```

	DAY_OF_WEEK <dbl>
DAY_OF_WEEK	1

1 row

```
$OP_CARRIER_FL_NUM
```

	OP_CARRIER_FL_NUM <dbl>
OP_CARRIER_FL_NUM	1

1 row

```
$CRS_DEP_TIME
```

	CRS_DEP_TIME <dbl>
CRS_DEP_TIME	1.00
DEP_TIME	0.96
WHEELS_OFF	0.93

3 rows

```
$DEP_TIME
```

	DEP_TIME <dbl>
CRS_DEP_TIME	0.96
DEP_TIME	1.00
WHEELS_OFF	0.97

3 rows

```
$DEP_DELAY
```

	DEP_DELAY <dbl>
DEP_DELAY	1.00
ARR_DELAY	0.96

2 rows

```
$TAXI_OUT
```

	TAXI_OUT <dbl>
TAXI_OUT	1

1 row

```
$WHEELS_OFF
```

	WHEELS_OFF <dbl>
CRS_DEP_TIME	0.93
DEP_TIME	0.97
WHEELS_OFF	1.00

3 rows

```
$WHEELS_ON
```

	WHEELS_ON <dbl>
WHEELS_ON	1.00
CRS_ARR_TIME	0.85
ARR_TIME	0.96

3 rows

```
$TAXI_IN
```

	TAXI_IN <dbl>
TAXI_IN	1

1 row

```
$CRS_ARR_TIME
```

	CRS_ARR_TIME <dbl>
WHEELS_ON	0.85
CRS_ARR_TIME	1.00
ARR_TIME	0.84

3 rows

```
$ARR_TIME
```

	ARR_TIME <dbl>
WHEELS_ON	0.96
CRS_ARR_TIME	0.84
ARR_TIME	1.00

3 rows

```
$ARR_DELAY
```

	ARR_DELAY <dbl>
DEP_DELAY	0.96
ARR_DELAY	1.00

2 rows

```
$CRS_ELAPSED_TIME
```

	CRS_ELAPSED_TIME <dbl>
CRS_ELAPSED_TIME	1.00
ACTUAL_ELAPSED_TIME	0.98
AIR_TIME	0.99
DISTANCE	0.98
DISTANCE_GROUP	0.97

5 rows

```
$ACTUAL_ELAPSED_TIME
```

	ACTUAL_ELAPSED_TIME <dbl>
CRS_ELAPSED_TIME	0.98
ACTUAL_ELAPSED_TIME	1.00
AIR_TIME	0.99
DISTANCE	0.97
DISTANCE_GROUP	0.96

5 rows

```
$AIR_TIME
```

	AIR_TIME <dbl>
CRS_ELAPSED_TIME	0.99
ACTUAL_ELAPSED_TIME	0.99
AIR_TIME	1.00
DISTANCE	0.98
DISTANCE_GROUP	0.97

5 rows

```
$DISTANCE
```

	DISTANCE <dbl>
CRS_ELAPSED_TIME	0.98
ACTUAL_ELAPSED_TIME	0.97
AIR_TIME	0.98
DISTANCE	1.00
DISTANCE_GROUP	0.99

5 rows

```
$DISTANCE_GROUP
```

	DISTANCE_GROUP <dbl>
CRS_ELAPSED_TIME	0.97
ACTUAL_ELAPSED_TIME	0.96
AIR_TIME	0.97
DISTANCE	0.99
DISTANCE_GROUP	1.00

5 rows

```
NA
```

- We can see that `CRS_ELAPSED_TIME` `ACTUAL_ELAPSED_TIME` `AIR_TIME` `DISTANCE` `DISTANCE_GROUP` have high corr each other, so we will just keep 1 of it

- `ACTUAL_ELAPSED_TIME` have highest corr with ARR_DELAY, so we will keep this and remove all

- `DEP_TIME` and `WHEELS_OFF` have same highest corr, so we can choose, keep `DEP_TIME`

- We will remove DEP_DELAY too

- We still have `WHEELS_ON` high corr with `ARR_TIME` and `CRS_ARR_TIME`, remove `ARR_TIME`, `CRS_ARR_TIME`

Hide

```
df3_numeric_reduce <- subset(df3_numeric, select = -c(CRS_ELAPSED_TIME, AIR_TIME, DISTAN
CE, DISTANCE_GROUP, CRS_DEP_TIME, WHEELS_OFF, DEP_DELAY, ARR_TIME, CRS_ARR_TIME))

df3_numeric_reduce
```

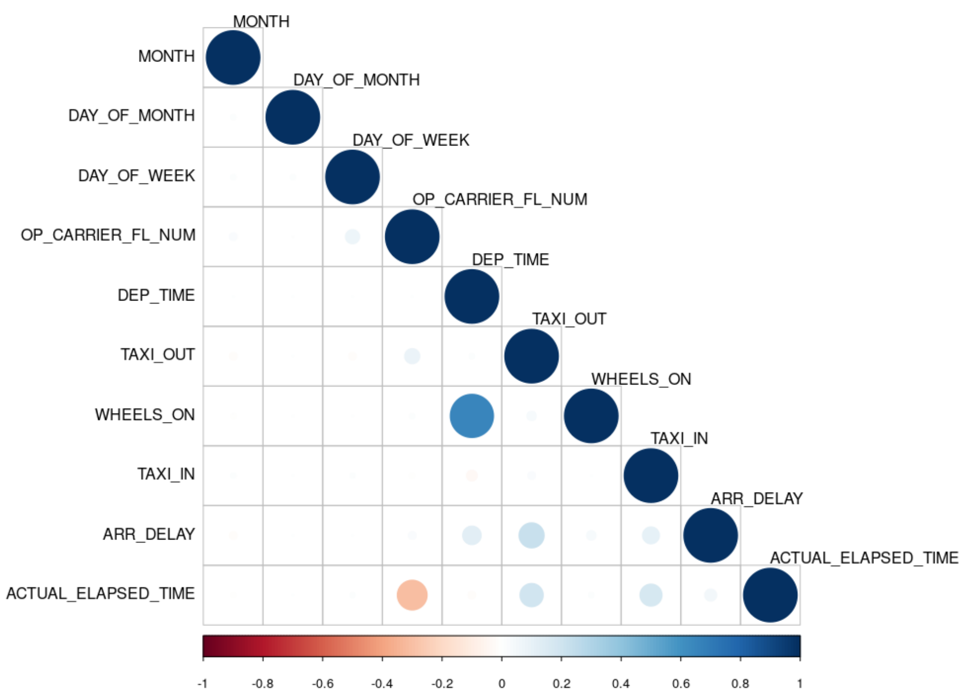
	M... <int>	DAY_OF_M... <int>	DAY_OF_... <int>	OP_CARRIER_FL_... <int>	DEP_T... <int>	TAXI_... <dbl>	WHEEL... <int>	TAXI_ <db
1	10	8	2	5120	1830	33	2001	
2	10	9	3	5120	1828	25	1951	
3	10	10	4	5120	1838	39	2018	
4	10	11	5	5120	1833	26	1958	
5	10	12	6	5120	1827	15	1944	
6	10	13	7	5120	1834	37	2015	
7	10	14	1	5120	1831	19	1948	
8	10	15	2	5120	2058	16	2213	
9	10	16	3	5120	1925	27	2051	
10	10	17	4	5120	1857	27	2022	

1-10 of 7,268,232 rows | 1-9 of 10 columns Previous **1** 2 3 4 5 6 ... 78 Next

Plot again

Hide

```
cormat_reduce <- round(cor(df3_numeric_reduce), 2)
corrplot(cormat_reduce, type = "lower", tl.srt = 360, tl.col = "black")
```



Ok, so we can see that no high correlation occur in out data now.

We can now concat to original data

Hide

```
df4 <- df3[,sapply(df3, is.factor)]
df4 <- cbind(df4, df3_numeric_reduce)

df4
```

	OP_UNIQUE_CAR... <fctr>	ORI... <fctr>	D... <fctr>	M... <int>	DAY_OF_M... <int>	DAY_OF_... <int>	OP_CARRIER_FL_... <int>	DE
1	OH	PHL	CAK	10	8	2	5120	
2	OH	PHL	CAK	10	9	3	5120	
3	OH	PHL	CAK	10	10	4	5120	
4	OH	PHL	CAK	10	11	5	5120	
5	OH	PHL	CAK	10	12	6	5120	
6	OH	PHL	CAK	10	13	7	5120	
7	OH	PHL	CAK	10	14	1	5120	
8	OH	PHL	CAK	10	15	2	5120	
9	OH	PHL	CAK	10	16	3	5120	
10	OH	PHL	CAK	10	17	4	5120	

1-10 of 7,268,232 rows | 1-9 of 13 columns Previous **1** 2 3 4 5 6 ... 78 Next

6. Save file

Hide

```
write.csv(df4, file = "/home/apolong72/ds/r/data/airline_2/all.csv", row.names = FALSE)
```

—————RELOAD ALL (retrieve RAM purpose) —————

Since my computer dont have too much ram, I have to reload frequenly to have some space

Hide

```
# remove some junk
rm(list = ls())
rm()
gc()
```

```
          used  (Mb) gc trigger   (Mb)  max used   (Mb)
Ncells 2782024 148.6    8039128  429.4  10652849  569.0
Vcells 9802570  74.8  376957098 2876.0 469868416 3584.9
```

Read data again

Hide

```
# df <- read.csv("/home/apolong72/ds/r/data/airline_2/all.csv")
# # get copy of data
# df_copy <- df

df
```

OP_UNIQUE_CAR...	ORI...	D...	M...	DAY_OF_M...	DAY_OF_...	OP_CARRIER_FL_...	DEP_T
<fctr>	<fctr>	<fctr>	<int>	<int>	<int>	<int>	<ir
OH	PHL	CAK	10	8	2	5120	18
OH	PHL	CAK	10	9	3	5120	18
OH	PHL	CAK	10	10	4	5120	18
OH	PHL	CAK	10	11	5	5120	18
OH	PHL	CAK	10	12	6	5120	18
OH	PHL	CAK	10	13	7	5120	18
OH	PHL	CAK	10	14	1	5120	18
OH	PHL	CAK	10	15	2	5120	20
OH	PHL	CAK	10	16	3	5120	19
OH	PHL	CAK	10	17	4	5120	18

1-10 of 7,268,232 rows | 1-9 of 13 columns Previous **1** 2 3 4 5 6 ... 78 Next

7. Write some useful functions

Hide

```
# write some function to plot histogram and bar chart

## hist plot
plot.hist <- function (df, name) {
  ggplot( data=df, aes(x=df[, name])) +
    geom_histogram(fill="skyblue", alpha=0.8, bins = 100) +
    labs(title="Histogram", subtitle=name, y="Count", x=name, caption="by Quan") +
    theme_minimal() +
    theme(text=element_text(size=14,  family="Arial", face = "bold"))
}

## box plot
plot.box <- function (df, name) {
  ggplot( data=df, aes(x=df[, name])) +
    geom_boxplot(fill="skyblue", alpha=0.8,) +
    labs(title="Box Plot", subtitle=name, y="Count", x=name, caption="by Quan") +
    theme_minimal() +
    theme(text=element_text(size=14,  family="Arial", face = "bold"))
}

## bar plot
plot.bar <- function(df) {
  ggplot( data=df, aes(x=ind, y=values)) +
    geom_bar(fill="skyblue",stat="identity") +
    labs(title="Bar Plot", subtitle="Most Frequence", y="Count", x="Name", caption="by Q
uan") +
    theme_minimal() +
    theme(text=element_text(size=14,  family="Arial", face = "bold"))
}


# create dist list
dist.list <- function(df) {
  #
  list_temp <- list()
  name.df <- names(df)
  j <- 1
  #
  for (i in df) {
    list_temp[[name.df[j]]] <- summary(as.factor(i))
    j <- j + 1
  }

  return(list_temp)
}
```

8. Next we will get more insight of this data

Create distinct values list for each feature

Hide

```
dist_list <- dist.list(df)
dist_list_pretty <- mclapply(dist_list, function(x) stack(x), mc.cores = 6)

dist_list_pretty
```

```
$OP_UNIQUE_CARRIER
```

values <int>	ind <fctr>
252348	9E
924242	AA
261015	AS
292724	B6
988025	DL
128180	EV
133061	F9
104443	G4
83689	HA
314658	MQ

1-10 of 17 rows Previous **1** 2 Next

```
$ORIGIN
```

values <int>	ind <fctr>
391701	ATL
328129	ORD
295645	DFW
246477	DEN
231325	CLT
216481	LAX
176601	IAH
172578	PHX
166750	SFO

values	ind
<int>	<fctr>
166297	LGA

1-10 of 100 rows Previous 1 2 3 4 5 6 ... 10 Next

```
$DEST
```

values	ind
<int>	<fctr>
391731	ATL
327659	ORD
294798	DFW
246132	DEN
230776	CLT
216677	LAX
176285	IAH
172460	PHX
166834	SFO
165823	LGA

1-10 of 100 rows Previous 1 2 3 4 5 6 ... 10 Next

```
$MONTH
```

values	ind
<int>	<fctr>
565963	1
516314	2
618505	3
596020	4
621339	5
620887	6
643781	7
645351	8

values	ind
<int>	<fctr>
594716	9
629637	10

1-10 of 12 rows Previous **1** 2 Next

```
$DAY_OF_MONTH
```

values	ind
<int>	<fctr>
236456	1
232030	2
237234	3
237920	4
235399	5
240084	6
238684	7
244481	8
236163	9
242425	10

1-10 of 31 rows Previous **1** 2 3 4 Next

```
$DAY_OF_WEEK
```

values	ind
<int>	<fctr>
1083750	1
1056064	2
1048502	3
1070661	4
1085374	5
889963	6
1033918	7

7 rows

$OP_CARRIER_FL_NUM

values <int>	ind <fctr>
3377	55
3370	403
3364	511
3317	546
3248	347
3191	352
3140	64
3122	478
3112	419
3023	676

1-10 of 100 rows Previous 1 2 3 4 5 6 ... 10 Next

$DEP_TIME

values <int>	ind <fctr>
19885	555
18882	557
18612	556
17828	558
16230	559
16135	554
15183	655
14963	600
14321	656
14098	657

1-10 of 100 rows Previous 1 2 3 4 5 6 ... 10 Next

```
$TAXI_OUT
```

values <int>	ind <fctr>
556821	12
541719	11
541223	13
503899	14
484105	10
456900	15
404107	16
382349	9
353190	17
307455	18

1-10 of 100 rows Previous **1** 2 3 4 5 6 ... 10 Next

```
$WHEELS_ON
```

values <int>	ind <fctr>
7852	1857
7825	1901
7823	1104
7818	1849
7816	1853
7796	1855
7791	1908
7783	1850
7773	1854
7768	1852

1-10 of 100 rows Previous **1** 2 3 4 5 6 ... 10 Next

```
$TAXI_IN
```

values <int>	ind <fctr>
1082525	4
1029153	5
850636	6
770088	3
685136	7
515654	8
402559	9
316963	10
249061	11
238771	2

1-10 of 100 rows Previous **1** 2 3 4 5 6 ... 10 Next

```
$ARR_DELAY
```

values <int>	ind <fctr>
217492	-11
216891	-10
216803	-12
214404	-9
213399	-13
208614	-8
207501	-14
200414	-7
199345	-15
191303	-6

1-10 of 100 rows Previous **1** 2 3 4 5 6 ... 10 Next

```
$ACTUAL_ELAPSED_TIME
```

values	ind
<int>	<fctr>
60983	79
60795	80
60609	81
60527	82
60351	83
60349	78
60291	77
59505	84
59504	76
58842	85

1-10 of 100 rows Previous **1** 2 3 4 5 6 ... 10 Next

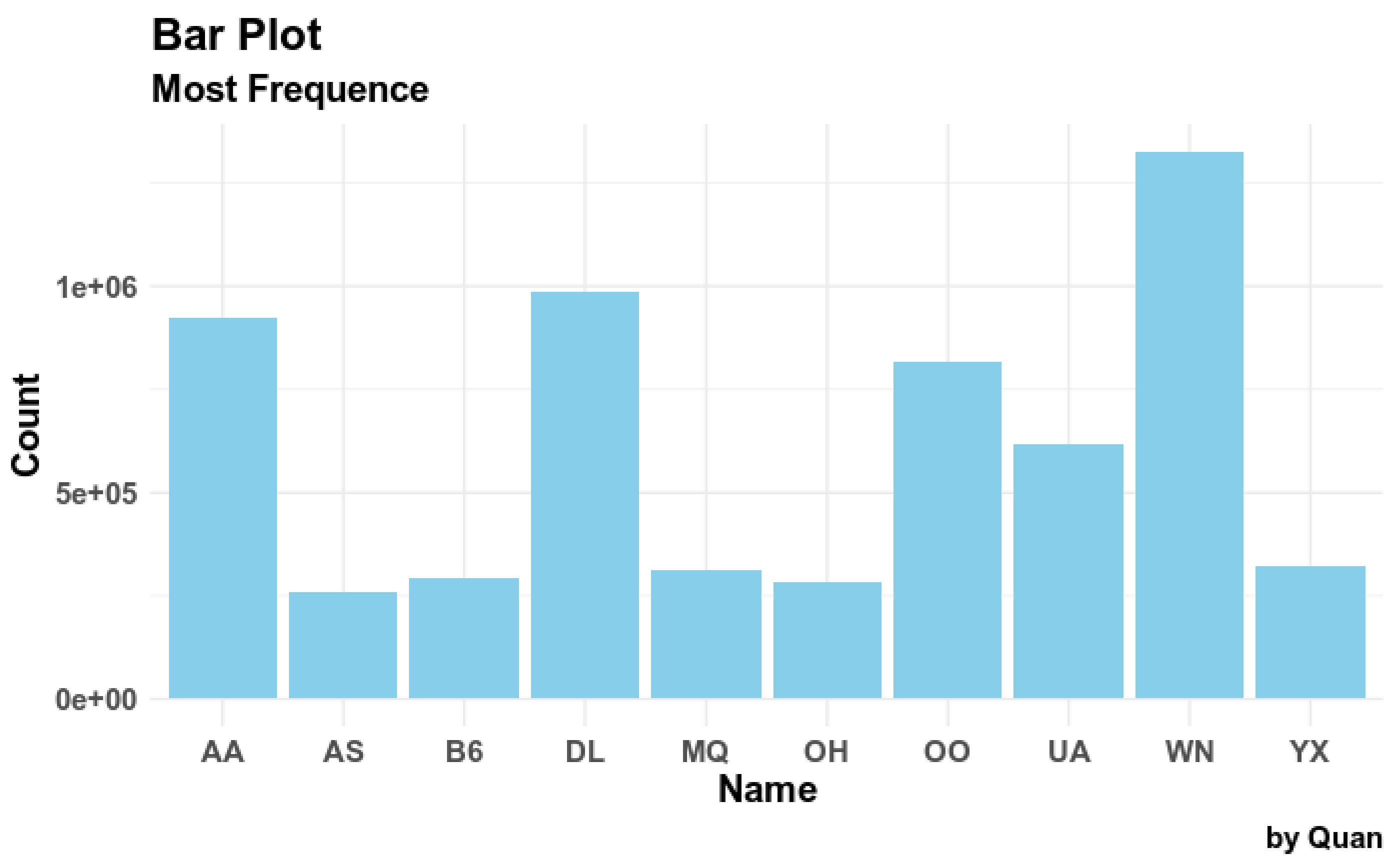
```
NA
```

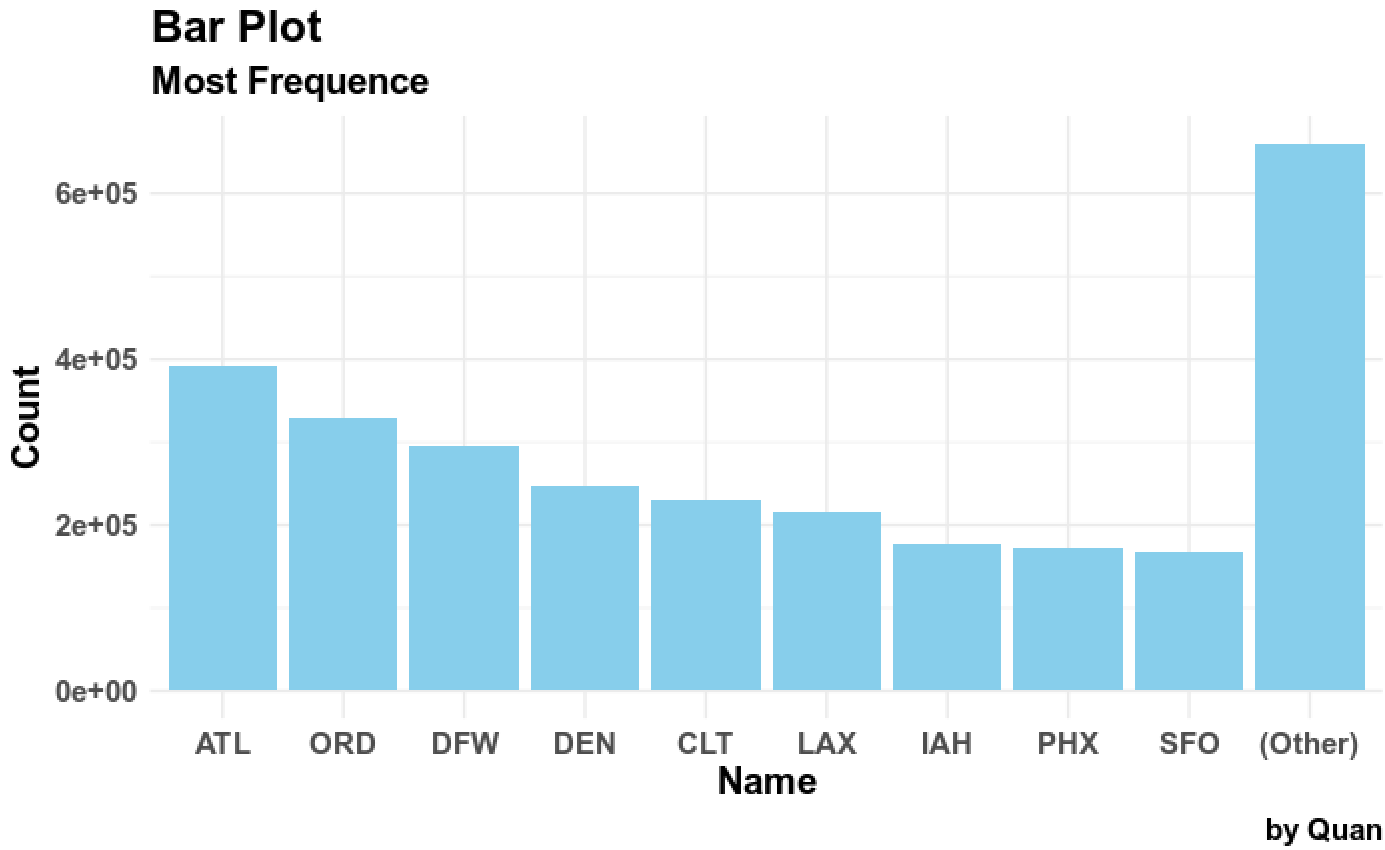
Use bar plot to show most 10 distinct value each features

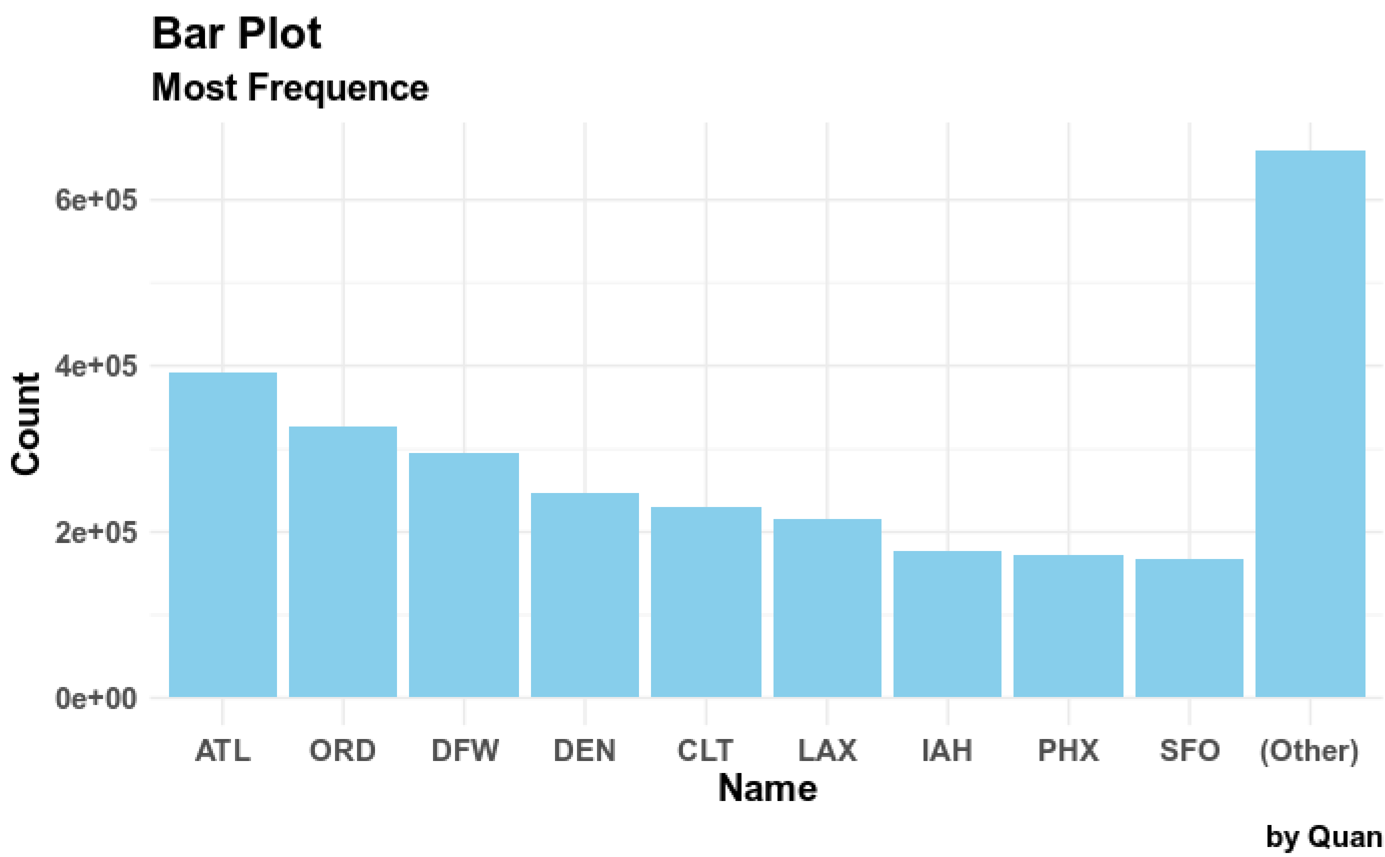
Hide

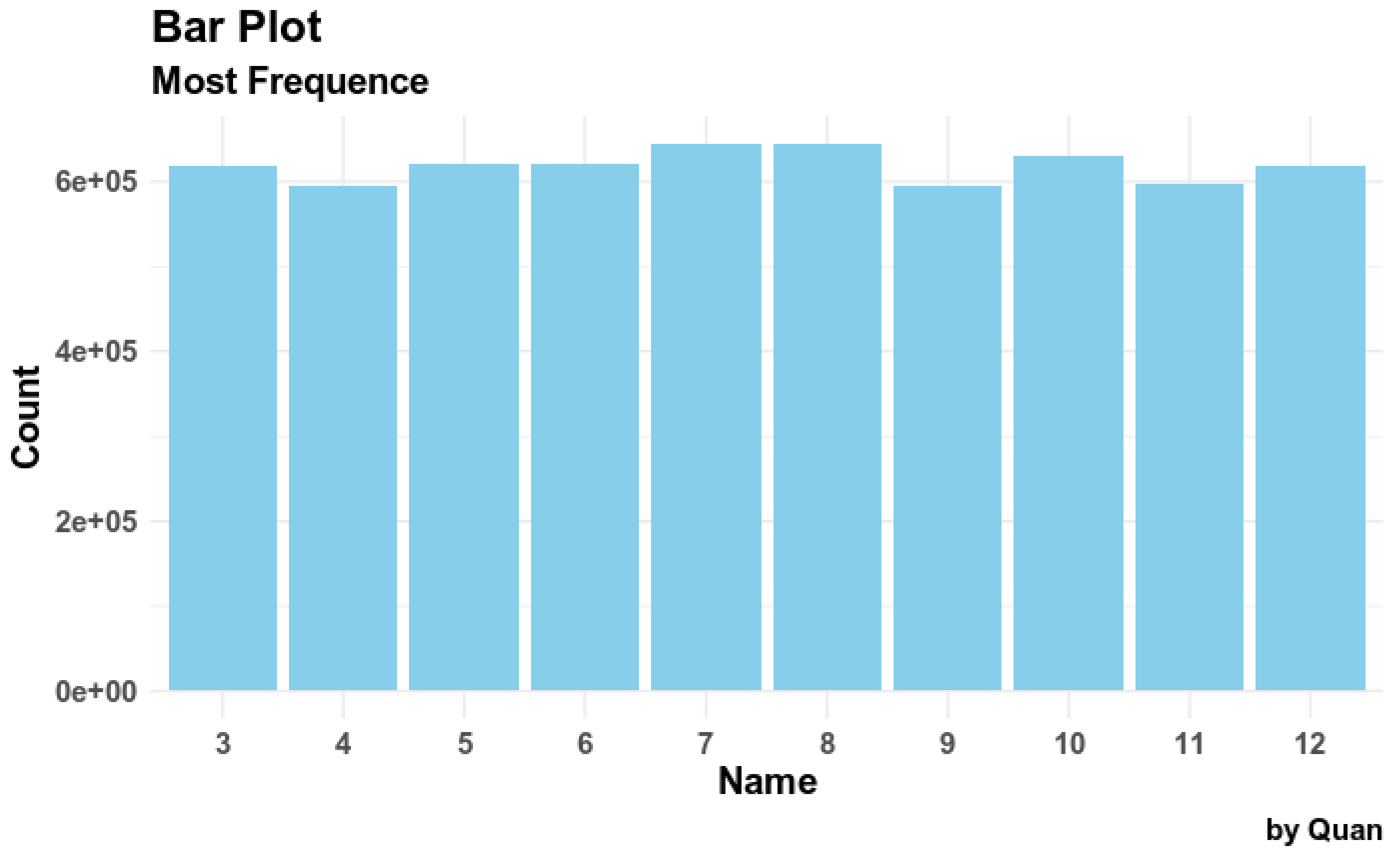
```
# # plot top 10 most frequence values in each feature
# plot_top_10_frequence <- mclapply(dist_list_pretty, function(x) {
#   # head(order(x$values, decreasing = T), 10) : sort by values, pick top 10 frequence
#   temp <- head(order(x$values, decreasing = T), 10)
#   print(plot.bar(x[temp, ]))
# }, mc.cores = 6)

plot_top_10_frequence
```

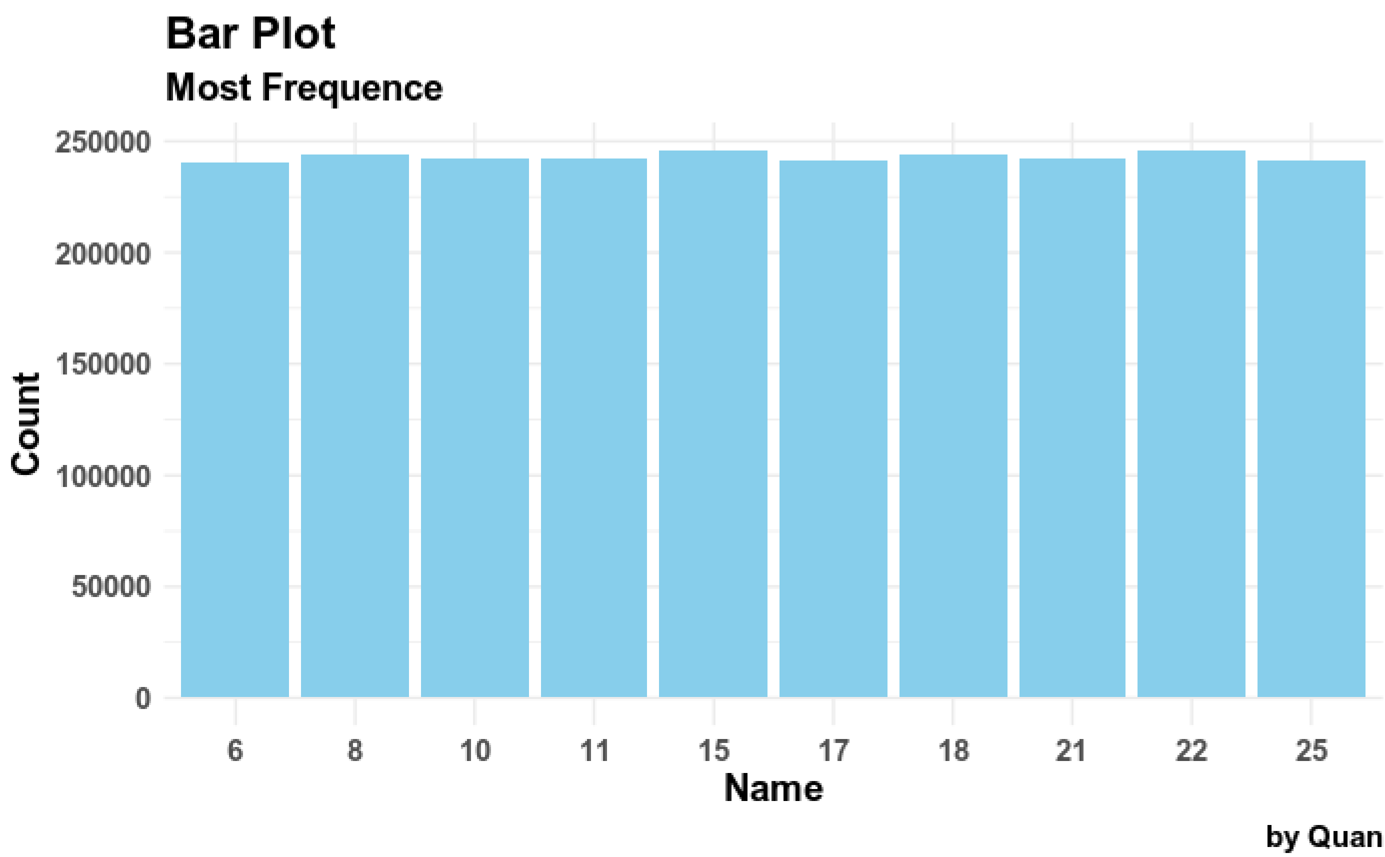
```
$OP_UNIQUE_CARRIER

$ORIGIN

$DEST

$MONTH

$DAY_OF_MONTH

$DAY_OF_WEEK

$OP_CARRIER_FL_NUM

$DEP_TIME

$TAXI_OUT

$WHEELS_ON

$TAXI_IN

$ARR_DELAY

$ACTUAL_ELAPSED_TIME
```

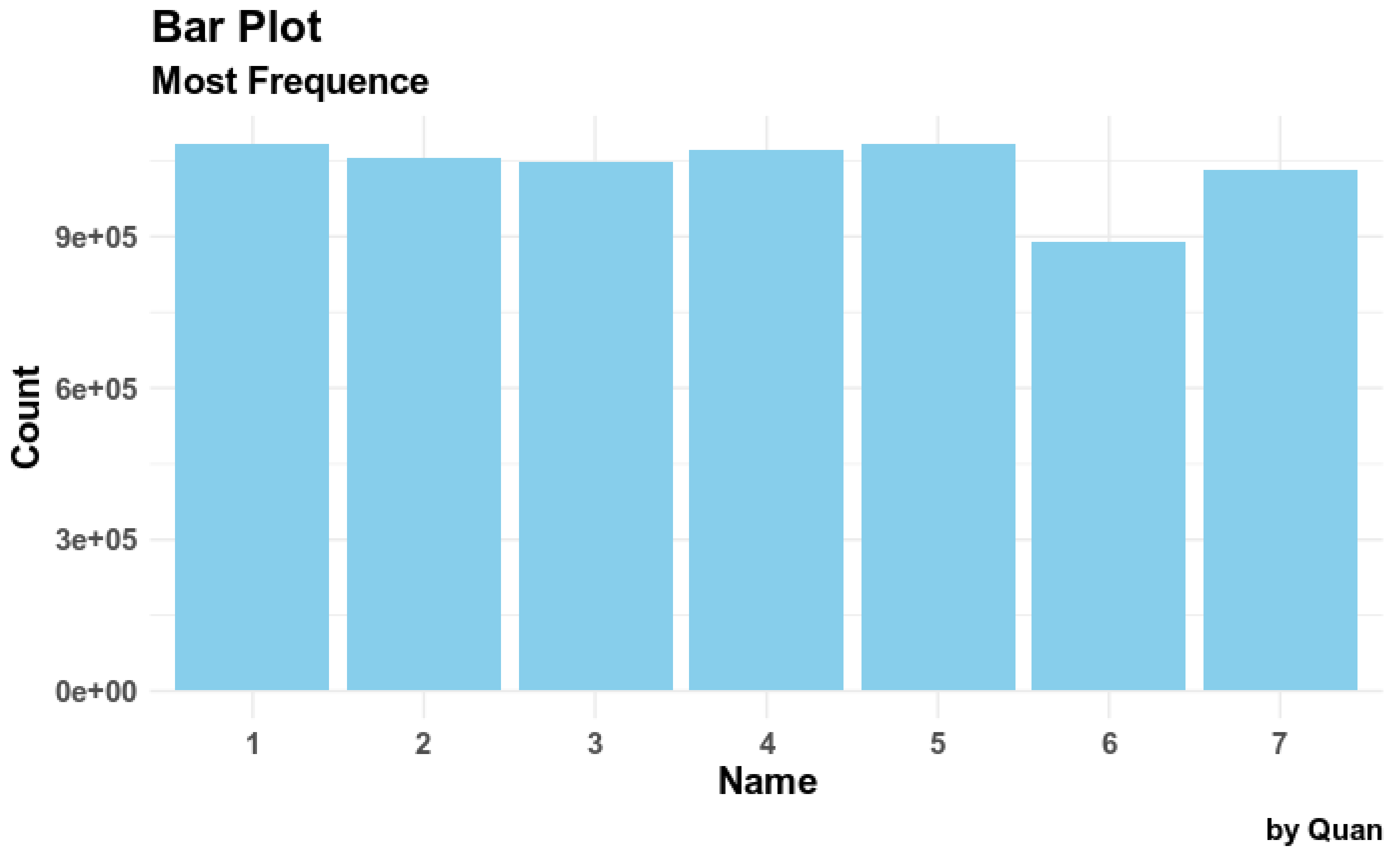


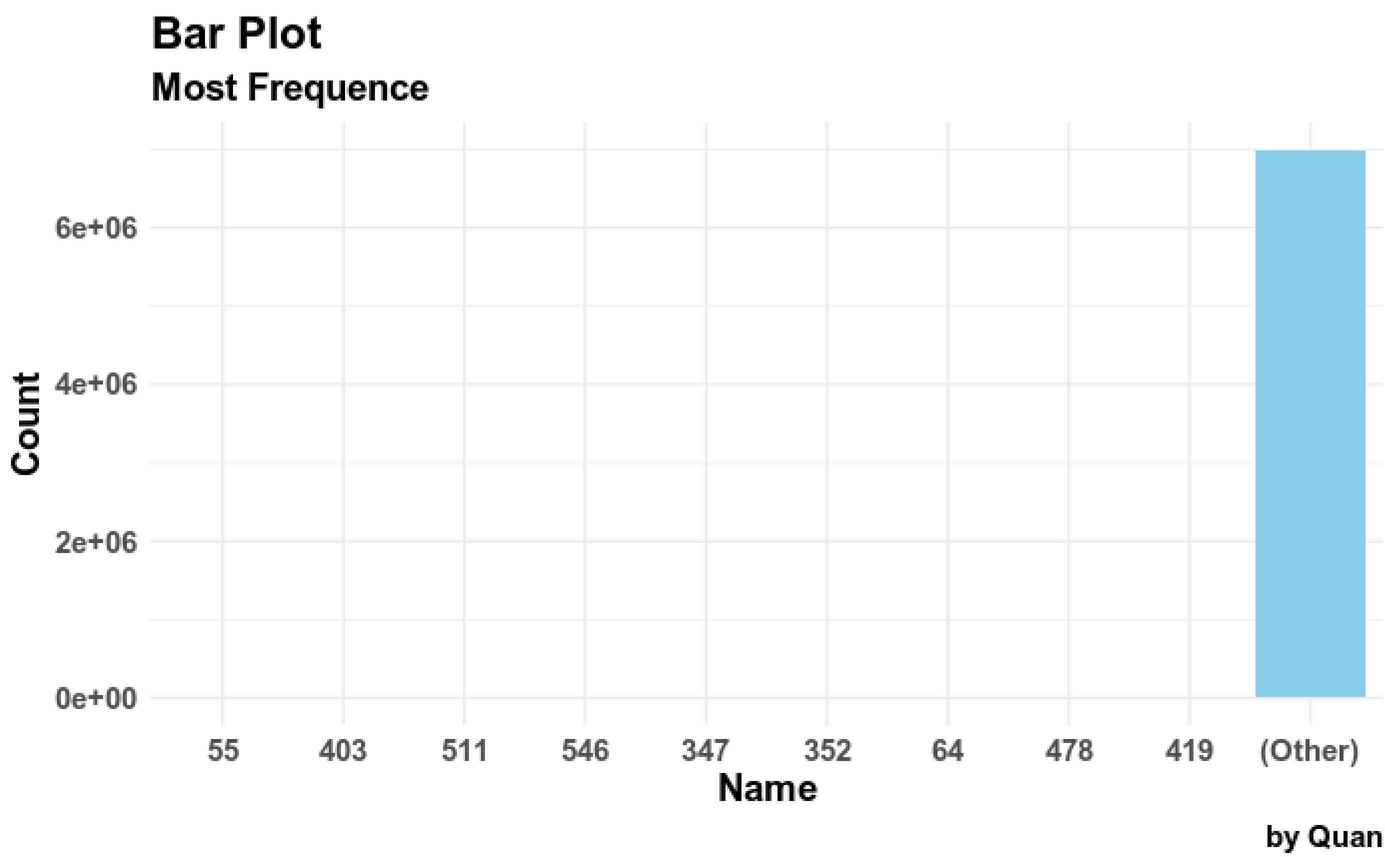

Bar Plot
Most Frequence
Count
6e+05
4e+05
2e+05
0e+00
ATL
ORD
DFW
DEN
CLT
LAX
IAH
PHX
SFO
(Other)
Name
by Quan

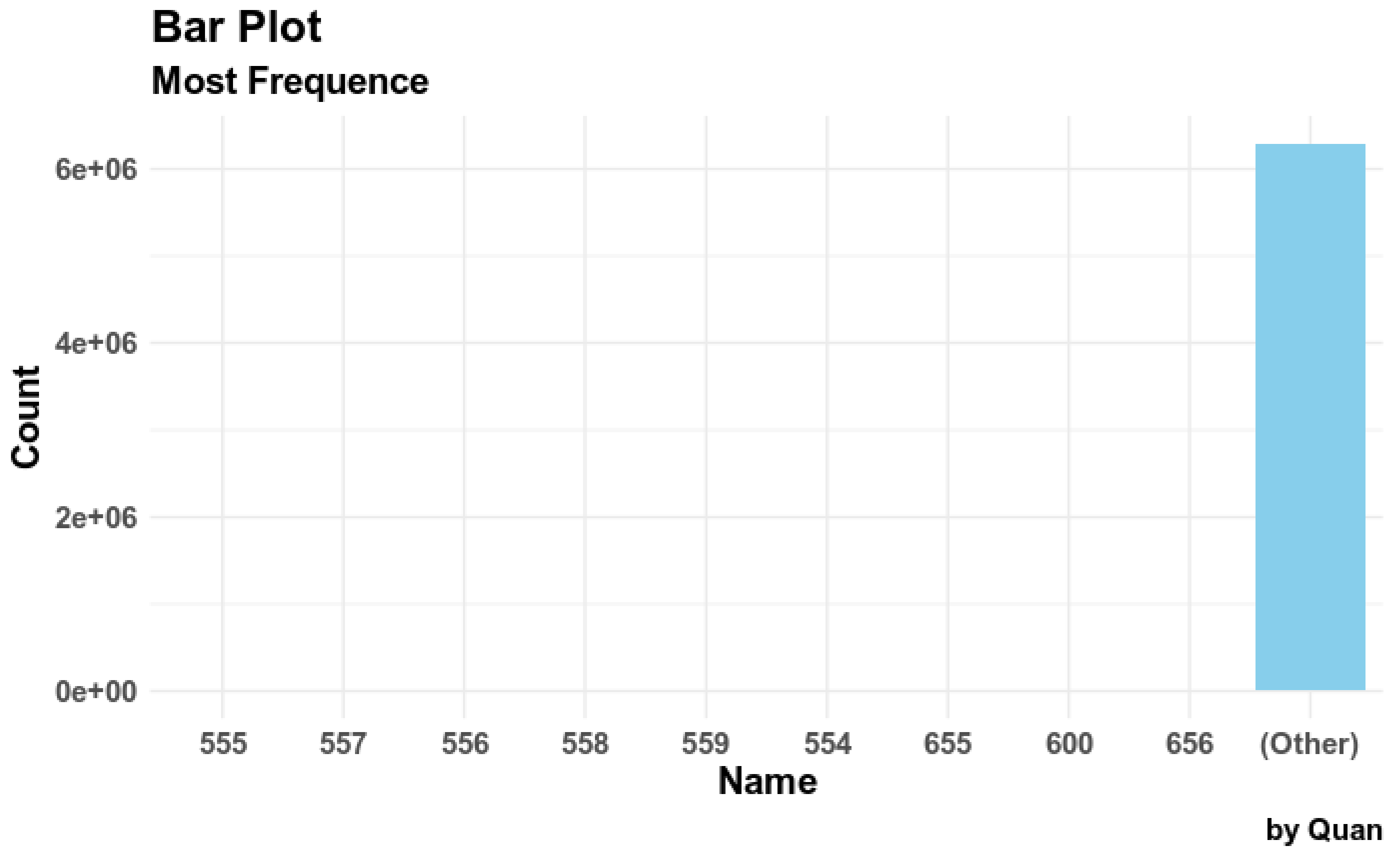

Bar Plot
Most Frequence
Count
6e+05
4e+05
2e+05
0e+00
ATL
ORD
DFW
DEN
CLT
LAX
IAH
PHX
SFO
(Other)
Name
by Quan

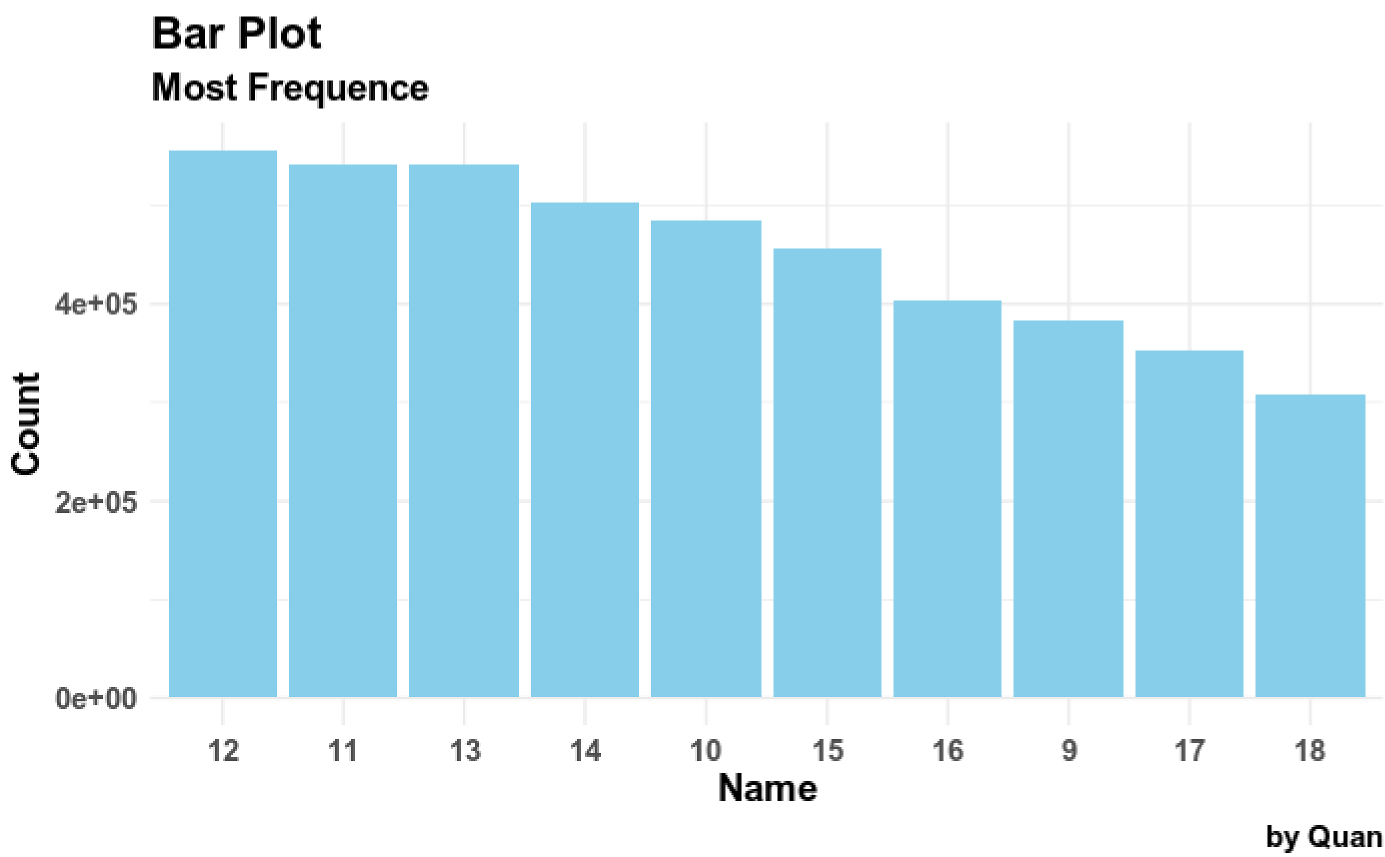

Bar Plot
Most Frequence
6e+05
4e+05
2e+05
0e+00
Count
3
4
5
6
7
8
9
10
11
12
Name
by Quan

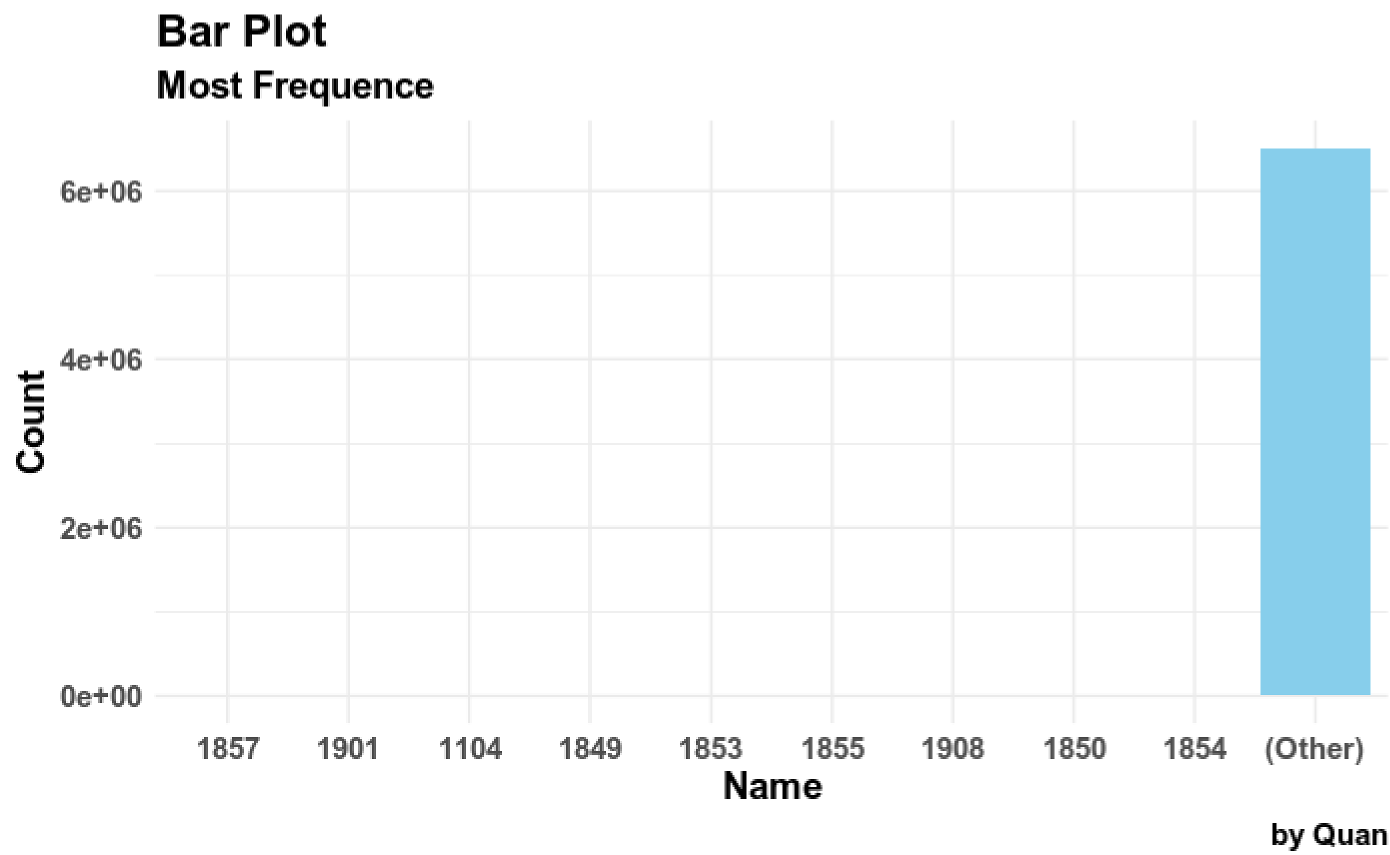

Bar Plot
Most Frequence
250000
200000
150000
100000
50000
0
Count
6
8
10
11
15
17
18
21
22
25
Name
by Quan


Bar Plot
Most Frequence
Count
9e+05
6e+05
3e+05
0e+00
1
2
3
4
5
6
7
Name
by Quan

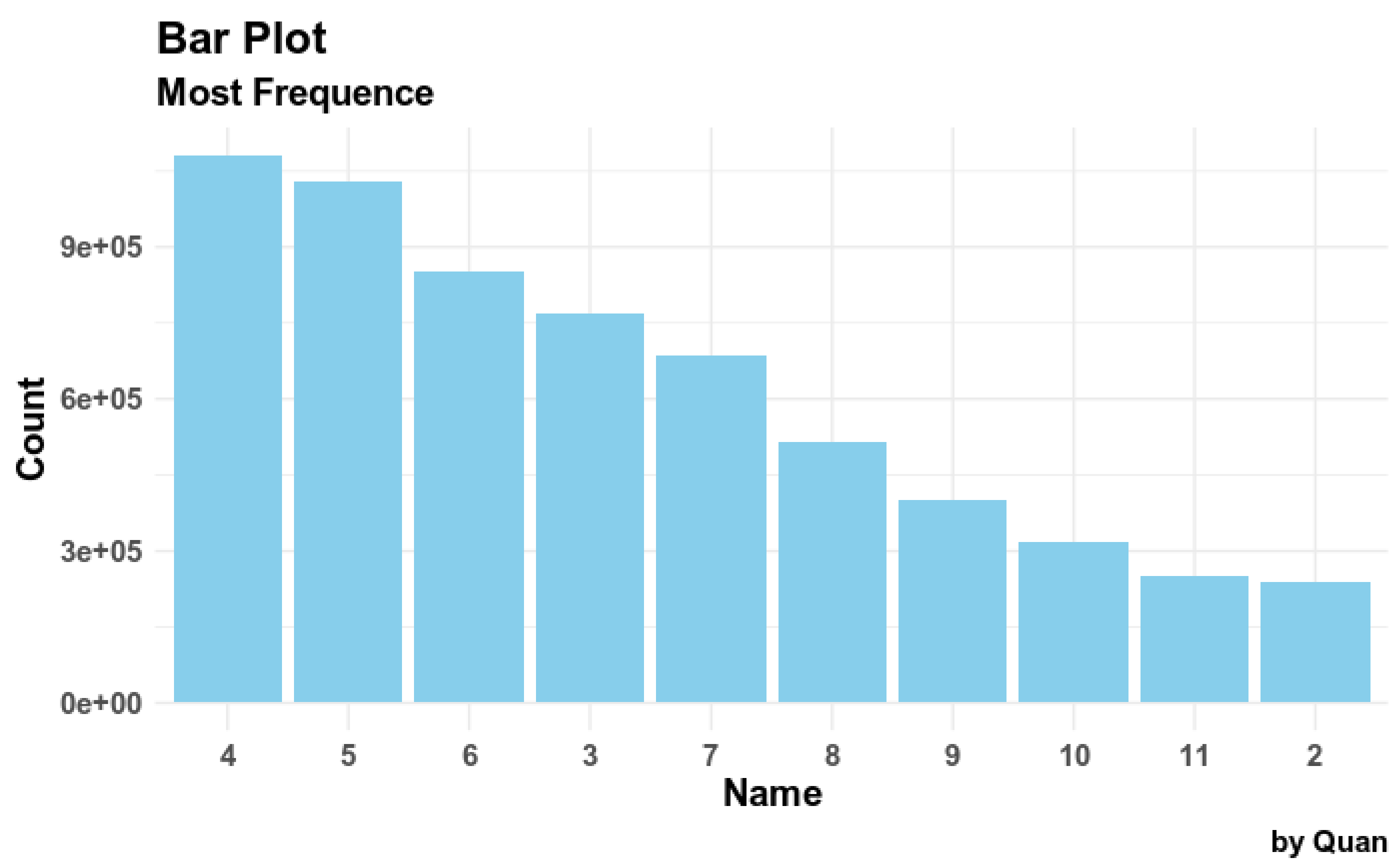

Bar Plot
Most Frequence
Count
6e+06
4e+06
2e+06
0e+00
55
403
511
546
347
352
64
478
419
(Other)
Name
by Quan


Bar Plot
Most Frequence
6e+06
4e+06
2e+06
0e+00
Count
555
557
556
558
559
554
655
600
656
(Other)
Name
by Quan

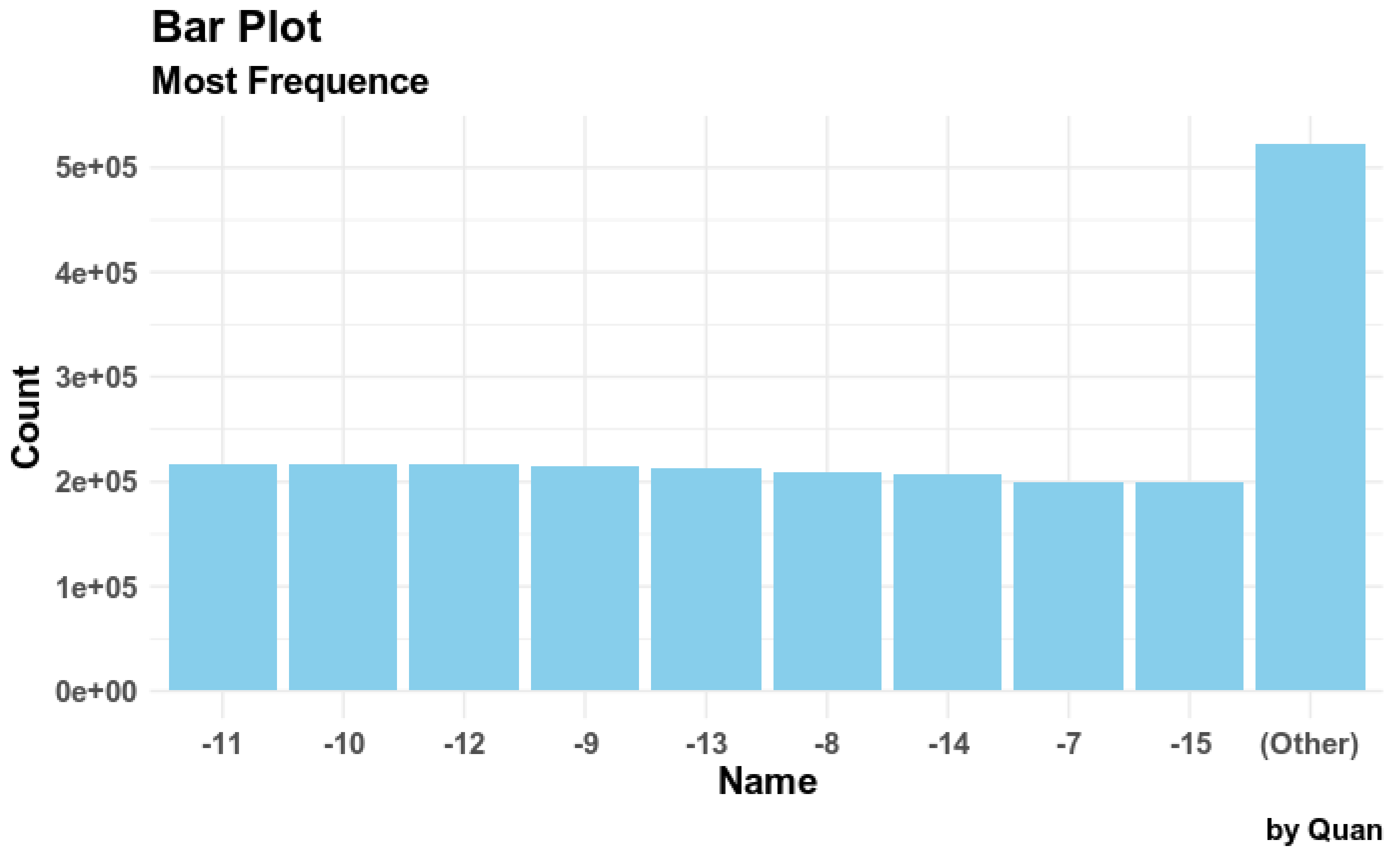

Bar Plot
Most Frequence
4e+05
2e+05
0e+00
Count
12
11
13
14
10
15
16
9
17
18
Name
by Quan

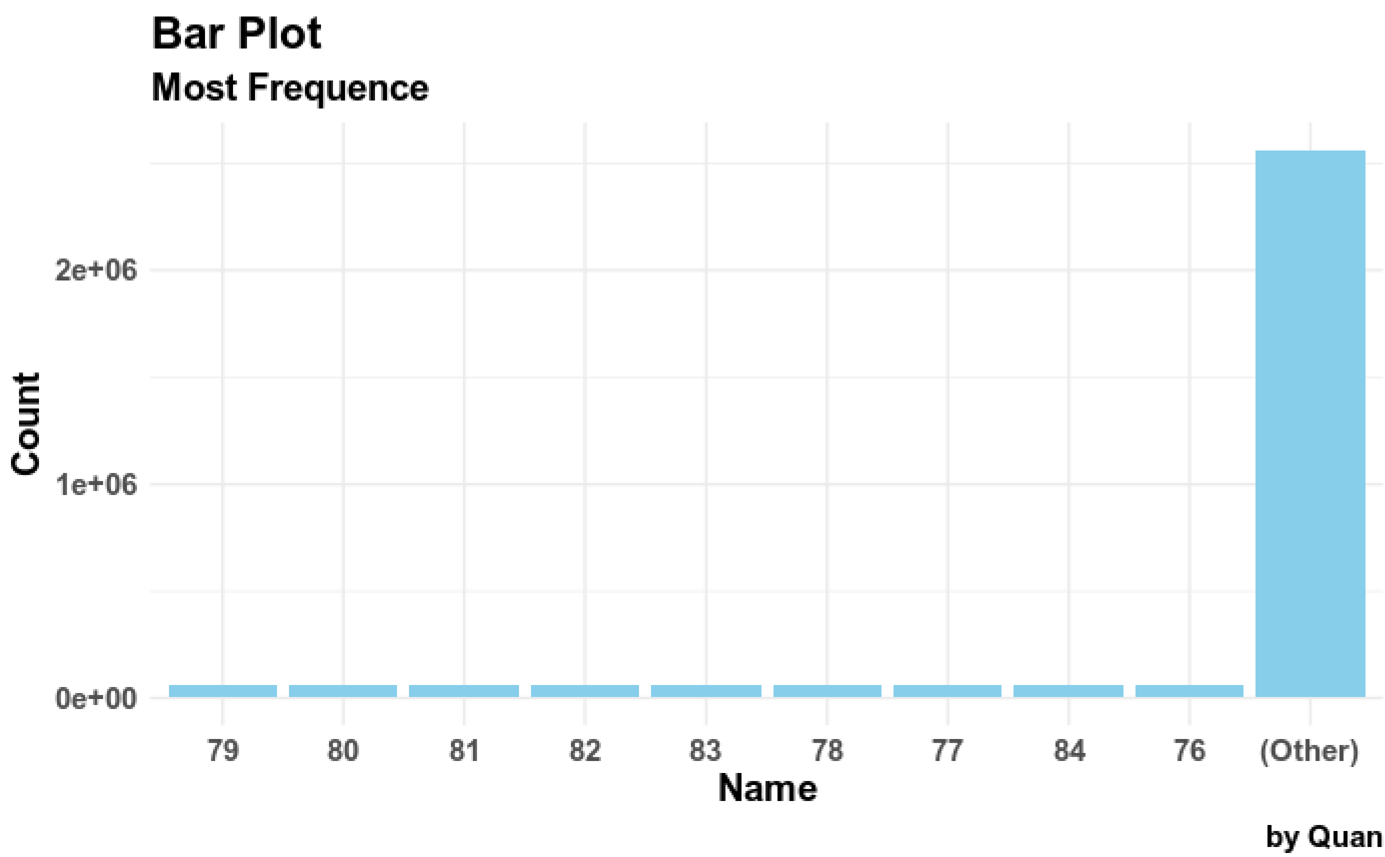


Bar Plot
Most Frequence
Count
6e+06
4e+06
2e+06
0e+00
1857
1901
1104
1849
1853
1855
1908
1850
1854
(Other)
Name
by Quan



Bar Plot
Most Frequence
Count
9e+05
6e+05
3e+05
0e+00
4
5
6
3
7
8
9
10
11
2
Name
by Quan





- We can see that some feature have very high `other` values, so its mean it have very much distinct value

- Not so much insight from this plot, but at least we will sure that no weird values at majority in any features

9. Next, we will convert `ARR_DELAY` to logical with meet code: 1 if later than 30 min, else 0

Hide

```
#
# temp <- ((df["ARR_DELAY"] > 30) * 1)
# temp <- as.factor(temp)
#
# # add new column LATE, remove ARR_DELAY
# df$LATE <- temp
# df$ARR_DELAY <- NULL

# we can see that `OP_CARRIER_FL_NUM` is similar to `OP_UNIQUE_CARRIER`, remove `OP_CARR
IER_FL_NUM`
df$OP_CARRIER_FL_NUM <- NULL

df
```

OP_UNIQUE_CAR... <fctr>	ORI... <fctr>	D... <fctr>	M... <int>	DAY_OF_M... <int>	DAY_OF_... <int>	DEP_T... <int>	TAXI_... <int>	WHEEL <ir
OH	PHL	CAK	10	8	2	1830	33	20
OH	PHL	CAK	10	9	3	1828	25	19
OH	PHL	CAK	10	10	4	1838	39	20
OH	PHL	CAK	10	11	5	1833	26	19
OH	PHL	CAK	10	12	6	1827	15	19
OH	PHL	CAK	10	13	7	1834	37	20
OH	PHL	CAK	10	14	1	1831	19	19
OH	PHL	CAK	10	15	2	2058	16	22
OH	PHL	CAK	10	16	3	1925	27	20
OH	PHL	CAK	10	17	4	1857	27	20

1-10 of 7,268,232 rows | 1-10 of 12 columns Previous **1** 2 3 4 5 6 ... 78 Next

10. We will then check outlier of all features

Transfrom `MONTH`, `DAY_OF_MONTH`, `DAY_OF_WEEK` to factor type

Hide

```
df <- transform(df, MONTH = as.factor(MONTH),
                      DAY_OF_MONTH = as.factor(DAY_OF_MONTH),
                      DAY_OF_WEEK = as.factor(DAY_OF_WEEK))
```

Hide

```
## first, we need to have all distinct values of factor features:
temp <- lapply(df, is.factor)
df_factor <- df[,unlist(temp)]

# get factor features
dist_factor <- lapply(df_factor, function(x) stack(summary(x, maxsum=9999999)))
```

Plot box plot

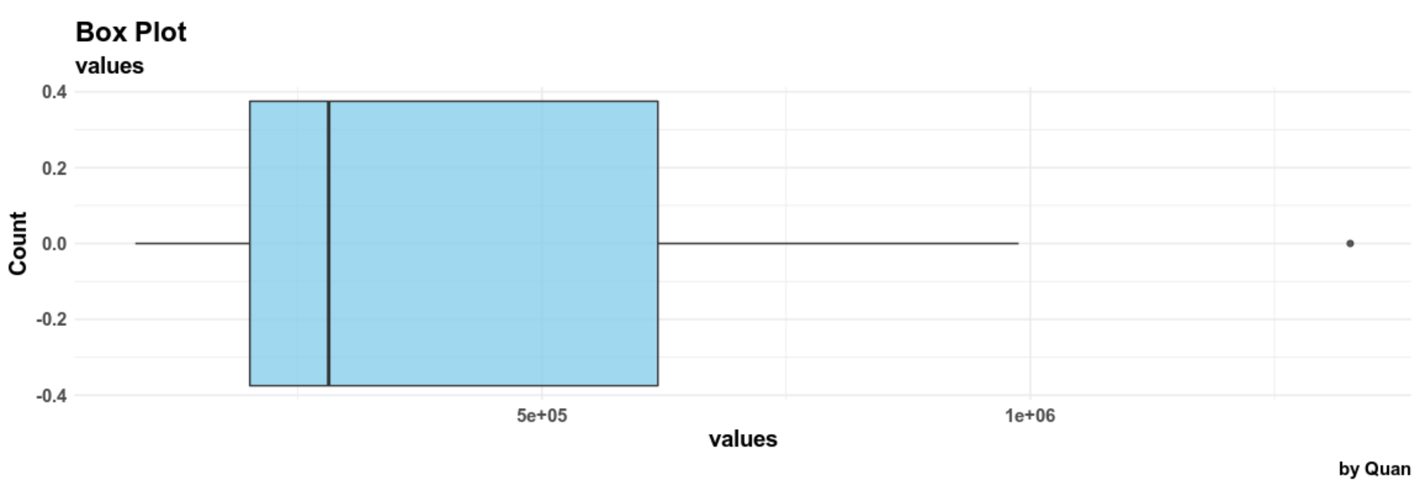
Hide

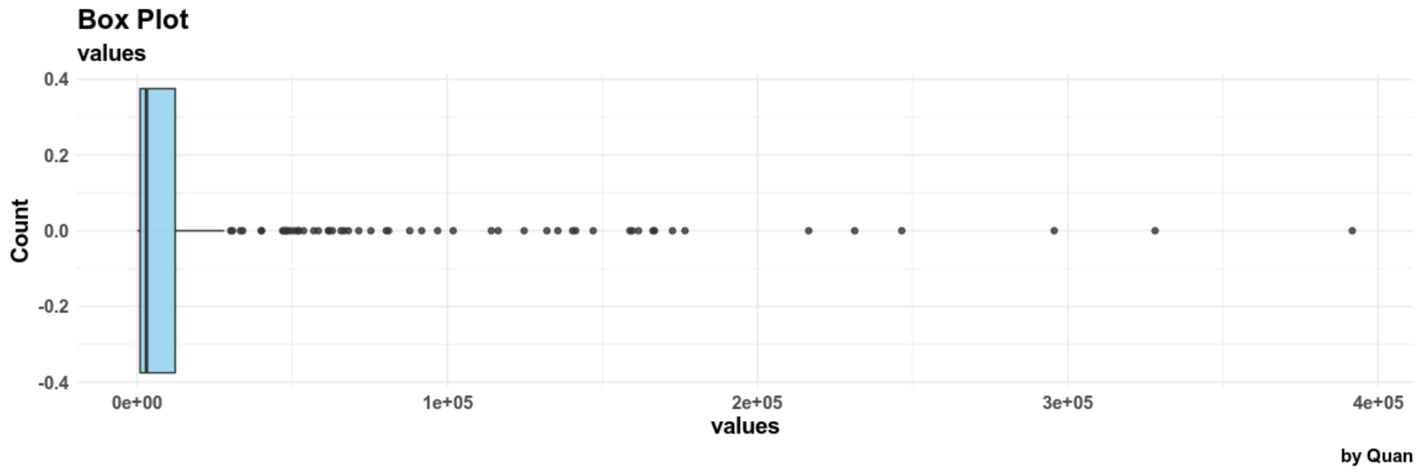
```
# check box plot all dist_factor features
print(names(dist_factor))
```

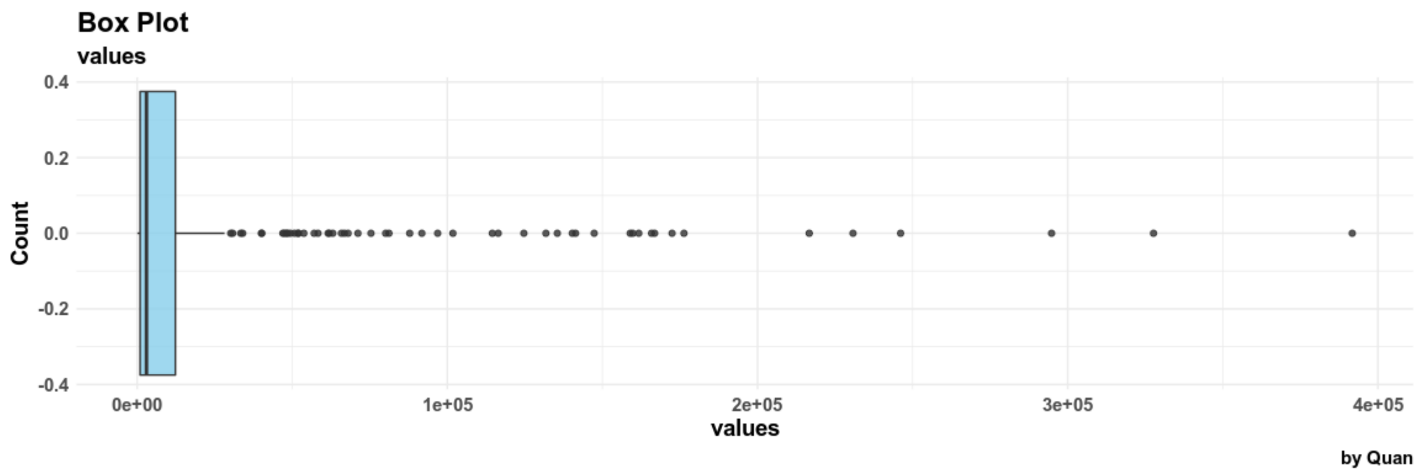
```
[1] "OP_UNIQUE_CARRIER" "ORIGIN"            "DEST"              "MONTH"             "DAY
_OF_MONTH"      "DAY_OF_WEEK"
[7] "LATE"
```

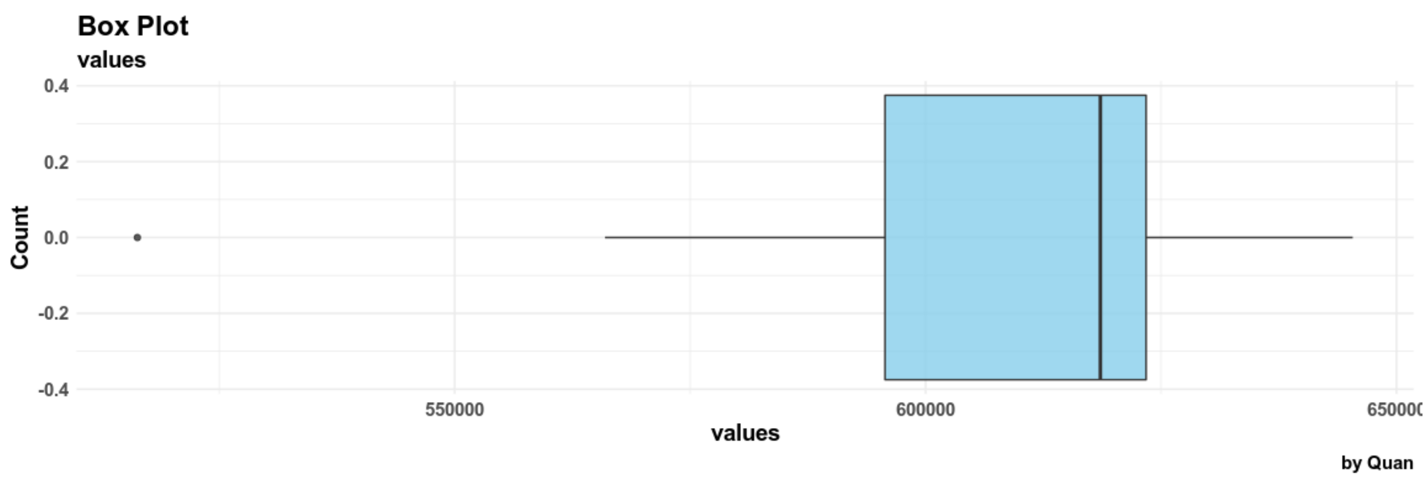
Hide

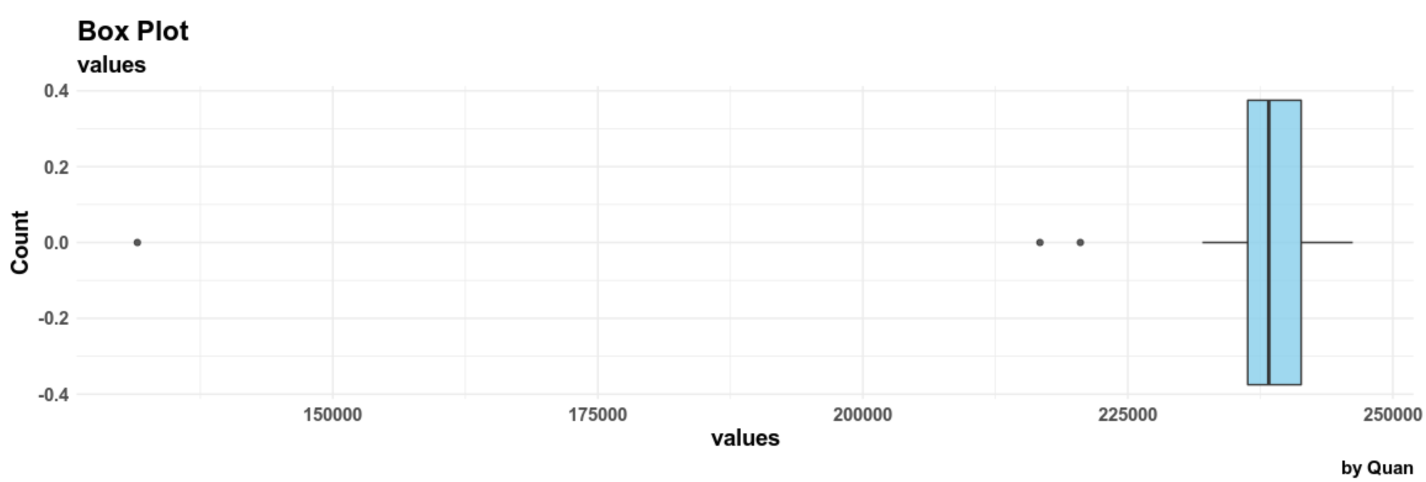
```
for (i in dist_factor) {
  print(plot.box(i, "values"))
}
```

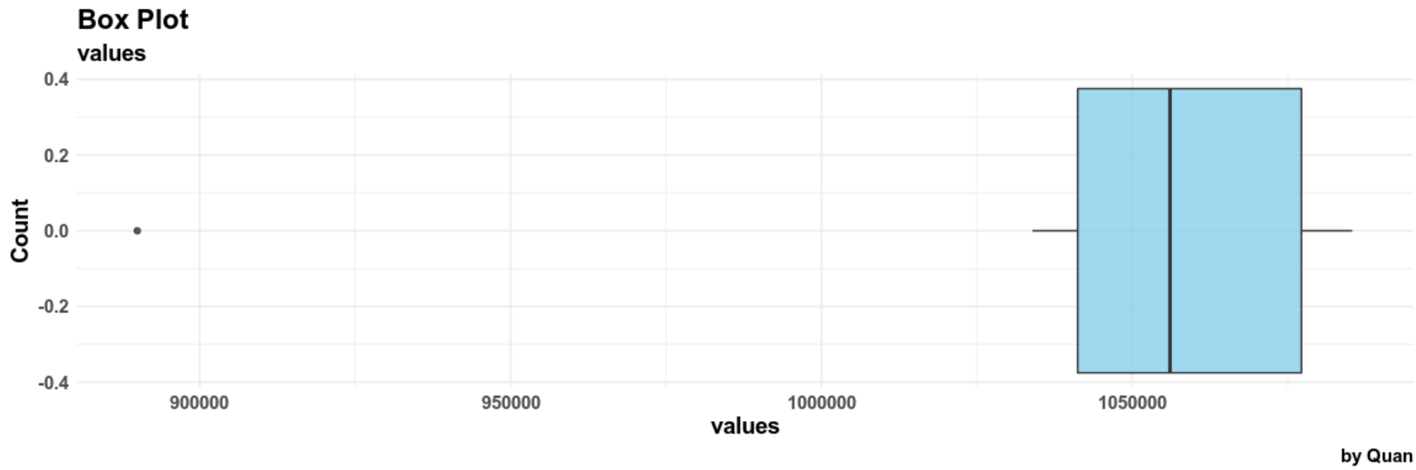


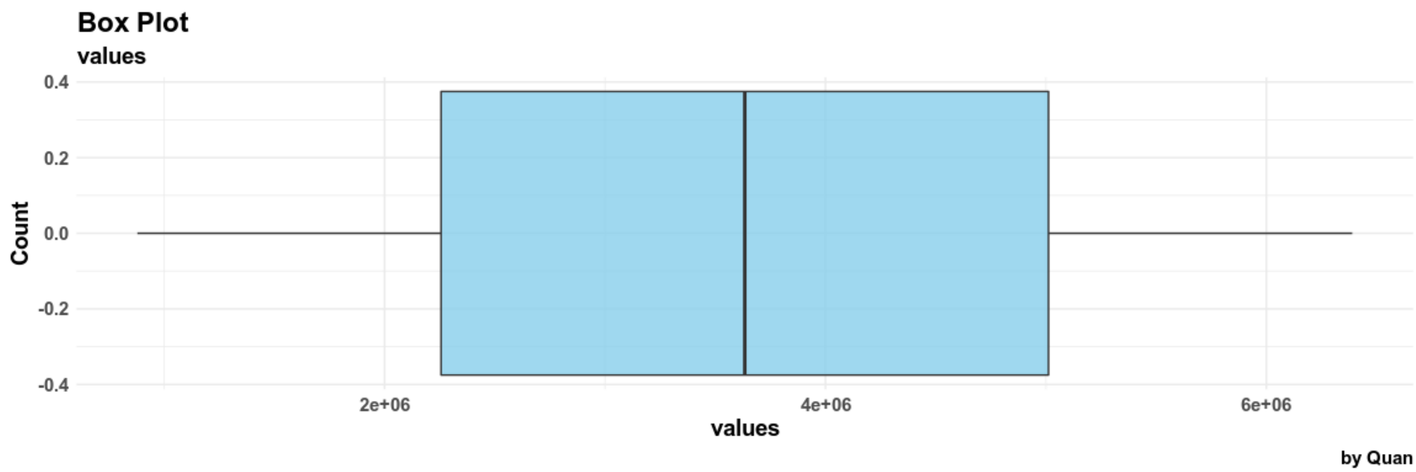

Box Plot
values
Count
0.4
0.2
0.0
-0.2
-0.4
0e+00
1e+05
2e+05
3e+05
4e+05
values
by Quan


Box Plot
values
Count
0.4
0.2
0.0
-0.2
-0.4
0e+00
1e+05
2e+05
3e+05
4e+05
values
by Quan


Box Plot
values
Count
0.4
0.2
0.0
-0.2
-0.4
550000
600000
650000
values
by Quan


Box Plot
values
Count
0.4
0.2
0.0
-0.2
-0.4
150000
175000
200000
225000
250000
values
by Quan





We see that some features have outliers

11. NORMAL DISTRIBUTION

Get numeric features

Hide

```
# # get all numeric features
# temp <- lapply(df,function(x) !is.factor(x))
# df_numeric <- df[,unlist(temp)]

df_numeric
```

DEP_TIME <int>	TAXI_OUT <int>	WHEELS_ON <int>	TAXI_IN <int>	ACTUAL_ELAPSED_TIME <int>
1830	33	2001	3	94
1828	25	1951	4	87
1838	39	2018	3	103
1833	26	1958	7	92
1827	15	1944	5	82

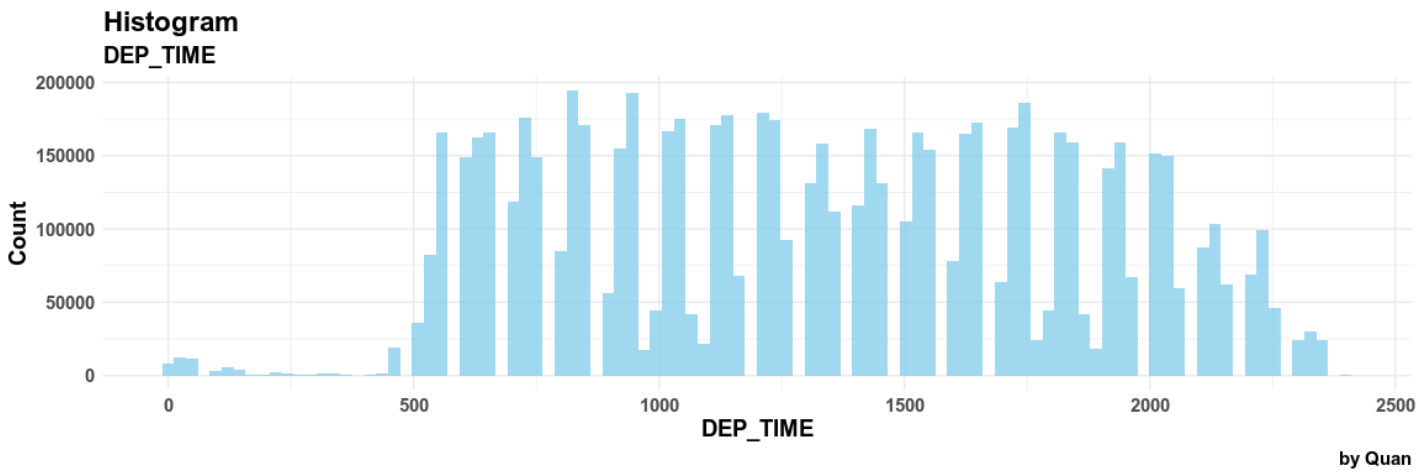
DEP_TIME <int>	TAXI_OUT <int>	WHEELS_ON <int>	TAXI_IN <int>	ACTUAL_ELAPSED_TIME <int>
1834	37	2015	8	109
1831	19	1948	8	85
2058	16	2213	5	80
1925	27	2051	4	90
1857	27	2022	3	88

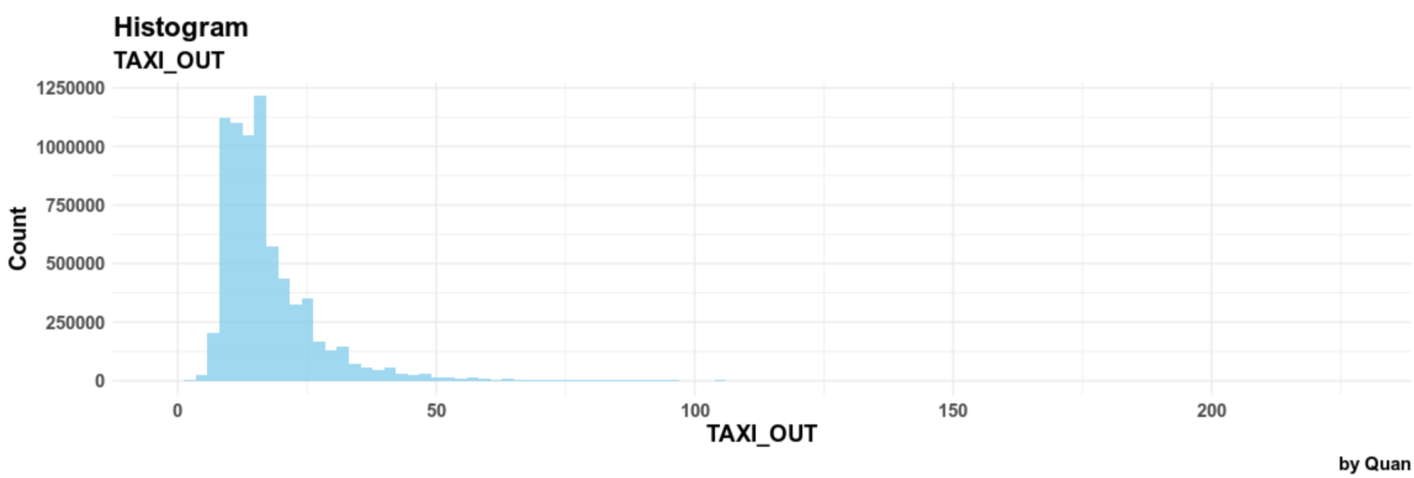
1-10 of 7,268,232 rows Previous **1** 2 3 4 5 6 ... 78 Next

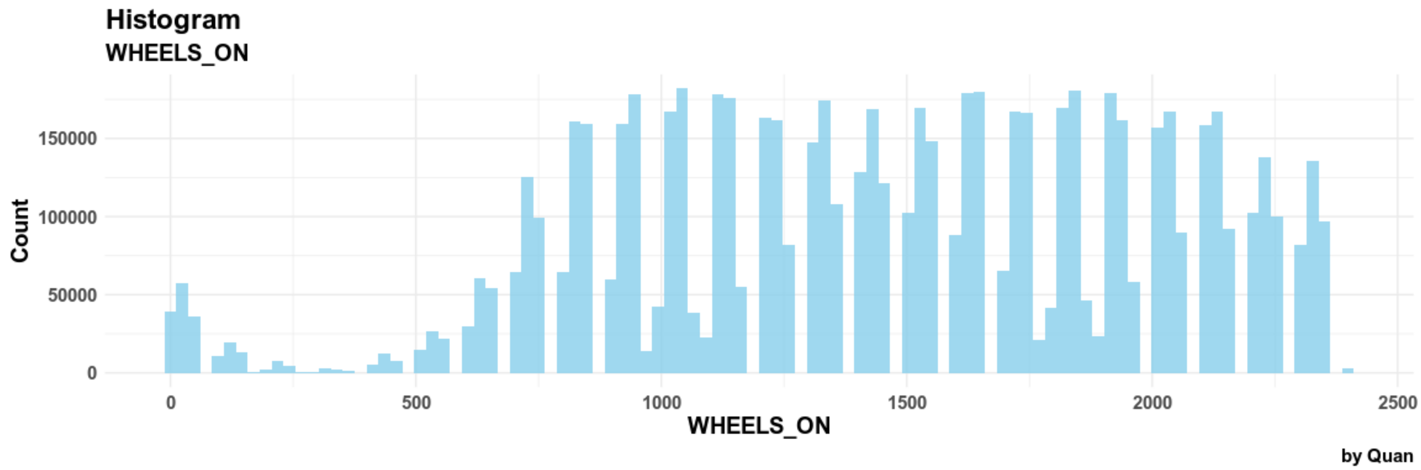
Hist plot

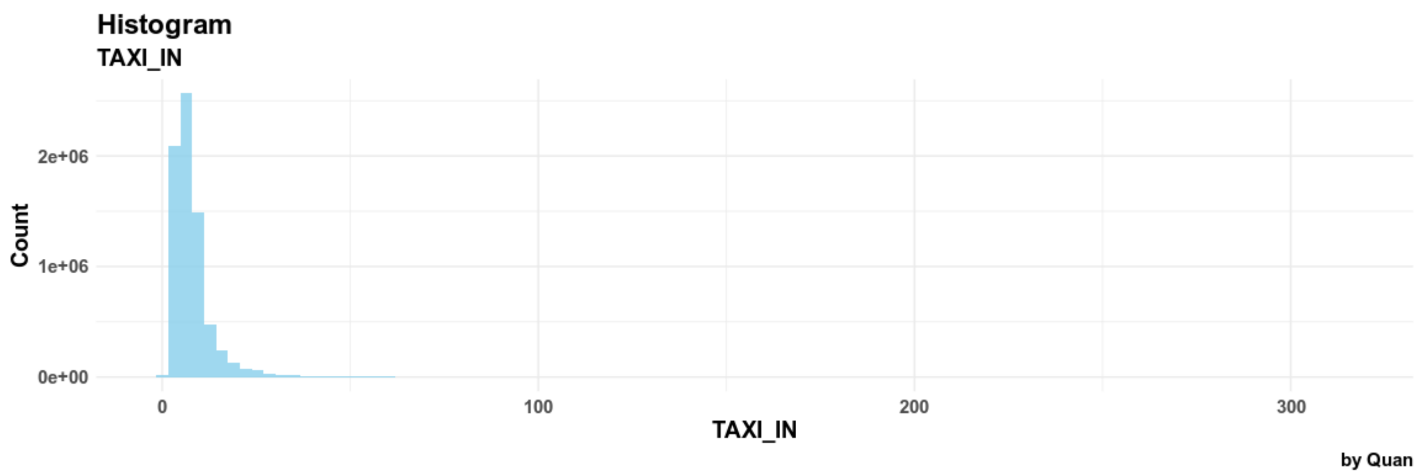
Hide

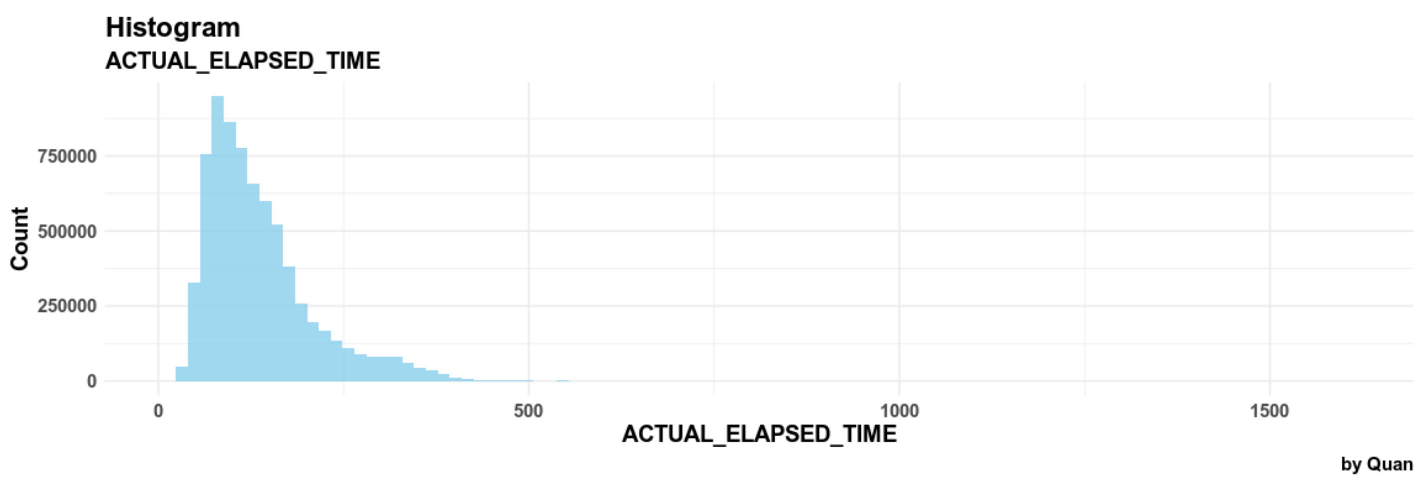
```
# hist plot all numeric features
for (i in names(df_numeric)){
  print(plot.hist(df_numeric, i))
}
```











We will use yeo johnson algorithm to apply transform

Hide

```
# # apply transform_yeo for all features in df_numeric
# transform_yeo <- mclapply(df_numeric, yeojohnson, mc.cores = 6)
#
# # create a dataframe contain all transformed values for all features
# system.time(
# df_transform_yeo <- lapply(transform_yeo, function(i){data.frame("transform" = i$x.
t)})
# )

df_transform_yeo
```

```
$DEP_TIME
```

transform <dbl>
0.972577021
0.968896584
0.987291646
0.978096340
0.967056098
0.979935757
0.974416972
1.387682842
1.146591978
1.022193472

1-10 of 7,268,232 rows Previous 1 2 3 4 5 6 ... 78 Next

```
$TAXI_OUT
```

transform <dbl>
1.5737812
1.0808882
1.8430484
1.1541093
0.0110519
1.7603124

transform
<dbl>
0.5336581
0.1585956
1.2234282
1.2234282

1-10 of 7,268,232 rows Previous **1** 2 3 4 5 6 ... 78 Next

```
$WHEELS_ON
```

transform
<dbl>
1.01704584
0.92061695
1.04987578
0.93410523
0.90713254
1.04408065
0.91483744
1.42799421
1.11366741
1.05760369

1-10 of 7,268,232 rows Previous **1** 2 3 4 5 6 ... 78 Next

```
$TAXI_IN
```

transform
<dbl>
-1.36713226
-0.79611837
-1.36713226
0.23723626
-0.36960666

transform
<dbl>
0.46427079
0.46427079
-0.36960666
-0.79611837
-1.36713226

1-10 of 7,268,232 rows Previous **1** 2 3 4 5 6 ... 78 Next

```
$ACTUAL_ELAPSED_TIME
```

transform
<dbl>
-0.483850770
-0.647872544
-0.292303481
-0.529257990
-0.774505711
-0.174907372
-0.697507697
-0.827638436
-0.575802659
-0.623539225

1-10 of 7,268,232 rows Previous **1** 2 3 4 5 6 ... 78 Next

```
NA
```

Plot transformed features by histplot

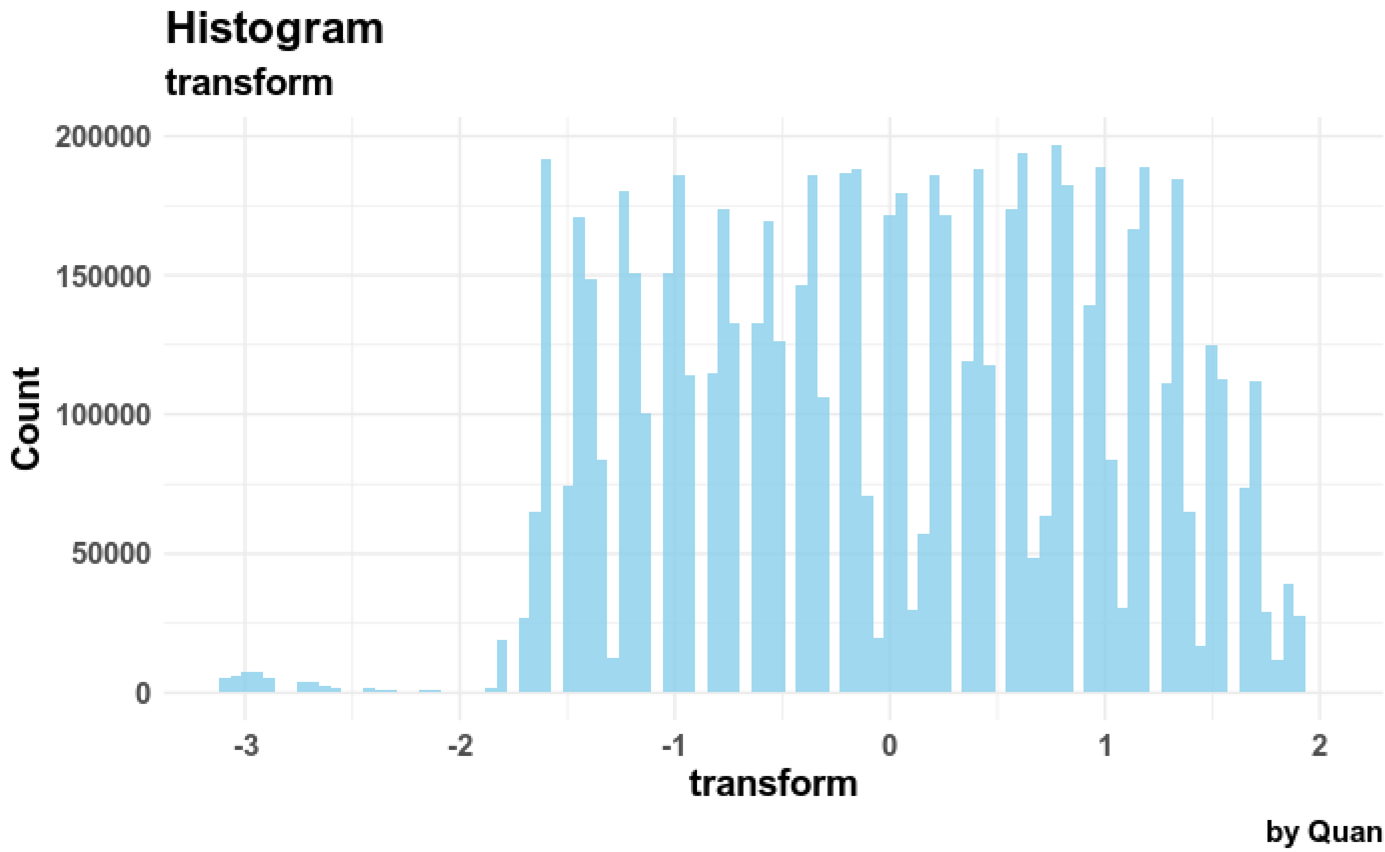
Hide

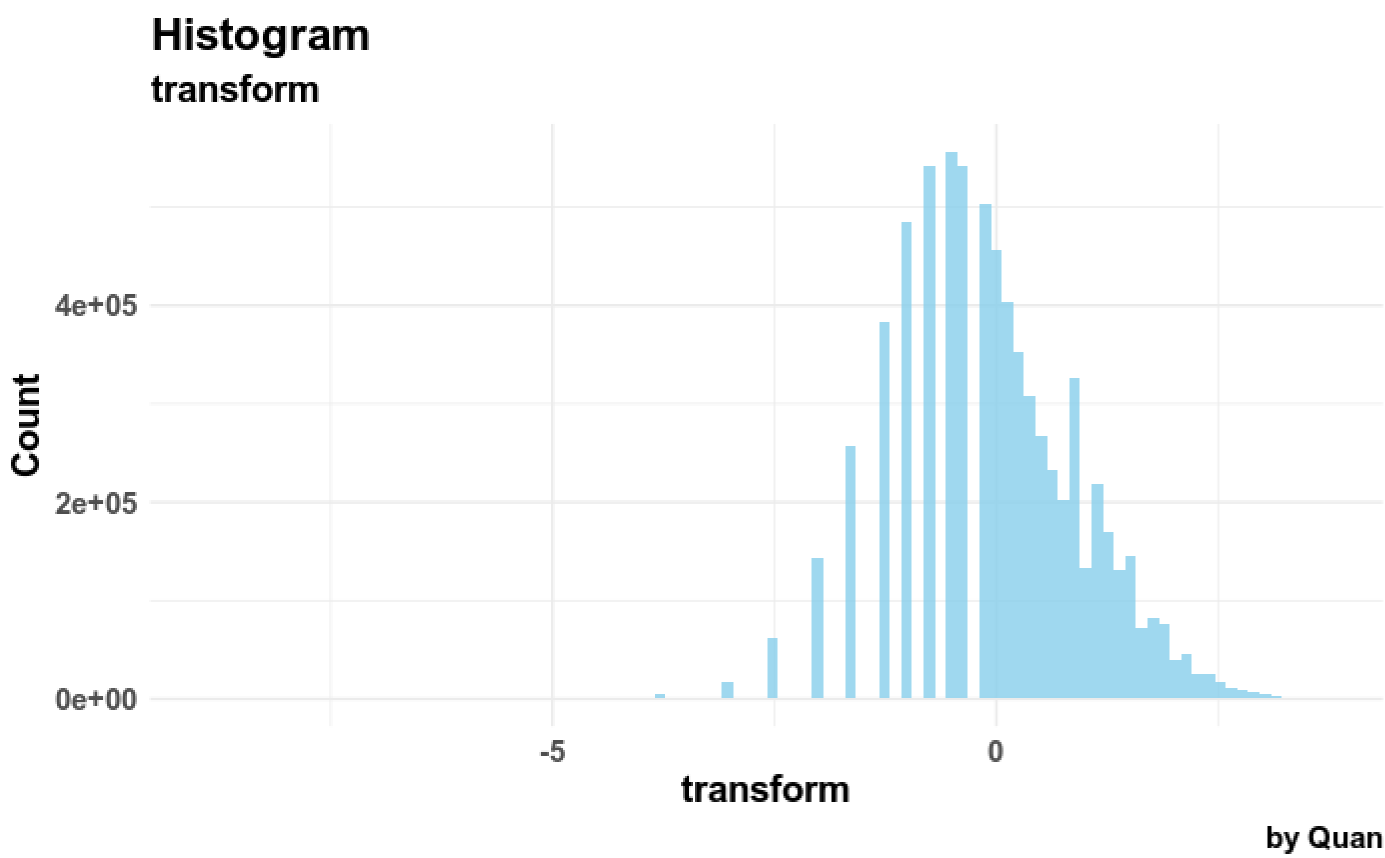
```
print(names(df_transform_yeo))
```

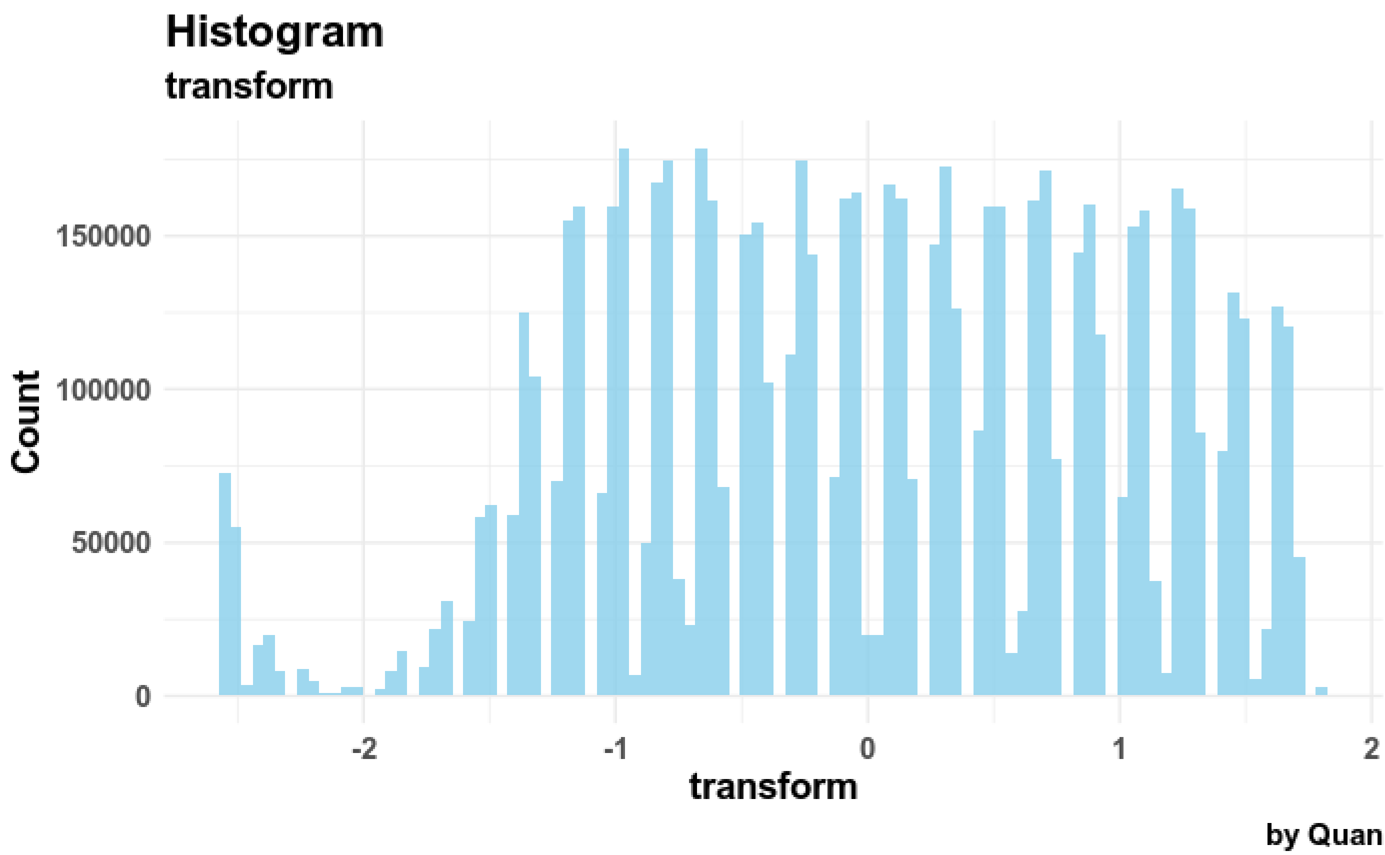
```
[1] "DEP_TIME"            "TAXI_OUT"            "WHEELS_ON"            "TAXI_IN"
"ACTUAL_ELAPSED_TIME"
```

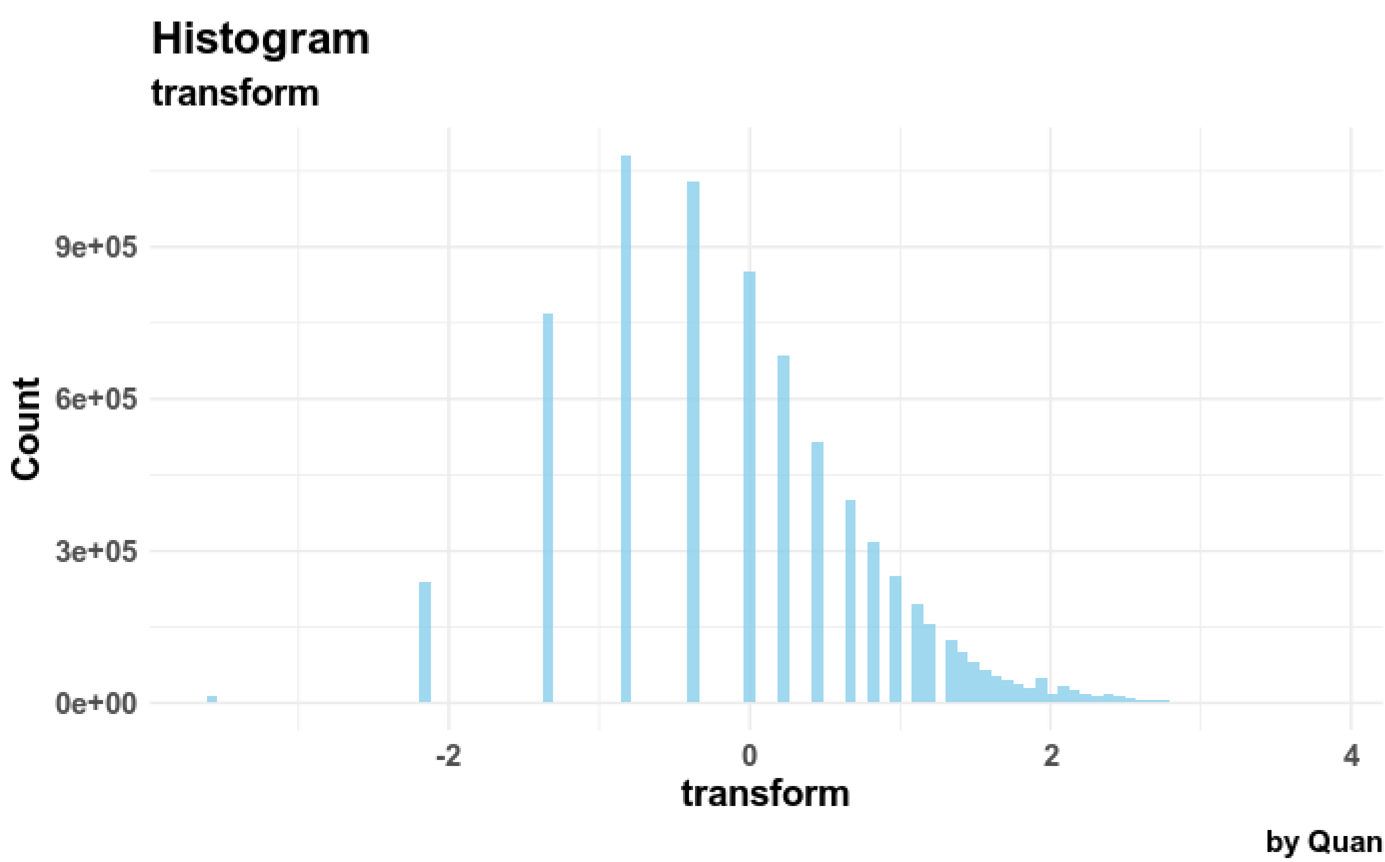
Hide

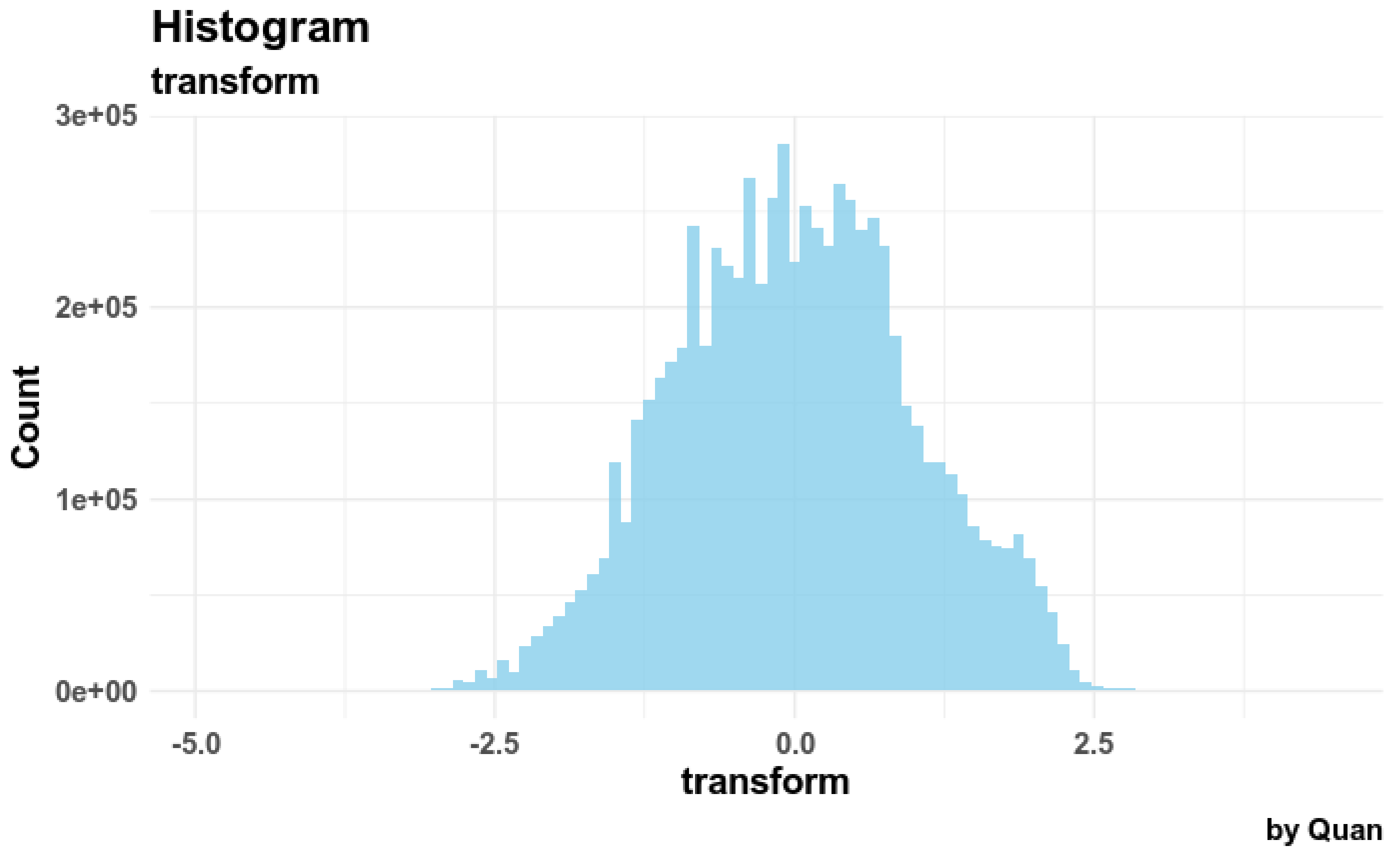
```
for (i in df_transform_yeo) print(plot.hist(i, name = "transform"))
```






Histogram
transform
Count
150000
100000
50000
0
-2
-1
0
1
2
transform
by Quan


Histogram
transform
Count
9e+05
6e+05
3e+05
0e+00
-2
0
2
4
transform
by Quan



We can see that features had normal distribution now, as well as well scaling

12. Next, we will find outlier of these features

Hide

```
# very useful trick to get outlier by using boxplot$out
outlier_list <- lapply(df_transform_yeo, function(x)boxplot(x, plot = F)$out)
```

Update new values of numeric features to orig data

Hide

```
df2 <- df
for (name in names(df_transform_yeo)){
  df2[,name] <- df_transform_yeo[[name]][,]
}

df2
```

OP_UNIQUE_CAR... <fctr>	ORI... <fctr>	D... <fctr>	M... <fctr>	DAY_OF_M... <fctr>	DAY_OF_... <fctr>	DEP_TIME <dbl>	TAXI_OUT <dbl>
OH	PHL	CAK	10	8	2	0.972577021	1.5737812
OH	PHL	CAK	10	9	3	0.968896584	1.0808882
OH	PHL	CAK	10	10	4	0.987291646	1.8430484

OP_UNIQUE_CAR...	ORI...	D...	M...	DAY_OF_M...	DAY_OF_...	DEP_TIME	TAXI_OUT
<fctr>	<fctr>	<fctr>	<fctr>	<fctr>	<fctr>	<dbl>	<dbl>
OH	PHL	CAK	10	11	5	0.978096340	1.1541093
OH	PHL	CAK	10	12	6	0.967056098	0.0110519
OH	PHL	CAK	10	13	7	0.979935757	1.7603124
OH	PHL	CAK	10	14	1	0.974416972	0.5336581
OH	PHL	CAK	10	15	2	1.387682842	0.1585956
OH	PHL	CAK	10	16	3	1.146591978	1.2234282
OH	PHL	CAK	10	17	4	1.022193472	1.2234282

1-10 of 7,268,232 rows | 1-9 of 12 columns Previous **1** 2 3 4 5 6 ... 78 Next

Remove outliers

Hide

```
# for (name in names(outlier_list)){
#   df2 <- df2[which(!df2[,name] %in% outlier_list[[name]]),]
# }

df2
```

	OP_UNIQUE_CAR...	ORI...	D...	M...	DAY_OF_M...	DAY_OF_...	DEP_TIME	TAXI_OU
	<fctr>	<fctr>	<fctr>	<fctr>	<fctr>	<fctr>	<dbl>	<db
1	OH	PHL	CAK	10	8	2	0.972577021	1.57378
2	OH	PHL	CAK	10	9	3	0.968896584	1.08088
3	OH	PHL	CAK	10	10	4	0.987291646	1.84304
4	OH	PHL	CAK	10	11	5	0.978096340	1.15410
5	OH	PHL	CAK	10	12	6	0.967056098	0.01105
6	OH	PHL	CAK	10	13	7	0.979935757	1.76031
7	OH	PHL	CAK	10	14	1	0.974416972	0.53365
8	OH	PHL	CAK	10	15	2	1.387682842	0.15859
9	OH	PHL	CAK	10	16	3	1.146591978	1.22342
10	OH	PHL	CAK	10	17	4	1.022193472	1.22342

1-10 of 7,185,818 rows | 1-9 of 12 columns Previous **1** 2 3 4 5 6 ... 78 Next

More than 80,000 rows have been removed

For future purposes, we will seperate target feature `LATE`

Hide

```
# we split dataframe by target
df_late <- df2["LATE"]
df2$LATE <- NULL
```

13. ONE HOT ENCODING (we faced many problems, since my computer RAM don't have enough to apply One Hot. but you can try by your own computer)

Hide

```
# option 1
dummy <- dummyVars(" ~ .", data=df2)
## Error: cannot allocate vector of size 42.5 Gb
## so that we didnt have enough RAM to store this variable ~
df3 <- data.frame(predict(dummy, newdata = df2))

# option 2
## Still error, problem occur because of RAM exceeded!
library(mltools)
library(data.table)
newdata <- one_hot(as.data.table(df2))
```

14. We have to just convert factor features to numerics distinct values

Hide

```
# convert all factor to numeric
temp <- df2[,which(sapply(df2,is.factor))]
numeric_factor <- lapply(temp, as.numeric)

# apply transform_yeo for these feature
transform_yeo_factor<- mclapply(numeric_factor, yeojohnson, mc.cores = 6)

# make dataframe of these feature
system.time(
df_transform_yeo_factor <- lapply(transform_yeo_factor, function(i){data.frame("transfor
m" = i$x.t)})
)
```

```
   user  system elapsed 
  0.004   0.000   0.004 
```

Plot these features

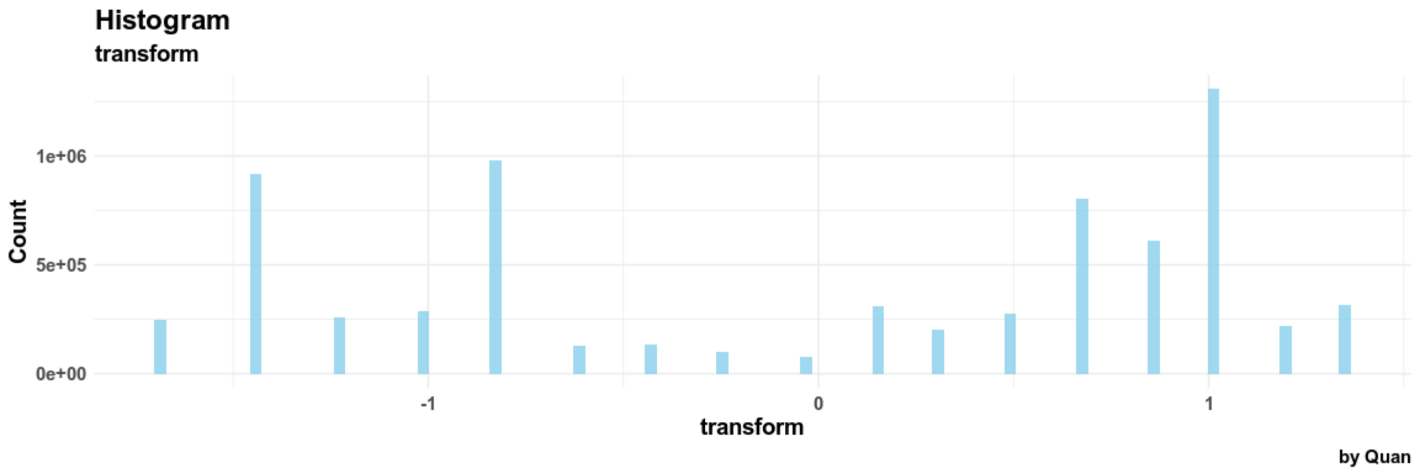
Hide

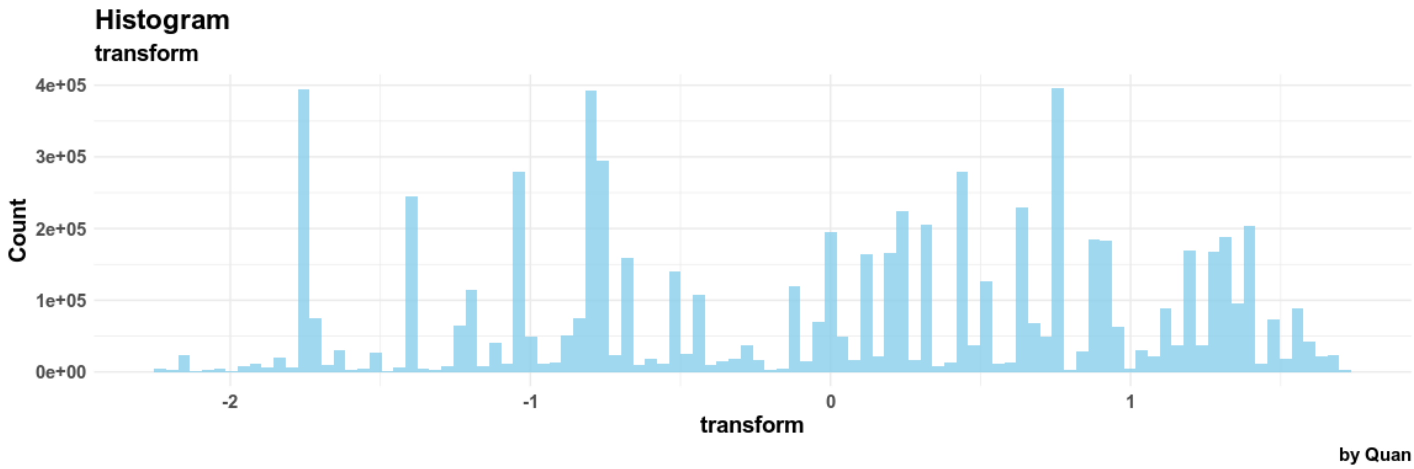
```
print(names(df_transform_yeo_factor))
```

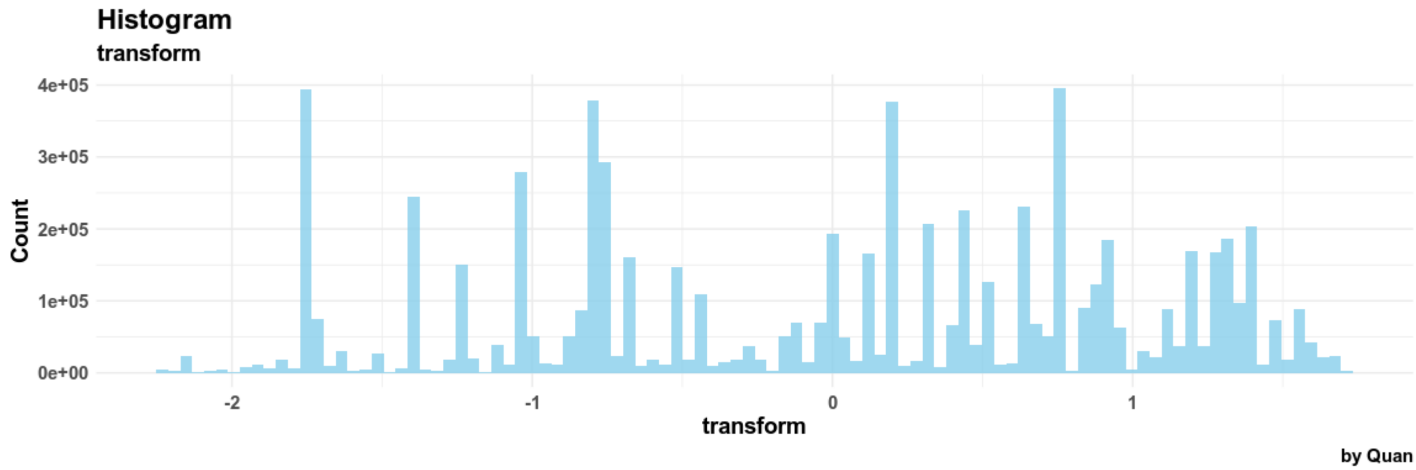
```
[1] "OP_UNIQUE_CARRIER" "ORIGIN"            "DEST"              "MONTH"             "DAY
_OF_MONTH"      "DAY_OF_WEEK"
```

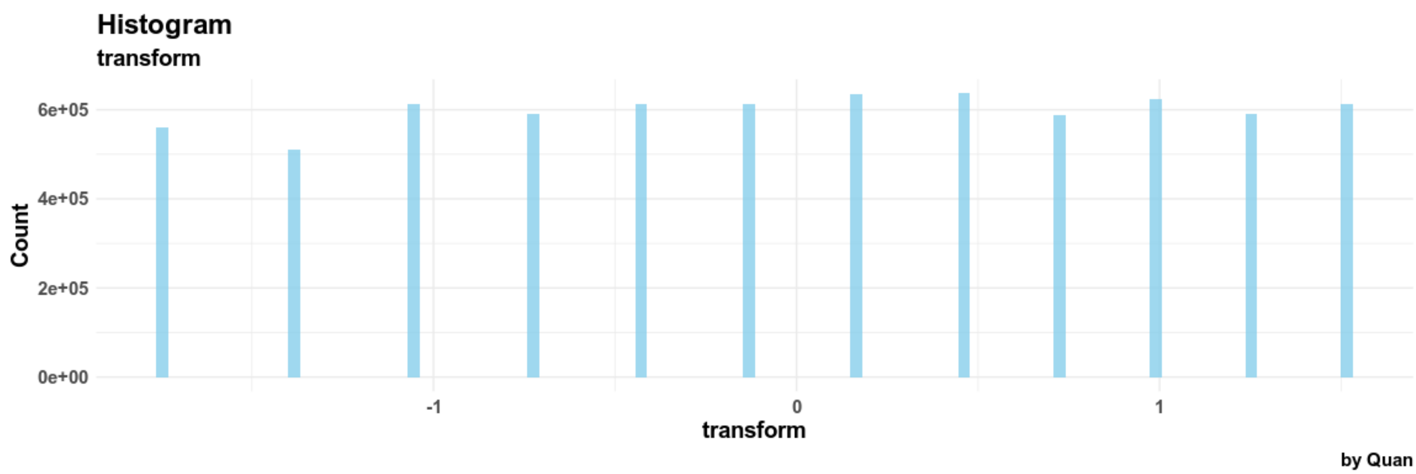
Hide

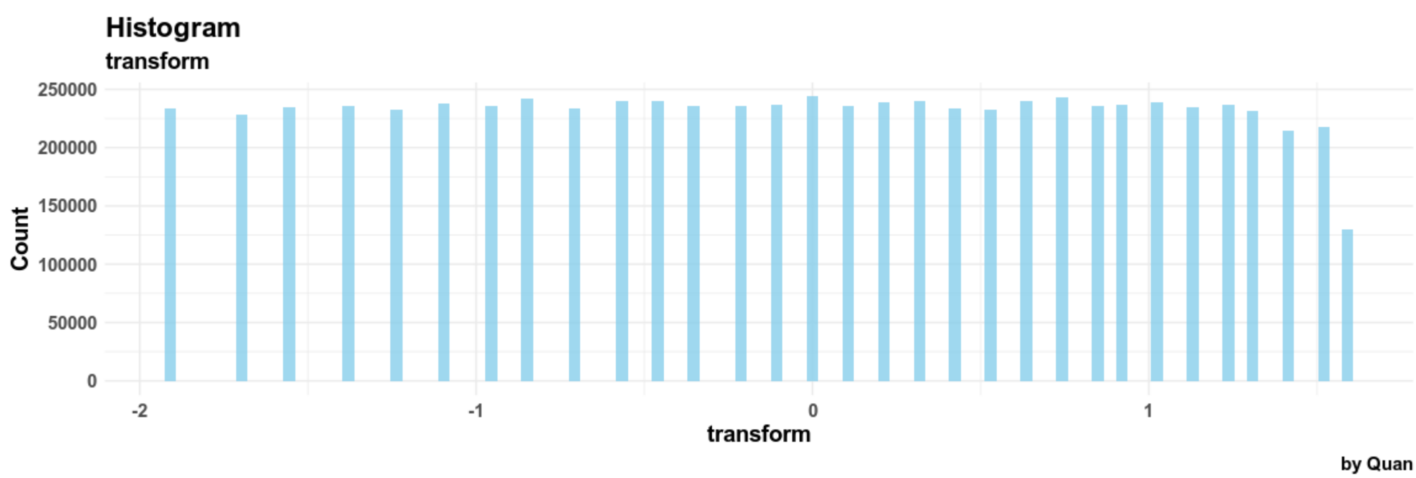
```
for (i in df_transform_yeo_factor) print(plot.hist(i, name = "transform"))
```

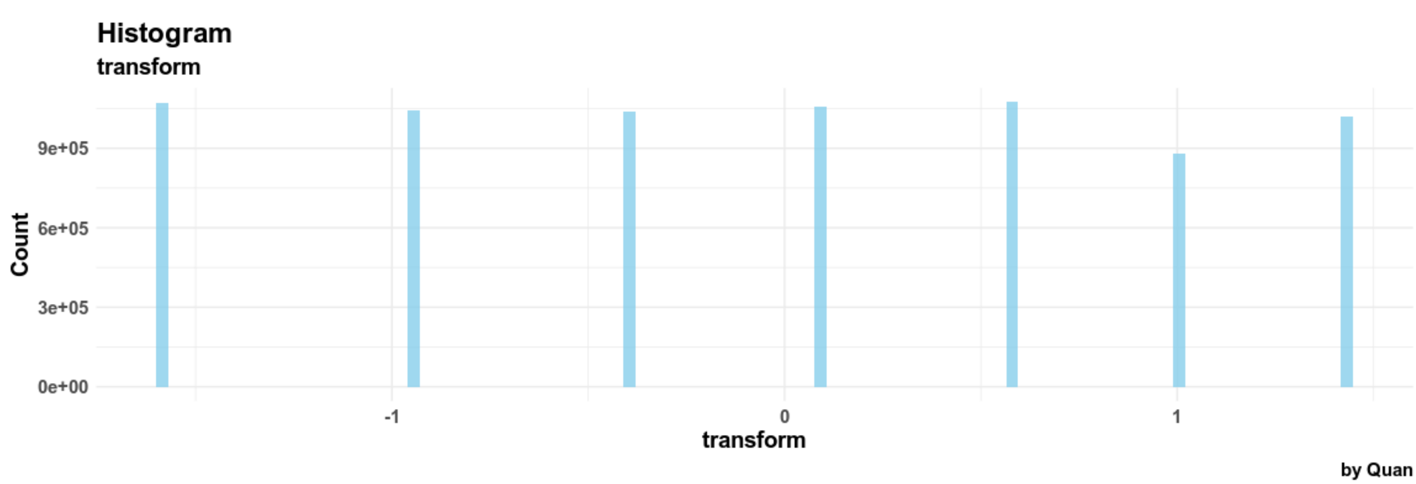





Histogram
transform
4e+05
3e+05
2e+05
1e+05
0e+00
Count
-2
-1
0
1
transform
by Quan


Histogram
transform
6e+05
4e+05
2e+05
0e+00
Count
-1
0
1
transform
by Quan


Histogram
transform
250000
200000
150000
100000
50000
0
Count
-2
-1
0
1
transform
by Quan


Histogram
transform
9e+05
6e+05
3e+05
0e+00
Count
-1
0
1
transform
by Quan

Add these feature to orig data

Hide

```
df3 <- df2
for (name in names(df_transform_yeo_factor)) {
  df3[,name] <- df_transform_yeo_factor[[name]]
}

# add target features again
df3$LATE <- df_late[,]

df3
```

	OP_UNIQUE_CAR... <dbl>	ORIGIN <dbl>	DEST <dbl>	MONTH <dbl>	DAY_OF_MO... <dbl>	DAY_OF_... <dbl>
1	0.4962763	0.89338399	-1.17885269	0.9882139	-8.337389e-01	-0.9545875
2	0.4962763	0.89338399	-1.17885269	0.9882139	-7.060647e-01	-0.4042598
3	0.4962763	0.89338399	-1.17885269	0.9882139	-5.817866e-01	0.1013186
4	0.4962763	0.89338399	-1.17885269	0.9882139	-4.605165e-01	0.5738773
5	0.4962763	0.89338399	-1.17885269	0.9882139	-3.419390e-01	1.0206343
6	0.4962763	0.89338399	-1.17885269	0.9882139	-2.257935e-01	1.4464296
7	0.4962763	0.89338399	-1.17885269	0.9882139	-1.118613e-01	-1.5715088
8	0.4962763	0.89338399	-1.17885269	0.9882139	4.366633e-05	-0.9545875
9	0.4962763	0.89338399	-1.17885269	0.9882139	1.100809e-01	-0.4042598
10	0.4962763	0.89338399	-1.17885269	0.9882139	2.183890e-01	0.1013186

1-10 of 7,185,818 rows | 1-8 of 12 columns Previous **1** 2 3 4 5 6 ... 78 Next

15. Save file

Hide

```
# add target column again
df3$LATE <- df_late[,]
```

B. Apply Machine Learning

Hide

```
# clean var
rm(list = ls())
rm()
gc()
```

```
           used  (Mb) gc trigger   (Mb)  max used   (Mb)
Ncells  2907585 155.3    8039128  429.4  10652849  569.0
Vcells 17312786 132.1  459561666 3506.2 574452082 4382.8
```

1. Load file

Hide

```
df <- read.csv(file = "/home/apolong72/ds/r/data/final.csv")
```

2. quick look at data

Hide

```
df
```

OP_UNIQUE_CAR... <dbl>	ORIGIN <dbl>	DEST <dbl>	MONTH <dbl>	DAY_OF_MO... <dbl>	DAY_OF_... <dbl>	DE
0.4962763	0.89338399	-1.17885269	0.9882139	-8.337389e-01	-0.9545875	0.972
0.4962763	0.89338399	-1.17885269	0.9882139	-7.060647e-01	-0.4042598	0.968
0.4962763	0.89338399	-1.17885269	0.9882139	-5.817866e-01	0.1013186	0.987
0.4962763	0.89338399	-1.17885269	0.9882139	-4.605165e-01	0.5738773	0.978
0.4962763	0.89338399	-1.17885269	0.9882139	-3.419390e-01	1.0206343	0.967
0.4962763	0.89338399	-1.17885269	0.9882139	-2.257935e-01	1.4464296	0.979
0.4962763	0.89338399	-1.17885269	0.9882139	-1.118613e-01	-1.5715088	0.974
0.4962763	0.89338399	-1.17885269	0.9882139	4.366633e-05	-0.9545875	1.387
0.4962763	0.89338399	-1.17885269	0.9882139	1.100809e-01	-0.4042598	1.146
0.4962763	0.89338399	-1.17885269	0.9882139	2.183890e-01	0.1013186	1.022

1-10 of 7,185,818 rows | 1-7 of 12 columns Previous 1 2 3 4 5 6 ... 78 Next

Convert type and Skim

Hide

```
# convert target to factor type
df$LATE <- as.factor(df$LATE)

# skim
skim(df)
```

```
── Data Summary ────────────────────────
                           Values
Name                       df
Number of rows             7185818
Number of columns          12
_______________________
Column type frequency:
  factor                   1
  numeric                  11
________________________
Group variables            None

── Variable type: factor ───────────────────────────────────────────────────────────────
──────────────────────────────────────────────────────────────
  skim_variable n_missing complete_rate ordered n_unique top_counts
1 LATE                  0             1 FALSE          2 0: 6343661, 1: 842157

── Variable type: numeric ──────────────────────────────────────────────────────────────
──────────────────────────────────────────────────────────────
   skim_variable       n_missing complete_rate     mean    sd     p0    p25      p50   p7
5  p100 hist
 1 OP_UNIQUE_CARRIER           0             1 -2.16e-8 1.00  -1.68 -0.813  0.319   1.02
1.35
 2 ORIGIN                      0             1 -1.36e-8 1.00  -2.23 -0.780  0.202   0.77
8  1.72
 3 DEST                        0             1  3.54e-9 1.00  -2.23 -0.783  0.199   0.77
5  1.72
 4 MONTH                       0             1  2.57e-8 1.00  -1.76 -0.727  0.167   0.98
8  1.51
 5 DAY_OF_MONTH                0             1  2.76e-9 1.00  -1.90 -0.834  0.110   0.83
8  1.60
 6 DAY_OF_WEEK                 0             1 -5.03e-9 1.00  -1.57 -0.955  0.101   1.02
1.45
 7 DEP_TIME                    0             1 -1.82e-3 1.00  -3.11 -0.810  0.0197  0.81
7  2.00
 8 TAXI_OUT                    0             1  3.32e-3 0.966 -2.49 -0.750  0.0111  0.64
1  2.72
 9 WHEELS_ON                   0             1 -1.15e-3 1.00  -2.56 -0.792 -0.0169  0.84
4  1.79
10 TAXI_IN                     0             1  4.01e-3 0.982 -2.19 -0.796 -0.0349  0.65
8  2.83
11 ACTUAL_ELAPSED_TIME         0             1  1.48e-3 0.990 -2.78 -0.723  0.00526 0.68
0  2.78
```

data is better now, no `NA` , `sd` equal to 1, `hist` seem good enough.

3. Split data to train/test

Hide

```
set.seed(123)
split <- sample.split(df$LATE, SplitRatio = 0.9)

train <- subset(df, split)
test <- subset(df, !split)

train
```

	OP_UNIQUE_CAR... <dbl>	ORIGIN <dbl>	DEST <dbl>	MONTH <dbl>	DAY_OF_MO... <dbl>	DAY_OF_... <dbl>
1	0.4962763	0.89338399	-1.17885269	0.9882139	-8.337389e-01	-0.9545875
2	0.4962763	0.89338399	-1.17885269	0.9882139	-7.060647e-01	-0.4042598
3	0.4962763	0.89338399	-1.17885269	0.9882139	-5.817866e-01	0.1013186
4	0.4962763	0.89338399	-1.17885269	0.9882139	-4.605165e-01	0.5738773
6	0.4962763	0.89338399	-1.17885269	0.9882139	-2.257935e-01	1.4464296
7	0.4962763	0.89338399	-1.17885269	0.9882139	-1.118613e-01	-1.5715088
8	0.4962763	0.89338399	-1.17885269	0.9882139	4.366633e-05	-0.9545875
9	0.4962763	0.89338399	-1.17885269	0.9882139	1.100809e-01	-0.4042598
10	0.4962763	0.89338399	-1.17885269	0.9882139	2.183890e-01	0.1013186
11	0.4962763	0.89338399	-1.17885269	0.9882139	3.250890e-01	0.5738773

1-10 of 6,467,236 rows | 1-8 of 12 columns Previous **1** 2 3 4 5 6 ... 78 Next

4. Check balance of target feature

Hide

```
# check target feature in train set

print("unique value:")
```

```
[1] "unique value:"
```

Hide

```
table(train$LATE)
```

```
       0       1
5709295  757941
```

Hide

```
print("percent:")
```

```
[1] "percent:"
```

Hide

```
table(train$LATE)[2]/table(train$LATE)[1] * 100
```

```
       1
13.27556
```

We can see that target feature is imbalance, so that we will approach difference from original

Try some resampling algorithms:

5. Downsampling

Hide

```
# down sample
set.seed(9560)
down_train <- downSample(x = train[, -ncol(train)], y = train$LATE)
# change target label to numeric
labels <- down_train$Class
y <- recode(labels, '0' = 0, "1" = 1)

table(down_train$Class)
```

```
     0      1
757941 757941
```

So out data downsampled to around 1,500,000 rows

We will then apply xgboost, for quick checking, we will just apply 20 round

Hide

```
set.seed(42)
system.time(
  xgb <- xgboost(data = data.matrix(down_train[,-ncol(down_train)]),
                 label = y,
                 eta = 0.6, # first learning rate
                 gamma = 0.1, # another learning rate
                 max_depth = 10, # max depth of each trees
                 nrounds = 20, # number of rounds to train
                 objective = "binary:logistic", # binary target
                 colsample_bytree = 0.6, # use 60% to resampling each rows
                 # verbose = 1,
                 nthread = 5, # control number of core to running parallel
  )
)
```

```
[10:24:53] WARNING: amalgamation/../src/learner.cc:1061: Starting in XGBoost 1.3.0, the
default evaluation metric used with the objective 'binary:logistic' was changed from 'er
ror' to 'logloss'. Explicitly set eval_metric if you'd like to restore the old behavior.
[1] train-logloss:0.628076
[2] train-logloss:0.585732
[3] train-logloss:0.569754
[4] train-logloss:0.552662
[5] train-logloss:0.544978
[6] train-logloss:0.541205
[7] train-logloss:0.536056
[8] train-logloss:0.533348
[9] train-logloss:0.525659
[10]    train-logloss:0.521849
[11]    train-logloss:0.520142
[12]    train-logloss:0.517126
[13]    train-logloss:0.511297
[14]    train-logloss:0.509166
[15]    train-logloss:0.507348
[16]    train-logloss:0.502363
[17]    train-logloss:0.501249
[18]    train-logloss:0.499082
[19]    train-logloss:0.497648
[20]    train-logloss:0.494267
   user  system elapsed
165.034   0.423  43.556
```

Hide

```
# save model
xgb.save(xgb, "/home/apolong72/ds/r/data/airline_2/model/xgb.model")
```

```
[1] TRUE
```

Predict in test set

Hide

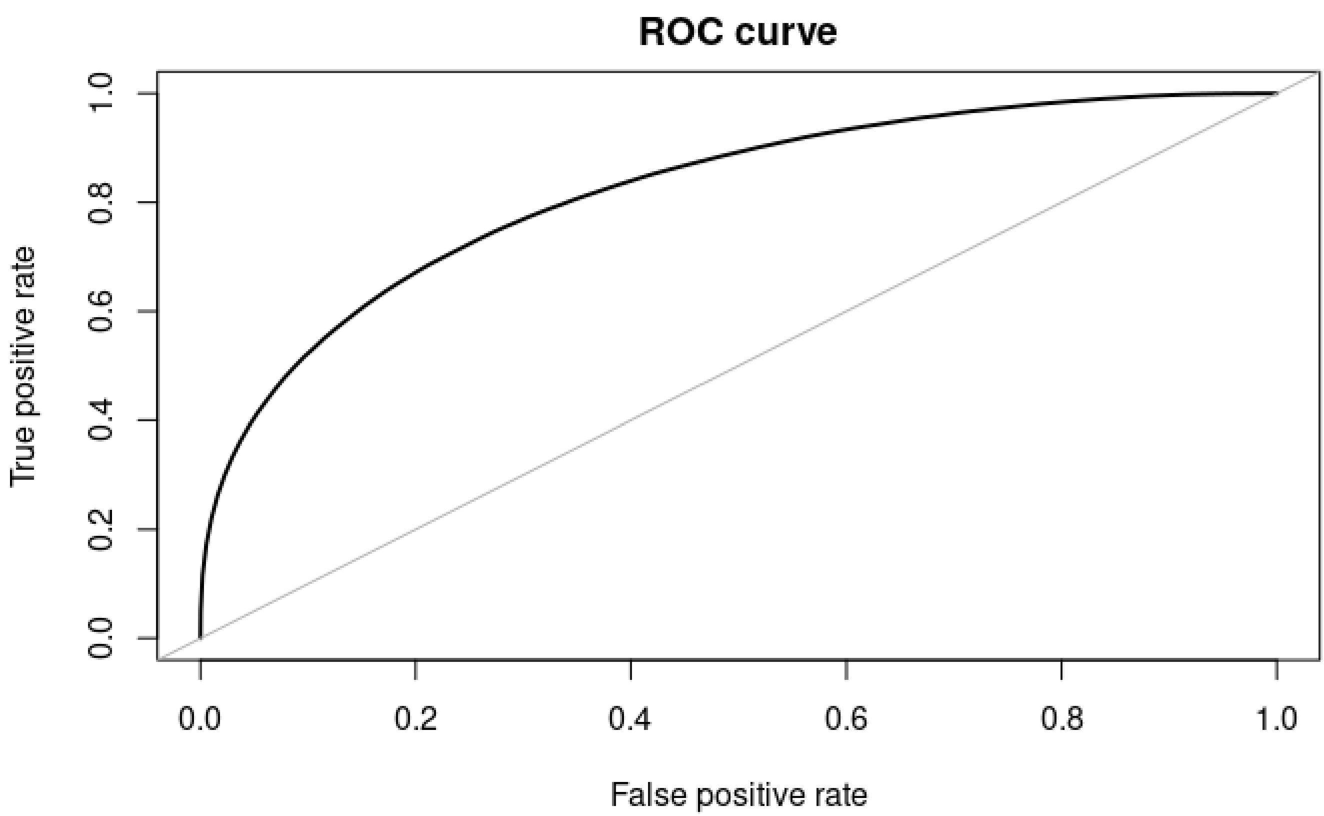
```
xgb_pred <- predict(xgb, data.matrix(test[,-length(test)]))

head(xgb_pred)
```

```
[1] 0.46753800 0.71803403 0.04974341 0.10220948 0.13261186 0.37312338
```

Plot ROC

Hide

```
roc_auc <- roc.curve(test$LATE, xgb_pred, plotit = TRUE)
```



In general, we will choose threshold have farest distant, we can see that around False positive ~= 0.23 have farest distant

Hide

```
# lets check false positive rate
roc_auc$false.positive.rate[roc_auc$false.positive.rate < 0.3]
```

```
 [1] 0.2942843721 0.2837967356 0.2734399385 0.2629586075 0.2527610244 0.2426060035 0.232
3690109 0.2221982263 0.2120274416 0.2020048363
[11] 0.1920358279 0.1821062289 0.1722743653 0.1624582654 0.1527824631 0.1431507994 0.133
6720442 0.1241523032 0.1146940410 0.1055746998
[21] 0.0964017618 0.0872808442 0.0784168761 0.0697515315 0.0612769285 0.0529441994 0.044
8416214 0.0369881110 0.0295113546 0.0225106642
[31] 0.0160522474 0.0103032004 0.0054385008 0.0019310619 0.0002427621 0.0000000000
```

Hide

```
# we will choose `0.2323690109`

# we cannot use logical `==` because `roc_auc$false.positive.rate` auto round some digi
t, hard to catch up with what, so we will use logical `<=`
get_num <- which(roc_auc$false.positive.rate <= 0.232369011)[1]
```

Hide

```
# check values of false positive
roc_auc$false.positive.rate[get_num]
```

```
[1] 0.232369
```

Hide

```
# check values of true positive
roc_auc$true.positive.rate[get_num]
```

```
[1] 0.7058397
```

Hide

```
# get threshold
th <- roc_auc$thresholds[get_num]
th
```

```
[1] 0.505042
```

Ok we had the best threshold until now , we will use it for calculate the accuracy of test set

Hide

```
# get predicted binary
pred_label <- (xgb_pred > th) * 1
distinct_pred <- table(test$LATE == pred_label)
distinct_pred
```

```
 FALSE   TRUE 
172180 546402 
```

Hide

```
# calculate accuracy
acc <- 100 - (distinct_pred[1] / distinct_pred[2] * 100)
print(paste("accuracy is:", as.character(unname(acc)),"%", sep = " "))
```

```
[1] "accuracy is: 68.488402311851 %"
```

6. Upsampling

Similar to downsampling

Hide

```
# down sample
set.seed(9560)
up_train <- upSample(x = train[, -ncol(train)], y = train$LATE)

# change target label to numeric
labels <- up_train$Class
y <- recode(labels, '0' = 0, "1" = 1)

table(up_train$Class)
```

```
      0       1 
5709295 5709295 
```

So out data downsampled to around 11,500,000 rows

You can now apply same method of up sample algorithms, I will not run it, because it will take so much time.

7. ROSE algorithms and GridsearchCV

We will down sampling to 50,000 samples to apply Gridsearch CV

Hide

```
# get sample data ( around 50,000)
split <- sample.split(train$LATE, SplitRatio = 0.01)
sample_df <- subset(train, split)

# split to train/test
split2 <- sample.split(sample_df$LATE, SplitRatio = 0.7)
s_train <- subset(sample_df, split2)
s_test <- subset(sample_df, !split2)

s_train
```

	OP_UNIQUE_CAR... <dbl>	ORIGIN <dbl>	DEST <dbl>	MONTH <dbl>	DAY_OF_MO... <dbl>	DAY_OF_... <dbl>
547	0.4962763	1.221785373	-0.795057446	0.9882139	2.183890e-01	0.101318
550	0.4962763	1.221785373	-0.795057446	0.9882139	5.340798e-01	1.446429
808	-0.8130907	1.381570176	0.313958969	0.9882139	-9.652974e-01	-1.571508
941	-0.8130907	0.651903868	0.120618752	0.9882139	-8.337389e-01	-0.954587
1028	-0.8130907	0.642815487	-0.533154708	0.9882139	-8.337389e-01	-0.954587
1115	-0.8130907	-0.686841311	-1.411537666	0.9882139	-8.337389e-01	-0.954587
1211	-0.8130907	-1.097889138	-1.770853482	0.9882139	-8.337389e-01	-0.954587
1319	-0.8130907	-0.686841311	0.639922141	0.9882139	-8.337389e-01	-0.954587
1508	-0.8130907	-0.441560987	-0.690663741	0.9882139	-8.337389e-01	-0.954587
1788	-0.8130907	-1.770296610	1.669051077	0.9882139	-8.337389e-01	-0.954587

1-10 of 45,270 rows | 1-7 of 12 columns Previous **1** 2 3 4 5 6 ... 78 Next

We will apply ROSE algorithms to resampling data

(ROSE and SMOTE algorithms is a very popular resampling algorithms, and they usually outperform original algorithms like Upsampling..)

Hide

```
# ROSE

# down sample
set.seed(9560)
rose_s_train <- ROSE(LATE ~., data=s_train)$data

# change target label to numeric
labels <- rose_s_train$LATE
y <- recode(labels, "0" = 0, "1" = 1)

table(rose_s_train$LATE)
```

```
    0     1 
22886 22384 
```

As you can see, ROSE keep rows remain, but balance target binary.

Next, we will set parameter for grid search

Hide

```
searchGridSubCol <- expand.grid(subsample = c(0.5, 1),
                                colsample_bytree = c(0.5, 0.6), # resampling each rows
                                max_depth = 10, # max depth of each trees
                                eta = c(0.2, 0.3, 0.5), # first learning rate
                                lambda = c(0.5, 1),
                                # min_child_weight = c(1,3),
                                gamma = c(0.1, 0.5) # another learning rate

)
```

Apply grid search

Hide

```
system.time(
rmseErrorsHyperparameters <- apply(searchGridSubCol, 1, function(parameterList){

  #Extract Parameters to test
  currentSubsampleRate <- parameterList[["subsample"]]
  currentColsampleRate <- parameterList[["colsample_bytree"]]
  currentDepth <- parameterList[["max_depth"]]
  currentEta <- parameterList[["eta"]]
  # currentMinChild <- parameterList[["min_child_weight"]]
  currentLambda <- parameterList[["lambda"]]
  currentGamma <- parameterList[["gamma"]]

  # Apply
  xgboostModelCV <- xgb.cv(data = data.matrix(rose_s_train[,-ncol(rose_s_train)]),
                           label = y,
                           nrounds = 50,
                           nfold = 5,
                           showsd = TRUE,
                           metrics = "rmse",
                           verbose = FALSE,
                           print_every_n = 10,
                           booster = "gbtree",
                           early_stopping_rounds = 10,
                           "eval_metric" = "rmse",
                           "objective" = "binary:logistic",
                           "max.depth" = currentDepth,
                           "eta" = currentEta,
                           "subsample" = currentSubsampleRate,
                           "colsample_bytree" = currentColsampleRate,
                           # "min_child_weight" = currentMinChild,
                           "gamma" = currentGamma,
                           "lambda" = currentLambda
                           )

  xvalidationScores <- as.data.frame(xgboostModelCV$evaluation_log)

  # get rmse
  rmse <- tail(xvalidationScores$test_rmse_mean, 1)
  trmse <- tail(xvalidationScores$train_rmse_mean,1)

  output <- return(c(rmse, trmse, currentSubsampleRate, currentColsampleRate, currentDep
th, currentEta, currentGamma, currentLambda))}

  )
)
```

```
   user  system elapsed 
948.300   2.349 182.280 
```

Create dataframe for results

Hide

```
output <- as.data.frame(t(rmseErrorsHyperparameters))
varnames <- c("TestRMSE", "TrainRMSE", "SubSampRate", "ColSampRate", "Depth", "eta", "Ga
mma", "Lambda")
names(output) <- varnames

output[order(output$TestRMSE),]
```

	TestRMSE <dbl>	TrainRMSE <dbl>	SubSampRate <dbl>	ColSampRate <dbl>	Depth <dbl>	eta <dbl>	Gam... <dbl>	Lambda <dbl>
16	0.4553010	0.3690764	1.0	0.6	10	0.2	0.1	1.0
38	0.4561654	0.3747560	1.0	0.5	10	0.2	0.5	1.0
4	0.4565702	0.3650474	1.0	0.6	10	0.2	0.1	0.5
28	0.4566676	0.3647398	1.0	0.6	10	0.2	0.5	0.5
26	0.4566844	0.3666248	1.0	0.5	10	0.2	0.5	0.5
40	0.4568618	0.3632208	1.0	0.6	10	0.2	0.5	1.0
14	0.4573782	0.3655976	1.0	0.5	10	0.2	0.1	1.0
2	0.4573912	0.3696224	1.0	0.5	10	0.2	0.1	0.5
8	0.4585760	0.3712124	1.0	0.6	10	0.3	0.1	0.5
15	0.4586070	0.3904512	0.5	0.6	10	0.2	0.1	1.0

1-10 of 48 rows Previous **1** 2 3 4 5 Next

We pick the best params (lowest RMSE test)

Hide

```
best_param <- output[order(output$TestRMSE),][1,-c(1,2)]
best_param
```

	SubSampRate <dbl>	ColSampRate <dbl>	Depth <dbl>	eta <dbl>	Gamma <dbl>	Lambda <dbl>
16	1	0.6	10	0.2	0.1	1

1 row

Ok now, we will use ROSE algorithms apply to original data, and use best params to predict target

8. ROSE algorithm

Hide

```
# ROSE

# down sample
set.seed(9560)
rose_train <- ROSE(LATE ~., data=train)$data

# change target label to numeric
labels <- rose_train$LATE
y <- recode(labels, "0" = 0, "1" = 1)

table(rose_train$LATE)
```

```

      0       1 
3231682 3235554 
```

We will apply xgboost, for quick checking, we will just apply 20 round

Hide

```
set.seed(42)
system.time(
  xgb_rose <- xgboost(data = data.matrix(rose_train[,-ncol(rose_train)]),
                 label = y,
                 eta = best_param$eta, # first learning rate
                 gamma = best_param$Gamma, # another learning rate
                 max_depth = best_param$Depth, # max depth of each trees
                 nrounds = 20, # number of rounds to train
                 objective = "binary:logistic", # binary target
                 colsample_bytree = best_param$ColSampRate, # resampling each rows
                 lambda = best_param$Lambda,
                 subsample = best_param$SubSampRate,
                 # verbose = 1,
                 nthread = 5, # control number of core to running parallel
  )
)
```

```
[11:32:46] WARNING: amalgamation/../src/learner.cc:1061: Starting in XGBoost 1.3.0, the
default evaluation metric used with the objective 'binary:logistic' was changed from 'er
ror' to 'logloss'. Explicitly set eval_metric if you'd like to restore the old behavior.
[1] train-logloss:0.675101
[2] train-logloss:0.654993
[3] train-logloss:0.640569
[4] train-logloss:0.627368
[5] train-logloss:0.622839
[6] train-logloss:0.619219
[7] train-logloss:0.608793
[8] train-logloss:0.606864
[9] train-logloss:0.603579
[10]	train-logloss:0.601265
[11]	train-logloss:0.600244
[12]	train-logloss:0.597531
[13]	train-logloss:0.593087
[14]	train-logloss:0.591079
[15]	train-logloss:0.588849
[16]	train-logloss:0.587778
[17]	train-logloss:0.586715
[18]	train-logloss:0.586158
[19]	train-logloss:0.583278
[20]	train-logloss:0.582245
    user   system  elapsed 
1624.245    4.372  410.355 
```

Hide

```
# save model
xgb.save(xgb_rose, "/home/apolong72/ds/r/data/airline_2/model/xgb_rose.model")
```

```
[1] TRUE
```

Predict in test set

Hide

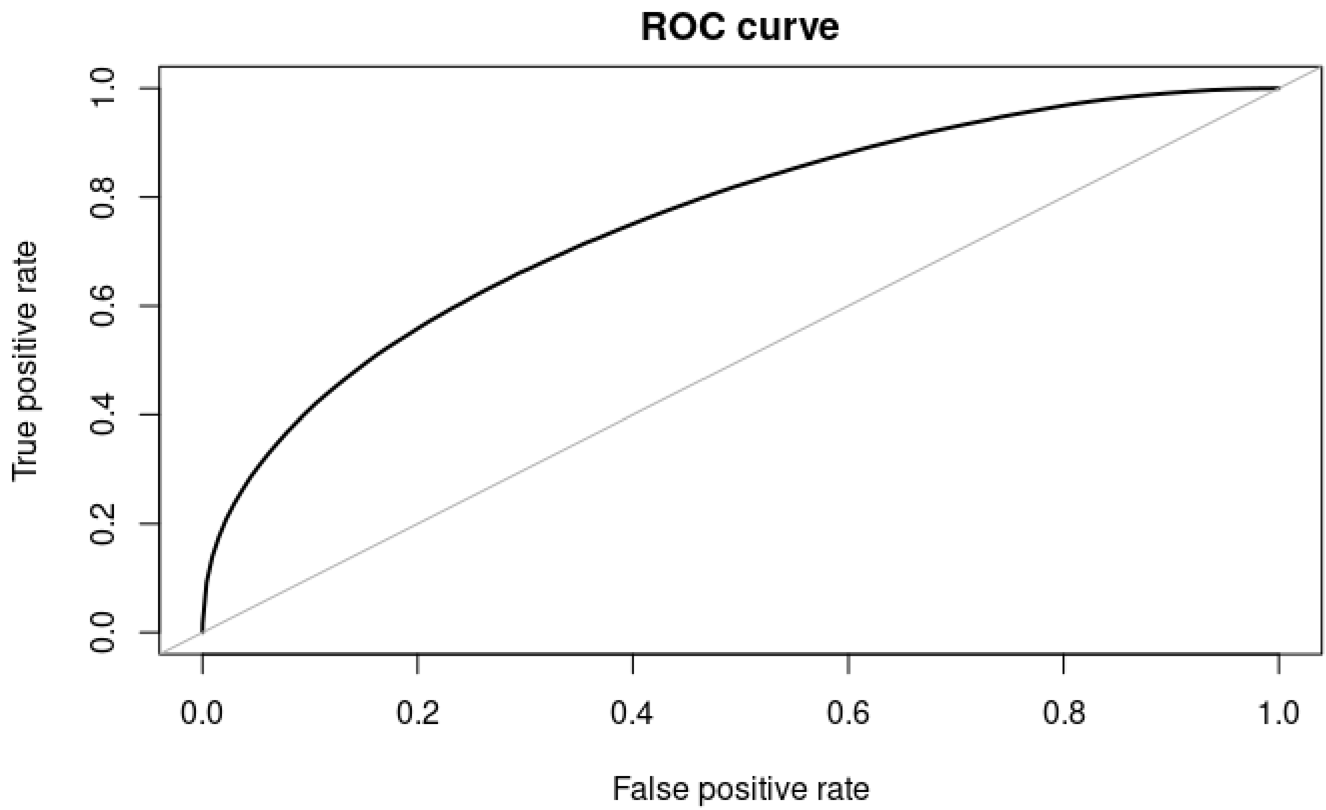
```
xgb_pred_rose <- predict(xgb_rose, data.matrix(test[,-length(test)]))

head(xgb_pred_rose)
```

```
[1] 0.4169802 0.5130195 0.1813435 0.1611036 0.2298483 0.4079071
```

Plot ROC

Hide

```
roc_auc <- roc.curve(test$LATE, xgb_pred_rose, plotit = TRUE)
```



In general, we will choose threshold have farest distant, we can see that around False positive ~= 0.3 have farest distant

Hide

```
# lets check false positive rate
roc_auc$false.positive.rate[roc_auc$false.positive.rate <= 0.3]
```

```
 [1] 0.2939281109 0.2835571263 0.2732365858 0.2629507256 0.2527137331 0.2423805816 0.232
1688111 0.2219397004 0.2118067488 0.2019181356
[11] 0.1919239051 0.1820620903 0.1722065811 0.1625087095 0.1528202962 0.1433431174 0.133
8060363 0.1243414685 0.1150156219 0.1056992336
[21] 0.0965010735 0.0875898141 0.0787557971 0.0700589250 0.0614172260 0.0531065662 0.044
7076293 0.0368714591 0.0293552933 0.0219447448
[31] 0.0152971628 0.0091871254 0.0038731584 0.0001734015 0.0000000000
```

Hide

```
# we will choose `0.2939281109`

# we cannot use logical `==` because `roc_auc$false.positive.rate` auto round some digi
t, hard to catch up with what, so we will use logical `<=`
get_num <- which(roc_auc$false.positive.rate <= 0.3)[1]
```

Hide

```
# check values of false positive
roc_auc$false.positive.rate[get_num]
```

```
[1] 0.2939281
```

Hide

```
# check values of true positive
roc_auc$true.positive.rate[get_num]
```

```
[1] 0.6598152
```

Hide

```
# get threshold
th <- roc_auc$thresholds[get_num]
th
```

```
[1] 0.4625081
```

We had the best threshold until now , we will use it for calculate the accuracy of test set

Hide

```
# get predicted binary
pred_label <- (xgb_pred > th) * 1
distinct_pred <- table(test$LATE == pred_label)
distinct_pred
```

```


 FALSE   TRUE 
204174 514408 
```

Hide

```
# calculate accuracy
acc <- 100 - (distinct_pred[1] / distinct_pred[2] * 100)
print(paste("accuracy is:", as.character(unname(acc)),"%", sep = " "))
```

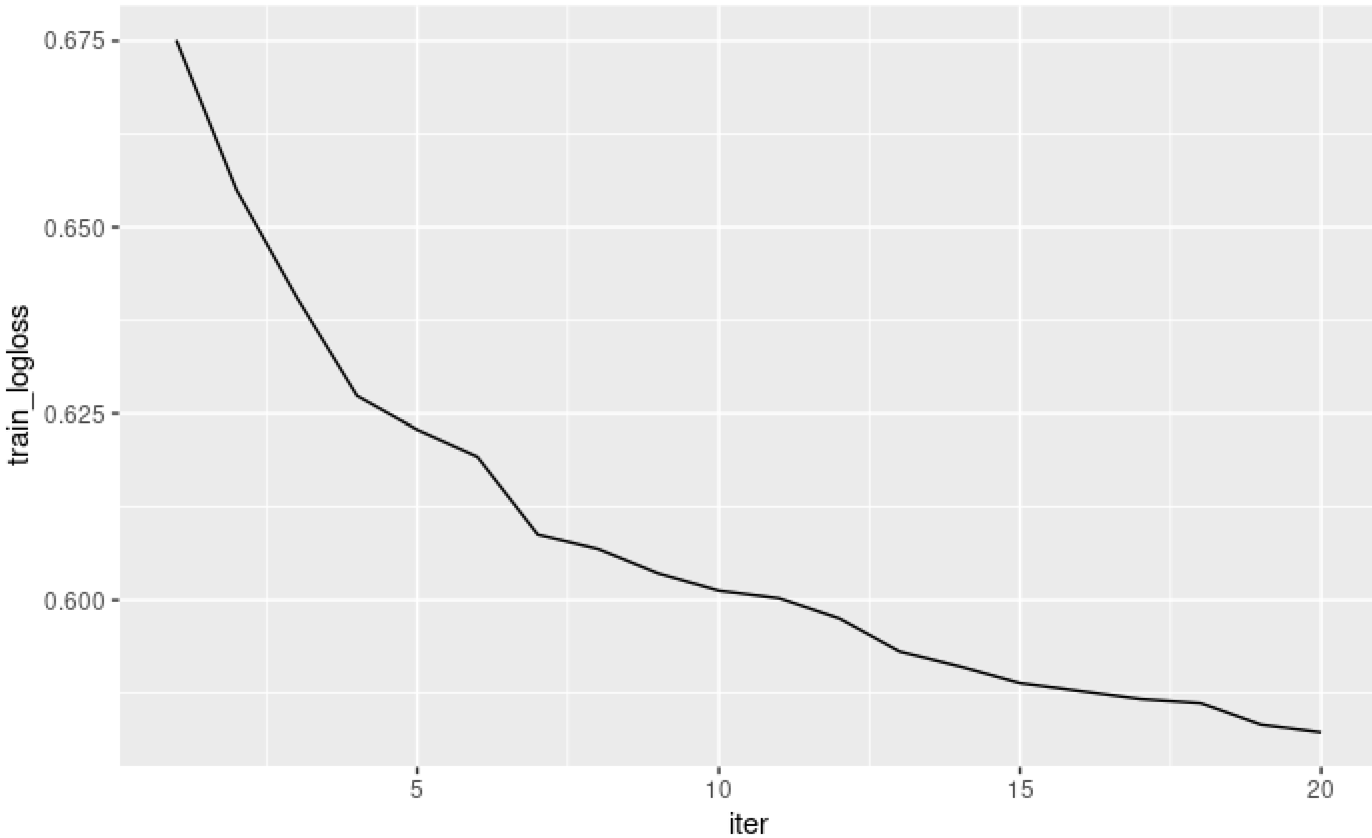
```
[1] "accuracy is: 60.3089376526026 %"
```

We can see that prediction is not too good, its because we set parameter `route` too small, and `learning rate` is quite small too.

We will then plot a chart to prove:

Hide

```
ggplot(data=xgb_rose$evaluation_log, aes(x=iter, y=train_logloss)) +
  geom_line()
```



Easy to see that loss is potentially reduce more, so in this case, we just need to incease number of rounds, so that our model will be better

END