Exploratory data analysis on TITANIC dataset

Data importing

Import the data into `r` by using function `read.csv`, we use the parameter `na.string` to specify the pattern of missing values.

```
df = read.csv('titanic_train.csv', na.strings = c("NA", "NaN", "", " "))
```

You also see some properties of dataset with `dim()` and `str()`.

```
dim(df)
```

```
## [1] 891  11
```

```
str(df)
```

```
## 'data.frame':    891 obs. of  11 variables:
##  $ survived: int  0 1 1 1 0 0 0 0 1 1 ...
##  $ pclass  : int  3 1 3 1 3 3 1 3 3 2 ...
##  $ name    : Factor w/ 891 levels "Abbing, Mr. Anthony",..: 109 191 358 277 16 559 520 629 417 581 .
##  $ sex     : Factor w/ 2 levels "female","male": 2 1 1 1 2 2 2 2 1 1 ...
##  $ age     : num  22 38 26 35 35 NA 54 2 27 14 ...
##  $ sibsp   : int  1 1 0 1 0 0 0 3 0 1 ...
##  $ parch   : int  0 0 0 0 0 0 0 1 2 0 ...
##  $ ticket  : Factor w/ 681 levels "110152","110413",..: 524 597 670 50 473 276 86 396 345 133 ...
##  $ fare    : num  7.25 71.28 7.92 53.1 8.05 ...
##  $ cabin   : Factor w/ 147 levels "A10","A14","A16",..: NA 82 NA 56 NA NA 130 NA NA NA ...
##  $ embarked: Factor w/ 3 levels "C","Q","S": 3 1 3 3 3 2 3 3 3 1 ...
```

The `dim()` function gives us a vector of 2 items including the number of rows/records/observation and the number of columns/features/fields. Meanwhile, function `str()` shows the basic characteristics of each columns such as data type and some values in the column.

Some common statistics are also calculated by using `summary()` function:

```
summary(df)
```

```
##     survived          pclass
##  Min.   :0.0000   Min.   :1.000
##  1st Qu.:0.0000   1st Qu.:2.000
##  Median :0.0000   Median :3.000
##  Mean   :0.3838   Mean   :2.309
##  3rd Qu.:1.0000   3rd Qu.:3.000
##  Max.   :1.0000   Max.   :3.000
##
##                                     name         sex           age
##  Abbing, Mr. Anthony                   :  1   female:314   Min.   : 0.42
##  Abbott, Mr. Rossmore Edward           :  1   male  :577   1st Qu.:20.12
##  Abbott, Mrs. Stanton (Rosa Hunt)      :  1                Median :28.00
##  Abelson, Mr. Samuel                   :  1                Mean   :29.70
##  Abelson, Mrs. Samuel (Hannah Wizosky):  1                3rd Qu.:38.00
##  Adahl, Mr. Mauritz Nils Martin        :  1                Max.   :80.00
##  (Other)                               :885                NA's   :177
##      sibsp           parch             ticket         fare
##  Min.   :0.000   Min.   :0.0000   1601    :  7   Min.   :  0.00
```

```
##  1st Qu.:0.000   1st Qu.:0.0000   347082  :  7   1st Qu.:  7.91
##  Median :0.000   Median :0.0000   CA. 2343:  7   Median : 14.45
##  Mean   :0.523   Mean   :0.3816   3101295 :  6   Mean   : 32.20
##  3rd Qu.:1.000   3rd Qu.:0.0000   347088  :  6   3rd Qu.: 31.00
##  Max.   :8.000   Max.   :6.0000   CA 2144 :  6   Max.   :512.33
##                                   (Other) :852
##          cabin     embarked
##  B96 B98    :  4   C   :168
##  C23 C25 C27:  4   Q   : 77
##  G6         :  4   S   :644
##  C22 C26    :  3   NA's:  2
##  D          :  3
##  (Other)    :186
##  NA's       :687
```

The function `object.size()` returns the memory allocation of data frame.

```
object.size(df)
```

```
## 187304 bytes
```

To reduct the size of data frame, we can specify the data type of each column after reading:

```
print(paste('Size of df before optimize:', object.size(df)))
```

```
## [1] "Size of df before optimize: 187304"
```

```
df$survived = as.logical(df$survived)
df$ticket = as.numeric(df$ticket)
print(paste('Size of df after optimize:', object.size(df)))
```

```
## [1] "Size of df after optimize: 145160"
```

Data cleaning

Detecting missing values

In `R`, function `is.na()` return the logical vector which indicate if the value is missing or not. Count the number of missing values in the dataset:

```
sum(is.na(df))
```

```
## [1] 866
```

To count missing values per columns:

```
colSums(is.na(df))
```

```
## survived   pclass     name      sex      age    sibsp    parch   ticket
##        0        0        0        0      177        0        0        0
##     fare    cabin embarked
##        0      687        2
```

Handling missing values

Drop the columns:

```
df <- subset(df, select=-c(age, cabin))
print(dim(df))
```

```
## [1] 891   9
```

Drop the rows:

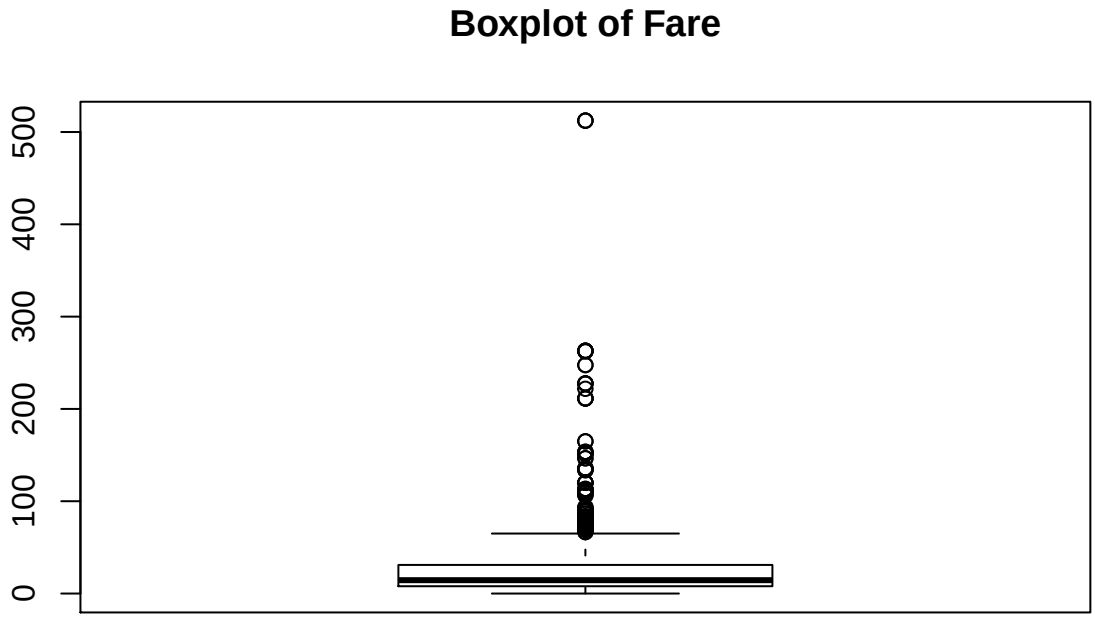
```
df <- subset(df, !is.na(df$embarked))
print(dim(df))
```

```
## [1] 889   9
```

Detecting the outliers

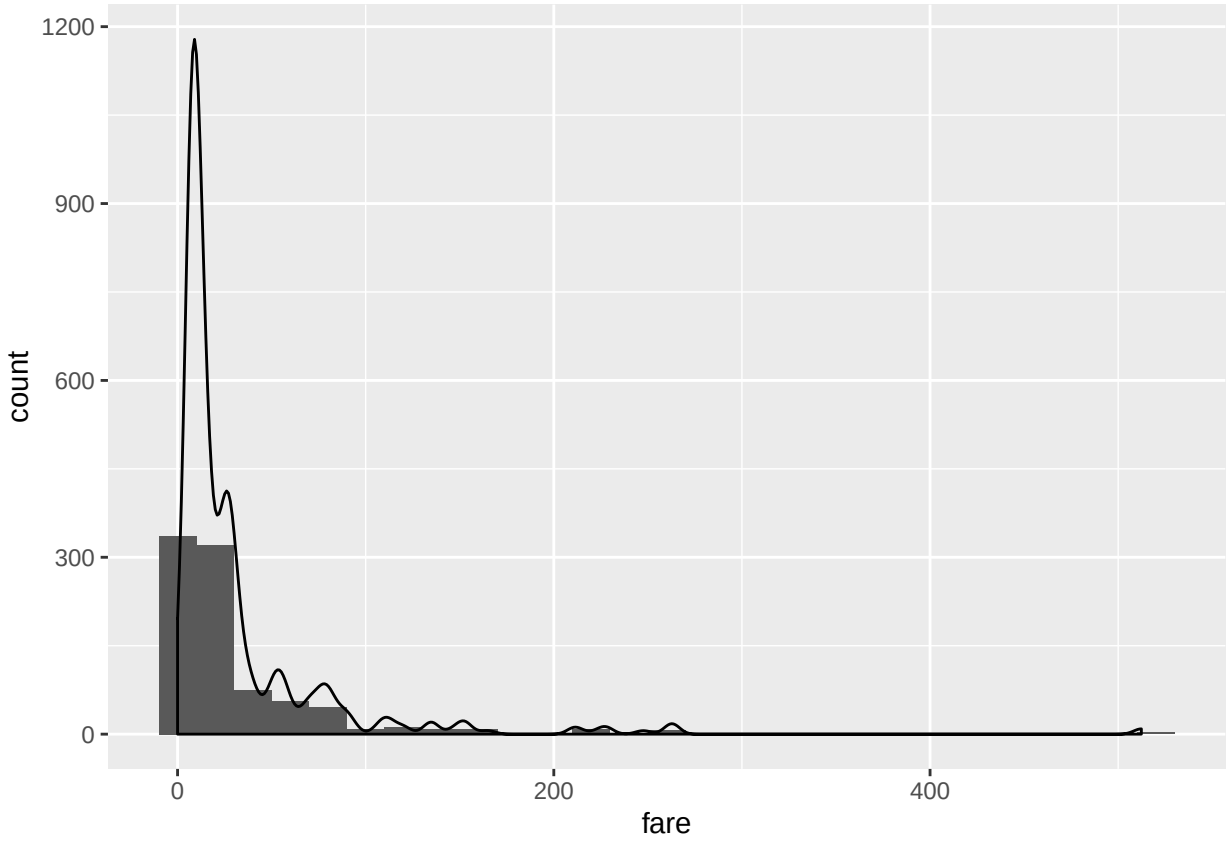
Visualize the variable by using Boxplot to investigate the potential outliers:

```
boxplot(df$fare,
        main='Boxplot of Fare')
```



Or using the histogram:

```
library(ggplot2)
g <- ggplot(df)
g <- g + geom_histogram(aes(x=fare), binwidth = 20)
g <- g + geom_density(aes(x=fare, y=..count.. * 30))
g
```

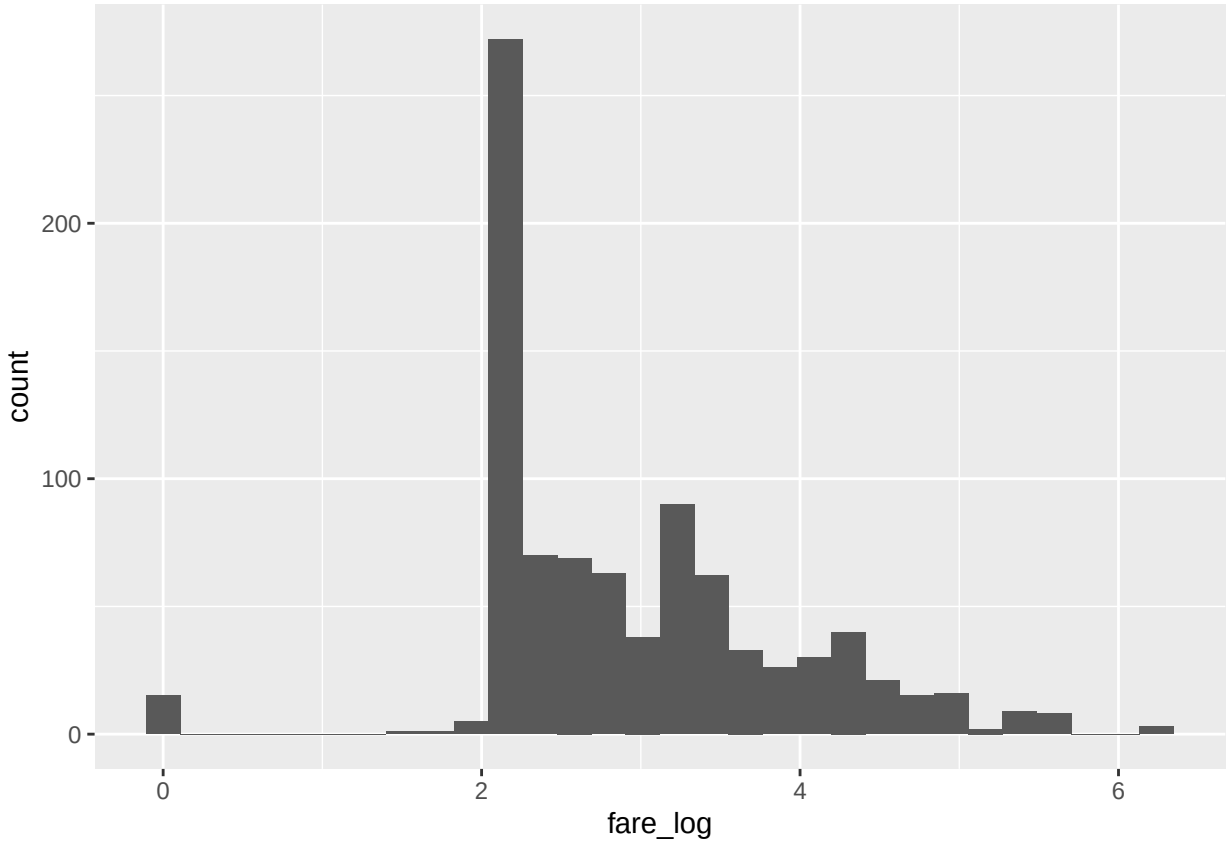


Data transformation

Apply log-transformation to `fare` features to normalize this variable:

```
fare_log <- log1p(df$fare)
g <- ggplot()
g <- g + geom_histogram(aes(x=fare_log))
g
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Remove the outliers using Z-Score

Calculating Z-Score of data series and remove the records whose Z-score greater than 2 or less than -2:

```
fare_zscore <- (fare_log - mean(fare_log))/sd(fare_log)
outlier <- abs(fare_zscore) > 2
df_non_outliers <- subset(df, !outlier)
print(dim(df_non_outliers))
```

```
## [1] 836   9
```

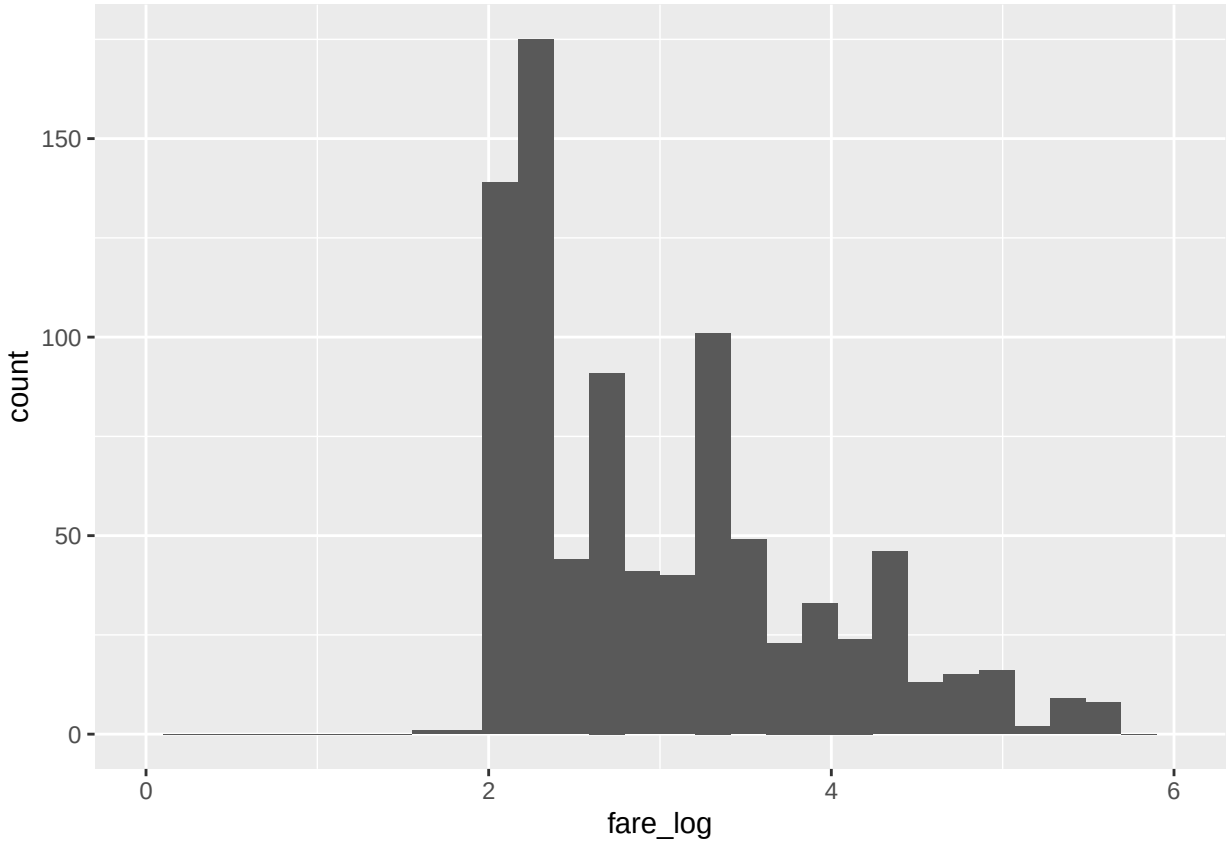
Visualize the data after remove outliers:

```
g <- ggplot()
g <- g + geom_histogram(aes(x=fare_log)) + xlim(0,6)
g
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## Warning: Removed 3 rows containing non-finite values (stat_bin).
```

```
## Warning: Removed 2 rows containing missing values (geom_bar).
```



Bivariate analysis

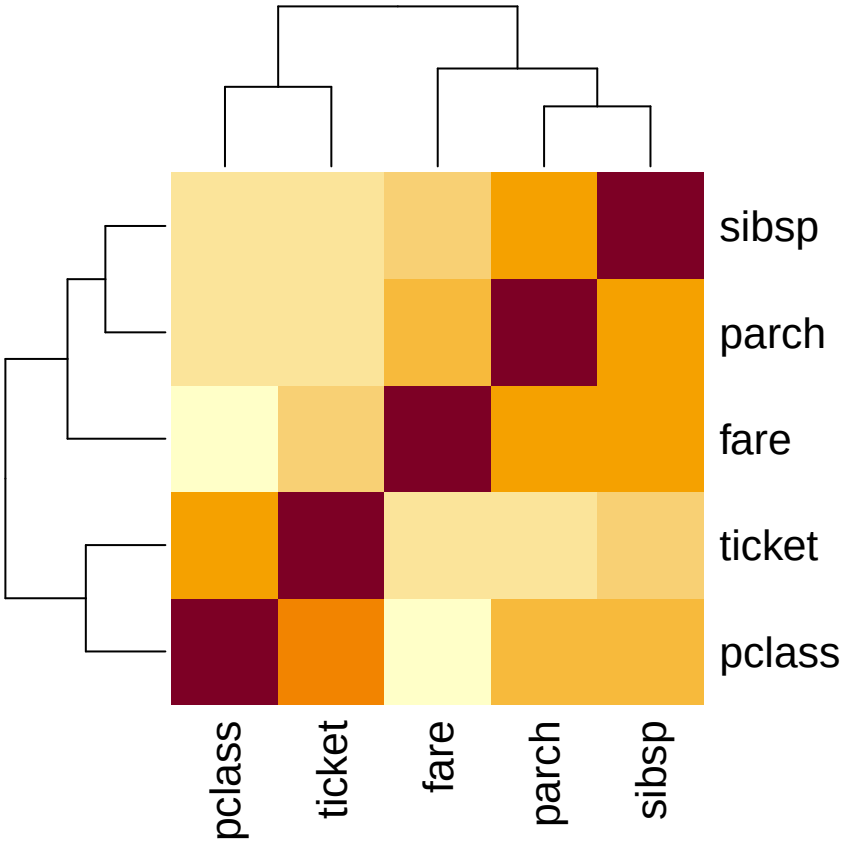
Calculating the correlation matrix:

```
df_numeric <- df[sapply(df, is.numeric)]
df_cor <- cor(df_numeric)
df_cor
```

```
##             pclass      sibsp      parch      ticket        fare
## pclass  1.00000000 0.08165562 0.01682449  0.31623767 -0.54819329
## sibsp   0.08165562 1.00000000 0.41454164  0.07802779  0.16088685
## parch   0.01682449 0.41454164 1.00000000  0.01824859  0.21753204
## ticket  0.31623767 0.07802779 0.01824859  1.00000000 -0.01064211
## fare   -0.54819329 0.16088685 0.21753204 -0.01064211  1.00000000
```

We can also visualize the correlation matrix by heatmap

```
heatmap(df_cor)
```


sibsp
parch
fare
ticket
pclass
pclass
ticket
fare
parch
sibsp