Data Reduction

Hazim Fitri

2024-12-17

Contents

Data reduction is a technique that we can use when the number of the data is too large and using full data requires a costly and time-consuming computational method. There are two techniques to reduce the data:

1) Dimension Data Reduction
2) Numerosity Data Reduction

Dimensional

Removing Attributes

A manual way to reduce attribute is by domain knowledge. If the attribute seem very similar or not relevant, we can just simply remove it. Other way is to see whether the attribute is significant or not. This can be done through regression as below.

```r
library(ISLR)
```

```
## Warning: package 'ISLR' was built under R version 4.4.2
```

```r
data(package='ISLR')
# Major league baseball data from the 1986 and 1987 seasons
data("Hitters")
hitters2 = na.omit(Hitters)
model.f = lm(Salary~., data=hitters2)
summary(model.f)
```

```
##
## Call:
## lm(formula = Salary ~ ., data = hitters2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -907.62 -178.35  -31.11  139.09 1877.04
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  163.10359   90.77854   1.797 0.073622 .
## AtBat         -1.97987    0.63398  -3.123 0.002008 **
## Hits           7.50077    2.37753   3.155 0.001808 **
## HmRun          4.33088    6.20145   0.698 0.485616
## Runs          -2.37621    2.98076  -0.797 0.426122
## RBI           -1.04496    2.60088  -0.402 0.688204
## Walks          6.23129    1.82850   3.408 0.000766 ***
## Years         -3.48905   12.41219  -0.281 0.778874
## CAtBat        -0.17134    0.13524  -1.267 0.206380
## CHits          0.13399    0.67455   0.199 0.842713
## CHmRun        -0.17286    1.61724  -0.107 0.914967
## CRuns          1.45430    0.75046   1.938 0.053795 .
## CRBI           0.80771    0.69262   1.166 0.244691
## CWalks        -0.81157    0.32808  -2.474 0.014057 *
## LeagueN       62.59942   79.26140   0.790 0.430424
## DivisionW   -116.84925   40.36695  -2.895 0.004141 **
## PutOuts        0.28189    0.07744   3.640 0.000333 ***
## Assists        0.37107    0.22120   1.678 0.094723 .
## Errors        -3.36076    4.39163  -0.765 0.444857
## NewLeagueN   -24.76233   79.00263  -0.313 0.754218
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 315.6 on 243 degrees of freedom
## Multiple R-squared:  0.5461, Adjusted R-squared:  0.5106
## F-statistic: 15.39 on 19 and 243 DF,  p-value: < 2.2e-16
```

As we can see from the above summary for linear model of salary with other variables, only certain variables can be considered significant as indicate by at least one '*' at the right of the column. This indicates that the variables are at least significance at alpha =0.05

```r
hitters3 = cbind(hitters2$AtBat, hitters2$Hits, hitters2$Walks, hitters2$CWalks,
                 hitters2$Division, hitters2$PutOuts)
head(hitters3)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]  315   81   39  375    2  632
## [2,]  479  130   76  263    2  880
## [3,]  496  141   37  354    1  200
## [4,]  321   87   30   33    1  805
## [5,]  594  169   35  194    2  282
## [6,]  185   37   21   24    1   76
```

Primary Component Analysis (PCA)

```r
reading = read.csv('./Data/READING120n.csv')
head(reading)
```

```
##   GEN rhyme Begsnd ABC LS Spelling COW
## 1   M    10     10   6  7        4   7
## 2   F    10     10  22 19        9  15
## 3   M     9     10  23 15        5   6
## 4   F     5     10  10  3        2   3
## 5   F     2     10   4  0        0   2
## 6   M     5      6  22  8       17   6
```

Remove non-numeric column.

```r
reading2 = reading[,-1]
head(reading2)
```

```
##   rhyme Begsnd ABC LS Spelling COW
## 1    10     10   6  7        4   7
## 2    10     10  22 19        9  15
## 3     9     10  23 15        5   6
## 4     5     10  10  3        2   3
## 5     2     10   4  0        0   2
## 6     5      6  22  8       17   6
```

```r
library(psych)
```

```
## Warning: package 'psych' was built under R version 4.4.2
```

```r
describe(reading2)
```

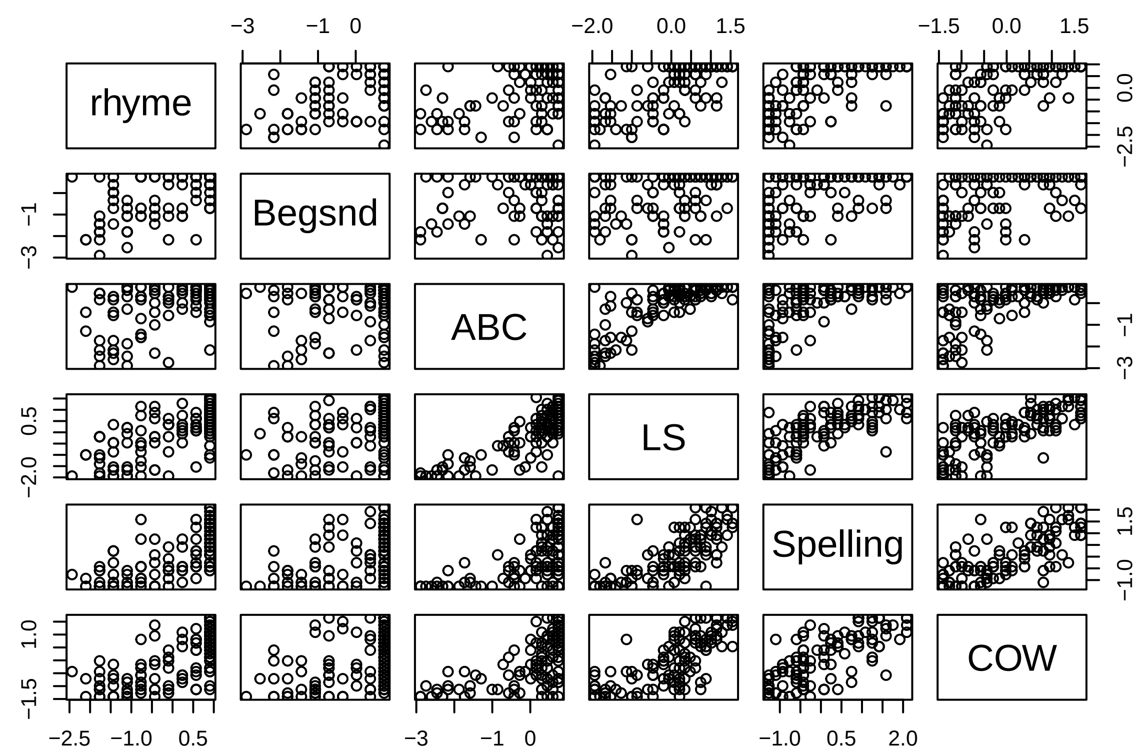
```
##          vars   n  mean   sd median trimmed  mad min max range  skew kurtosis
## rhyme       1 120  7.29 2.99      9    7.65 1.48   0  10    10 -0.65    -1.02
## Begsnd      2 120  7.94 2.74     10    8.41 0.00   0  10    10 -1.03    -0.23
## ABC         3 120 20.92 6.89     24   22.36 2.97   1  26    25 -1.54     1.19
```

```
## LS           4 120 14.46 7.45     16    14.92 7.41    0  26     26 -0.53     -0.77
## Spelling     5 120  7.55 5.96      6     7.18 7.41    0  20     20  0.39     -1.03
## COW          6 120 10.15 7.21     10     9.96 9.64    0  22     22  0.13     -1.33
##            se
## rhyme    0.27
## Begsnd   0.25
## ABC      0.63
## LS       0.68
## Spelling 0.54
## COW      0.66
```

```r
str(reading2)
```

```
## 'data.frame':    120 obs. of  6 variables:
##  $ rhyme   : int  10 10 9 5 2 5 8 4 3 9 ...
##  $ Begsnd  : int  10 10 10 10 10 6 5 3 7 10 ...
##  $ ABC     : int  6 22 23 10 4 22 25 26 18 26 ...
##  $ LS      : int  7 19 15 3 0 8 20 16 8 17 ...
##  $ Spelling: int  4 9 5 2 0 17 12 3 3 15 ...
##  $ COW     : int  7 15 6 3 2 6 4 0 0 15 ...
```

```r
z = scale(reading2)
pairs(~., data=z)
```



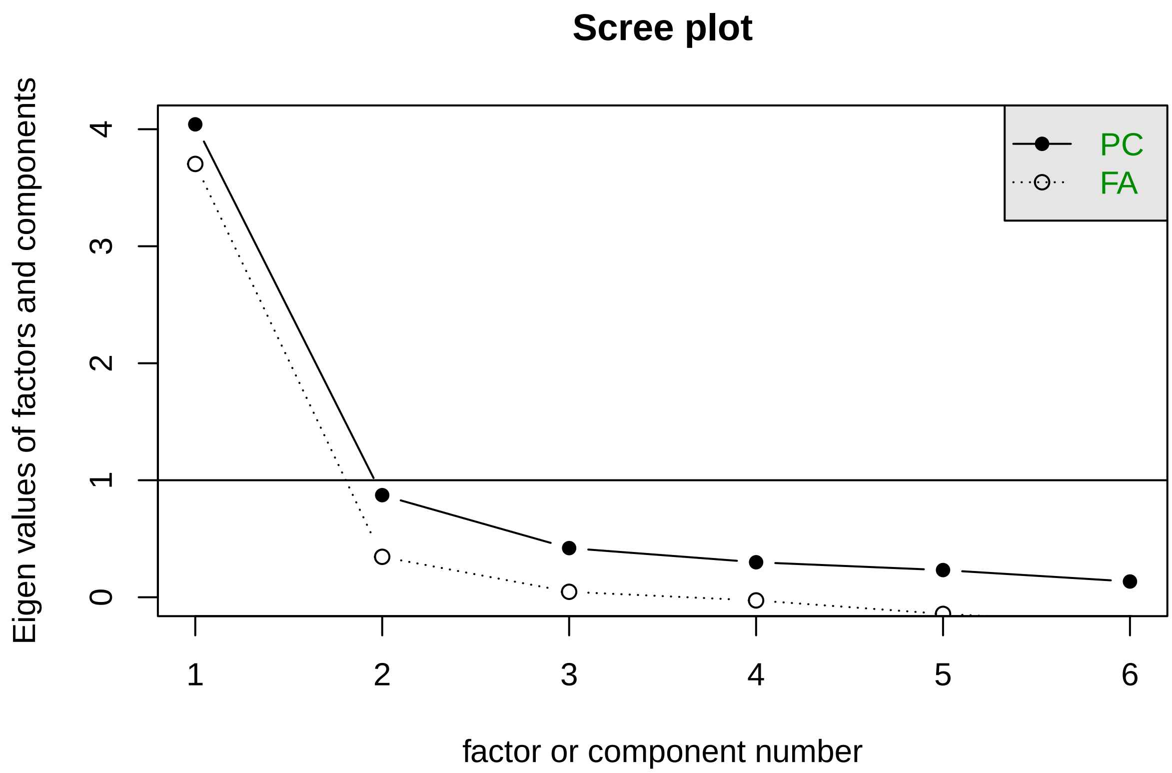
```r
cor_z = cor(z)
cor_z
```

```
##              rhyme    Begsnd       ABC        LS  Spelling       COW
## rhyme    1.0000000 0.6161831 0.4994385 0.6769710 0.6682135 0.6929980
## Begsnd   0.6161831 1.0000000 0.2850706 0.3467132 0.4688980 0.4694738
## ABC      0.4994385 0.2850706 1.0000000 0.7955943 0.5888044 0.5981786
## LS       0.6769710 0.3467132 0.7955943 1.0000000 0.7579600 0.7492896
## Spelling 0.6682135 0.4688980 0.5888044 0.7579600 1.0000000 0.7668598
## COW      0.6929980 0.4694738 0.5981786 0.7492896 0.7668598 1.0000000
```

```r
eigen(cor_z)
```

```
## eigen() decomposition
## $values
## [1] 4.0417265 0.8725973 0.4200022 0.2990629 0.2322152 0.1343960
##
## $vectors
##            [,1]        [,2]        [,3]        [,4]        [,5]        [,6]
## [1,] -0.4202540  0.29934149 -0.09269853  0.80020266 -0.12282334  0.26415587
## [2,] -0.3068973  0.75974276  0.43140561 -0.33291224  0.02990871 -0.17541132
## [3,] -0.3849778 -0.46782622  0.65714698 -0.08464742  0.06601473  0.43539111
## [4,] -0.4458305 -0.33461651  0.06528679  0.14407269 -0.13081189 -0.80444760
## [5,] -0.4358068 -0.03894126 -0.43902727 -0.43785381 -0.60459527  0.24199105
## [6,] -0.4385206 -0.02897612 -0.42005561 -0.17090073  0.77266730  0.06474189
```

```r
scree(cor_z)
```




```
# variance percentage for each variable in PCA
prop.var = eigen(cor_z)$values / length(eigen(cor_z)$values)
prop.var
```

```
## [1] 0.67362108 0.14543288 0.07000036 0.04984381 0.03870254 0.02239933
```

```
cumsum(prop.var)
```

```
## [1] 0.6736211 0.8190540 0.8890543 0.9388981 0.9776007 1.0000000
```

Interpretation :

- If keep 1 variable, we will be able to explain 67.4% of the data
- By keeping 2, we already able to explain 81.9% of the data.

```
v = eigen(cor_z)$vectors
y = scale(reading2) %*% eigen(cor_z)$vectors
colnames(y) = c('PCA1', 'PCA2', 'PCA3', 'PCA4', 'PCA5', 'PCA6')
head(y)
```

```
##             PCA1       PCA2         PCA3      PCA4        PCA5        PCA6
## [1,]   1.1193495  2.2262055 -0.802398485  0.8485456 -0.07816090 -0.20279050
```

```
## [2,] -1.3445992  0.5362251 -0.005566031  0.3270445  0.21458787 -0.21216009
## [3,] -0.1808963  0.6101463  0.904678375  0.4770719 -0.22322576 -0.04872692
## [4,]  2.2270880  1.6629857  0.079348607 -0.3737515  0.00991986 -0.07692304
## [5,]  3.3703245  1.9219481 -0.220657112 -0.9899433  0.22398274 -0.48736015
## [6,]  0.4270255 -0.5972152 -0.642725583 -1.1107536 -1.20664830  1.03409235
```

```r
#we only keeping PCA1 and PCA2 for further analysis
data2 = y[c(1,2)]
head(data2)
```

```
## [1]  1.119349 -1.344599
```

Factor Analysis

```r
# make the first column as row names
foodtexture = read.csv('./Data/food-texture.csv', row.names = 1)

# alternatively
# rownames(foodtexture) = foodtexture$X
head(foodtexture)
```

```
##       Oil Density Crispy Fracture Hardness
## B110 16.5    2955     10       23       97
## B136 17.7    2660     14        9      139
## B171 16.2    2870     12       17      143
## B192 16.7    2920     10       31       95
## B225 16.3    2975     11       26      143
## B237 19.1    2790     13       16      189
```

```r
str(foodtexture)
```

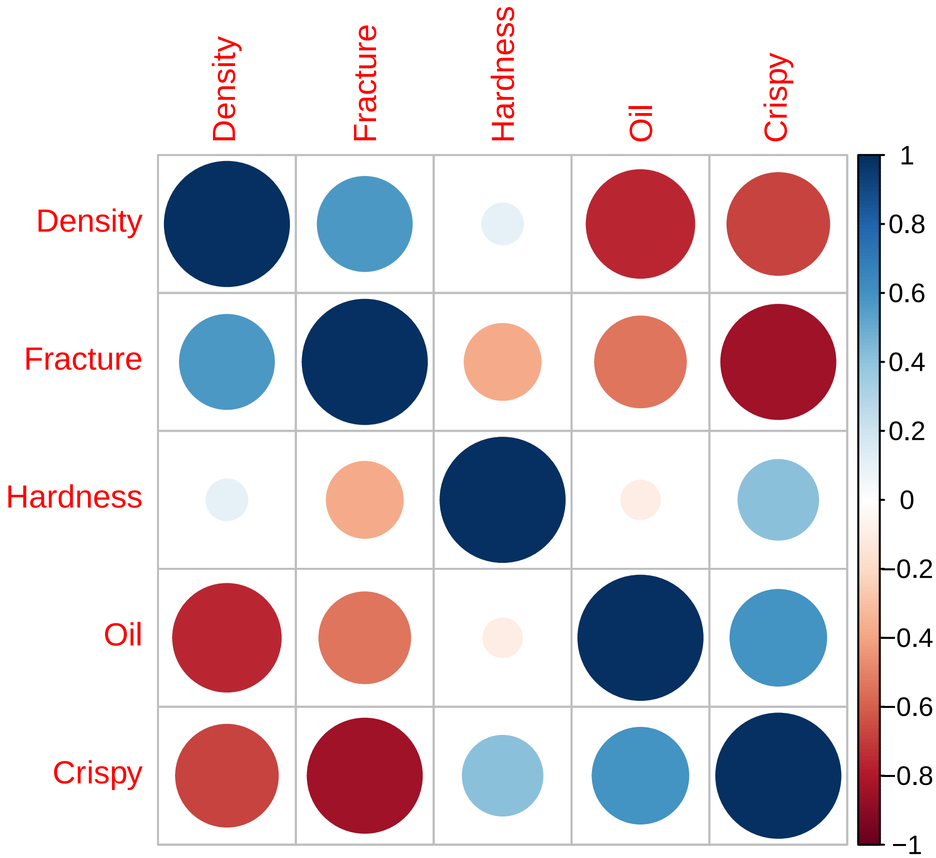
```
## 'data.frame':    50 obs. of  5 variables:
##  $ Oil     : num  16.5 17.7 16.2 16.7 16.3 19.1 18.4 17.5 15.7 16.4 ...
##  $ Density : int  2955 2660 2870 2920 2975 2790 2750 2770 2955 2945 ...
##  $ Crispy  : int  10 14 12 10 11 13 13 10 11 11 ...
##  $ Fracture: int  23 9 17 31 26 16 17 26 23 24 ...
##  $ Hardness: int  97 139 143 95 143 189 114 63 123 132 ...
```

```r
library(corrplot)
```

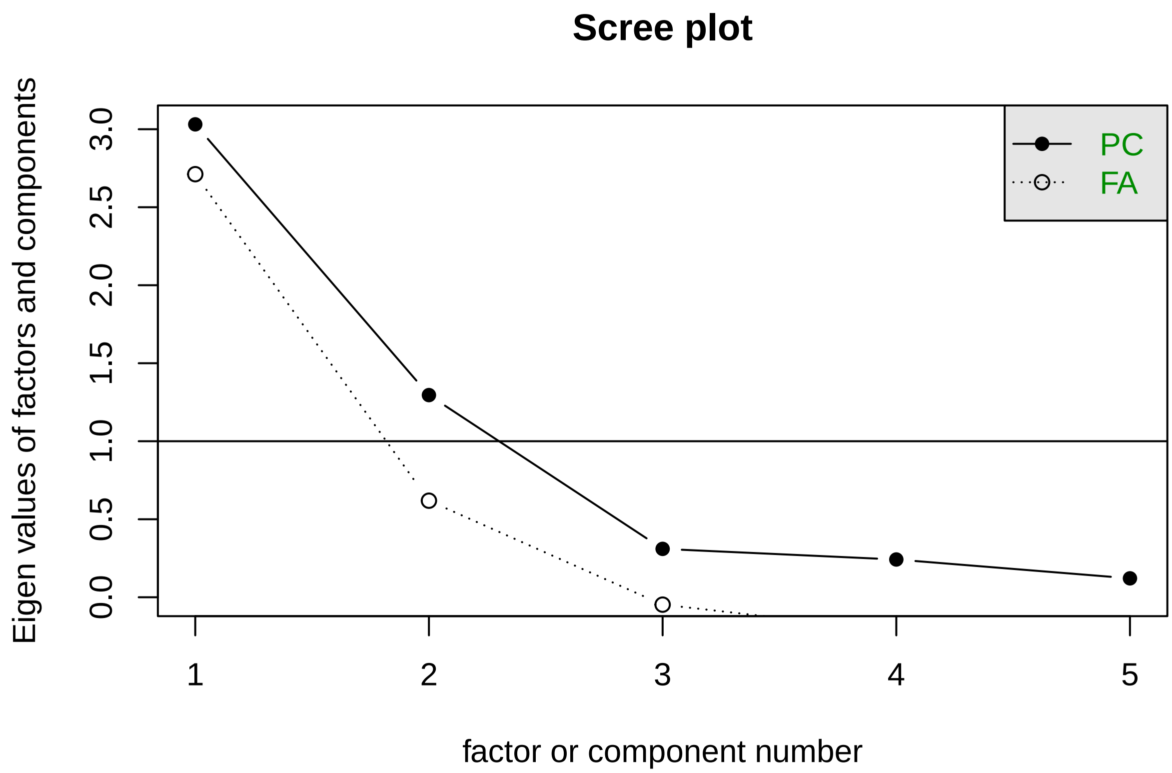
```
## Warning: package 'corrplot' was built under R version 4.4.2
```

```
## corrplot 0.95 loaded
```

```r
corrplot(cor(scale(foodtexture)), order = 'hclust')
```




```
# scree plot
scree(scale(foodtexture))
```




```r
# factor analysis
f.a = factanal(scale(foodtexture), factors = 2, rotation = 'varimax')
f.a
```

```
##
## Call:
## factanal(x = scale(foodtexture), factors = 2, rotation = "varimax")
##
## Uniquenesses:
##      Oil  Density   Crispy Fracture Hardness
##    0.334    0.156    0.042    0.256    0.407
##
## Loadings:
##          Factor1 Factor2
## Oil      -0.816
## Density   0.919
## Crispy   -0.745   0.635
## Fracture  0.645  -0.573
## Hardness          0.764
##
##                Factor1 Factor2
## SS loadings      2.490   1.316
## Proportion Var   0.498   0.263
## Cumulative Var   0.498   0.761
##
## Test of the hypothesis that 2 factors are sufficient.
```

```
## The chi square statistic is 0.27 on 1 degree of freedom.
## The p-value is 0.603
```

Interpretation :

- We can keep 76.1% of the original data using 2 factor from the original 5 variables
- Dominant variables from factor 1 : Oil, Density, Crispy, Fracture
- Dominant variables from factor 2 : Crispy, Fracture, Hardness

```
f.a_score = factanal(scale(foodtexture), factors = 2,
                     scores = c('regression'), rotation = 'varimax')
f.a_score
```

```
##
## Call:
## factanal(x = scale(foodtexture), factors = 2, scores = c("regression"),     rotation = "varimax")
##
## Uniquenesses:
##      Oil  Density   Crispy Fracture Hardness
##    0.334    0.156    0.042    0.256    0.407
##
## Loadings:
##          Factor1 Factor2
## Oil      -0.816
## Density   0.919
## Crispy   -0.745   0.635
## Fracture  0.645  -0.573
## Hardness          0.764
##
##                Factor1 Factor2
## SS loadings      2.490   1.316
## Proportion Var   0.498   0.263
## Cumulative Var   0.498   0.761
##
## Test of the hypothesis that 2 factors are sufficient.
## The chi square statistic is 0.27 on 1 degree of freedom.
## The p-value is 0.603
```

Numerosity

Parametric Model

Regression Model

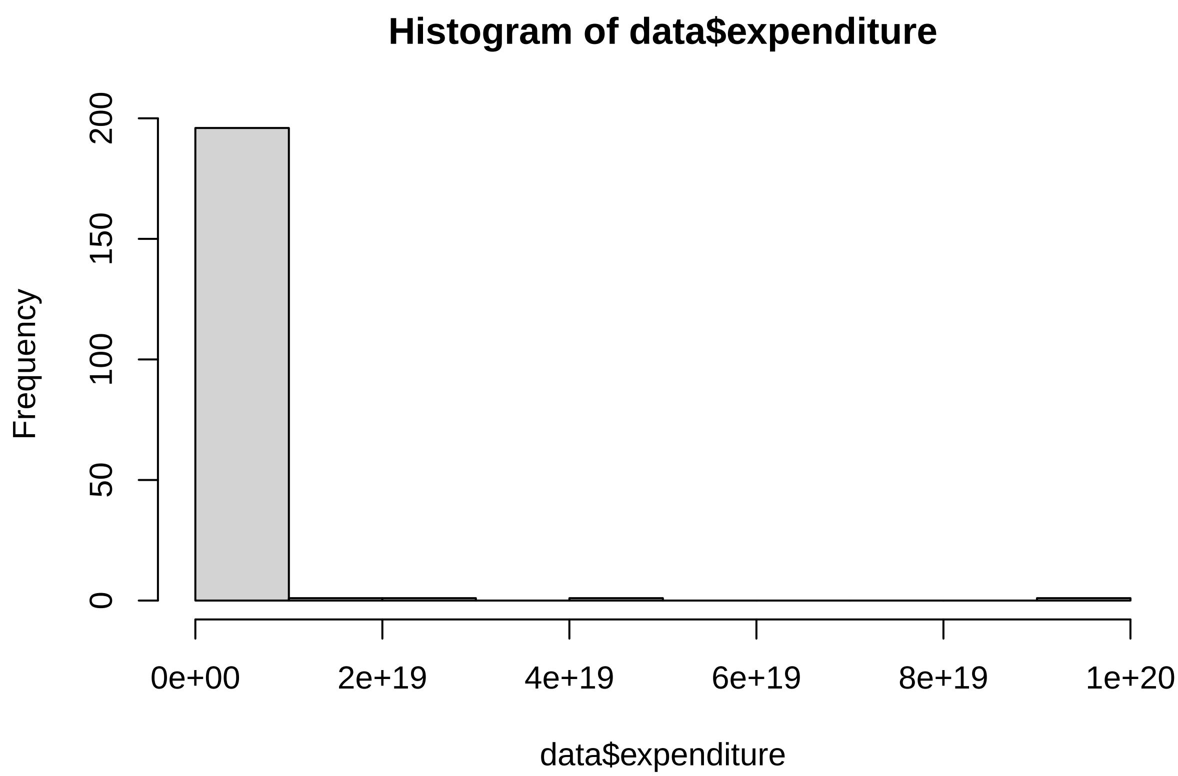
```
data = read.csv('./Data/data.csv', sep=';')
head(data)
```

```
##     income education_level work_experience  expenditure
## 1 45435.43               3       13.568200 2.743065e+10
```

```
## 2 36910.20               1        6.407732 4.532608e+08
## 3 16836.11               1        7.943813 2.658155e+04
## 4 47458.35               5       20.478526 8.593200e+11
## 5 17016.09               2       15.450881 4.224400e+04
## 6 46910.73               1        4.273132 2.991605e+10
```

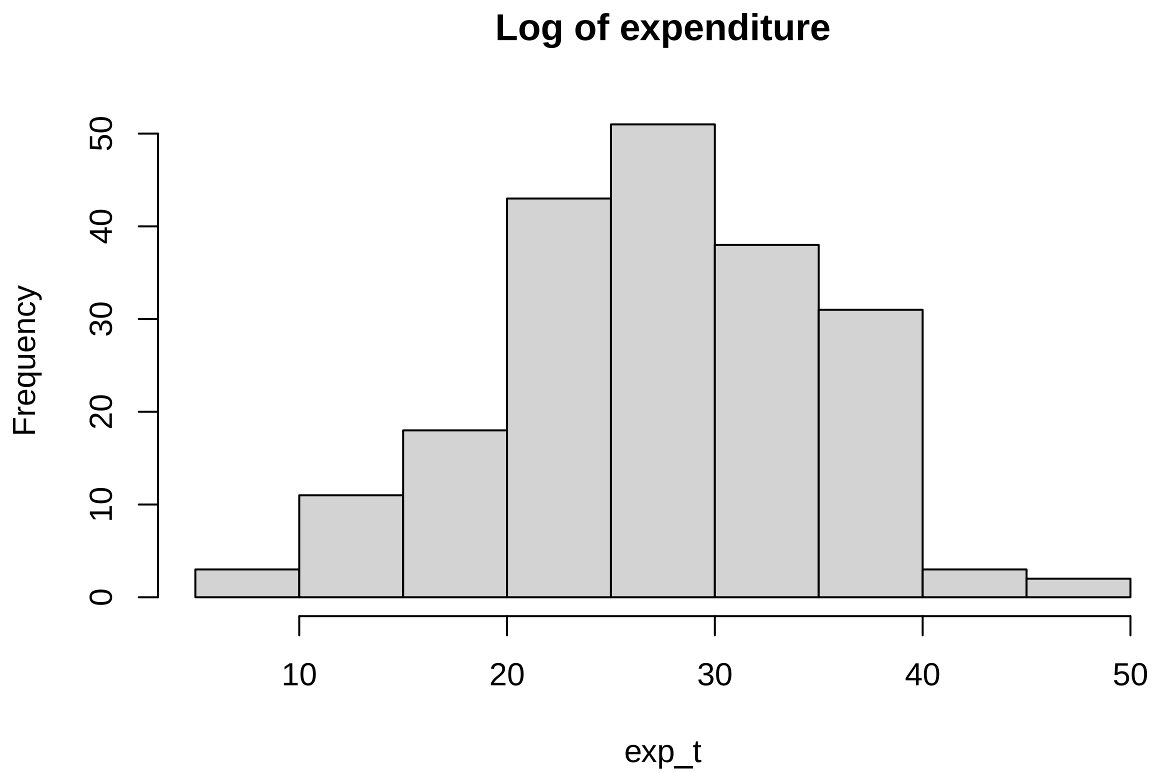
In this model, we're trying to predict the expenditure using 3 predictor variable which is the income, education level, and work experience. We assume the **Y is nearly normally distributed**

```r
hist(data$expenditure)
```



Since the response variable is not normal, we need to transform the data to be normal. For this example, we use log to transform the data

```r
exp_t = log(data$expenditure)
hist(exp_t, main='Log of expenditure')
```




```r
str(data)
```

```
## 'data.frame':    200 obs. of  4 variables:
##  $ income          : num  45435 36910 16836 47458 17016 ...
##  $ education_level: int  3 1 1 5 2 1 2 3 1 5 ...
##  $ work_experience: num  13.57 6.41 7.94 20.48 15.45 ...
##  $ expenditure    : num  2.74e+10 4.53e+08 2.66e+04 8.59e+11 4.22e+04 ...
```

Next, we notice that the education level is supposed to be a categorical variable instead of numerical variable. Thus, we need to change the data type first

```r
data$education_level = as.factor(data$education_level)
str(data)
```

```
## 'data.frame':    200 obs. of  4 variables:
##  $ income          : num  45435 36910 16836 47458 17016 ...
##  $ education_level: Factor w/ 5 levels "1","2","3","4",..: 3 1 1 5 2 1 2 3 1 5 ...
##  $ work_experience: num  13.57 6.41 7.94 20.48 15.45 ...
##  $ expenditure    : num  2.74e+10 4.53e+08 2.66e+04 8.59e+11 4.22e+04 ...
```

Now, we can start to fit the data into linear regression model.

```r
model_reg = lm(log(expenditure)~income+education_level+work_experience,
               data=data)
summary(model_reg)
```

```
##
## Call:
## lm(formula = log(expenditure) ~ income + education_level + work_experience,
##     data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.05108 -0.35025 -0.00016  0.31069  1.21984
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      9.621e-01  1.544e-01   6.233 2.82e-09 ***
## income           4.987e-04  2.316e-06 215.315  < 2e-16 ***
## education_level2 1.873e-01  1.080e-01   1.734 0.084517 .
## education_level3 3.292e-01  1.055e-01   3.120 0.002087 **
## education_level4 4.297e-01  1.186e-01   3.623 0.000373 ***
## education_level5 8.723e-01  1.111e-01   7.852 2.76e-13 ***
## work_experience  8.311e-02  7.239e-03  11.479  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5001 on 193 degrees of freedom
## Multiple R-squared:  0.9959, Adjusted R-squared:  0.9958
## F-statistic:  7782 on 6 and 193 DF,  p-value: < 2.2e-16
```

$R^2 > 0.99$ shows that this model is suitable for represent the original data. Save information related to this model

1. Parameter coefficient:
2.

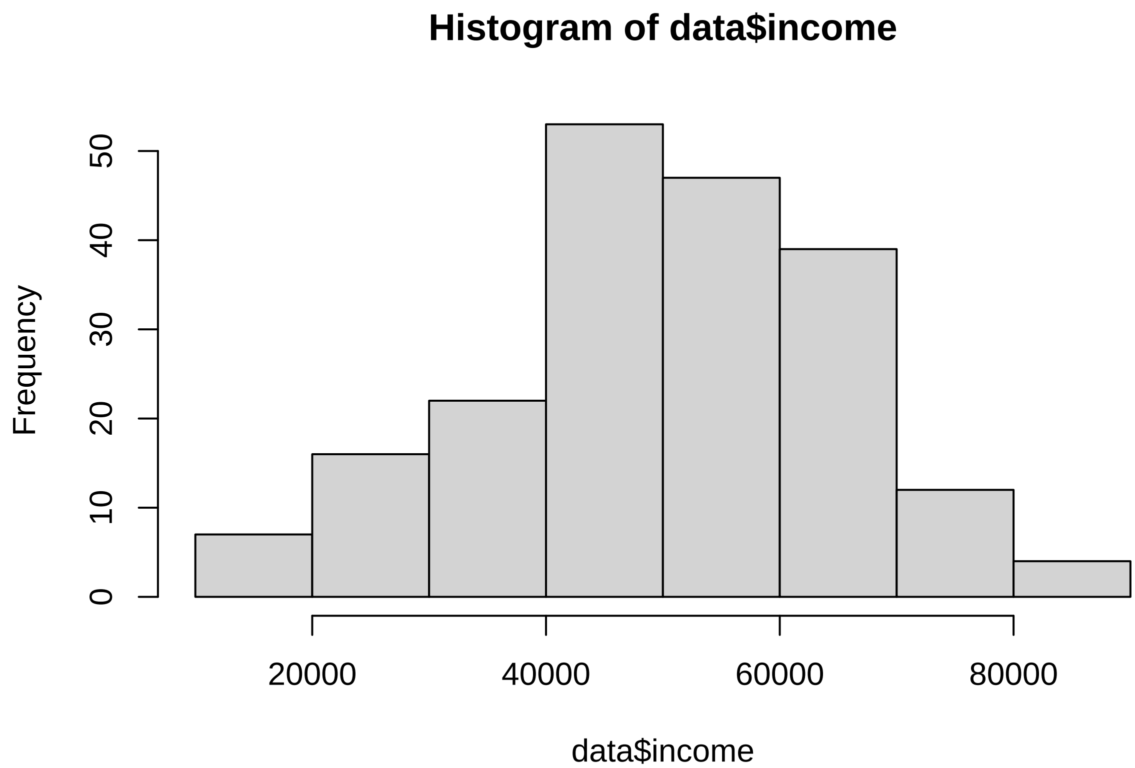
$$log(expenditure) = 0.09621 + 0.0005(income) + 0.1872(education_level) + 0.3292(education_level3$$

3. Feature information
4.

```r
coef(model_reg)
```

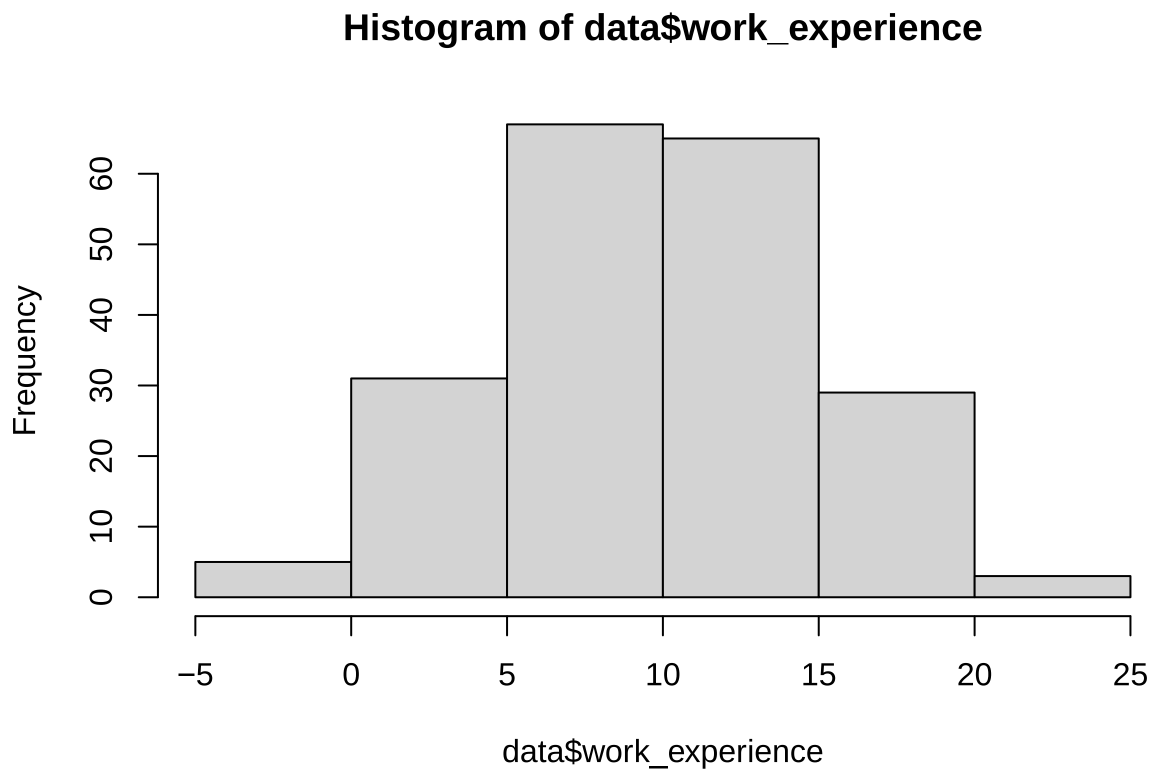
```
##      (Intercept)           income education_level2 education_level3
##     0.9621264920     0.0004986787     0.1872654922     0.3292185117
## education_level4 education_level5  work_experience
##     0.4296501743     0.8722609773     0.0831053838
```

```r
# feature information
# X1 = income
muIn = mean(data$income)
sdIn = sd(data$income)
rangeIn = range(data$income)
hist(data$income)
```



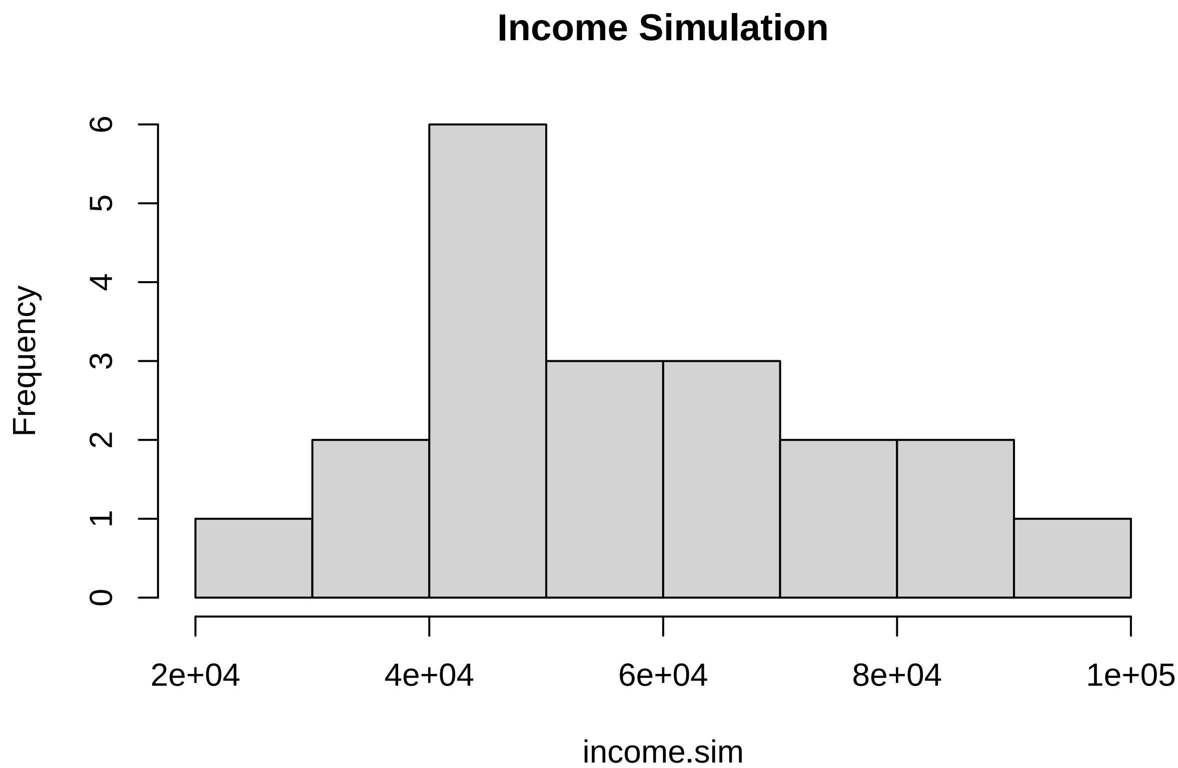

```
# X2 = Education level
edu_range = 1:5
```

```
# X3 = Work Experience
muWe = mean(data$work_experience)
sdWe = sd(data$work_experience)
rangeWe = range(data$work_experience)
hist(data$work_experience)
```



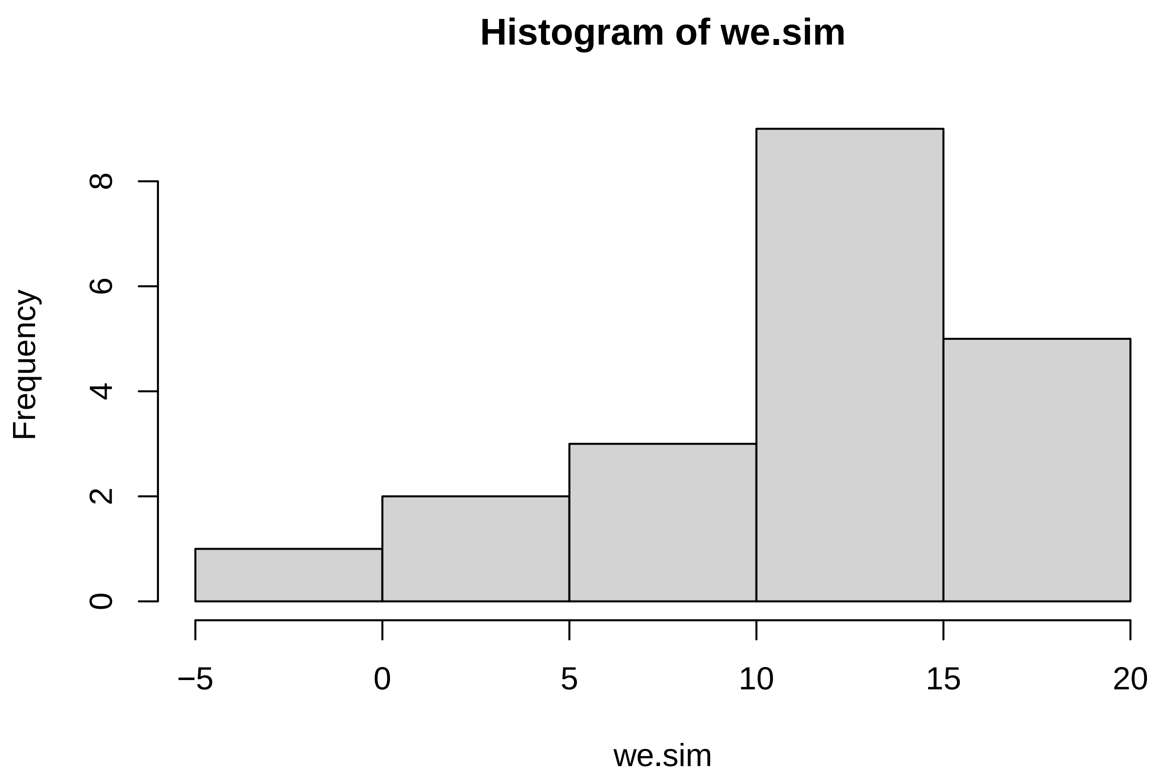

```r
# feature simulation
n = 20
```

```r
# X1 = income
income.sim = rnorm(n, mean = muIn, sd = sdIn)
hist(income.sim, main = 'Income Simulation')
```




```r
# X2 = Education level
education.sim = sample(1:5, n, replace = T)
```

```r
# X3 = Work Experience
we.sim = rnorm(n, mean = muWe, sd = sdWe)
hist(we.sim)
```




```
feature.sim = data.frame(work_experience = we.sim, income = income.sim,
                         education_level = education.sim)
feature.sim$education_level = as.factor(feature.sim$education_level)
head(feature.sim)
```

```
##   work_experience   income education_level
## 1      8.46428078 44680.76               2
## 2     13.38379452 75583.06               4
## 3     12.05915381 43255.32               1
## 4     -0.09849299 27327.05               5
## 5     11.84959590 96880.73               1
## 6     17.14408352 53610.39               4
```

```
y.sim  = predict(model_reg, feature.sim)
head(y.sim)
```

```
##        1        2        3        4        5        6
## 24.13416 40.19570 23.53481 15.45362 50.25924 29.55090
```

Log-linear Model

Probability Distribution

Non-Parametric Model

Histogram

Resampling

```
kewangan = read.table('./Data/Kewangan.D.txt')
head(kewangan)
```

```
##   ID Bangsa      Hutang Pendapatan.Tahunan
## 1  1   Cina  -255.41849           3919.225
## 2  2  India  -550.95988           2023.781
## 3  3   Cina   -74.77182           2480.774
## 4  4 Melayu -3144.75019           2907.829
## 5  5   Cina -1423.13386           2481.821
## 6  6  India -1092.24217           3750.682
```

```
table(kewangan$Bangsa)/length(kewangan$Bangsa)
```

```
##
##   Cina  India Melayu
## 0.3301 0.3382 0.3317
```

From the data above, we can see that the data is not properly represent Malaysia population. Thus, we can use our general knowledge for the proportion. However, in the real world cases, this is the idea on how to calculate the proportion of each category. For this exercise, we'll use 60% Malay, 30% Chinese, 10% Indian.

```
sample_size = 3000
sm = sample_size * 0.6
sc = sample_size * 0.3
si = sample_size * 0.1
# separate according to race
dm = subset(kewangan, Bangsa=='Melayu')
dc = subset(kewangan, Bangsa=='Cina')
di = subset(kewangan, Bangsa=='India')
# bootstrap resampling
nm = sample(nrow(dm), size=sm, replace=F)
nc = sample(nrow(dc), size=sc, replace=F)
ni = sample(nrow(di), size=si, replace=F)
#
snm = dm[nm,]
snc = dc[nc,]
sni = di[ni,]

# create new data
newdata = rbind(snm, snc, sni)
rownames(newdata) = NULL
head(newdata)
```

```
##      ID Bangsa     Hutang Pendapatan.Tahunan
## 1 28439 Melayu -1648.1484           2944.727
## 2 12697 Melayu  -327.3863           2549.149
## 3 32382 Melayu -1530.0249           2281.347
## 4 13092 Melayu  -849.0996           3105.685
## 5 20853 Melayu -1945.6000           2215.130
## 6 26292 Melayu  -828.9663           2207.673
```

Clustering

Types of sampling

Simple