DA5030.P2.Verma

Shivam Verma

Problem 1

Part 1 Download the data set. Add headers to the dataset.

```
# reading the dataset
census <- read.csv("adult.data", header = FALSE, stringsAsFactors = TRUE)
# reading the dataset for description of the each column
info <- read.csv("old.adult.names", header = FALSE)
colnames(census) <- c("age", "workclass", "fnlwgt", "education", "education-num", "marital_status", "occ
```

Part 2 Explore the data set as you see fit and that allows you to get a sense of the data and get comfortable with it.

```
#getting structure of data
str(census)
```

```
## 'data.frame':    32561 obs. of  15 variables:
##  $ age           : int  39 50 38 53 28 37 49 52 31 42 ...
##  $ workclass     : Factor w/ 9 levels " ?"," Federal-gov",..: 8 7 5 5 5 5 5 7 5 5 ...
##  $ fnlwgt        : int  77516 83311 215646 234721 338409 284582 160187 209642 45781 159449 ...
##  $ education     : Factor w/ 16 levels " 10th"," 11th",..: 10 10 12 2 10 13 7 12 13 10 ...
##  $ education-num : int  13 13 9 7 13 14 5 9 14 13 ...
##  $ marital_status: Factor w/ 7 levels " Divorced"," Married-AF-spouse",..: 5 3 1 3 3 3 4 3 5 3 ...
##  $ occupation    : Factor w/ 15 levels " ?"," Adm-clerical",..: 2 5 7 7 11 5 9 5 11 5 ...
##  $ relationship  : Factor w/ 6 levels " Husband"," Not-in-family",..: 2 1 2 1 6 6 2 1 2 1 ...
##  $ race          : Factor w/ 5 levels " Amer-Indian-Eskimo",..: 5 5 5 3 3 5 3 5 5 5 ...
##  $ sex           : Factor w/ 2 levels " Female"," Male": 2 2 2 2 1 1 1 2 1 2 ...
##  $ capital_gain  : int  2174 0 0 0 0 0 0 0 14084 5178 ...
##  $ capital_loss  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ hours_per_week: int  40 13 40 40 40 40 16 45 50 40 ...
##  $ native_country: Factor w/ 42 levels " ?"," Cambodia",..: 40 40 40 40 6 40 24 40 40 40 ...
##  $ class         : Factor w/ 2 levels " <=50K"," >50K": 1 1 1 1 1 1 1 2 2 2 ...
```

```
#getting summary of data
summary(census)
```

```
##       age                    workclass         fnlwgt
##  Min.   :17.00    Private         :22696   Min.   :  12285
##  1st Qu.:28.00    Self-emp-not-inc: 2541   1st Qu.: 117827
##  Median :37.00    Local-gov       : 2093   Median : 178356
##  Mean   :38.58    ?               : 1836   Mean   : 189778
##  3rd Qu.:48.00    State-gov       : 1298   3rd Qu.: 237051
##  Max.   :90.00    Self-emp-inc    : 1116   Max.   :1484705
```

```
##                  (Other)          :  981
##          education     education-num                  marital_status
##   HS-grad     :10501   Min.   : 1.00    Divorced             : 4443
##   Some-college: 7291   1st Qu.: 9.00    Married-AF-spouse    :   23
##   Bachelors   : 5355   Median :10.00    Married-civ-spouse   :14976
##   Masters     : 1723   Mean   :10.08    Married-spouse-absent:  418
##   Assoc-voc   : 1382   3rd Qu.:12.00    Never-married        :10683
##   11th        : 1175   Max.   :16.00    Separated            : 1025
##  (Other)      : 5134                    Widowed              :  993
##             occupation          relationship                   race
##   Prof-specialty :4140    Husband       :13193    Amer-Indian-Eskimo:  311
##   Craft-repair   :4099    Not-in-family : 8305    Asian-Pac-Islander: 1039
##   Exec-managerial:4066    Other-relative:  981    Black             : 3124
##   Adm-clerical   :3770    Own-child     : 5068    Other             :  271
##   Sales          :3650    Unmarried     : 3446    White             :27816
##   Other-service  :3295    Wife          : 1568
##  (Other)         :9541
##      sex          capital_gain    capital_loss    hours_per_week
##   Female:10771   Min.   :    0   Min.   :   0.0   Min.   : 1.00
##   Male  :21790   1st Qu.:    0   1st Qu.:   0.0   1st Qu.:40.00
##                  Median :    0   Median :   0.0   Median :40.00
##                  Mean   : 1078   Mean   :  87.3   Mean   :40.44
##                  3rd Qu.:    0   3rd Qu.:   0.0   3rd Qu.:45.00
##                  Max.   :99999   Max.   :4356.0   Max.   :99.00
##
##        native_country     class
##   United-States:29170    <=50K:24720
##   Mexico       :  643    >50K : 7841
##   ?            :  583
##   Philippines  :  198
##   Germany      :  137
##   Canada       :  121
##  (Other)       : 1709
```

Part 3 Split the data set 75/25 so you retain 25% for testing using random sampling.

```
# segregating the age column to 4 bins(A,B,C,D)
census$age[which(census$age>16 & census$age<=35)] <- "A"
census$age[which(census$age>35 & census$age<=54)] <- "B"
census$age[which(census$age>54 & census$age<=73)] <- "C"
census$age[which(census$age>73 & census$age<=92)] <- "D"

# creating random sample of 75/25
set.seed(123)
random_sample <- sample(nrow(census), nrow(census)*0.75)
# selecting features
census_select_val <- census[,c("age", "education", "workclass", "sex", "race", "native_country", "class
# creating train dataset
census_select_val_train <- census_select_val[random_sample,]
# creating test dataset
census_select_val_test <- census_select_val[-random_sample,]
```

Part 4 Using the Naive Bayes Classification algorithm from the KlaR, naivebayes, and e1071 packages, build an ensemble classifier that predicts whether an individual earns more than or less than US$50,000. Only use the features age, education, workclass, sex, race, and native-country. Ignore any other features in your model. You need to transform continuous variables into categorical variables by binning (use equal size bins from in to max).

Applying specific algorithms and feature selection, binning already done in previous steps.

```r
library(e1071)
library(gmodels)

# Applying e1071 Naive Bayes Classification algorithm, traing on train data
e1071_model <- naiveBayes(class~. , data = census_select_val_train)
# predicting test data
prediction1 <- predict(e1071_model, census_select_val_test[,-7])
# preparing the crosstable of applied model
CrossTable(prediction1, census_select_val_test$class)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## | Chi-square contribution |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  8141
##
##
##              | census_select_val_test$class
##  prediction1 |     <=50K |      >50K | Row Total |
## -------------|-----------|-----------|-----------|
##        <=50K |      5670 |      1182 |      6852 |
##              |    51.584 |   156.620 |           |
##              |     0.827 |     0.173 |     0.842 |
##              |     0.926 |     0.586 |           |
##              |     0.696 |     0.145 |           |
## -------------|-----------|-----------|-----------|
##         >50K |       454 |       835 |      1289 |
##              |   274.209 |   832.552 |           |
##              |     0.352 |     0.648 |     0.158 |
##              |     0.074 |     0.414 |           |
##              |     0.056 |     0.103 |           |
## -------------|-----------|-----------|-----------|
## Column Total |      6124 |      2017 |      8141 |
##              |     0.752 |     0.248 |           |
## -------------|-----------|-----------|-----------|
##
##
```

```r
library(klaR)
```

```
## Loading required package: MASS
```

```
## Warning: package 'MASS' was built under R version 3.6.2
```

```r
# Applying klaR Naive Bayes Classification algorithm, traing on train data
klaR_model <- NaiveBayes (class ~., data = census_select_val_train)
# predicting test data
prediction2 <- predict(klaR_model, census_select_val_test[,-7])
```

```
## Warning in data.matrix(newdata): NAs introduced by coercion
```

```
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 2832
```

```
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 3710
```

```
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 6364
```

```
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 7608
```

```r
# preparing the crosstable of applied model
CrossTable(prediction2$class, census_select_val_test$class)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## | Chi-square contribution |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  8141
##
##
##                   | census_select_val_test$class
## prediction2$class |     <=50K |      >50K | Row Total |
## ------------------|-----------|-----------|-----------|
##             <=50K |      5731 |      1381 |      7112 |
##                   |    27.141 |    82.406 |           |
##                   |     0.806 |     0.194 |     0.874 |
##                   |     0.936 |     0.685 |           |
```

```
##                  |     0.704 |     0.170 |           |
## -----------------|-----------|-----------|-----------|
##             >50K |       393 |       636 |      1029 |
##                  |   187.589 |   569.555 |           |
##                  |     0.382 |     0.618 |     0.126 |
##                  |     0.064 |     0.315 |           |
##                  |     0.048 |     0.078 |           |
## -----------------|-----------|-----------|-----------|
##     Column Total |      6124 |      2017 |      8141 |
##                  |     0.752 |     0.248 |           |
## -----------------|-----------|-----------|-----------|
##
##
```

```
library(naivebayes)
```

```
## naivebayes 0.9.7 loaded
```

```
# Applying klaR Naive Bayes Classification algorithm, traing on train data
NB_model <- naive_bayes(class~ age + education + workclass + sex + race + native_country ,data = census_
# predicting test data
prediction3 <- predict(NB_model, census_select_val_test[,-7])
# preparing the crosstable of applied model
CrossTable(prediction3, census_select_val_test$class)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## | Chi-square contribution |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  8141
##
##
##              | census_select_val_test$class
##  prediction3 |     <=50K |      >50K | Row Total |
## -------------|-----------|-----------|-----------|
##        <=50K |      5669 |      1182 |      6851 |
##              |    51.542 |   156.492 |           |
##              |     0.827 |     0.173 |     0.842 |
##              |     0.926 |     0.586 |           |
##              |     0.696 |     0.145 |           |
## -------------|-----------|-----------|-----------|
##         >50K |       455 |       835 |      1290 |
##              |   273.733 |   831.108 |           |
##              |     0.353 |     0.647 |     0.158 |
##              |     0.074 |     0.414 |           |
```

```
##               |     0.056 |     0.103 |           |
## --------------|-----------|-----------|-----------|
## Column Total |      6124 |      2017 |      8141 |
##               |     0.752 |     0.248 |           |
## --------------|-----------|-----------|-----------|
##
##
```

```
pred <- predict(NB_model, census_select_val_test[8142,-7])
```

```
# storing all model predictions in a new data frame
model_prediction <- data.frame(as.character(prediction1), as.character(prediction2$class), as.character
# naming the columns
colnames(model_prediction)<-c("e1071_pred", "klaR_pred", "NB_pred")
```

```
# adding final predictions column
model_prediction$final<- 0
# creating mode function to make ensemble model
calculate_mode <- function(x) {
uniqx <- unique(x)
uniqx[which.max(tabulate(match(x, uniqx)))]
}

# passing each row to ensemble for final predictions
for (i in 1:nrow(model_prediction))
{
  # getting mode of each row for final predictions
  model_prediction$final[i] <- calculate_mode(model_prediction[i,])
  # Results stored in 1,0 format so converting the results
  ifelse(model_prediction$final[i] == "1", model_prediction$final[i] <- " <=50K", model_prediction$fina

}
```

Part 5 Create a full logistic regression model of the same features as in (4) (i.e., do not eliminate any features regardless of p-value). Be sure to either use dummy coding for categorical features or convert them to factor variables and ensure that the glm function does the dummy coding.

```
#library(dummies)
#age <- census_select_val_train$age
#a <- dummy(age, sep = "_")
# Code for dummy conversion although glm takes care of factor variables and does dummy coding and imple

# creating a logistic model with selected feature values
model.glm <- glm(class ~ age + education + workclass + sex + race + native_country, data = census_selec
#summarizing the model
summary(model.glm)
```

```
##
## Call:
## glm(formula = class ~ age + education + workclass + sex + race +
##     native_country, family = "binomial", data = census_select_val_train)
##
```

```
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2983  -0.6475  -0.4283  -0.1249   3.1861
##
## Coefficients:
##                                        Estimate Std. Error z value Pr(>|z|)
## (Intercept)                            -6.04712    0.32128 -18.822  < 2e-16
## ageB                                    1.37086    0.04022  34.081  < 2e-16
## ageC                                    1.27035    0.05681  22.362  < 2e-16
## ageD                                    0.41641    0.19824   2.101 0.035677
## education 11th                         -0.02648    0.22181  -0.119 0.904957
## education 12th                          0.51607    0.26178   1.971 0.048684
## education 1st-4th                      -0.69684    0.54573  -1.277 0.201639
## education 5th-6th                      -0.15893    0.34024  -0.467 0.640421
## education 7th-8th                      -0.29225    0.24812  -1.178 0.238851
## education 9th                          -0.26166    0.27710  -0.944 0.345022
## education Assoc-acdm                    1.64613    0.17996   9.147  < 2e-16
## education Assoc-voc                     1.63661    0.17429   9.390  < 2e-16
## education Bachelors                     2.36086    0.16126  14.640  < 2e-16
## education Doctorate                     3.50955    0.21099  16.634  < 2e-16
## education HS-grad                       0.93309    0.16019   5.825 5.71e-09
## education Masters                       2.74183    0.16963  16.164  < 2e-16
## education Preschool                   -11.66558  140.19310  -0.083 0.933684
## education Prof-school                   3.34811    0.19495  17.174  < 2e-16
## education Some-college                  1.29912    0.16142   8.048 8.42e-16
## workclass Federal-gov                   1.13531    0.13695   8.290  < 2e-16
## workclass Local-gov                     0.75699    0.12105   6.253 4.02e-10
## workclass Never-worked                -11.18562  419.49882  -0.027 0.978728
## workclass Private                       0.79457    0.10527   7.548 4.42e-14
## workclass Self-emp-inc                  1.57373    0.12994  12.111  < 2e-16
## workclass Self-emp-not-inc              0.61088    0.11735   5.206 1.93e-07
## workclass State-gov                     0.51805    0.13406   3.864 0.000111
## workclass Without-pay                 -12.32162  252.35886  -0.049 0.961058
## sex Male                                1.25432    0.04375  28.672  < 2e-16
## race Asian-Pac-Islander                 0.37554    0.26524   1.416 0.156821
## race Black                              0.06281    0.23069   0.272 0.785405
## race Other                              0.09031    0.35320   0.256 0.798181
## race White                              0.55594    0.22047   2.522 0.011681
## native_country Cambodia                 0.96098    0.69303   1.387 0.165555
## native_country Canada                   0.40155    0.28546   1.407 0.159531
## native_country China                    0.03103    0.39664   0.078 0.937633
## native_country Columbia                -1.91596    0.78260  -2.448 0.014358
## native_country Cuba                     0.52007    0.33279   1.563 0.118112
## native_country Dominican-Republic      -0.50807    0.77261  -0.658 0.510796
## native_country Ecuador                  0.33882    0.79257   0.427 0.669017
## native_country El-Salvador             -0.14561    0.47011  -0.310 0.756765
## native_country England                  0.39875    0.33909   1.176 0.239623
## native_country France                   0.36857    0.55148   0.668 0.503919
## native_country Germany                  0.65796    0.28456   2.312 0.020764
## native_country Greece                  -0.01812    0.58462  -0.031 0.975269
## native_country Guatemala               -0.76012    0.75846  -1.002 0.316255
## native_country Haiti                   -0.59454    0.77506  -0.767 0.443028
## native_country Holand-Netherlands     -11.16858  882.74339  -0.013 0.989905
## native_country Honduras               -12.26176  248.43952  -0.049 0.960636
```

```
## native_country Hong                          0.19268    0.77570   0.248 0.803829
## native_country Hungary                      -0.47363    1.08519  -0.436 0.662511
## native_country India                         0.39996    0.34056   1.174 0.240221
## native_country Iran                          0.37557    0.41738   0.900 0.368205
## native_country Ireland                       0.12783    0.72503   0.176 0.860054
## native_country Italy                         0.96650    0.36786   2.627 0.008605
## native_country Jamaica                       0.51679    0.42738   1.209 0.226585
## native_country Japan                         0.76289    0.40550   1.881 0.059924
## native_country Laos                         -0.03218    0.90588  -0.036 0.971666
## native_country Mexico                       -0.46757    0.25772  -1.814 0.069638
## native_country Nicaragua                    -0.30293    0.77306  -0.392 0.695166
## native_country Outlying-US(Guam-USVI-etc) -12.53447  301.77093  -0.042 0.966868
## native_country Peru                         -0.57630    0.79760  -0.723 0.469961
## native_country Philippines                   0.29301    0.28209   1.039 0.298938
## native_country Poland                        0.01784    0.41490   0.043 0.965701
## native_country Portugal                     -0.08364    0.71179  -0.118 0.906462
## native_country Puerto-Rico                  -0.08663    0.37574  -0.231 0.817651
## native_country Scotland                     -0.14257    1.15196  -0.124 0.901506
## native_country South                         0.18915    0.39681   0.477 0.633587
## native_country Taiwan                        0.36290    0.44141   0.822 0.411002
## native_country Thailand                      0.28033    0.75378   0.372 0.709966
## native_country Trinadad&Tobago              -0.83696    1.21438  -0.689 0.490693
## native_country United-States                 0.34994    0.13409   2.610 0.009062
## native_country Vietnam                      -1.42004    0.66688  -2.129 0.033223
## native_country Yugoslavia                    0.83644    0.66174   1.264 0.206229
##
## (Intercept)                               ***
## ageB                                      ***
## ageC                                      ***
## ageD                                      *
## education 11th
## education 12th                            *
## education 1st-4th
## education 5th-6th
## education 7th-8th
## education 9th
## education Assoc-acdm                      ***
## education Assoc-voc                       ***
## education Bachelors                       ***
## education Doctorate                       ***
## education HS-grad                         ***
## education Masters                         ***
## education Preschool
## education Prof-school                     ***
## education Some-college                    ***
## workclass Federal-gov                     ***
## workclass Local-gov                       ***
## workclass Never-worked
## workclass Private                         ***
## workclass Self-emp-inc                    ***
## workclass Self-emp-not-inc                ***
## workclass State-gov                       ***
## workclass Without-pay
## sex Male                                  ***
```

```
## race Asian-Pac-Islander
## race Black
## race Other
## race White                                  *
## native_country Cambodia
## native_country Canada
## native_country China
## native_country Columbia                     *
## native_country Cuba
## native_country Dominican-Republic
## native_country Ecuador
## native_country El-Salvador
## native_country England
## native_country France
## native_country Germany                      *
## native_country Greece
## native_country Guatemala
## native_country Haiti
## native_country Holand-Netherlands
## native_country Honduras
## native_country Hong
## native_country Hungary
## native_country India
## native_country Iran
## native_country Ireland
## native_country Italy                        **
## native_country Jamaica
## native_country Japan                        .
## native_country Laos
## native_country Mexico                       .
## native_country Nicaragua
## native_country Outlying-US(Guam-USVI-etc)
## native_country Peru
## native_country Philippines
## native_country Poland
## native_country Portugal
## native_country Puerto-Rico
## native_country Scotland
## native_country South
## native_country Taiwan
## native_country Thailand
## native_country Trinadad&Tobago
## native_country United-States                **
## native_country Vietnam                      *
## native_country Yugoslavia
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 26830  on 24419  degrees of freedom
## Residual deviance: 20676  on 24347  degrees of freedom
## AIC: 20822
##
```

```
## Number of Fisher Scoring iterations: 13
```

```
# predicting the test data using model
prediction4 <- predict(model.glm, census_select_val_test, type = "response")
# this model gives out probability, so setting threshold to 0.5 making predictions
prediction4 <- ifelse(prediction4 > 0.5, 1,0)
# Creating levels to convert to actual predictions
prediction4 <- factor(prediction4, levels = c(0,1), labels = c(" <=50K", " >50K"))
# generating crosstable for the model
CrossTable((prediction4), (census_select_val_test$class), dnn = c("predicted", "actual"))
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## | Chi-square contribution |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  8141
##
##
##              | actual
##    predicted |     <=50K |      >50K | Row Total |
## -------------|-----------|-----------|-----------|
##        <=50K |      5771 |      1260 |      7031 |
##              |    43.924 |   133.361 |           |
##              |     0.821 |     0.179 |     0.864 |
##              |     0.942 |     0.625 |           |
##              |     0.709 |     0.155 |           |
## -------------|-----------|-----------|-----------|
##         >50K |       353 |       757 |      1110 |
##              |   278.223 |   844.738 |           |
##              |     0.318 |     0.682 |     0.136 |
##              |     0.058 |     0.375 |           |
##              |     0.043 |     0.093 |           |
## -------------|-----------|-----------|-----------|
## Column Total |      6124 |      2017 |      8141 |
##              |     0.752 |     0.248 |           |
## -------------|-----------|-----------|-----------|
##
##
```

Part 6 Add the logistic regression model to the ensemble built in (4).

```
# adding new model predictions to data frame
model_prediction$glm_pred <- prediction4
# making the final prediction column to 0
model_prediction$final <- 0
```

```r
# passing each row for final predictions
for (i in 1:nrow(model_prediction))
{
  # calling mode function for final model predictions
  model_prediction$final[i] <- calculate_mode(model_prediction[i,])
  # results were in 1,0 format converting them to actual predictions
  ifelse(model_prediction$final[i] == "1", model_prediction$final[i] <- " <=50K", model_prediction$fina

}

# generating crosstable for final ensemble predictions and test data
CrossTable(unlist(model_prediction$final), census_select_val_test$class)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## | Chi-square contribution |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  8141
##
##
##                              | census_select_val_test$class
## unlist(model_prediction$final) |     <=50K |      >50K | Row Total |
## -------------------------------|-----------|-----------|-----------|
##                          <=50K |      5670 |      1182 |      6852 |
##                                |    51.584 |   156.620 |           |
##                                |     0.827 |     0.173 |     0.842 |
##                                |     0.926 |     0.586 |           |
##                                |     0.696 |     0.145 |           |
## -------------------------------|-----------|-----------|-----------|
##                           >50K |       454 |       835 |      1289 |
##                                |   274.209 |   832.552 |           |
##                                |     0.352 |     0.648 |     0.158 |
##                                |     0.074 |     0.414 |           |
##                                |     0.056 |     0.103 |           |
## -------------------------------|-----------|-----------|-----------|
##                   Column Total |      6124 |      2017 |      8141 |
##                                |     0.752 |     0.248 |           |
## -------------------------------|-----------|-----------|-----------|
##
##
```

```r
# generating confusion matrix for final ensemble predictions and test data
caret::confusionMatrix(as.factor(unlist(model_prediction$final)), as.factor(census_select_val_test$clas
```

```
## Confusion Matrix and Statistics
```

```
##
##           Reference
## Prediction  <=50K  >50K
##      <=50K   5670  1182
##      >50K     454   835
##
##                Accuracy : 0.799
##                  95% CI : (0.7902, 0.8077)
##     No Information Rate : 0.7522
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3866
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9259
##             Specificity : 0.4140
##          Pos Pred Value : 0.8275
##          Neg Pred Value : 0.6478
##              Prevalence : 0.7522
##          Detection Rate : 0.6965
##    Detection Prevalence : 0.8417
##       Balanced Accuracy : 0.6699
##
##        'Positive' Class :  <=50K
##
```

Part 7 Using the ensemble model from (6), predict whether a 35-year-old white female adult who is a local government worker with a doctorate who immigrated from Portugal earns more or less than US$50,000.

```r
# creating new data for prediction
newData <- data.frame(age = "A",education = as.factor(" Doctorate"), workclass = " Local-gov", sex = " F
# creating a temporary data equal to train data
census_train_temp  <- census_select_val_train
# adding new case to temporary data
census_train_temp <- rbind(census_train_temp , newData)
# Predicting newData class using klaR model
new_klar <- predict(klaR_model, newData)
# Predicting newData class using Naive Bayes model
new_NB <- predict(NB_model, census_train_temp[24421,-7])
# Predicting newData class using e1071 model
new_e <- predict(e1071_model, newData)
# Predicting newData class using logistic regression model
new_glm <- predict(model.glm, newData)
# this model gives out probability, so setting threshold to 0.5 making predictions
new_glm <- ifelse(new_glm >0.5, 1, 0)
# Creating levels to convert to actual predictions
new_glm <- factor(new_glm, levels = c(0, 1), labels = c(" <=50K", " >50K"))
# Creating new data predictions in a new data frame
new_pred_data <- data.frame(new_klar$class, new_e, new_NB, final = NA, new_glm)
# naming columns
colnames(new_pred_data) <- c("e1071_pred", "klaR_pred", "NB_pred", "final", "glm_pred")
# Adding new predictions to previous data frame
model_prediction <- rbind(model_prediction, new_pred_data)
```

```r
# setting final predictions to 0
model_prediction$final <- 0
# Passing each row for getting final predictions
for (i in 1:nrow(model_prediction))
{
  # calling mode function for final predictions
  model_prediction$final[i] <- calculate_mode(model_prediction[i,])
  # results were in 1,0 format converting it to actual predictions
  ifelse(model_prediction$final[i] == "1", model_prediction$final[i] <- " <=50K", model_prediction$fina

}
print("The class for 35-year-old white female adult who is a local government worker with a doctorate wh
```

```
## [1] "The class for 35-year-old white female adult who is a local government worker with a doctorate w
```

```r
model_prediction$final[8142]
```

```
## [[1]]
## [1] " <=50K"
```

Part 8 Calculate accuracy and prepare confusion matrices for all three Bayes implementations (KlaR, naive-bayes, e1071) and the logistic regression model. Compare the implementations and comment on differences. Be sure to use the same training data set for all three.

```r
library(caret)
```

```
## Loading required package: lattice
```

```
## Warning: package 'lattice' was built under R version 3.6.2
```

```
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 3.6.2
```

```r
# confusion matrix for e1071 predictions
a <- confusionMatrix(prediction1, census_select_val_test$class)
# printing matrix
a
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  <=50K  >50K
##      <=50K   5670  1182
##      >50K     454   835
##
##                Accuracy : 0.799
##                  95% CI : (0.7902, 0.8077)
##     No Information Rate : 0.7522
##     P-Value [Acc > NIR] : < 2.2e-16
```

```
##
##                   Kappa : 0.3866
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9259
##             Specificity : 0.4140
##          Pos Pred Value : 0.8275
##          Neg Pred Value : 0.6478
##              Prevalence : 0.7522
##          Detection Rate : 0.6965
##    Detection Prevalence : 0.8417
##       Balanced Accuracy : 0.6699
##
##        'Positive' Class :  <=50K
##
```

```r
# printing accuracy
paste0("Accuracy for e1071 is : ", a$overall[1]*100, "%")
```

```
## [1] "Accuracy for e1071 is : 79.90418867461%"
```

```r
# confusion matrix for klaR predictions
b <- confusionMatrix(prediction2$class, census_select_val_test$class)
# printing matrix
b
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  <=50K  >50K
##      <=50K   5731  1381
##      >50K     393   636
##
##                Accuracy : 0.7821
##                  95% CI : (0.773, 0.791)
##     No Information Rate : 0.7522
##     P-Value [Acc > NIR] : 1.372e-10
##
##                   Kappa : 0.3005
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9358
##             Specificity : 0.3153
##          Pos Pred Value : 0.8058
##          Neg Pred Value : 0.6181
##              Prevalence : 0.7522
##          Detection Rate : 0.7040
##    Detection Prevalence : 0.8736
##       Balanced Accuracy : 0.6256
##
##        'Positive' Class :  <=50K
##
```

```r
# printing accuracy
paste0("Accuracy for klaR is : ", b$overall[1]*100, "%")
```

```
## [1] "Accuracy for klaR is : 78.2090652254023%"
```

```r
# confusion matrix for Naive Bayes predictions
c <- confusionMatrix(prediction3, census_select_val_test$class)
# printing matrix
c
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  <=50K  >50K
##      <=50K   5669  1182
##      >50K     455   835
##
##                Accuracy : 0.7989
##                  95% CI : (0.79, 0.8076)
##     No Information Rate : 0.7522
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3864
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9257
##             Specificity : 0.4140
##          Pos Pred Value : 0.8275
##          Neg Pred Value : 0.6473
##              Prevalence : 0.7522
##          Detection Rate : 0.6964
##    Detection Prevalence : 0.8415
##       Balanced Accuracy : 0.6698
##
##        'Positive' Class :  <=50K
##
```

```r
# printing accuracy
paste0("Accuracy for Naive Bayes is : ", c$overall[1]*100, "%")
```

```
## [1] "Accuracy for Naive Bayes is : 79.8919051713549%"
```

```r
# confusion matrix for logistic regression predictions
d <- confusionMatrix(prediction4, census_select_val_test$class)
# printing matrix
d
```

```
## Confusion Matrix and Statistics
##
##           Reference
```

```
## Prediction  <=50K  >50K
##      <=50K   5771  1260
##      >50K     353   757
##
##                Accuracy : 0.8019
##                  95% CI : (0.793, 0.8105)
##     No Information Rate : 0.7522
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3741
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9424
##             Specificity : 0.3753
##          Pos Pred Value : 0.8208
##          Neg Pred Value : 0.6820
##              Prevalence : 0.7522
##          Detection Rate : 0.7089
##    Detection Prevalence : 0.8637
##       Balanced Accuracy : 0.6588
##
##        'Positive' Class :  <=50K
##
```

```r
 # printing accuracy
paste0("Accuracy for logistic regression is : ", d$overall[1]*100, "%")
```

```
## [1] "Accuracy for logistic regression is : 80.186709249478%"
```

Problem 2

Part 1 Load and then explore the data set on car sales referenced by the article Shonda Kuiper (2008) Introduction to Multiple Regression

```r
library(readxl)
# Reading the dataset using readxl library
cars_sales_price <- read_excel("kellycarsalesdata.xlsx")
# exploring basic stats of data
str(cars_sales_price)
```

```
## tibble [804 x 9] (S3: tbl_df/tbl/data.frame)
##  $ Price   : num [1:804] 17314 17542 16219 16337 16339 ...
##  $ Mileage : num [1:804] 8221 9135 13196 16342 19832 ...
##  $ Make    : chr [1:804] "Buick" "Buick" "Buick" "Buick" ...
##  $ Cylinder: num [1:804] 6 6 6 6 6 6 6 6 6 6 ...
##  $ Liter   : num [1:804] 3.1 3.1 3.1 3.1 3.1 3.1 3.1 3.1 3.1 3.1 ...
##  $ Doors   : num [1:804] 4 4 4 4 4 4 4 4 4 4 ...
##  $ Cruise  : num [1:804] 1 1 1 1 1 1 1 1 1 1 ...
##  $ Sound   : num [1:804] 1 1 1 0 0 1 1 1 0 1 ...
##  $ Leather : num [1:804] 1 0 0 0 1 0 0 0 1 1 ...
```

```r
# summarizing the data
summary(cars_sales_price)
```

```
##      Price          Mileage           Make              Cylinder
##  Min.   : 8639   Min.   :  266   Length:804         Min.   :4.000
##  1st Qu.:14273   1st Qu.:14624   Class :character   1st Qu.:4.000
##  Median :18025   Median :20914   Mode  :character   Median :6.000
##  Mean   :21343   Mean   :19832                      Mean   :5.269
##  3rd Qu.:26717   3rd Qu.:25213                      3rd Qu.:6.000
##  Max.   :70755   Max.   :50387                      Max.   :8.000
##      Liter           Doors           Cruise           Sound
##  Min.   :1.600   Min.   :2.000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:2.200   1st Qu.:4.000   1st Qu.:1.0000   1st Qu.:0.0000
##  Median :2.800   Median :4.000   Median :1.0000   Median :1.0000
##  Mean   :3.037   Mean   :3.527   Mean   :0.7525   Mean   :0.6791
##  3rd Qu.:3.800   3rd Qu.:4.000   3rd Qu.:1.0000   3rd Qu.:1.0000
##  Max.   :6.000   Max.   :4.000   Max.   :1.0000   Max.   :1.0000
##     Leather
##  Min.   :0.0000
##  1st Qu.:0.0000
##  Median :1.0000
##  Mean   :0.7239
##  3rd Qu.:1.0000
##  Max.   :1.0000
```

Part 2 Are there outliers in the data set? How do you identify outliers and how do you deal with them? Remove them but create a second data set with outliers removed. Keep the original data set.

```r
# creating second dataset
cars_outlier <- cars_sales_price

paste0("Total number of rows before outlier removal : ", nrow(cars_outlier))
```

```
## [1] "Total number of rows before outlier removal : 804"
```

```r
# loop for checking outliers
for (i in 1:ncol(cars_outlier)){

  # getting mean and sd of each row
  meanc <- mean(as.numeric(unlist(cars_outlier[,i])))
  sdc <- sd(as.numeric(unlist(cars_outlier[,i])))
  # setting threshold as 3 Sd
  sdc <- sdc * 3

  print(paste0("Column Name : ", colnames(cars_outlier[i])))

  # printing row numbers which include outliers in each column
  outlier <- (which(cars_outlier[,i] > meanc + sdc | cars_outlier[,i] < meanc - sdc))
  print(outlier)

  # checking if there is any outlier in this column or not
  if(length(outlier)!=0){
```

```r
    # if there is outlier removing it
    cars_outlier <- cars_outlier[-outlier,]

    print(paste0("Rows in data after outliers removed : ", nrow(cars_outlier)))
  }

}
```

```
## [1] "Column Name : Price"
##  [1]  81 151 152 153 154 155 156 157 158 159 160
## [1] "Rows in data after outliers removed : 793"
## [1] "Column Name : Mileage"
## [1] 639 669
## [1] "Rows in data after outliers removed : 791"

## Warning in mean(as.numeric(unlist(cars_outlier[, i]))): NAs introduced by
## coercion

## Warning in is.data.frame(x): NAs introduced by coercion

## [1] "Column Name : Make"
## integer(0)
## [1] "Column Name : Cylinder"
## integer(0)
## [1] "Column Name : Liter"
## integer(0)
## [1] "Column Name : Doors"
## integer(0)
## [1] "Column Name : Cruise"
## integer(0)
## [1] "Column Name : Sound"
## integer(0)
## [1] "Column Name : Leather"
## integer(0)
```

Part 3 What are the distributions of each of the features in the data set with outliers removed? Are they reasonably normal so you can apply a statistical learner such as regression? Can you normalize features through a log, inverse, or square-root transform? Transform as needed.

Checking the distribution for each feature values using the Fitness of Good model under library fitdistrplus. Following are the observations each of the feature values: **Price is Log normal distribution**, **Mileage is Normal Distribution**, **Make is Normal Distribution**, **Cylinder is Uniform Distribution**, **Liter is Uniform Distribution**, **Doors is Beta Distribution**, **Cruise is Beta Distribution**, **Sound is Beta Distribution** and **Leather is also Beta Distribution**. Although few features are not normal but they are categorical values that is either 1 or 0 values, so they don't make much difference on model. Price column needs transformation, thus trying it with inverse transform.

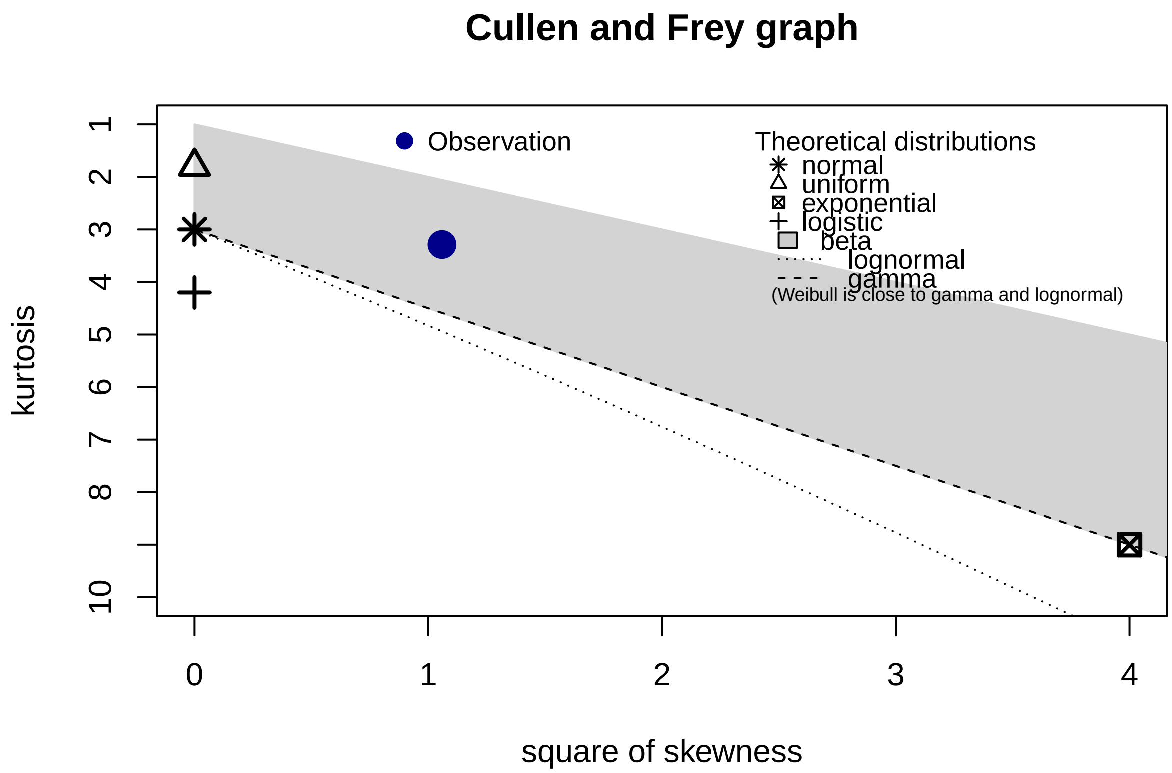
```r
library(fitdistrplus)
```

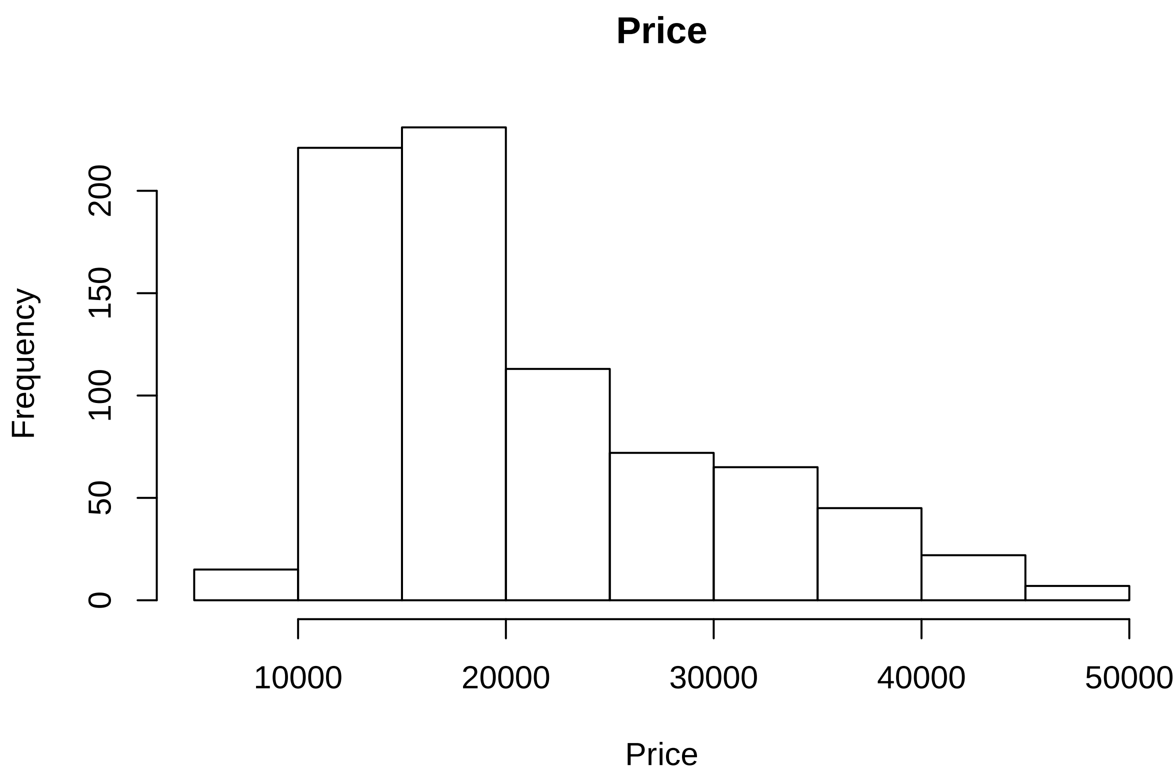
```
## Warning: package 'fitdistrplus' was built under R version 3.6.2

## Loading required package: survival
```

```
## Warning: package 'survival' was built under R version 3.6.2

##
## Attaching package: 'survival'

## The following object is masked from 'package:caret':
##
##     cluster
```

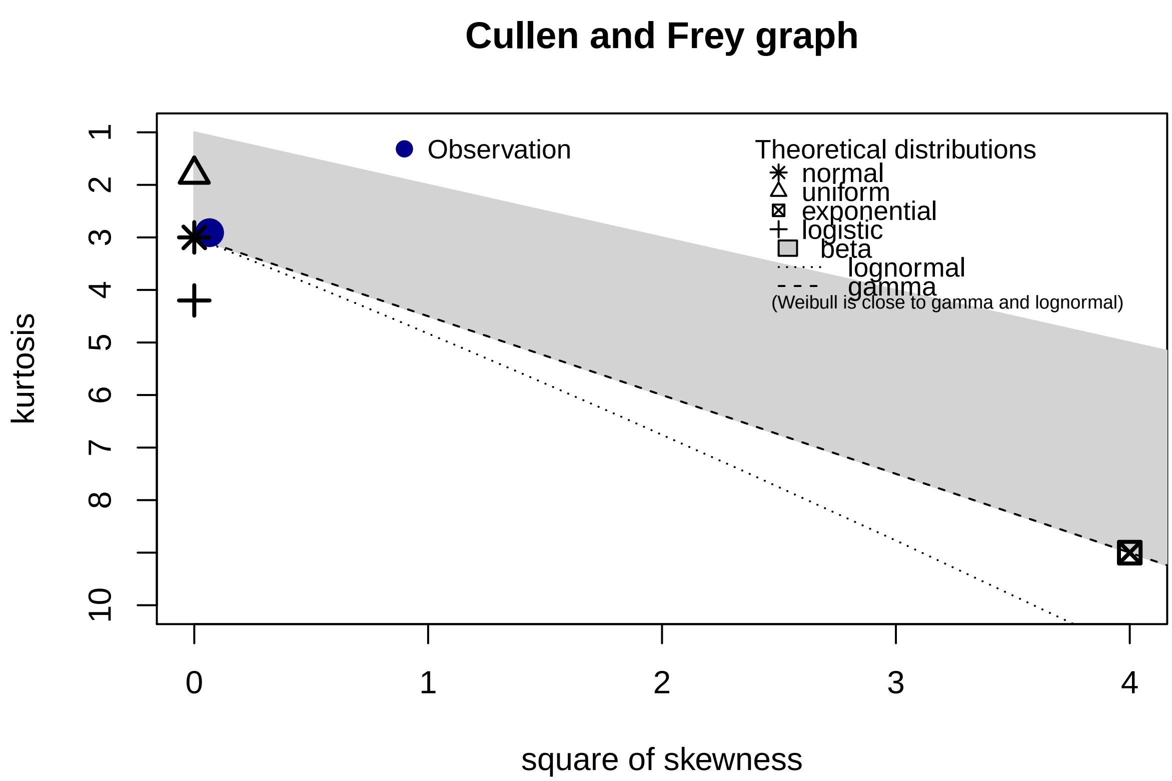
```
# loop for traversing each feature
for(i in 1:ncol(cars_outlier))
{
  # Printing varaible name
print(paste0("This graph is for : ", colnames(cars_outlier[i])))
  # Make is a categorical variable so not plotting histogram
  if(i==3){
    # demonstrating graph
descdist(as.numeric(as.factor(unlist(cars_outlier[,i]))), discrete = FALSE)
  }
  else{
    # demonstrating graph and histogram
    descdist(as.numeric((unlist(cars_outlier[,i]))), discrete = FALSE)
    hist(as.numeric((unlist(cars_outlier[,i]))), xlab = colnames(cars_outlier[i]), main=colnames(cars_ou
  }
}
```

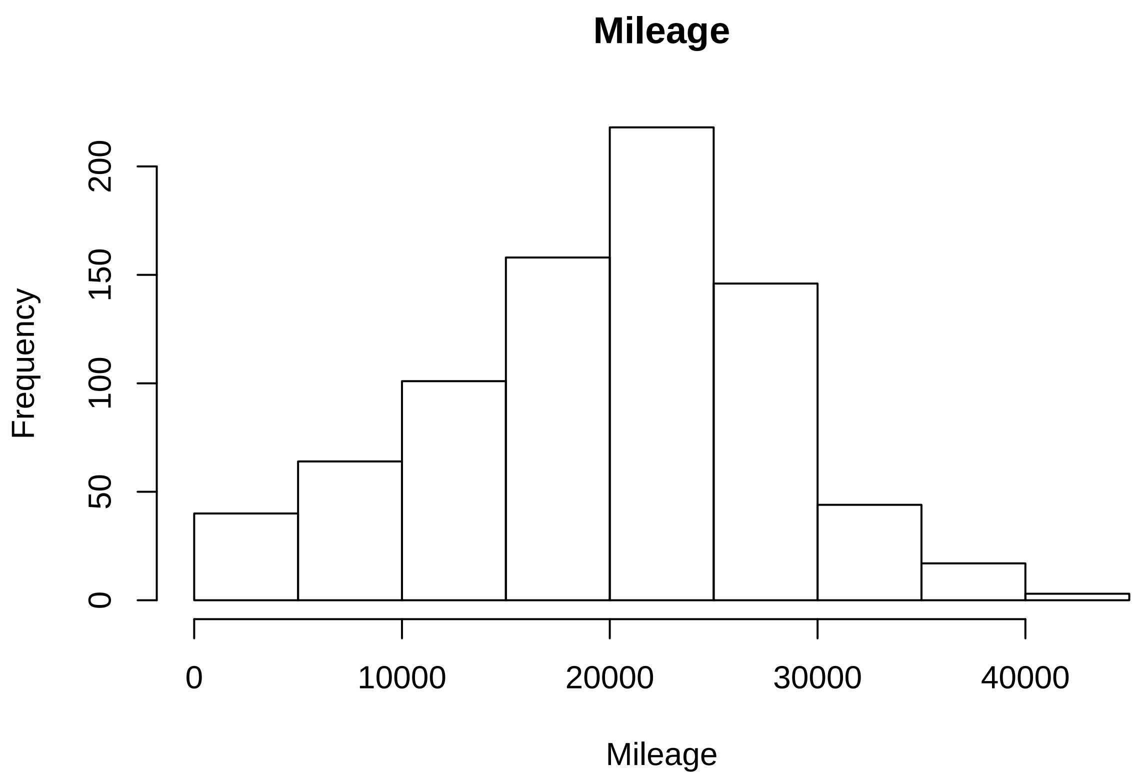
```
## [1] "This graph is for : Price"
```



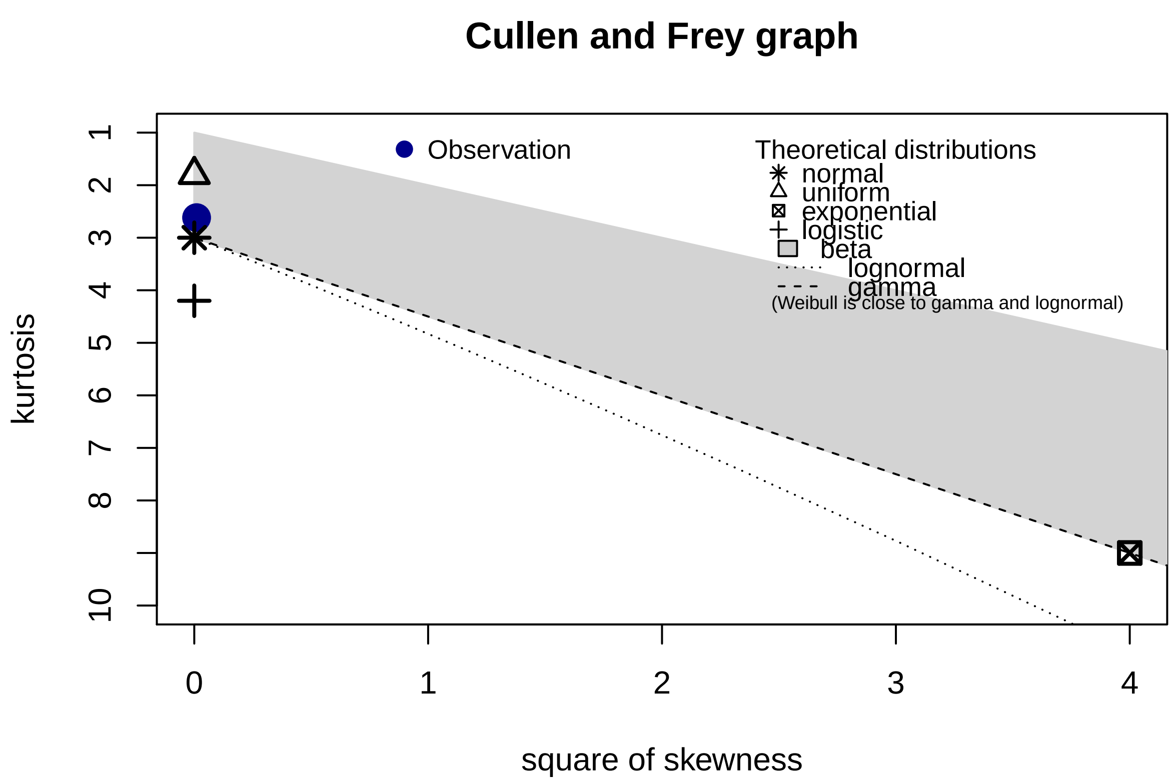



```
## [1] "This graph is for : Mileage"
```

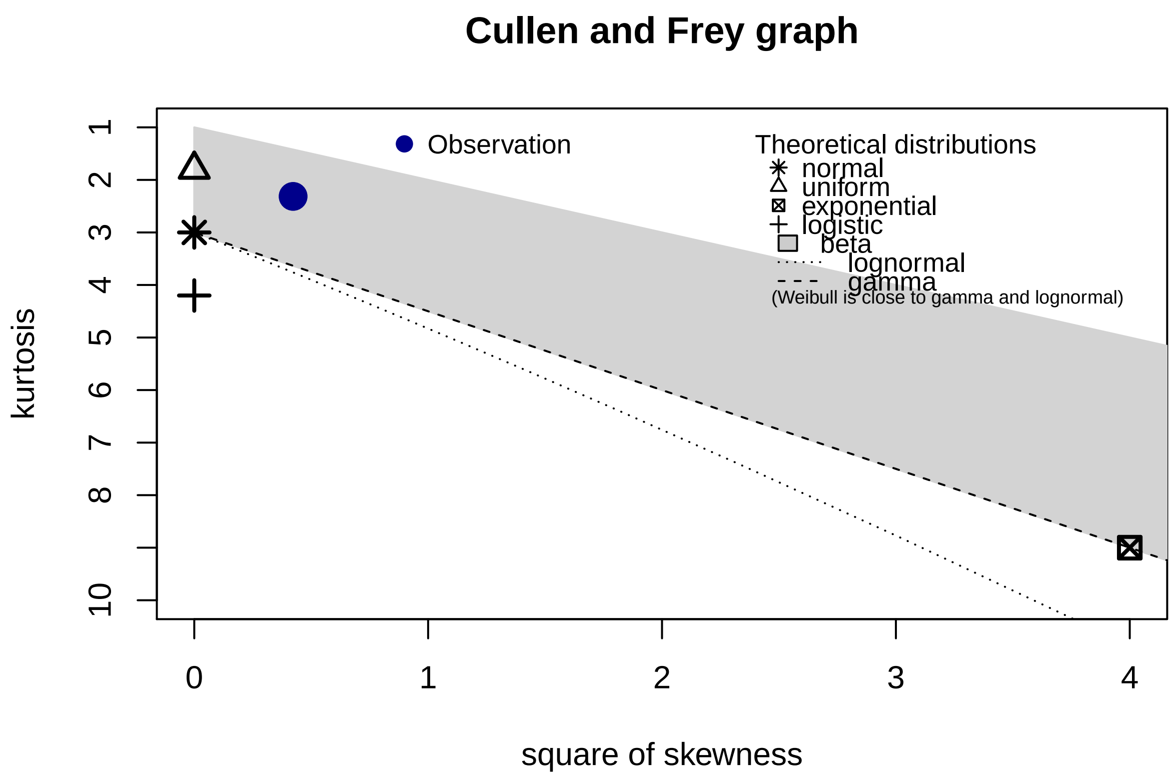


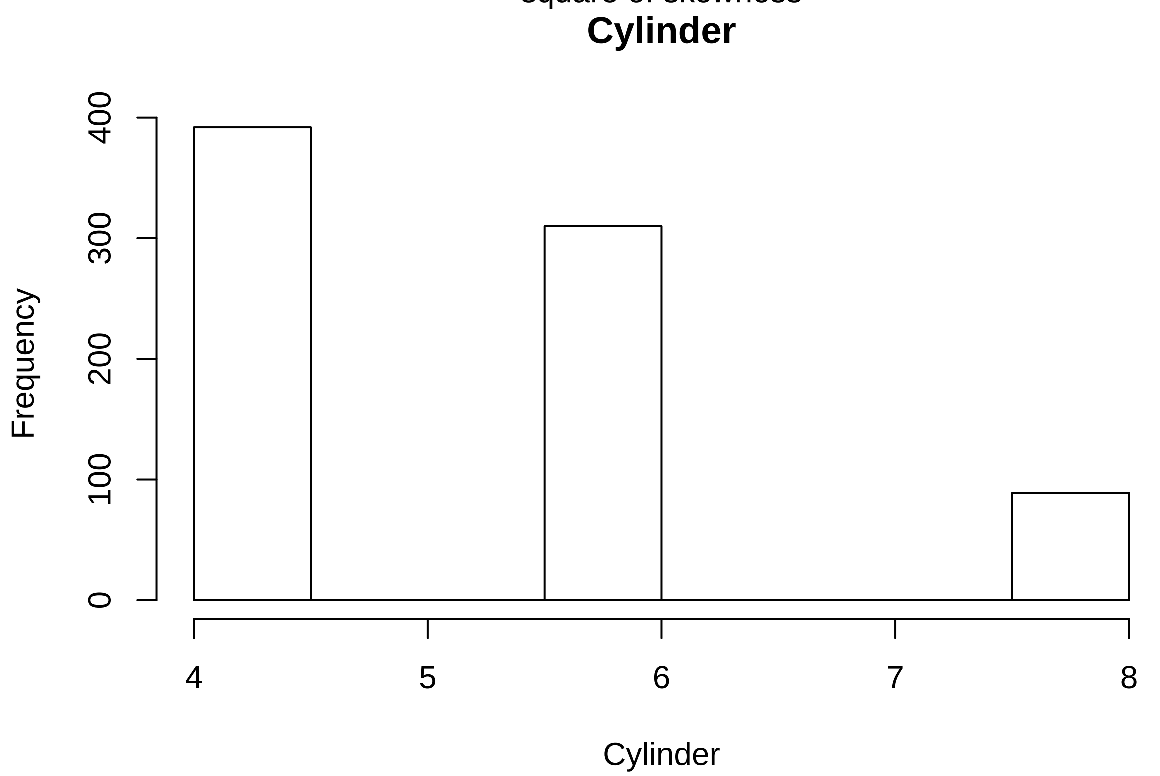



```
## [1] "This graph is for : Make"
```

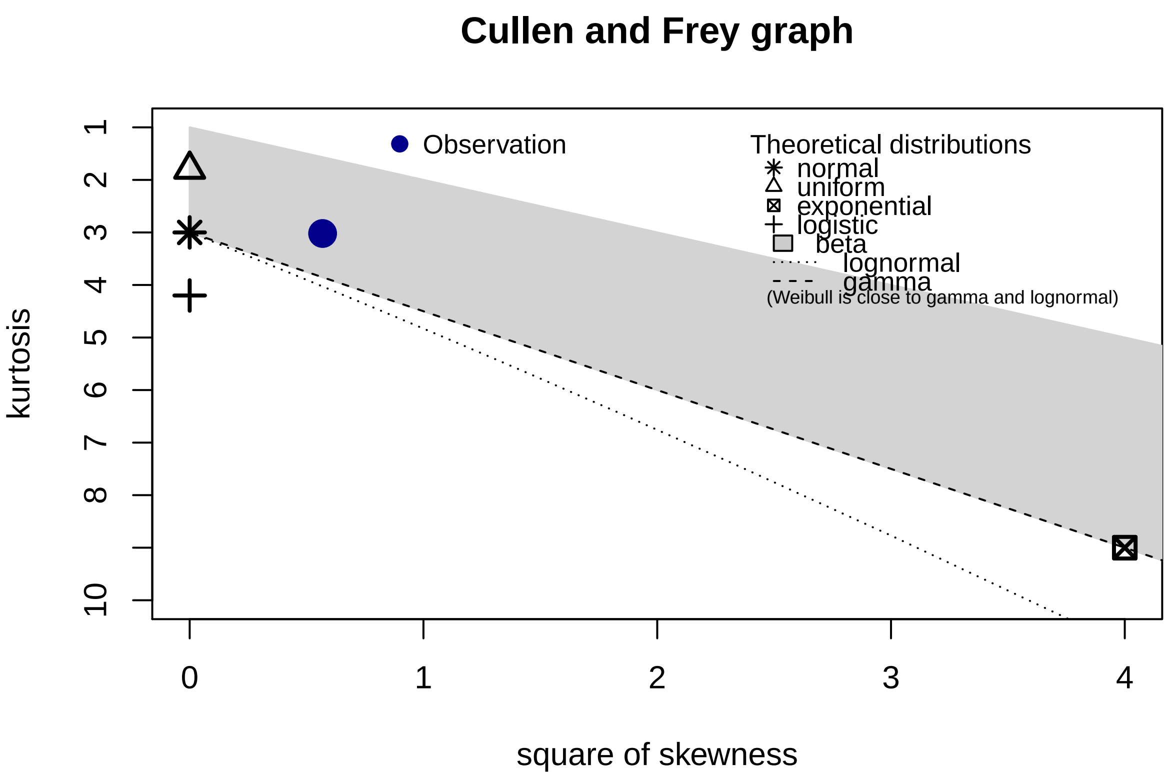


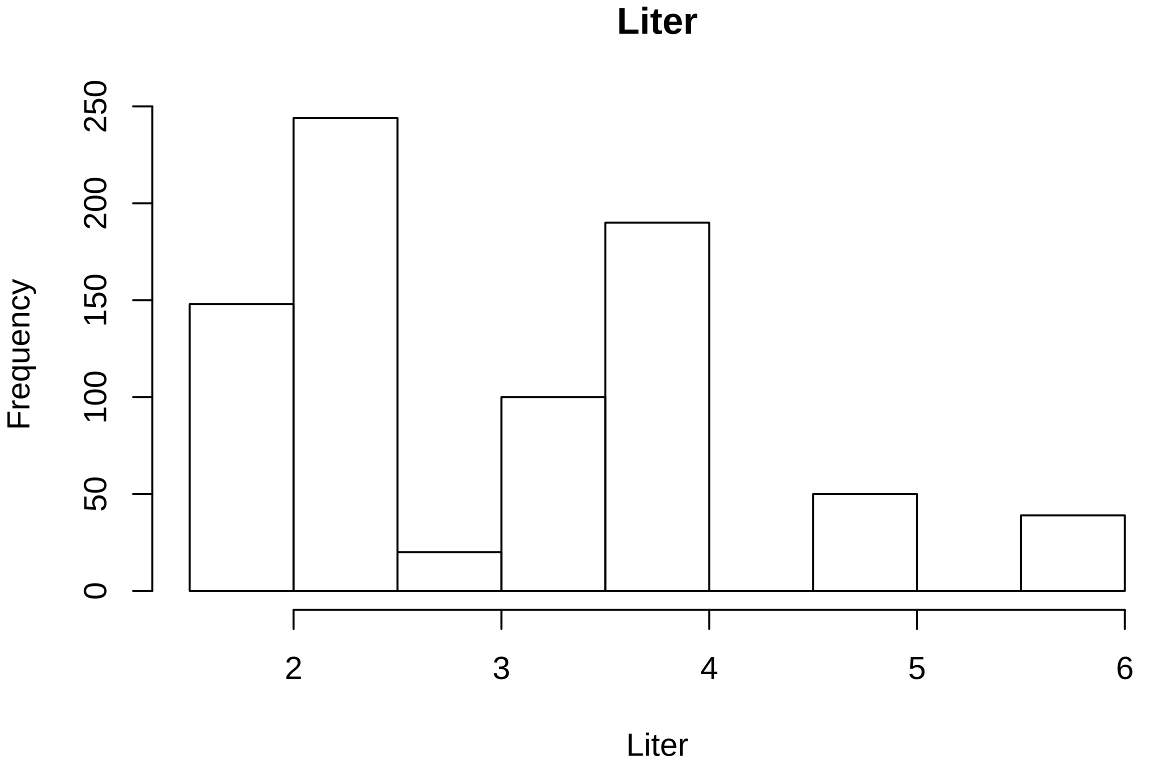

```
## [1] "This graph is for : Cylinder"
```



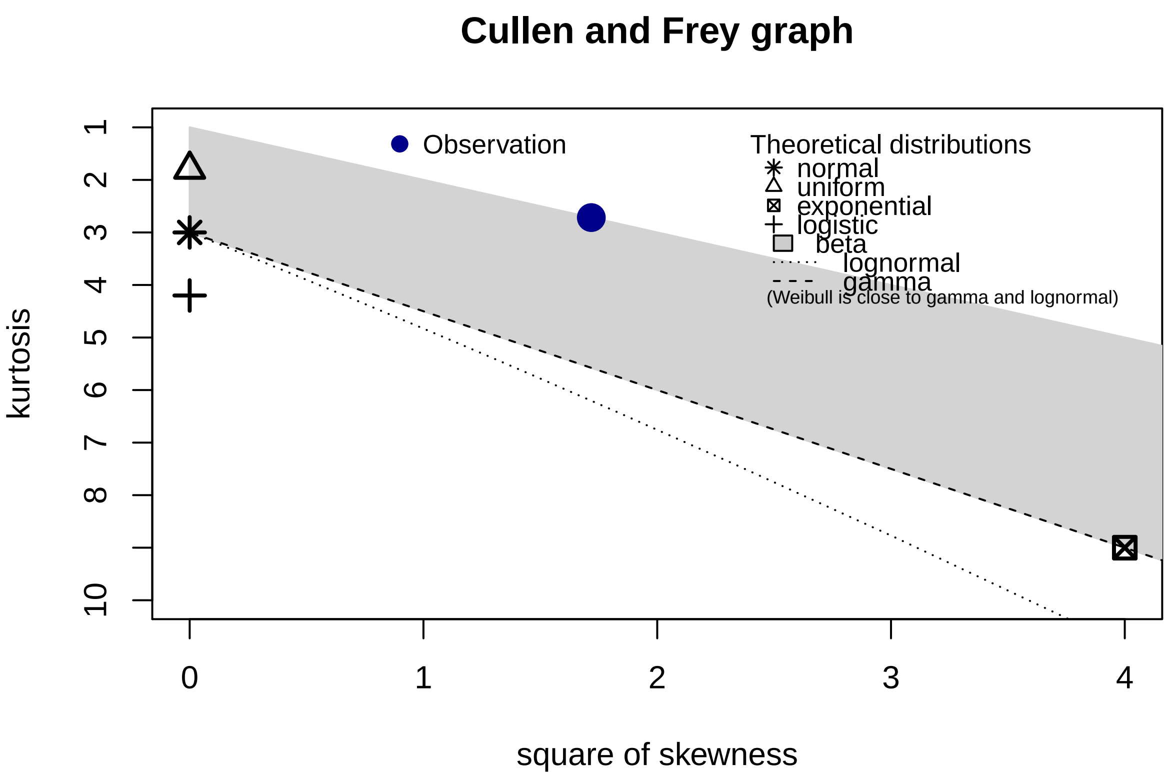


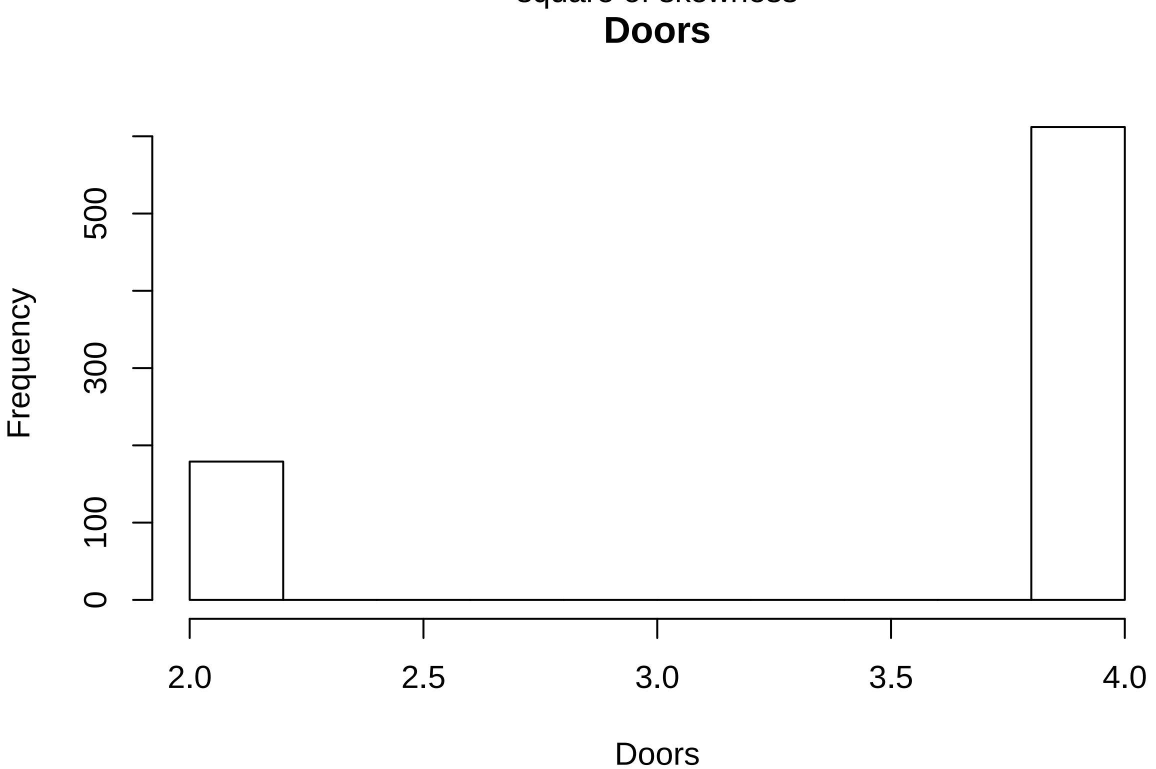

```
## [1] "This graph is for : Liter"
```



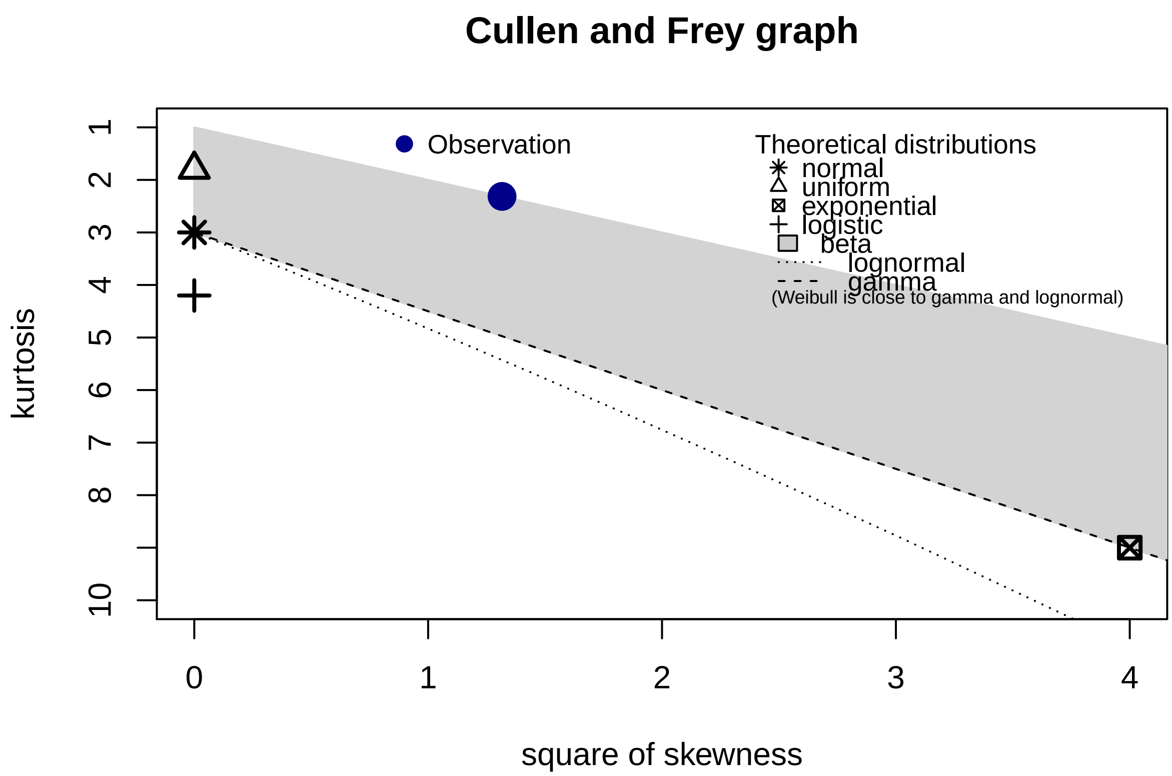


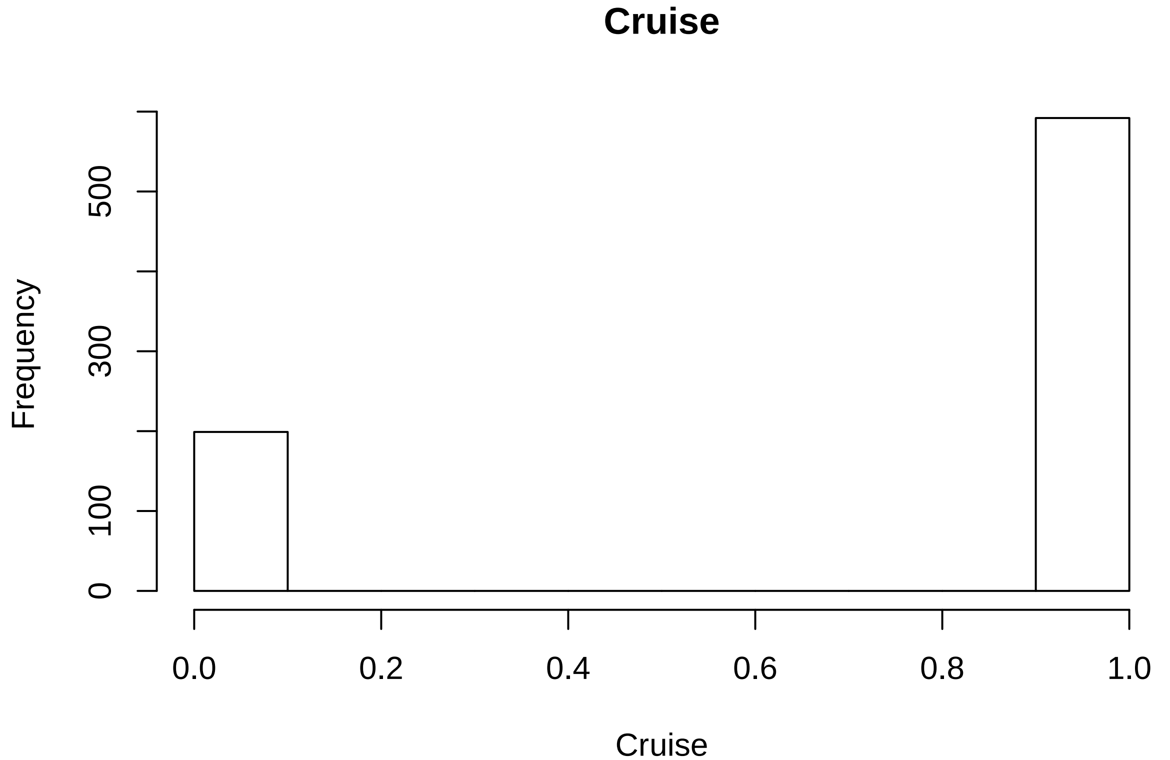

```
## [1] "This graph is for : Doors"
```



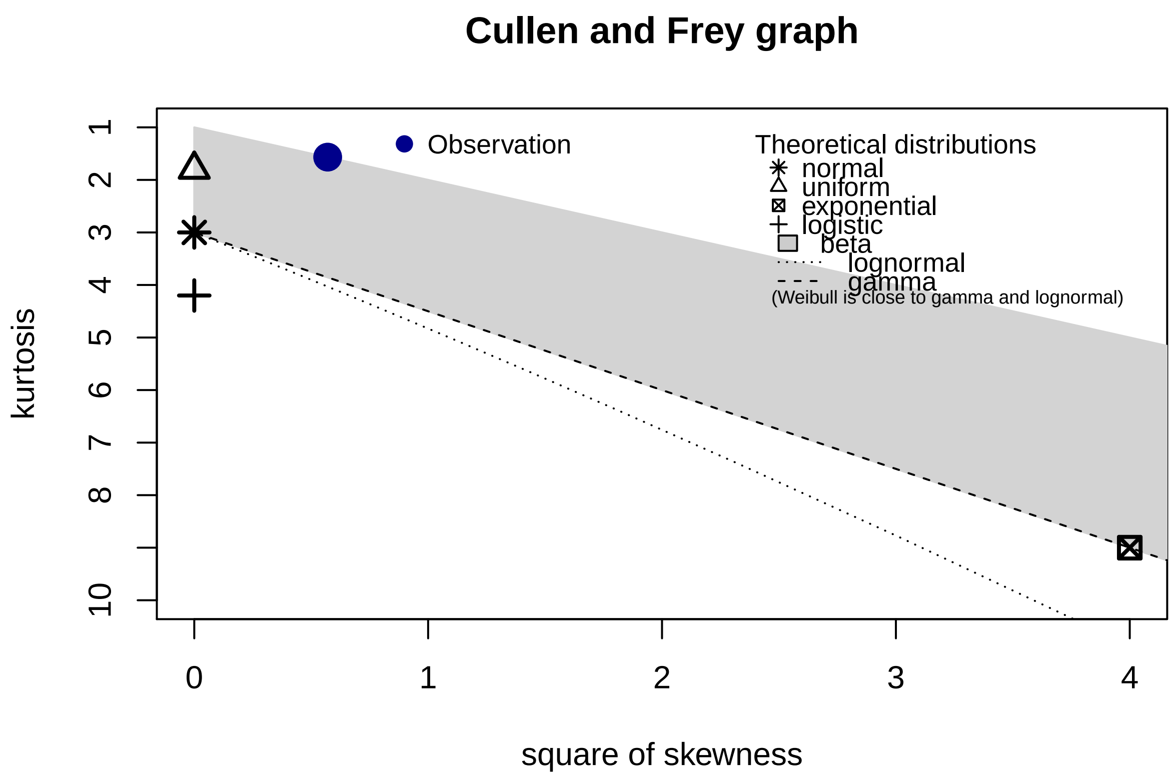


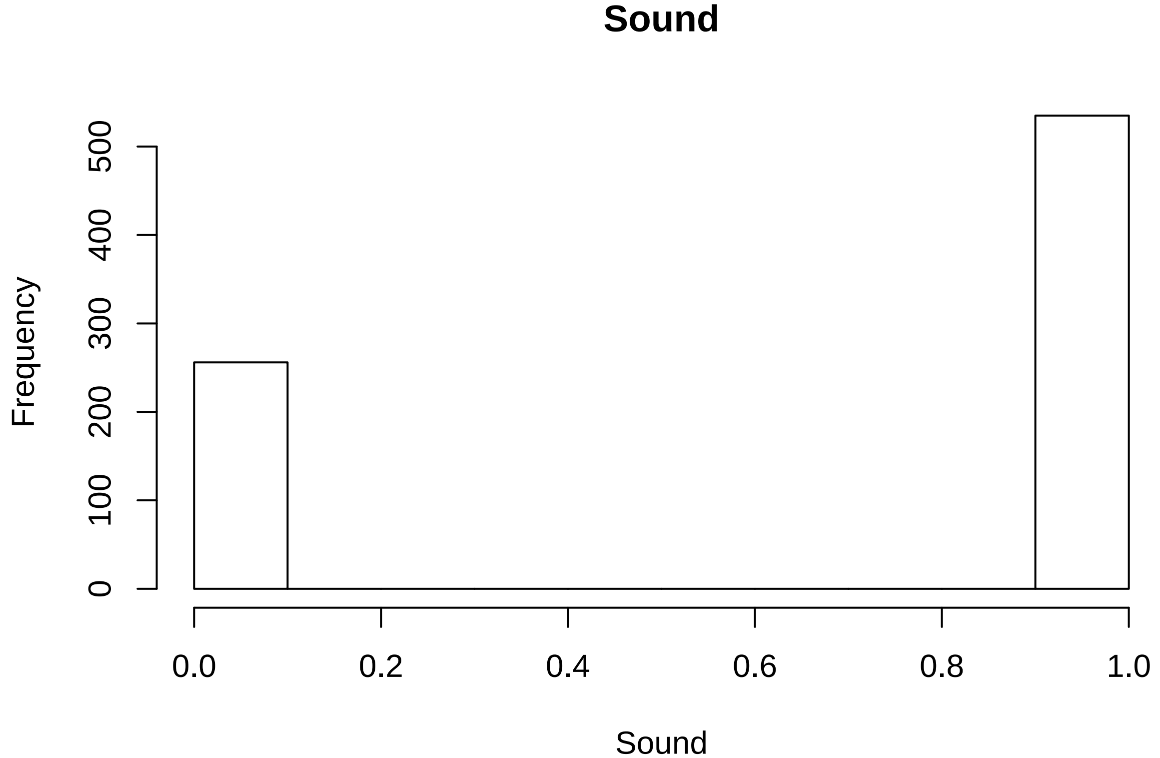

```
## [1] "This graph is for : Cruise"
```



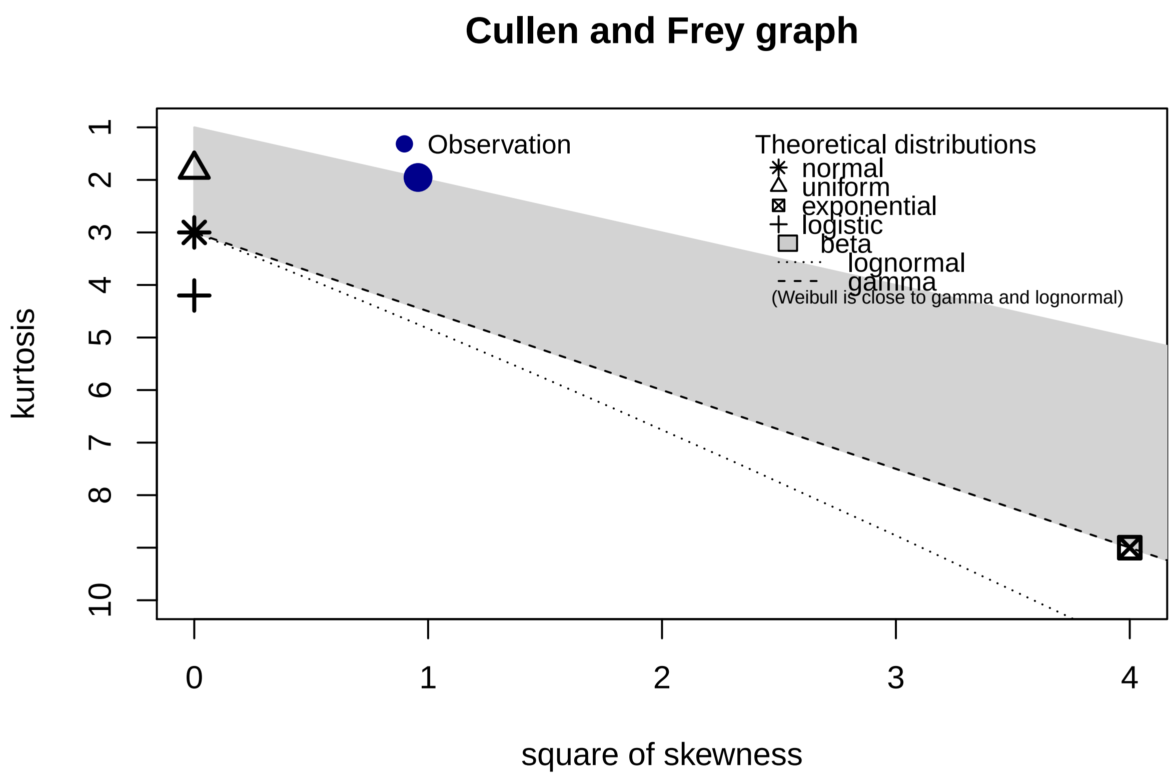


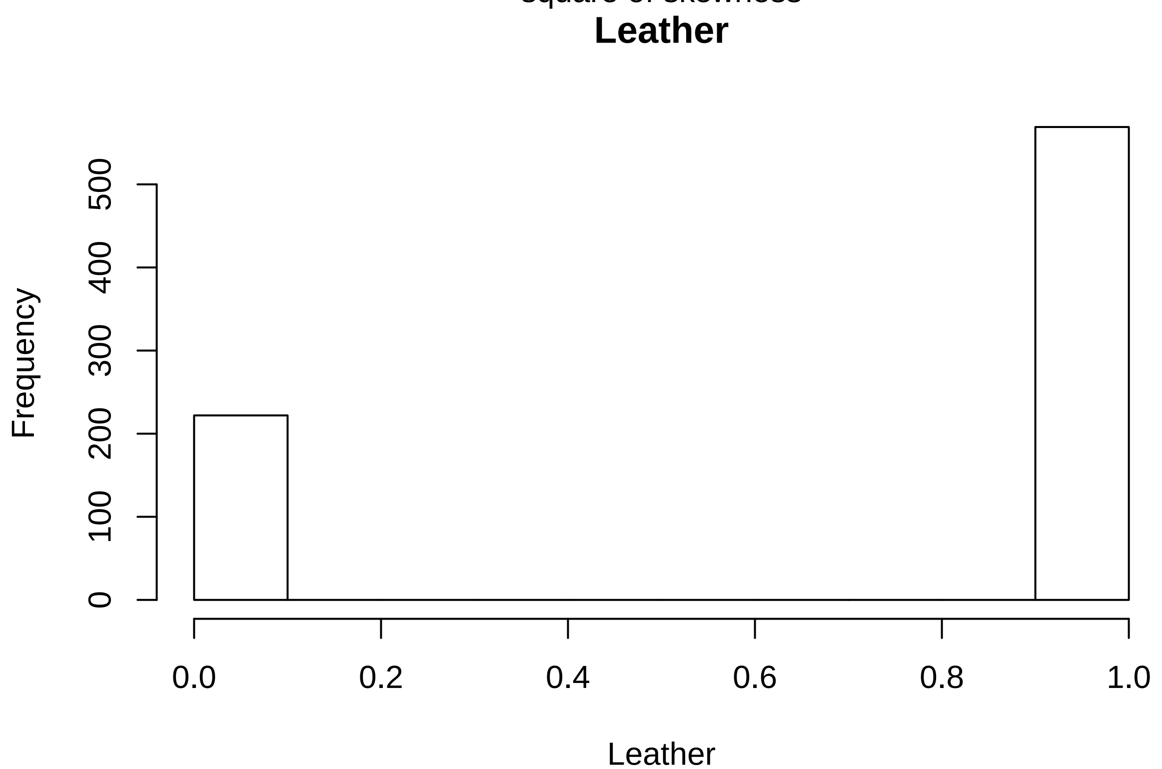

```
## [1] "This graph is for : Sound"
```



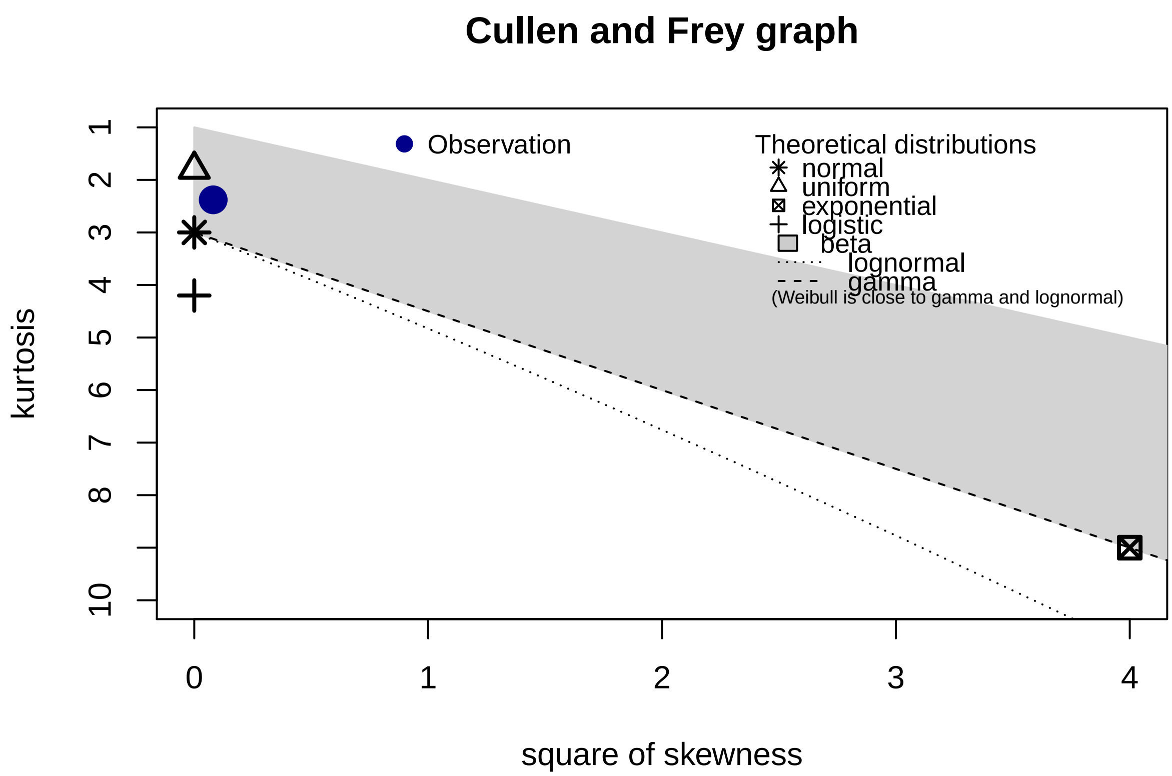



```
## [1] "This graph is for : Leather"
```






```
# Applying the inverse transform on Price to make it normally distributed
descdist((1/(as.numeric(unlist(cars_outlier[,1])))), discrete = F)
```

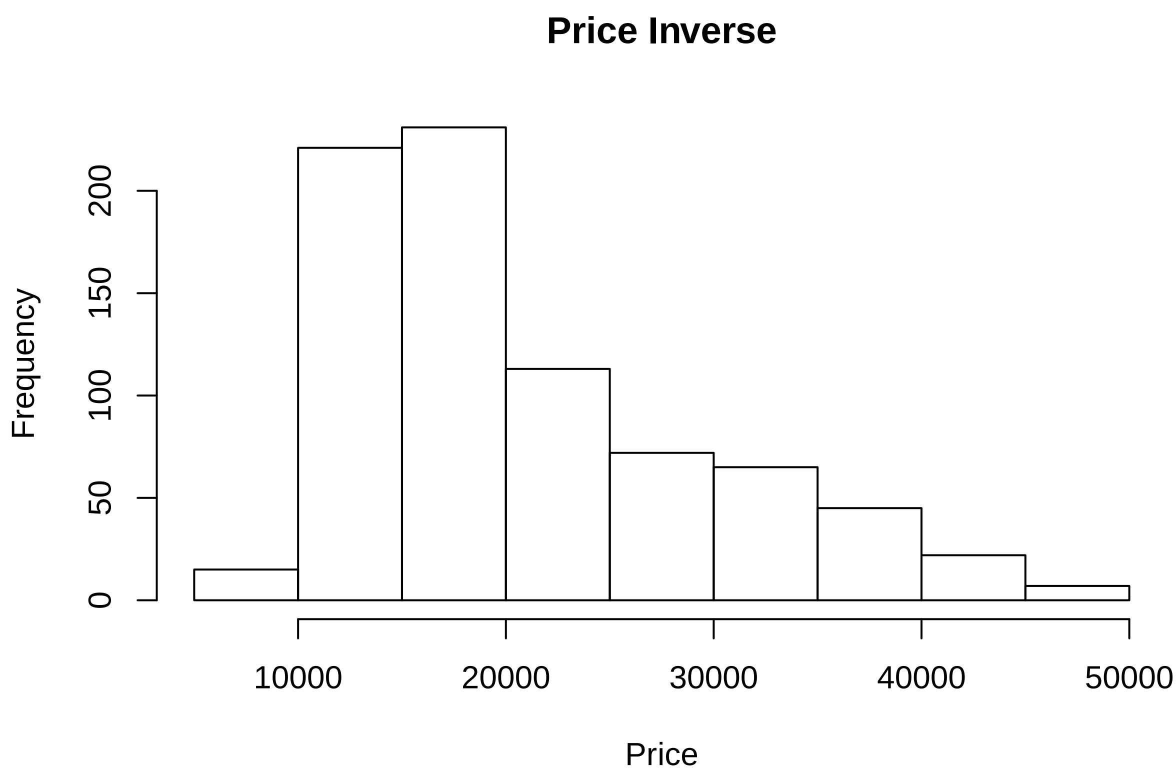



```
## summary statistics
## ------
## min:  2.030533e-05   max:  0.0001157551
## median:  5.589204e-05
## mean:  5.596088e-05
## estimated sd:  2.031868e-05
## estimated skewness:  0.2846514
## estimated kurtosis:  2.379437
```

```
print("Histogram for price column after applying inverse transform:")
```

```
## [1] "Histogram for price column after applying inverse transform:"
```

```
hist(as.numeric((unlist(cars_outlier[,1]))), xlab = "Price", main="Price Inverse")
```



Part 4 What are the correlations to the response variable (car sales price) and are there collinearities? Build a full correlation matrix.

```r
print("Correlation Matrix")
```

```
## [1] "Correlation Matrix"
```

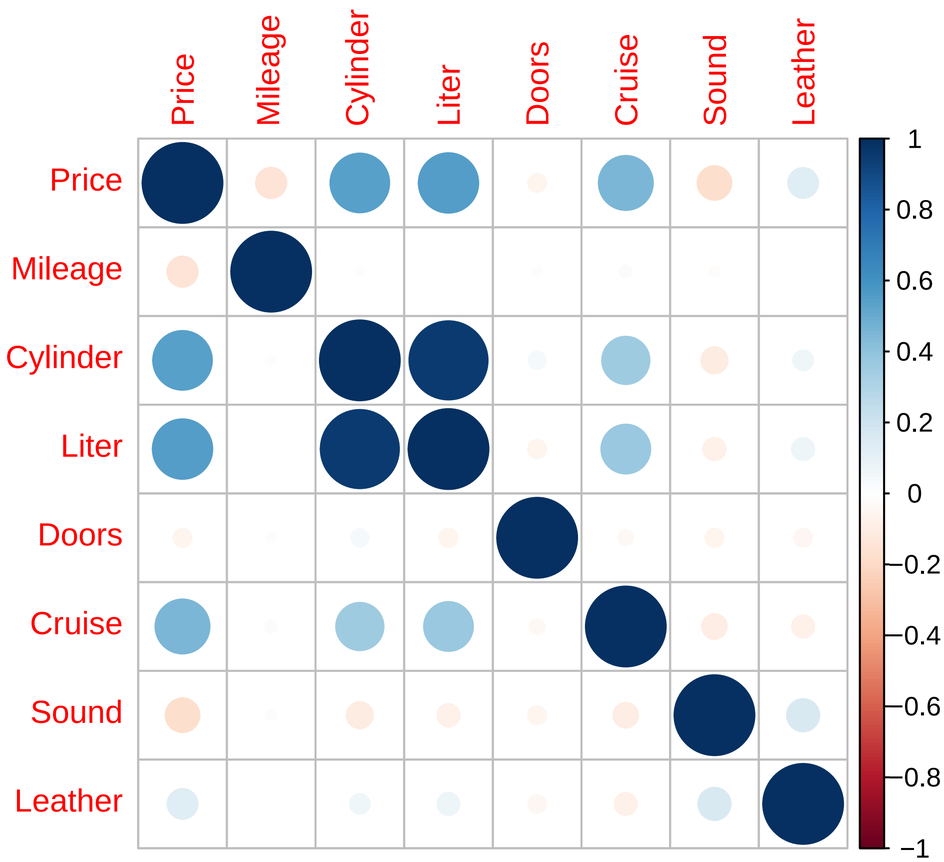
```r
# Generating correlation matrix
matrix <- cor(cars_outlier[sapply(cars_outlier, function (x) !is.character(x))])
matrix
```

```
##                Price      Mileage    Cylinder        Liter       Doors
## Price      1.0000000 -0.143589980  0.54090548  0.554673549 -0.05135330
## Mileage   -0.1435900  1.000000000 -0.01366780 -0.001556543 -0.01364391
## Cylinder   0.5409055 -0.013667802  1.00000000  0.958119540  0.04640472
## Liter      0.5546735 -0.001556543  0.95811954  1.000000000 -0.05242405
## Doors     -0.0513533 -0.013643906  0.04640472 -0.052424050  1.00000000
## Cruise     0.4573726  0.023107437  0.35074745  0.374405142 -0.03504310
## Sound     -0.1783239 -0.016272150 -0.10766242 -0.078747236 -0.05122069
## Leather    0.1396294 -0.002067665  0.06197323  0.077465407 -0.04866768
##                 Cruise       Sound      Leather
## Price      0.45737259 -0.17832388  0.139629388
## Mileage    0.02310744 -0.01627215 -0.002067665
## Cylinder   0.35074745 -0.10766242  0.061973231
## Liter      0.37440514 -0.07874724  0.077465407
## Doors     -0.03504310 -0.05122069 -0.048667680
## Cruise     1.00000000 -0.09592709 -0.076843111
## Sound     -0.09592709  1.00000000  0.163284185
## Leather   -0.07684311  0.16328419  1.000000000
```

```r
library('corrplot')
```

```
## corrplot 0.84 loaded
```

```r
# plotting matrix
corrplot(corr = matrix , method = "circle")
```



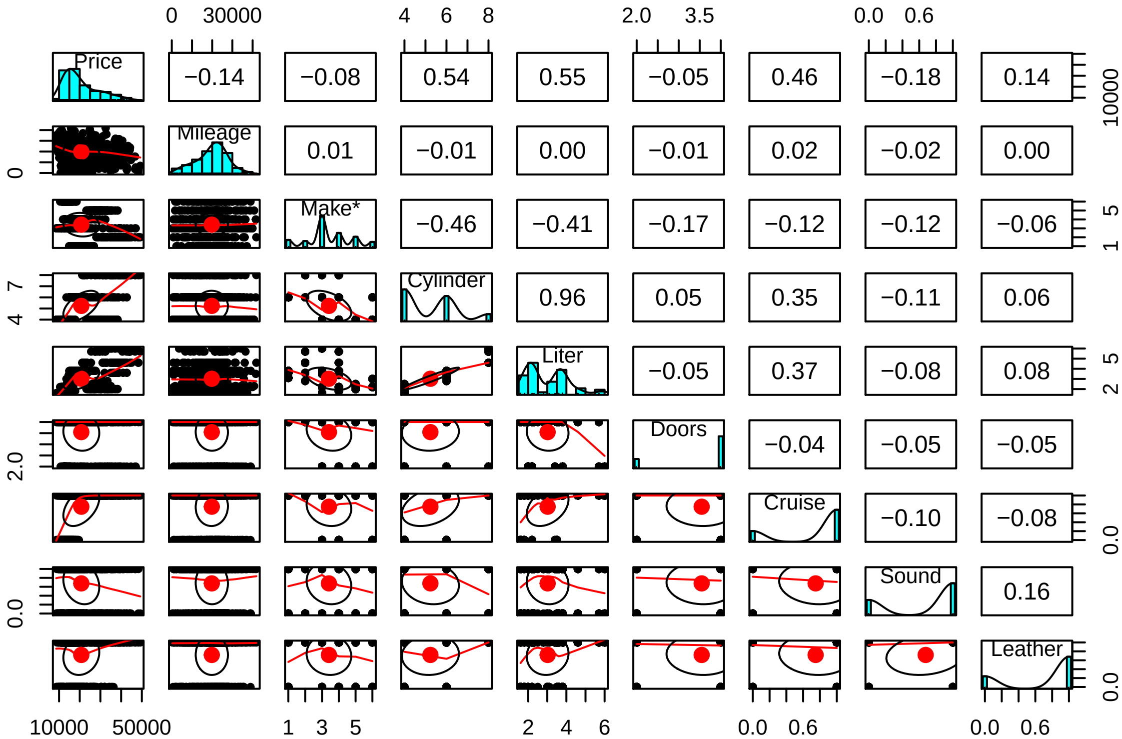

```r
library(psych)
```

```
##
## Attaching package: 'psych'

## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

```r
print("Colinearity Graph:")
```

```
## [1] "Colinearity Graph:"
```

```r
# generating Colinearity graph and matrix
pairs.panels(cars_outlier)
```



It is observed form Both the plots that features Cylinder and Liter are highly correlated and has a high colinearity factor.

Part 5 Split the data set 75/25 so you retain 25% for testing using random sampling.

```
set.seed(123)
# Creating random sample of 75/25 for data with outlier
random_sample2 <- sample(nrow(cars_sales_price), nrow(cars_sales_price)*0.75)
# splitting into train and test data
cars_train <- cars_sales_price[random_sample2,]
cars_test <- cars_sales_price[-random_sample2,]
# Creating random sample of 75/25 for data without outlier
random_sample3 <- sample(nrow(cars_outlier), nrow(cars_outlier)*0.75)
# Splitting into train and test data
cars_outlier_train <- cars_outlier[random_sample3,]
cars_outlier_test <- cars_outlier[-random_sample3,]
```

Part 6 Build a full multiple regression model for predicting car sales prices in this data set using the complete training data set (no outliers removed), i.e., a regression model that contains all features regardless of their p-values.

```
# generating the model
multiple_reg_model <- lm(Price~., data = cars_train)
# summarizing the model
summary(multiple_reg_model)
```

```
##
## Call:
## lm(formula = Price ~ ., data = cars_train)
##
```

```
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9683.6 -1963.8  -189.5  1425.5 22348.1
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    1.531e+04  1.500e+03  10.207  < 2e-16 ***
## Mileage       -1.815e-01  1.734e-02 -10.469  < 2e-16 ***
## MakeCadillac   1.584e+04  7.760e+02  20.416  < 2e-16 ***
## MakeChevrolet -2.483e+03  5.699e+02  -4.357 1.55e-05 ***
## MakePontiac   -2.261e+03  5.823e+02  -3.882 0.000115 ***
## MakeSAAB       1.424e+04  6.924e+02  20.561  < 2e-16 ***
## MakeSaturn    -2.495e+03  7.585e+02  -3.289 0.001066 **
## Cylinder       1.155e+02  4.657e+02   0.248 0.804158
## Liter          4.391e+03  5.296e+02   8.291 7.61e-16 ***
## Doors         -1.750e+03  1.809e+02  -9.673  < 2e-16 ***
## Cruise        -4.823e+02  4.038e+02  -1.194 0.232764
## Sound          1.066e+02  3.234e+02   0.330 0.741750
## Leather        1.170e+02  3.491e+02   0.335 0.737609
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3487 on 590 degrees of freedom
## Multiple R-squared:  0.8767, Adjusted R-squared:  0.8742
## F-statistic: 349.5 on 12 and 590 DF,  p-value: < 2.2e-16
```

The linear model generated using lm() function predicts the Price for cars based on all feature values in the dataset present. It is observed features like Cylinder, Cruise, Sound, and Leather, are not closely correlated to Price of the car, as the p-value of there significance is greater than 0.05 level threshold, thus we can get rid of them by backward elimination method. Model has 0.8742 as Adjusted R-squared value its proportional to the improvement in model due to feature values. For RMSE we obtain 3487, which is basically standard deviation of the residuals, Residuals account for prediction errors. It depends on how far is the measure points from regresion line. RMSE is the measure how far they are spread, RMSE is high here, becuase we have a lots of features included.

Part 7 Build an ideal multiple regression model using backward elimination based on p-value for predicting car sales prices in this data set using the complete training data set with outliers removed (Question 2) and features transformed (Question 3). Provide a detailed analysis of the model using the training data set with outliers removed and features transformed, including Adjusted R-Squared, RMSE, and p-values of all coefficients.

```r
# generating linear model using outlier removed and transformed data
model_mult_outlier <- lm(Price~., data = cars_outlier_train)
# summarizing the model
summary(model_mult_outlier)
```

```
##
## Call:
## lm(formula = Price ~ ., data = cars_outlier_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7712.6 -1508.1  -197.9  1129.6 11657.9
##
```

```
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)    1.453e+04  1.160e+03  12.522  < 2e-16 ***
## Mileage       -1.640e-01  1.365e-02 -12.016  < 2e-16 ***
## MakeCadillac   1.424e+04  6.091e+02  23.382  < 2e-16 ***
## MakeChevrolet -1.683e+03  4.238e+02  -3.971 8.05e-05 ***
## MakePontiac   -1.307e+03  4.332e+02  -3.017 0.002663 **
## MakeSAAB       1.467e+04  5.228e+02  28.066  < 2e-16 ***
## MakeSaturn    -2.047e+03  5.846e+02  -3.501 0.000499 ***
## Cylinder      -1.469e+03  3.702e+02  -3.968 8.17e-05 ***
## Liter          6.189e+03  4.200e+02  14.737  < 2e-16 ***
## Doors         -8.788e+02  1.436e+02  -6.121 1.72e-09 ***
## Cruise        -3.452e+02  3.017e+02  -1.144 0.252919
## Sound         -6.968e+02  2.477e+02  -2.813 0.005075 **
## Leather        2.379e+02  2.607e+02   0.913 0.361875
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2640 on 580 degrees of freedom
## Multiple R-squared:  0.9074, Adjusted R-squared:  0.9055
## F-statistic: 473.6 on 12 and 580 DF,  p-value: < 2.2e-16
```

```r
# removing insignificant features, based on p value > 0.05 i.e Leather
model_mult_outlier1 <- lm(Price~ Mileage+ Make+ Cylinder+ Liter+ Doors+Cruise, data = cars_outlier_train
# Summarizing the model
summary(model_mult_outlier1)
```

```
##
## Call:
## lm(formula = Price ~ Mileage + Make + Cylinder + Liter + Doors +
##     Cruise, data = cars_outlier_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7880.7 -1533.3  -210.7  1165.2 11563.6
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)   13870.0279  1106.5316  12.535  < 2e-16 ***
## Mileage          -0.1618     0.0137 -11.810  < 2e-16 ***
## MakeCadillac  14356.2166   583.9133  24.586  < 2e-16 ***
## MakeChevrolet -1721.6672   417.6110  -4.123 4.29e-05 ***
## MakePontiac   -1250.8594   432.6093  -2.891  0.00398 **
## MakeSAAB      14849.1921   519.8042  28.567  < 2e-16 ***
## MakeSaturn    -1825.6957   582.1254  -3.136  0.00180 **
## Cylinder      -1378.7085   364.3619  -3.784  0.00017 ***
## Liter          6124.8146   413.2188  14.822  < 2e-16 ***
## Doors          -876.0500   144.3464  -6.069 2.32e-09 ***
## Cruise         -360.0679   300.4998  -1.198  0.23132
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2655 on 582 degrees of freedom
## Multiple R-squared:  0.9061, Adjusted R-squared:  0.9045
```

```
## F-statistic: 561.4 on 10 and 582 DF,  p-value: < 2.2e-16
```

```r
# removing insignificant features, based on p value > 0.05 i.e Cruise
model_mult_outlier2 <- lm(Price~ Mileage+ Make+ Cylinder+ Liter+ Doors , data = cars_outlier_train)
# Summarizing the model
summary(model_mult_outlier2)
```

```
##
## Call:
## lm(formula = Price ~ Mileage + Make + Cylinder + Liter + Doors,
##     data = cars_outlier_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7872.5 -1515.4  -180.3  1182.1 11604.9
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)   13682.8504  1095.8591  12.486  < 2e-16 ***
## Mileage          -0.1621     0.0137 -11.830  < 2e-16 ***
## MakeCadillac  14396.6038   583.1575  24.687  < 2e-16 ***
## MakeChevrolet -1606.2178   406.4960  -3.951 8.72e-05 ***
## MakePontiac   -1170.0218   427.4762  -2.737 0.006388 **
## MakeSAAB      14766.9532   515.4457  28.649  < 2e-16 ***
## MakeSaturn    -1673.9547   568.3962  -2.945 0.003358 **
## Cylinder      -1388.7864   364.4009  -3.811 0.000153 ***
## Liter          6078.0976   411.5291  14.770  < 2e-16 ***
## Doors          -859.8230   143.7635  -5.981 3.87e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2656 on 583 degrees of freedom
## Multiple R-squared:  0.9058, Adjusted R-squared:  0.9044
## F-statistic: 623.2 on 9 and 583 DF,  p-value: < 2.2e-16
```

The ideal linear model generated using lm() function predicts the Price for cars based on selected feature values in the dataset present. It is observed features like Cruise and Leather, are not closely correlated to Price of the car in the initial model, as the p-value of there significance is greater than 0.05 level threshold, thus we can get rid of them by backward elimination system to obtain optimal results in final model. Model has 0.9044 as Adjusted R-squared value, significantly better than previous models, Adjusted R-squared value is proportional to the improvement in model due to feature values. For RMSE we obtained 2656, which is basically standard deviation of the residuals, Residuals account for prediction errors. It depends on how far is the measure points from regresion line. RMSE is the measure how far they are spread. Thus, all significant values are better than previously built models, so this results in an ideal model.

(model_mult_outlier2) is the ideal linear model.

Part 8 On average, by how much do we expect a leather interior to change the resale value of a car based on the models built in (6) and in (7)? Note that 1 indicates the presence of leather in the car.

For Model in (6) to identify the change in cost of car, I multiplied minimum and maximum values present of Leather in model to the Leather coefficient, based on model. Then I subtracted them to get the average of each model, this approach works on **range** i.e(max-min). Although minimum value is Null this change depends only on maximum values for this model. For model in (7), its an ideal model, with no outlier and only closely significant variables plus we dont have any leather coefficient in that model. Concludingly, on average based on Leather interior car cost will affect approximately, plus or minus +-117.0178.

```
# getting min and max
min_leather <- min(cars_train$Leather)
max_leather <- max(cars_train$Leather)
#for model in (6)
print("On average car cost affects on presence of leather interior : ")
```

```
## [1] "On average car cost affects on presence of leather interior : "
```

```
multiple_reg_model$coefficients[13] * max_leather - multiple_reg_model$coefficients[13] * min_leather
```

```
##  Leather
## 117.0178
```

Part 9 Using the regression models of (6) and (7) what are the predicted resale prices of a 2005 4-door Saab with 61,435 miles with a leather interior, a 4-cylinder 2.3 liter engine, cruise control, and a premium sound system? Why are the predictions different?

Difference between these predictions is due to the fact that Model in (7) has higher accuracy because the data we used has no outliers to affects prediction, and it is the ideal model with significant variables only which have p-value less then 0.05 significance level. This model also has high adjusted R- squared value and low RMSE values.

```
# generating new dataset for predictions
newcar <- data.frame("Price" = 0, "Mileage" = 61435, "Make" = "SAAB", "Cylinder" = 4, "Liter" = 2.3, "Do
# prediction using two models
paste0("Prediction using model in Part 6 : ", predict(multiple_reg_model, newcar))
```

```
## [1] "Prediction using model in Part 6 : 21696.6599905943"
```

```
paste0("Prediction using model in Part 7 : ", predict(model_mult_outlier2, newcar))
```

```
## [1] "Prediction using model in Part 7 : 23477.2891104563"
```

Part 10 For the regression model of (7), calculate the 95% prediction interval for the car in (9).

```
# predicting price for new car
new_car_pred <- predict(model_mult_outlier2, newcar)
# This is standard error, from the summary of the model
se <- 2656
# Finding the confidence interval
upper <- unname(new_car_pred + 1.96*(se))
lower <- unname(new_car_pred - 1.96*(se))
cat(sprintf("Predicted price: %f \n95%% Confidence interval: [%f, %f]", new_car_pred, lower, upper))
```

```
## Predicted price: 23477.289110
## 95% Confidence interval: [18271.529110, 28683.049110]
```