DA5030: Practicum 2

Code

Virly Ananda

Problem 1

1. Download the data set Census Income Data for Adults along with its explanation (Links to an external site.). There are two data sets (adult.data and adult.test). Note that the data file does not contain header names; you may wish to add those. The description of each column can be found in the data set explanation. Combine the two data sets into a single data set.

Hide

```
# Load RCurl to obtain data directly from the web address
library(RCurl)
```

Hide

```
# Download adult.data and store into current directory
url<-"http://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.data"
destfile<-"/Users/virlyananda/Desktop/DA5030/DA5030.P2.Ananda/adult.data"
download.file(url, destfile)
```

```
trying URL 'http://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.data
'
Content type 'application/x-httpd-php' length 3974305 bytes (3.8 MB)
==================================================
downloaded 3.8 MB
```

Hide

```
# Load adult.data into R: Apply table since .data can be read with .txt/tab deliminat
ed file
AdultDataDF<-read.table("adult.data",sep = ",", fill = F, strip.white = T)
```

Hide

```
# Check adult.data
str(AdultDataDF)
```

```
'data.frame':   32561 obs. of  15 variables:
 $ V1 : int  39 50 38 53 28 37 49 52 31 42 ...
 $ V2 : Factor w/ 9 levels "?","Federal-gov",..: 8 7 5 5 5 5 5 7 5 5 ...
 $ V3 : int  77516 83311 215646 234721 338409 284582 160187 209642 45781 159449 ...
 $ V4 : Factor w/ 16 levels "10th","11th",..: 10 10 12 2 10 13 7 12 13 10 ...
 $ V5 : int  13 13 9 7 13 14 5 9 14 13 ...
 $ V6 : Factor w/ 7 levels "Divorced","Married-AF-spouse",..: 5 3 1 3 3 3 4 3 5 3 ...
 $ V7 : Factor w/ 15 levels "?","Adm-clerical",..: 2 5 7 7 11 5 9 5 11 5 ...
 $ V8 : Factor w/ 6 levels "Husband","Not-in-family",..: 2 1 2 1 6 6 2 1 2 1 ...
 $ V9 : Factor w/ 5 levels "Amer-Indian-Eskimo",..: 5 5 5 3 3 5 3 5 5 5 ...
 $ V10: Factor w/ 2 levels "Female","Male": 2 2 2 2 1 1 1 2 1 2 ...
 $ V11: int  2174 0 0 0 0 0 0 0 14084 5178 ...
 $ V12: int  0 0 0 0 0 0 0 0 0 0 ...
 $ V13: int  40 13 40 40 40 40 16 45 50 40 ...
 $ V14: Factor w/ 42 levels "?","Cambodia",..: 40 40 40 40 6 40 24 40 40 40 ...
 $ V15: Factor w/ 2 levels "<=50K",">50K": 1 1 1 1 1 1 1 2 2 2 ...
```

Hide

```
# Download adult.test
url<-"http://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.test"
destfile<-"/Users/virlyananda/Desktop/DA5030/DA5030.P2.Ananda/adult.test"
download.file(url, destfile)
```

```
trying URL 'http://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.test
'
Content type 'application/x-httpd-php' length 2003153 bytes (1.9 MB)
==================================================
downloaded 1.9 MB
```

Hide

```
#install.packages("tm")
library(tm)
```

Hide

```
# Load adult.data into R: Apply table since .test can be read with .txt/tab deliminat
ed file
AdultTestDF<-read.table("adult.test", fill = F, sep = ",", skip = 1, strip.white = T)

# Remove the "."
AdultTestDF$V15<-sub('[[:punct:]]+$', '', AdultTestDF$V15)

# Convert back to factor
AdultTestDF$V15<-as.factor(AdultTestDF$V15)

# Check adult.test dataset
str(AdultTestDF)
```

```
'data.frame':   16281 obs. of  15 variables:
 $ V1 : int  25 38 28 44 18 34 29 63 24 55 ...
 $ V2 : Factor w/ 9 levels "?","Federal-gov",..: 5 5 3 5 1 5 1 7 5 5 ...
 $ V3 : int  226802 89814 336951 160323 103497 198693 227026 104626 369667 104996 ...
 $ V4 : Factor w/ 16 levels "10th","11th",..: 2 12 8 16 16 1 12 15 16 6 ...
 $ V5 : int  7 9 12 10 10 6 9 15 10 4 ...
 $ V6 : Factor w/ 7 levels "Divorced","Married-AF-spouse",..: 5 3 3 3 5 5 5 3 5 3 ...
 $ V7 : Factor w/ 15 levels "?","Adm-clerical",..: 8 6 12 8 1 9 1 11 9 4 ...
 $ V8 : Factor w/ 6 levels "Husband","Not-in-family",..: 4 1 1 1 4 2 5 1 5 1 ...
 $ V9 : Factor w/ 5 levels "Amer-Indian-Eskimo",..: 3 5 5 3 5 5 3 5 5 5 ...
 $ V10: Factor w/ 2 levels "Female","Male": 2 2 2 2 1 2 2 2 1 2 ...
 $ V11: int  0 0 0 7688 0 0 0 3103 0 0 ...
 $ V12: int  0 0 0 0 0 0 0 0 0 0 ...
 $ V13: int  40 50 40 40 30 30 40 32 40 10 ...
 $ V14: Factor w/ 41 levels "?","Cambodia",..: 39 39 39 39 39 39 39 39 39 39 ...
 $ V15: Factor w/ 2 levels "<=50K",">50K": 1 1 2 2 1 1 1 2 1 1 ...
```

Once we have the 2 datasets loaded into our R environment. We then merge them together by applying rbind() since the variables within the 2 datasets contain same variables and values(elements). Thus, our goal here to update the elements from the 1st dataset:

Hide

```
AdultDF<-rbind(AdultDataDF, AdultTestDF)

# Check the properties and dimension of the updated dataset
str(AdultDF)
```

```
'data.frame':	48842 obs. of  15 variables:
 $ V1 : int  39 50 38 53 28 37 49 52 31 42 ...
 $ V2 : Factor w/ 9 levels "?","Federal-gov",..: 8 7 5 5 5 5 5 7 5 5 ...
 $ V3 : int  77516 83311 215646 234721 338409 284582 160187 209642 45781 159449 ...
 $ V4 : Factor w/ 16 levels "10th","11th",..: 10 10 12 2 10 13 7 12 13 10 ...
 $ V5 : int  13 13 9 7 13 14 5 9 14 13 ...
 $ V6 : Factor w/ 7 levels "Divorced","Married-AF-spouse",..: 5 3 1 3 3 3 4 3 5 3 ...
 $ V7 : Factor w/ 15 levels "?","Adm-clerical",..: 2 5 7 7 11 5 9 5 11 5 ...
 $ V8 : Factor w/ 6 levels "Husband","Not-in-family",..: 2 1 2 1 6 6 2 1 2 1 ...
 $ V9 : Factor w/ 5 levels "Amer-Indian-Eskimo",..: 5 5 5 3 3 5 3 5 5 5 ...
 $ V10: Factor w/ 2 levels "Female","Male": 2 2 2 2 1 1 1 2 1 2 ...
 $ V11: int  2174 0 0 0 0 0 0 0 14084 5178 ...
 $ V12: int  0 0 0 0 0 0 0 0 0 0 ...
 $ V13: int  40 13 40 40 40 40 16 45 50 40 ...
 $ V14: Factor w/ 42 levels "?","Cambodia",..: 40 40 40 40 6 40 24 40 40 40 ...
 $ V15: Factor w/ 2 levels "<=50K",">50K": 1 1 1 1 1 1 1 2 2 2 ...
```

Fix the Combined Dataset:

In this section, in order to better view our dataset, we rename our columns accordingly as shown below:

Hide

```
# View original column names
colnames(AdultDF)
```

```
 [1] "V1"  "V2"  "V3"  "V4"  "V5"  "V6"  "V7"  "V8"  "V9"  "V10" "V11" "V12" "V13" "V
14" "V15"
```

Hide

```
# Rename Columns
names(AdultDF)<-c("age","workclass","fnlwgt","education","education-num","marital-sta
tus","occupation","relationship","race","sex","capital-gain","capital-loss","hours-pe
r-week","native.country", "income")

# Check the updated columns
head(AdultDF)
```

	...	workclass	fnlwgt	education	education-num	marital-status	occupation
	<int>	<fctr>	<int>	<fctr>	<int>	<fctr>	<fctr>
1	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical
2	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-manag

3	38	Private	215646	HS-grad	9	Divorced	Handlers-cle
4	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cle
5	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialt
6	37	Private	284582	Masters	14	Married-civ-spouse	Exec-manag

6 rows | 1-8 of 15 columns

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

In this section, we view the combined dataset where all columns have been renamed. In total we have 48842 observations out of 15 features/variables.

Hide

```
# View updated column names
str(AdultDF)
```

```
'data.frame':	48842 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 1
59449 ...
 $ 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 4
0 ...
 $ income        : Factor w/ 2 levels "<=50K",">50K": 1 1 1 1 1 1 1 2 2 2 ...
```

Hide

```
# Check the summary of each variables
summary(AdultDF)
```

```
      age                    workclass          fnlwgt              education     educ
ation-num
 Min.   :17.00   Private         :33906   Min.   :  12285   HS-grad     :15784   Min.
: 1.00
 1st Qu.:28.00   Self-emp-not-inc: 3862   1st Qu.: 117550   Some-college:10878   1st
Qu.: 9.00
 Median :37.00   Local-gov       : 3136   Median : 178144   Bachelors   : 8025   Medi
an :10.00
 Mean   :38.64   ?               : 2799   Mean   : 189664   Masters     : 2657   Mean
:10.08
 3rd Qu.:48.00   State-gov       : 1981   3rd Qu.: 237642   Assoc-voc   : 2061   3rd
Qu.:12.00
 Max.   :90.00   Self-emp-inc    : 1695   Max.   :1490400   11th        : 1812   Max.
:16.00
                 (Other)         : 1463                     (Other)     : 7625
               marital-status            occupation            relationship
race
 Divorced             : 6633   Prof-specialty : 6172   Husband       :19716   Amer-In
dian-Eskimo:  470
 Married-AF-spouse    :   37   Craft-repair   : 6112   Not-in-family :12583   Asian-P
ac-Islander: 1519
 Married-civ-spouse   :22379   Exec-managerial: 6086   Other-relative: 1506   Black
: 4685
 Married-spouse-absent:  628   Adm-clerical   : 5611   Own-child     : 7581   Other
:  406
 Never-married        :16117   Sales          : 5504   Unmarried     : 5125   White
:41762
 Separated            : 1530   Other-service  : 4923   Wife          : 2331
 Widowed              : 1518   (Other)        :14434
     sex         capital-gain    capital-loss    hours-per-week       native.country
income
 Female:16192   Min.   :    0   Min.   :   0.0   Min.   : 1.00   United-States:43832
<=50K:37155
 Male  :32650   1st Qu.:    0   1st Qu.:   0.0   1st Qu.:40.00   Mexico       :  951
>50K :11687
                Median :    0   Median :   0.0   Median :40.00   ?            :  857
                Mean   : 1079   Mean   :  87.5   Mean   :40.42   Philippines  :  295
                3rd Qu.:    0   3rd Qu.:   0.0   3rd Qu.:45.00   Germany      :  206
                Max.   :99999   Max.   :4356.0   Max.   :99.00   Puerto-Rico  :  184
                                                                 (Other)      : 2517
```

In the code chunk below, we determine whether our dataset contain any missing values:

Hide

```
anyNA(AdultDF)
```

```
[1] FALSE
```

CLEAN DATA

Hide

```
# Check the elements within workclass values
table(AdultDF$workclass)
```

```


               ?      Federal-gov        Local-gov     Never-worked          Private
Self-emp-inc
            2799             1432             3136               10            33906
1695
Self-emp-not-inc        State-gov      Without-pay
            3862             1981               21
```

Based on the result shown above, we can see that “?” can be seen as unknown element. Since unknown value could create problematic analysis, we decide to consider “?” as NA and drop these values.

Hide

```
AdultDF$workclass <- as.character(AdultDF$workclass)
AdultDF$workclass[AdultDF$workclass == "?"] <- "Unknown"
table(AdultDF$workclass)
```

```


     Federal-gov        Local-gov     Never-worked          Private     Self-emp-inc
Self-emp-not-inc
            1432             3136               10            33906             1695
3862
       State-gov          Unknown      Without-pay
            1981             2799               21
```

Hide

```
# Convert workclass back to factor
AdultDF$workclass<-as.factor(AdultDF$workclass)
head(AdultDF)
```

...	workclass	fnlwgt	education	education-num	marital-status	occupation

	<int>	<fctr>	<int>	<fctr>	<int>	<fctr>	<fctr>
1	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical
2	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-manag
3	38	Private	215646	HS-grad	9	Divorced	Handlers-cle
4	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cle
5	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialt
6	37	Private	284582	Masters	14	Married-civ-spouse	Exec-manag

6 rows | 1-8 of 15 columns

Hide

```
# Replace all ? values within each variables and consider them as NA
AdultDF[AdultDF == "?"] <- NA
```

Hide

```
# Remove all NA values
AdultDF<-na.omit(AdultDF)

# Check if the dataset contains NA values
anyNA(AdultDF)
```

```
[1] FALSE
```

3. Split the combined data set 70/30% so you retain 30% for validation and tuning using random sampling with replacement. Use a fixed seed so you produce the same results each time you run the code. Going forward you will use the 70% data set for training and the 30% data set for validation and determine accuracy.

Hide

```
library(caret)
```

Hide

```
# Apply set seed
set.seed(123)

# Split the combined dataset to 70% training and 30% validation with random sampling:
RandomSample<-createDataPartition(AdultDF$income, p= 0.7, list = F)

# Randomly split
trainSample<-AdultDF[RandomSample,]
validateSample<-AdultDF[-RandomSample,]
```

4. Using the Naive Bayes Classification algorithm from the KlaR package, build a binary 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 min to max).

Hide

```
# Install and Load KlaR Package
#install.packages("klaR")
library("klaR")
```

Hide

```
# Load DPLYR to manipulate data
library(dplyr)
```

To proceed, we first check which of our variables contain continous variable:

Hide

```
# Build a the model
nb_model<-trainSample
```

Based on the str() result shown above, we can see that column age is a continuous variable. From here, we transform column age to categorical by using cut().

Hide

```
# Observe first 5 age
nb_model$age[1:5]
```

```
[1] 39 50 38 53 28
```

Hide

```
# Convert age to categorical variables using cut() to 4 levels
nb_model$age<-cut(nb_model$age, breaks = 4)

# Check and observe the first 5 age
nb_model$age[1:5]
```

```
[1] (35.2,53.5] (35.2,53.5] (35.2,53.5] (35.2,53.5] (16.9,35.2]
Levels: (16.9,35.2] (35.2,53.5] (53.5,71.8] (71.8,90.1]
```

Build Naive Bayes Model:

Hide

```
binary_model<-NaiveBayes(income ~ age + education + workclass + sex + race + native.c
ountry , data = nb_model)

binary_model
```

Predict Model:

Hide

```
bm_Pred<-predict(binary_model, validateSample)
```

```
Numerical 0 probability for all classes with observation 1885Numerical 0 probability
for all classes with observation 1922Numerical 0 probability for all classes with obs
ervation 2722Numerical 0 probability for all classes with observation 3993Numerical 0
probability for all classes with observation 4669Numerical 0 probability for all clas
ses with observation 4965Numerical 0 probability for all classes with observation 664
8Numerical 0 probability for all classes with observation 6772Numerical 0 probability
for all classes with observation 7329Numerical 0 probability for all classes with obs
ervation 7462Numerical 0 probability for all classes with observation 7551Numerical 0
probability for all classes with observation 8495Numerical 0 probability for all clas
ses with observation 9334Numerical 0 probability for all classes with observation 934
6Numerical 0 probability for all classes with observation 10413Numerical 0 probabilit
y for all classes with observation 11110Numerical 0 probability for all classes with
observation 11742Numerical 0 probability for all classes with observation 12287Numeri
cal 0 probability for all classes with observation 13317
```

5. Build a confusion matrix for the classifier from (4) and comment on it, e.g., explain what it means.

Hide

```
confusionMatrix(bm_Pred$class, validateSample$income)
```

```
Confusion Matrix and Statistics

          Reference
Prediction <=50K >50K
     <=50K  9397 1972
     >50K    807 1390

               Accuracy : 0.7951
                 95% CI : (0.7883, 0.8019)
    No Information Rate : 0.7522
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.3783

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9209
            Specificity : 0.4134
         Pos Pred Value : 0.8265
         Neg Pred Value : 0.6327
             Prevalence : 0.7522
         Detection Rate : 0.6927
   Detection Prevalence : 0.8381
      Balanced Accuracy : 0.6672

       'Positive' Class : <=50K

```

To apply confusion matrix, we directly applied bm_Pred$class which already is a factor variable containing levels of income from bm_Pred. Based on the result, we can see that we obtainex 79.5% accuracy.

6. 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.

Hide

```
glmFit<-glm(income ~ age + education + workclass + sex + race + native.country , fami
ly = binomial, data = nb_model)
```

Hide

```
summary(glmFit)
```

Hide

```
# Convert validate sample age to categorical so that it can match with our model
validateSample$age<-cut(validateSample$age, breaks = 4)
glmPredict<-predict(glmFit, validateSample, type = "response")
```

7. Build a confusion matrix for the classifier from (6) and comment on it, e.g., explain what it means.

Hide

```
# Converting from probability to actual output: If values are greater than 50%, we co
nsider them as greater than 50K otherwise <=50K.
glmPrediction<-as.factor(ifelse(glmPredict >= 0.5, ">50K", "<=50K"))
```

Hide

```
confusionMatrix(glmPrediction, validateSample$income)
```

```
Confusion Matrix and Statistics

          Reference
Prediction <=50K >50K
     <=50K  9558 2089
     >50K    646 1273

               Accuracy : 0.7984
                 95% CI : (0.7915, 0.8051)
    No Information Rate : 0.7522
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.3683

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9367
            Specificity : 0.3786
         Pos Pred Value : 0.8206
         Neg Pred Value : 0.6634
             Prevalence : 0.7522
         Detection Rate : 0.7046
   Detection Prevalence : 0.8585
      Balanced Accuracy : 0.6577

       'Positive' Class : <=50K
```

Based on our Confusion Matrix accuracy test, our prediction model obtain 80% accuracy with 9558 cases under less than or equal to 50K and 1273 cases under greater than 50K accurate.

8. Create a Decision Tree model from rpart package, build a classifier that predicts whether an individual earns more than or less than US$50,000. Use the same features as (4). Make sure to transform categorical variables.

Hide

```
# Install and load rpart package
library(rpart)
```

Hide

```
# Train the model
rpartmodel<-rpart(income ~ age + education + workclass + sex + race + native.country,
data = nb_model)
rpartmodel
```

```
n= 31656

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 31656 7846 <=50K (0.7521481 0.2478519)
   2) education=10th,11th,12th,1st-4th,5th-6th,7th-8th,9th,Assoc-acdm,Assoc-voc,HS-gr
ad,Preschool,Some-college 23660 3946 <=50K (0.8332206 0.1667794) *
   3) education=Bachelors,Doctorate,Masters,Prof-school 7996 3900 <=50K (0.5122561 0.
4877439)
     6) age=(16.9,35.2] 3007  894 <=50K (0.7026937 0.2973063) *
     7) age=(35.2,53.5],(53.5,71.8],(71.8,90.1] 4989 1983 >50K (0.3974744 0.6025256)
      14) sex=Female 1252  446 <=50K (0.6437700 0.3562300) *
      15) sex=Male 3737 1177 >50K (0.3149585 0.6850415) *
```

Hide

```
# Make prediction
rpartPredict<-predict(rpartmodel, validateSample, type = "class")
```

9. Build a confusion matrix for the classifier from (8) and comment on it, e.g., explain what it means.

Hide

```
confusionMatrix(rpartPredict, validateSample$income)
```

```
Confusion Matrix and Statistics

          Reference
Prediction <=50K >50K
     <=50K  9670 2227
     >50K    534 1135
                                          
               Accuracy : 0.7965          
                 95% CI : (0.7896, 0.8032)
    No Information Rate : 0.7522          
    P-Value [Acc > NIR] : < 2.2e-16       
                                          
                  Kappa : 0.3432          
                                          
 Mcnemar's Test P-Value : < 2.2e-16       
                                          
            Sensitivity : 0.9477          
            Specificity : 0.3376          
         Pos Pred Value : 0.8128          
         Neg Pred Value : 0.6800          
             Prevalence : 0.7522          
         Detection Rate : 0.7128          
   Detection Prevalence : 0.8770          
      Balanced Accuracy : 0.6426          
                                          
       'Positive' Class : <=50K           
```

Based on the confusion matrix shown above, we are able to obtain an accuracy of 79.6% accuracy from the decision tree model through rpart application. Notice that during application of predict() function, we applied type = “class” since we want to obtain prediction on categorical income level.

Thus far, glm linear regression model shows the highest accuracy compared to naivebayes and decision trees model.

10. Build a function called predictEarningsClass() that predicts whether an individual makes more or less than US$50,000 and that combines the three predictive models from (4), (6), and (8) into a simple ensemble. If all three models disagree on a prediction, then the prediction should be the one from the model with the higher accuracy – make sure you do not hard code that as the training data may change over time and the same model may not be the more accurate forever.

Hide

```
library(caret)
```

Hide

```
# predictors and outcome

Predictors<-c("age", "education", "workclass", "sex", "race", "native.country")
Result<-'income'
```

Hide

```
# Groupt Predicion
pred_nb<-bm_Pred$class
pred_glm<-glmPrediction
pred_dt<-rpartPredict
```

```
# Put glm as the first layer
predictlayers<-c('pred_nb_prob', 'pred_dt')
```

11. Using the ensemble model from (10), predict whether a 47-year-old black female adult who is a local government worker with a Bacherlor's degree who immigrated from Honduras earns more or less than US$50,000.

Problem 2

1. Load and then explore this data set on car sales download into a dataframe called cars.df. Exclude name (manufacturer and model) from the data – do not use in any of the modeling going forward.

Hide

```
# Load CSV file from local
cars.df<-read.csv("CarDataSet.csv")

# Exclued name
str(cars.df)
```

```
'data.frame':	4340 obs. of  8 variables:
 $ name         : Factor w/ 1491 levels "Ambassador CLASSIC 1500 DSL AC",..: 774 1040
566 120 278 811 605 1257 391 833 ...
 $ year         : int  2007 2007 2012 2017 2014 2007 2016 2014 2015 2017 ...
 $ selling_price: int  60000 135000 600000 250000 450000 140000 550000 240000 850000
365000 ...
 $ km_driven    : int  70000 50000 100000 46000 141000 125000 25000 60000 25000 78000
...
 $ fuel         : Factor w/ 5 levels "CNG","Diesel",..: 5 5 2 5 2 5 5 5 5 1 ...
 $ seller_type  : Factor w/ 3 levels "Dealer","Individual",..: 2 2 2 2 2 2 2 2 2 2 ..
.
 $ transmission : Factor w/ 2 levels "Automatic","Manual": 2 2 2 2 2 2 2 2 2 2 ...
 $ owner        : Factor w/ 5 levels "First Owner",..: 1 1 1 1 3 1 1 3 1 1 ...
```

Hide

```
# Drop the name variable
cars.df<-cars.df[-1]

# Check the dataframe
str(cars.df)
```

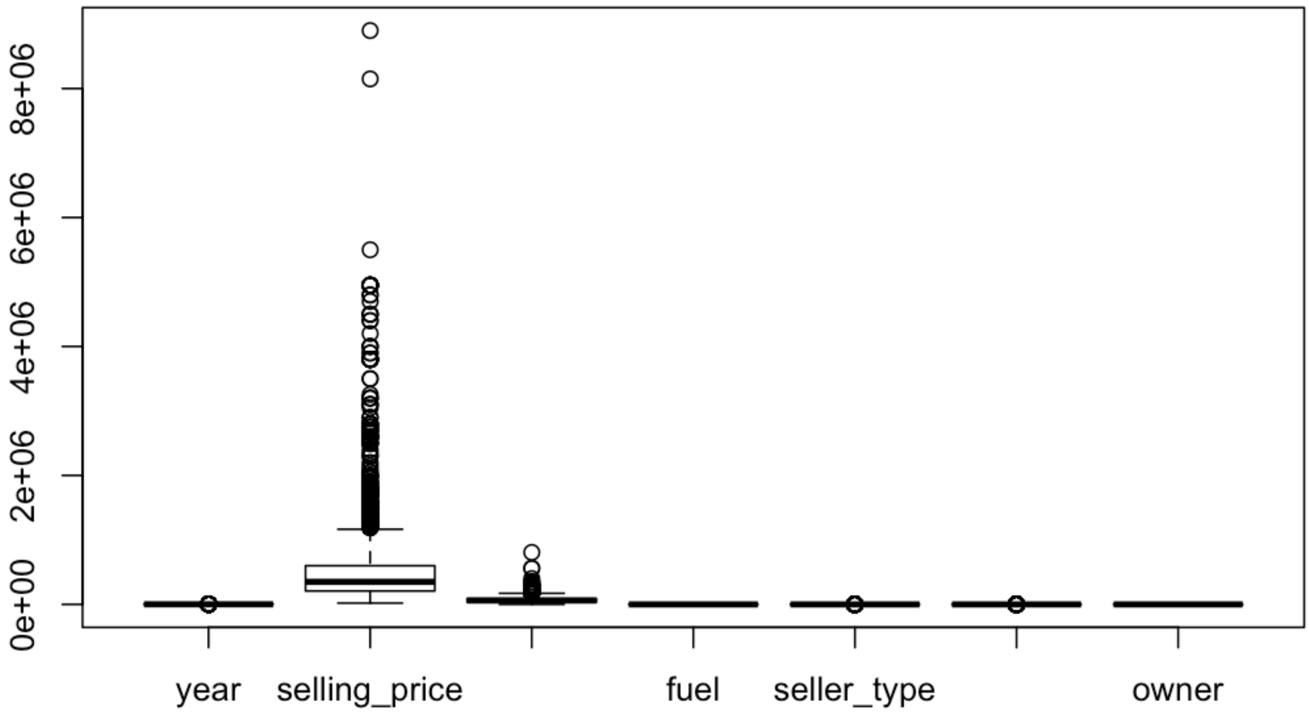
```
'data.frame':	4340 obs. of  7 variables:
 $ year         : int  2007 2007 2012 2017 2014 2007 2016 2014 2015 2017 ...
 $ selling_price: int  60000 135000 600000 250000 450000 140000 550000 240000 850000
365000 ...
 $ km_driven    : int  70000 50000 100000 46000 141000 125000 25000 60000 25000 78000
...
 $ fuel         : Factor w/ 5 levels "CNG","Diesel",..: 5 5 2 5 2 5 5 5 5 1 ...
 $ seller_type  : Factor w/ 3 levels "Dealer","Individual",..: 2 2 2 2 2 2 2 2 2 2 ..
.
 $ transmission : Factor w/ 2 levels "Automatic","Manual": 2 2 2 2 2 2 2 2 2 2 ...
 $ owner        : Factor w/ 5 levels "First Owner",..: 1 1 1 1 3 1 1 3 1 1 ...
```

2. Are there outliers in any one of the features in the data set? How do you identify outliers? Remove them but create a second data set with outliers removed called cars.no.df. Keep the original data set cars.df.

To detect outliers, we can apply visualization or by Z-Score calculation. Since Z-calculation needs a specific

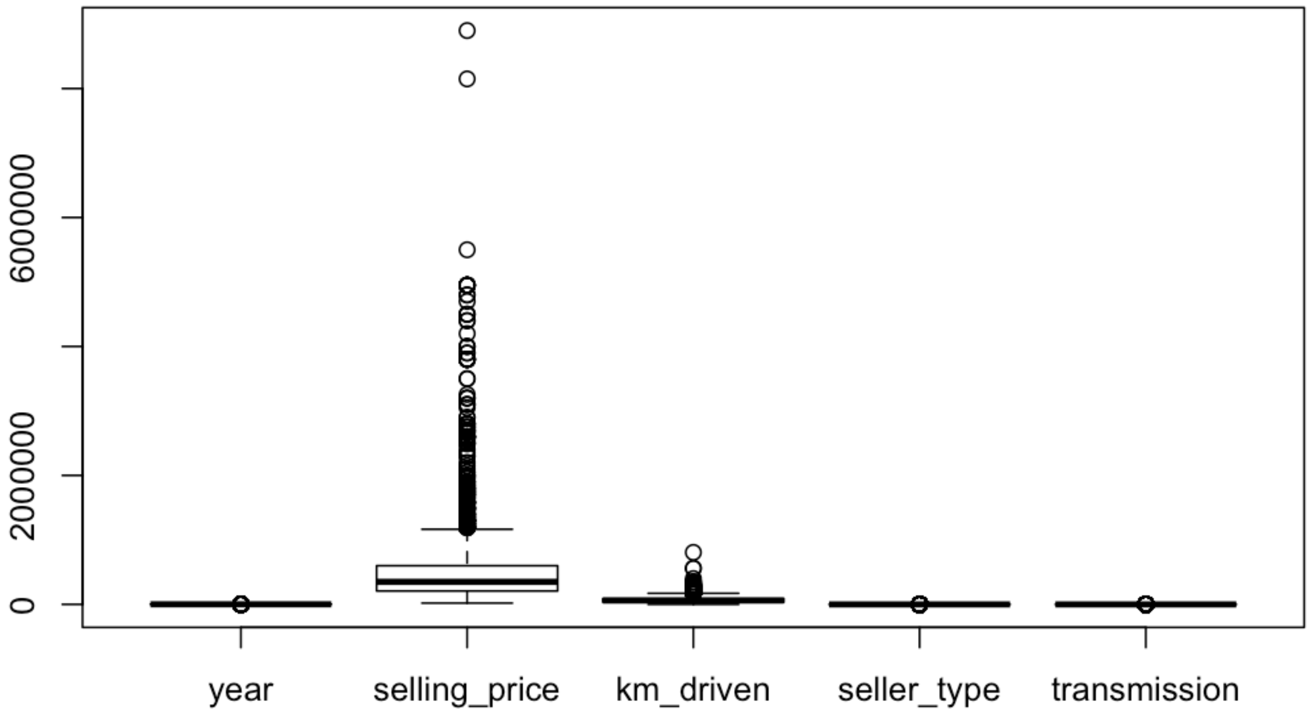
Hide

```
# Visualize outliers from the original dataset
boxplot(cars.df)
```



Hide

```
options(scipen = 999)
outliers<-cars.df[,c(-4, -7)]
boxplot(outliers, scale_y_continuous)
```



Based on the boxplot shown above, there are dots/points outside the box range representing outliers in year, selling_price, km_driven, seller_type, and transmission. Since seller_type and transmission are factors, we ignore them. To handle these outliers, we need to check the summary of each of the variables mentioned above to obtain information on the IQR.

Hide

```
# Summary of Year
summary(cars.df$year)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
   1992    2011    2014    2013    2016    2020 
```

Hide

```
# Calculate IQR for Year
IQR_Year<-2016-2011
Est_IQRYear<- 2011 + 1.5 * IQR_Year
Est_IQRYear
```

```
[1] 2018.5
```

Hide

```
# Summary of selling_price
summary(cars.df$selling_price)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  20000  208750  350000  504127  600000 8900000 
```

Hide

```
# Calculate IQR for selling_price
IQR_sellingPrice<-600000-208750
Est_IQRSPR<- 600000 + 1.5 * IQR_sellingPrice
Est_IQRSPR
```

```
[1] 1186875
```

Hide

```
# Summary of km_driven
summary(cars.df$km_driven)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
      1   35000   60000   66216   90000  806599 
```

Hide

```
# Calculate IQR for km_drive
IQR_kmdrive<-90000 - 35000
Est_IQRkmDrive<- 90000 + 1.5 * IQR_kmdrive
Est_IQRkmDrive
```

```
[1] 172500
```

Hide

```
summary(cars.df)
```

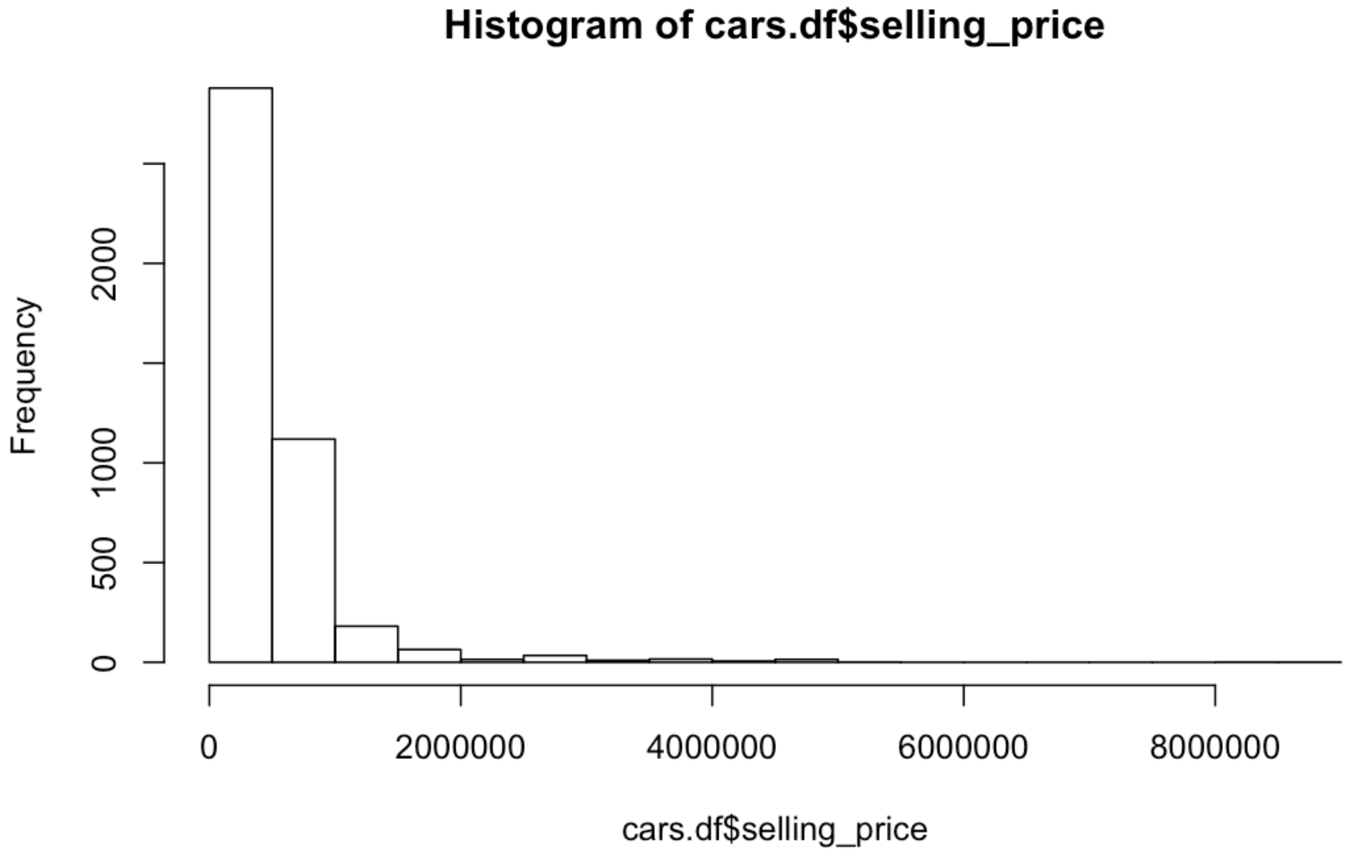
```
      year      selling_price       km_driven           fuel               seller_t
ype
 Min.   :1992   Min.   :  20000   Min.   :      1   CNG     :  40   Dealer          :
994
 1st Qu.:2011   1st Qu.: 208750   1st Qu.: 35000   Diesel  :2153   Individual      :3
244
 Median :2014   Median : 350000   Median : 60000   Electric:   1   Trustmark Dealer:
102
 Mean   :2013   Mean   : 504127   Mean   : 66216   LPG     :  23
 3rd Qu.:2016   3rd Qu.: 600000   3rd Qu.: 90000   Petrol  :2123
 Max.   :2020   Max.   :8900000   Max.   :806599
    transmission                   owner
 Automatic: 448   First Owner         :2832
 Manual   :3892   Fourth & Above Owner:  81
                  Second Owner        :1106
                  Test Drive Car      :  17
                  Third Owner         : 304
```

Hide

```
cars.no.df<-subset(cars.df, year<=2018.5 & selling_price<=1186875 & km_driven<=172500
)
```

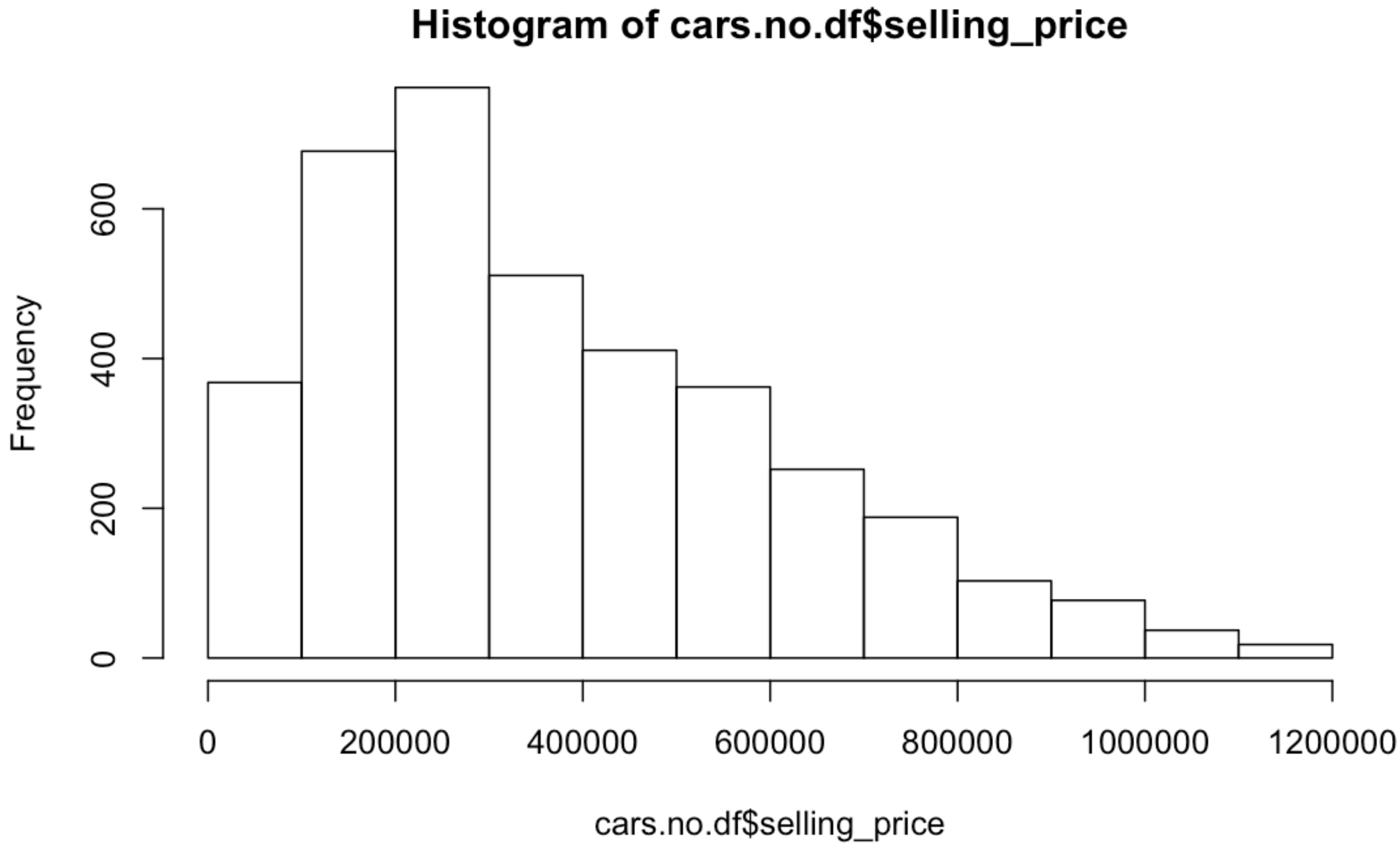
Hide

```
# Compare original data with outliers vs. without outliers through histogram
hist(cars.df$selling_price)
```



Hide

```
hist(cars.no.df$selling_price)
```



As we can see from the 2 histograms shown above, histogram with original data is considered to be have outliers due to its extremely low frequency in the selling_price. While the cars.no.df selling_price shows no outliers but is still positively skewed.

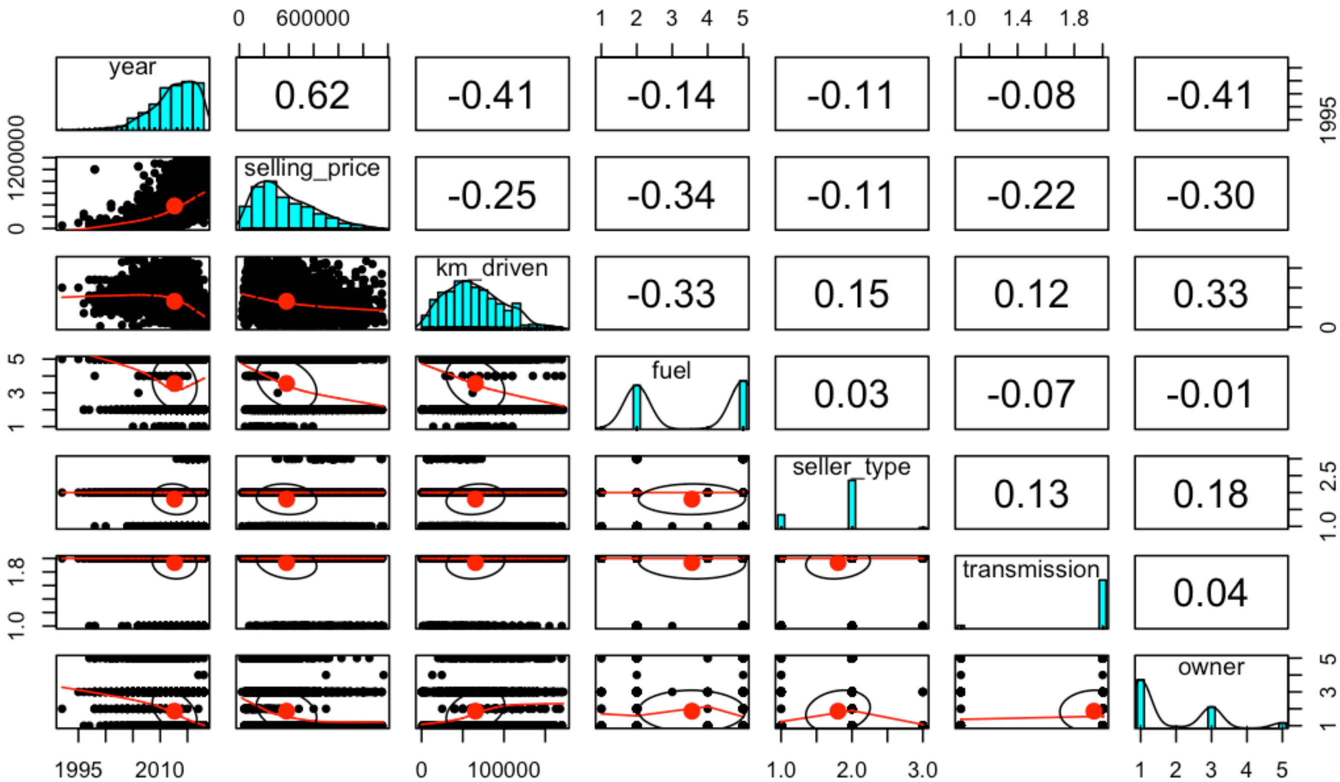
3. Using pairs.panel, what are the distributions of each of the features in the data set with outliers removed (cars.no.df)? Are they reasonably normal so you can apply a statistical learner such as regression? Can you normalize features through a log, inverse, or square-root transform? State which features should be transformed and then transform as needed and build a new data set, cars.tx.

Hide

```
# Load psych to apply pairs.panel
library(psych)
```

Hide

```
pairs.panels(cars.no.df)
```



Based on the pairs.panel visualization shown above, we can see that both year and selling_price columns seem to be skewed(year is negatively skewed while selling_price is positively skewed). However, we can also notice that km_driven column shows a lightly bell-curved pattern. Below we check further:

Hide

```
# Apply shapiro test
shapiro.test(cars.no.df$km_driven)
```

```
	Shapiro-Wilk normality test

data:  cars.no.df$km_driven
W = 0.97522, p-value < 0.00000000000000022
```

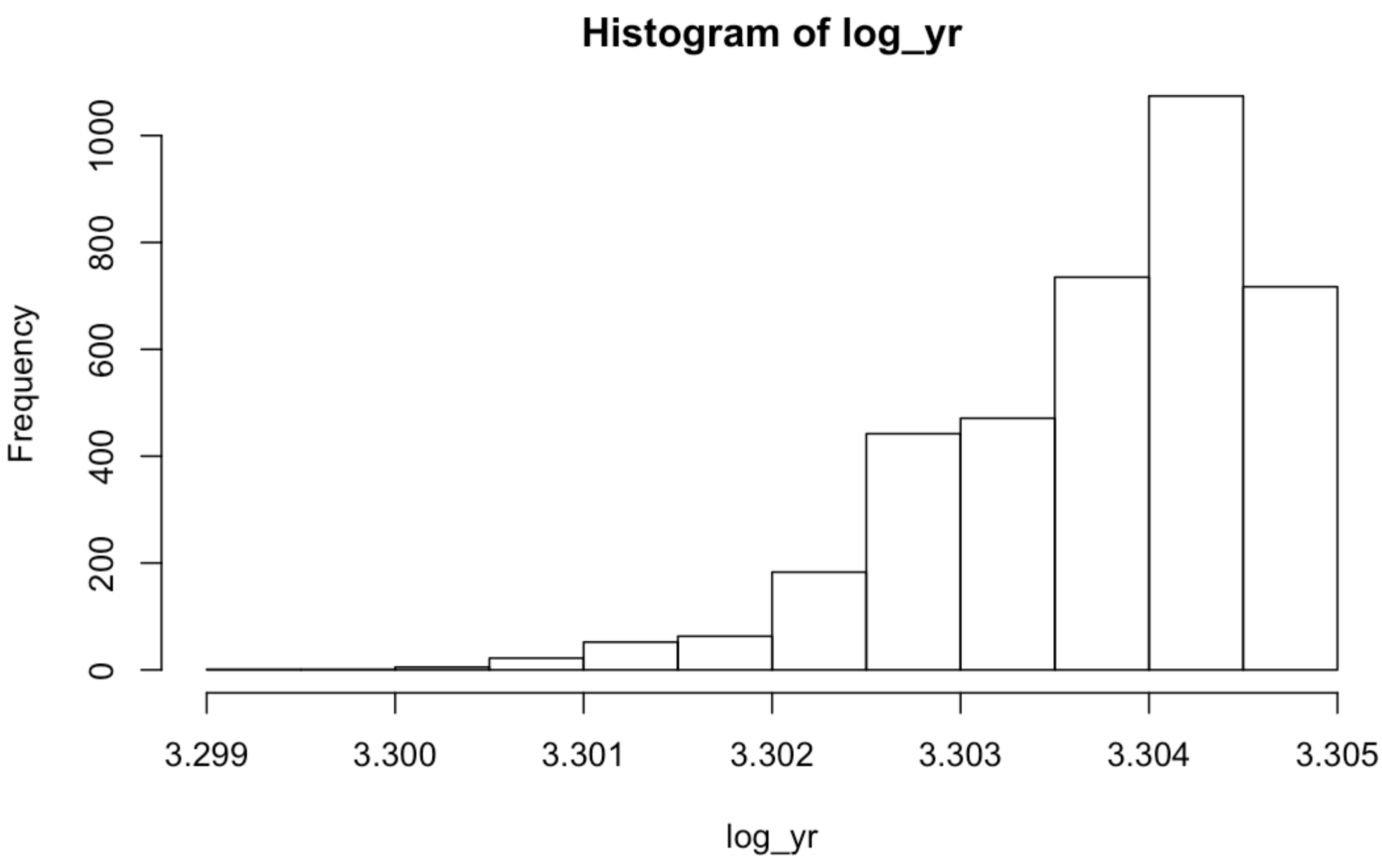
Based on the Shapiro normality test, p-value is less than 0.05, then the null hypothesis stating data are normally distributed is rejected. Thus, km_driven is not considered to be completely normal.

We can also analyze how the relationship between year and selling_price has a strong correlation of 0.62, followed by low but positive correlation betwen km_driven and owner with a score of 0.33. seller_type, transmission, and owner seem to have a very weak but positive correlation as well, leaving the rest of the scores mostly negative.

POSITIVE CORRELATION: year and selling_price = 0.62 seller_type and owner = 0.18 seller_type and km_driven = 0.15 seller_type and transmission = 0.13

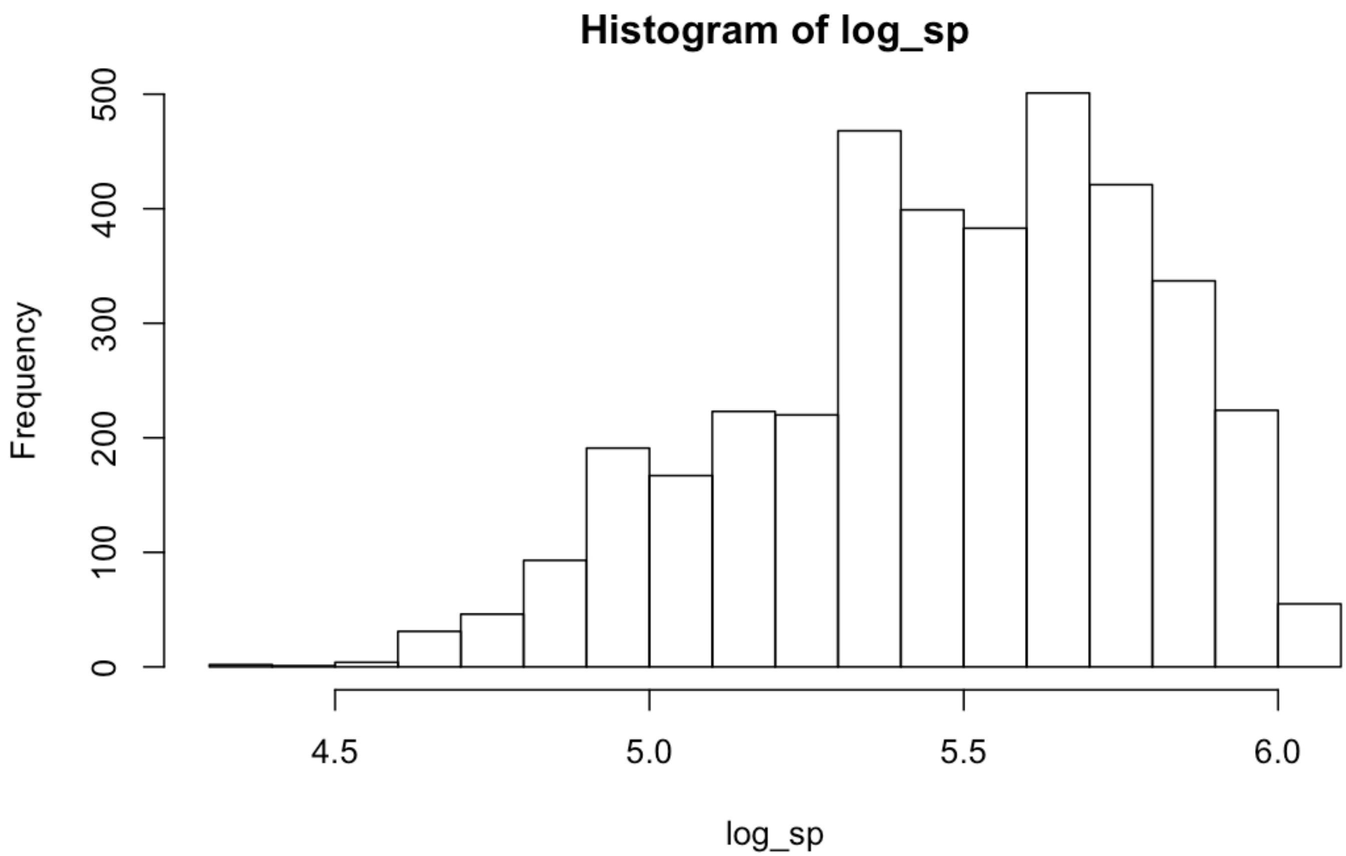
Hide

```
# Transform Data into log
log_yr<-log10(cars.no.df$year)
log_sp<-log10(cars.no.df$selling_price)
log_kmD<-log10(cars.no.df$km_driven)

# Show histogram
hist(log_yr)
```



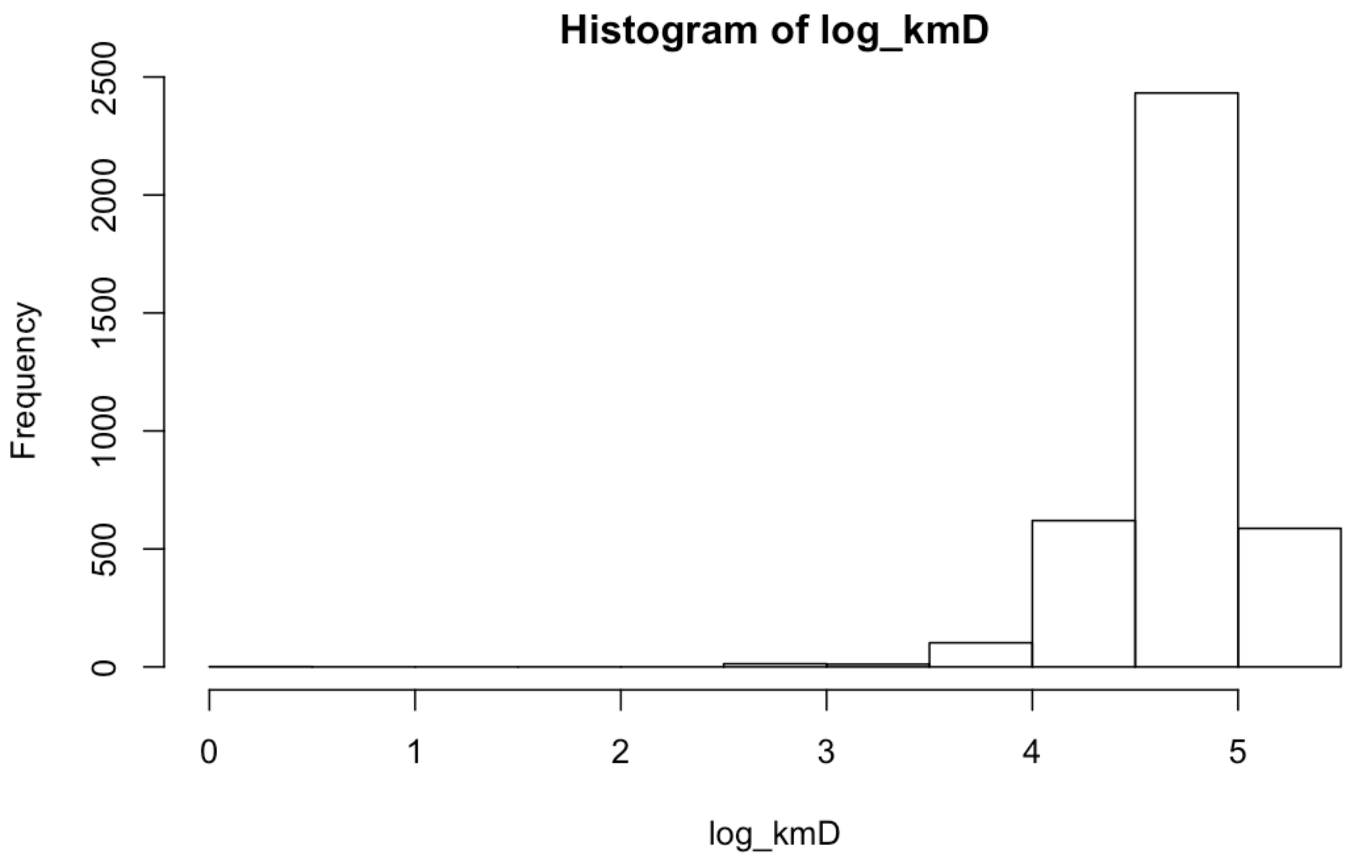
Hide

```
hist(log_sp)
```



Hide

```
hist(log_kmD)
```



Hide

```
# Shapiro Test for log
shapiro.test(log_yr)
```

```
	Shapiro-Wilk normality test

data:  log_yr
W = 0.93126, p-value < 0.00000000000000022
```

Hide

```
shapiro.test(log_sp)
```

```
	Shapiro-Wilk normality test

data:  log_sp
W = 0.97495, p-value < 0.00000000000000022
```

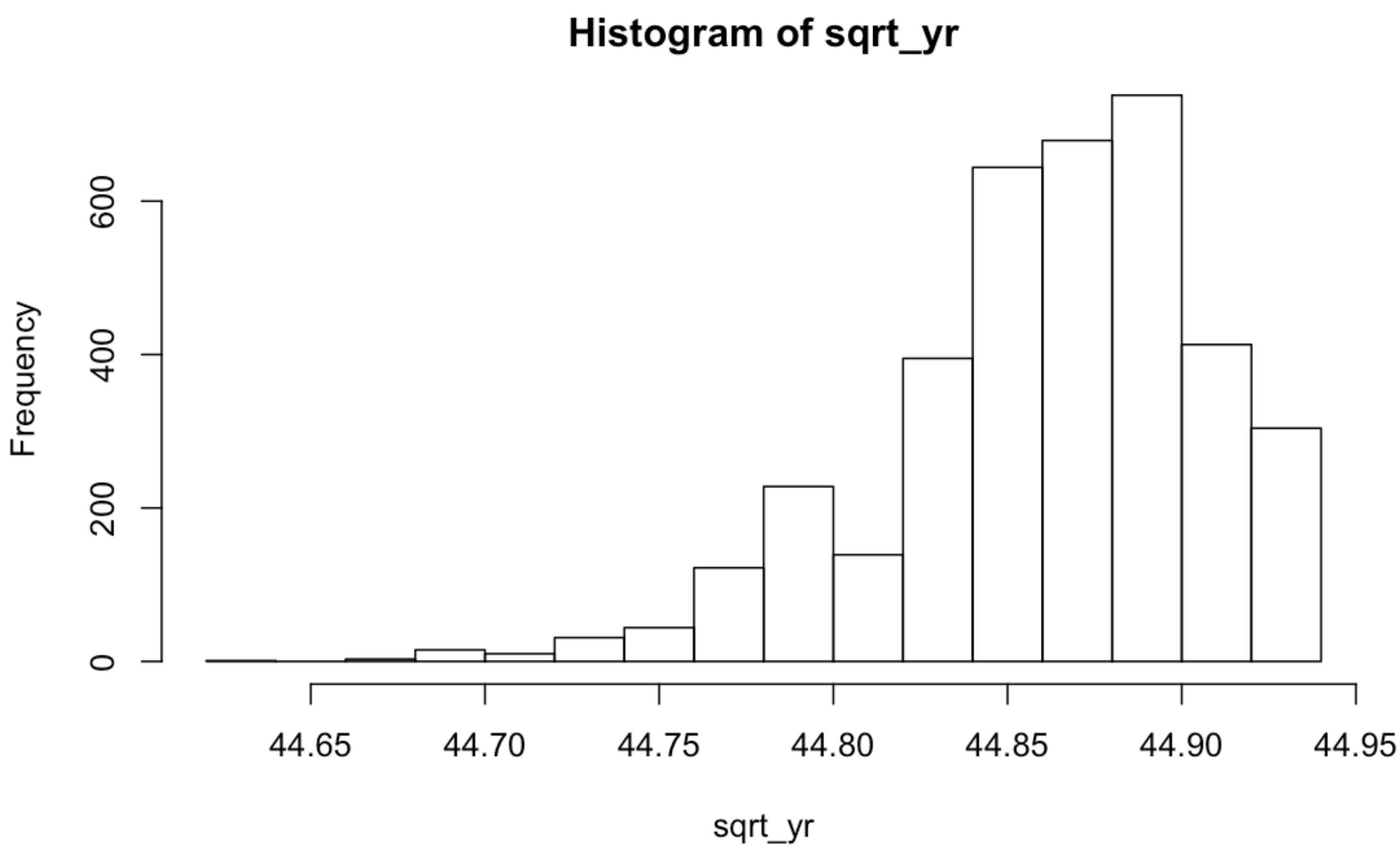
Hide

```
shapiro.test(log_kmD)
```

```
	Shapiro-Wilk normality test

data:  log_kmD
W = 0.86795, p-value < 0.00000000000000022
```

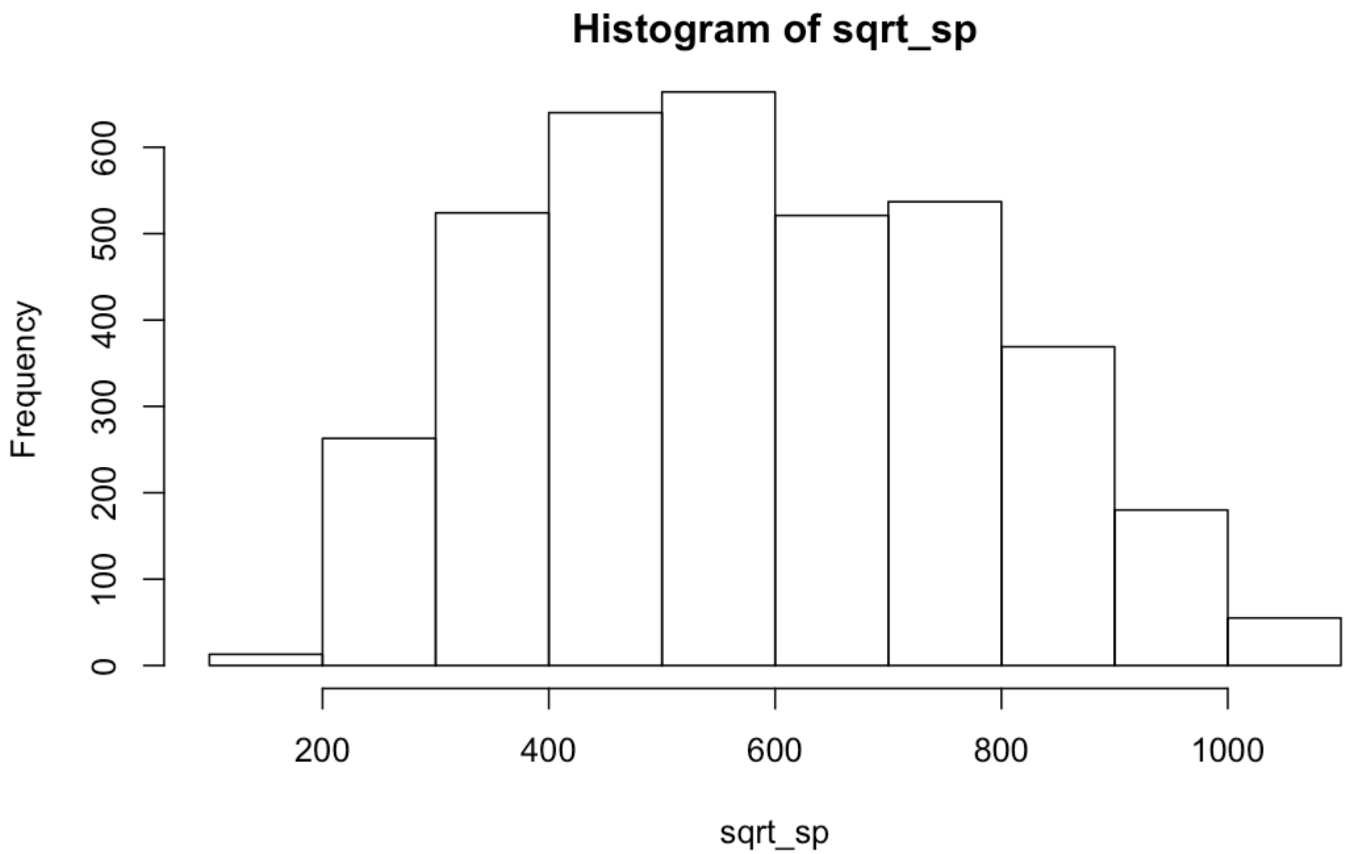
Hide

```
# Transform Data into Square Root
sqrt_yr<-sqrt(cars.no.df$year)
sqrt_sp<-sqrt(cars.no.df$selling_price)
sqrt_kmD<-sqrt(cars.no.df$km_driven)

# Show histogram
hist(sqrt_yr)
```



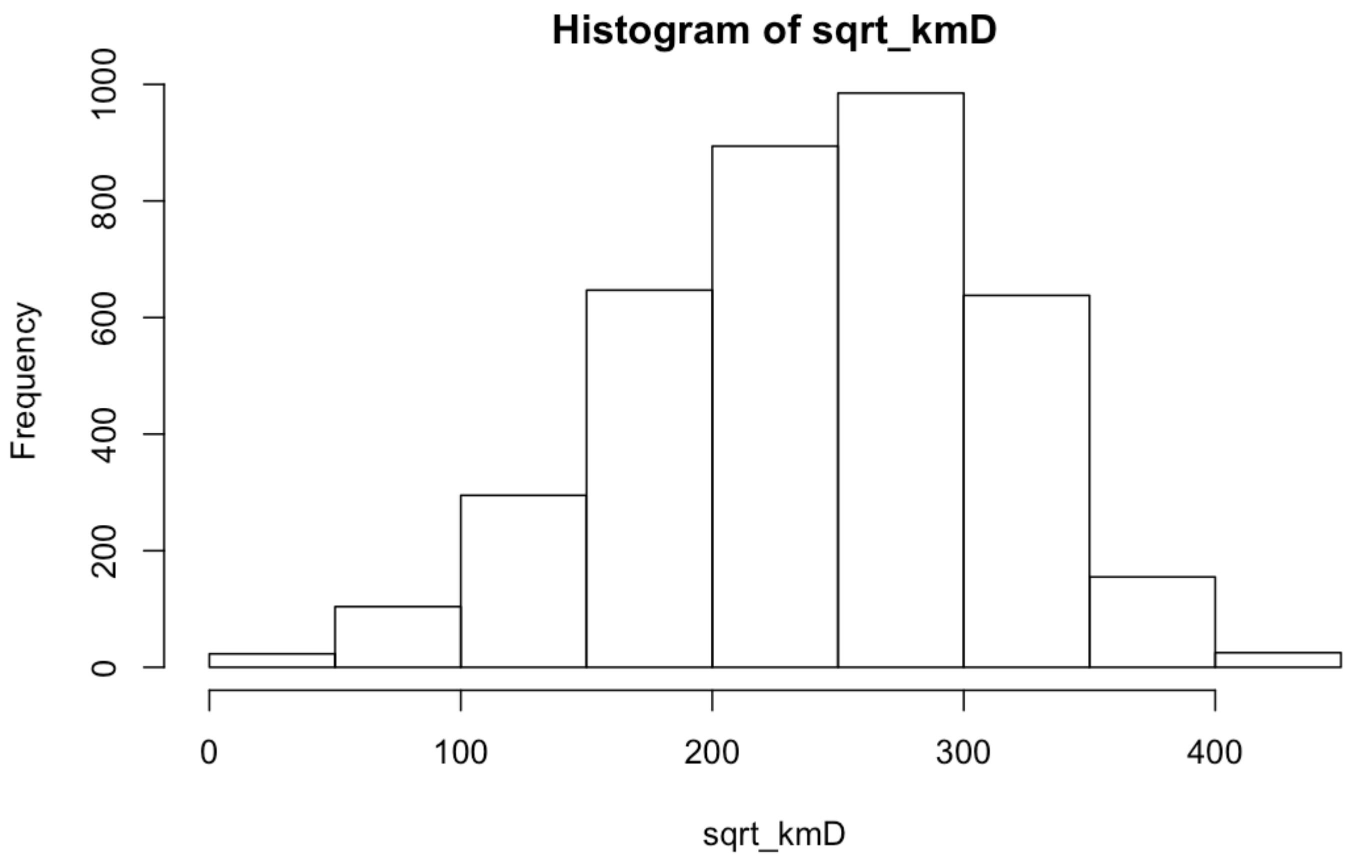
Hide

```
hist(sqrt_sp)
```



Hide

```
hist(sqrt_kmD)
```



Hide

```
# Shapiro Test for Square Root
shapiro.test(sqrt_yr)
```

```
	Shapiro-Wilk normality test

data:  sqrt_yr
W = 0.93152, p-value < 0.00000000000000022
```

Hide

```
shapiro.test(sqrt_sp)
```

```
	Shapiro-Wilk normality test

data:  sqrt_sp
W = 0.98357, p-value < 0.00000000000000022
```

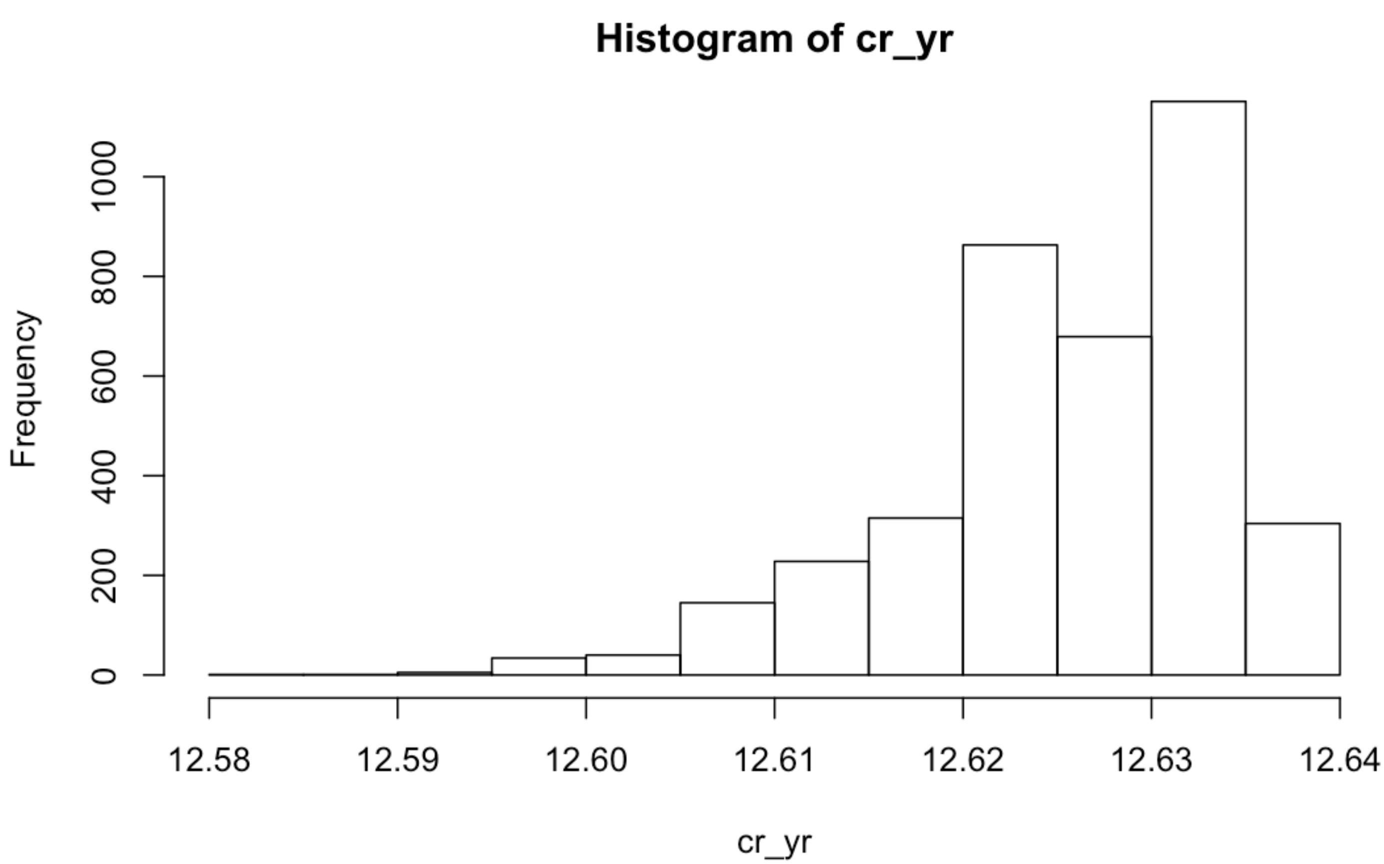
Hide

```
shapiro.test(sqrt_kmD)
```

```

	Shapiro-Wilk normality test

data:  sqrt_kmD
W = 0.99056, p-value = 0.000000000000003902
```

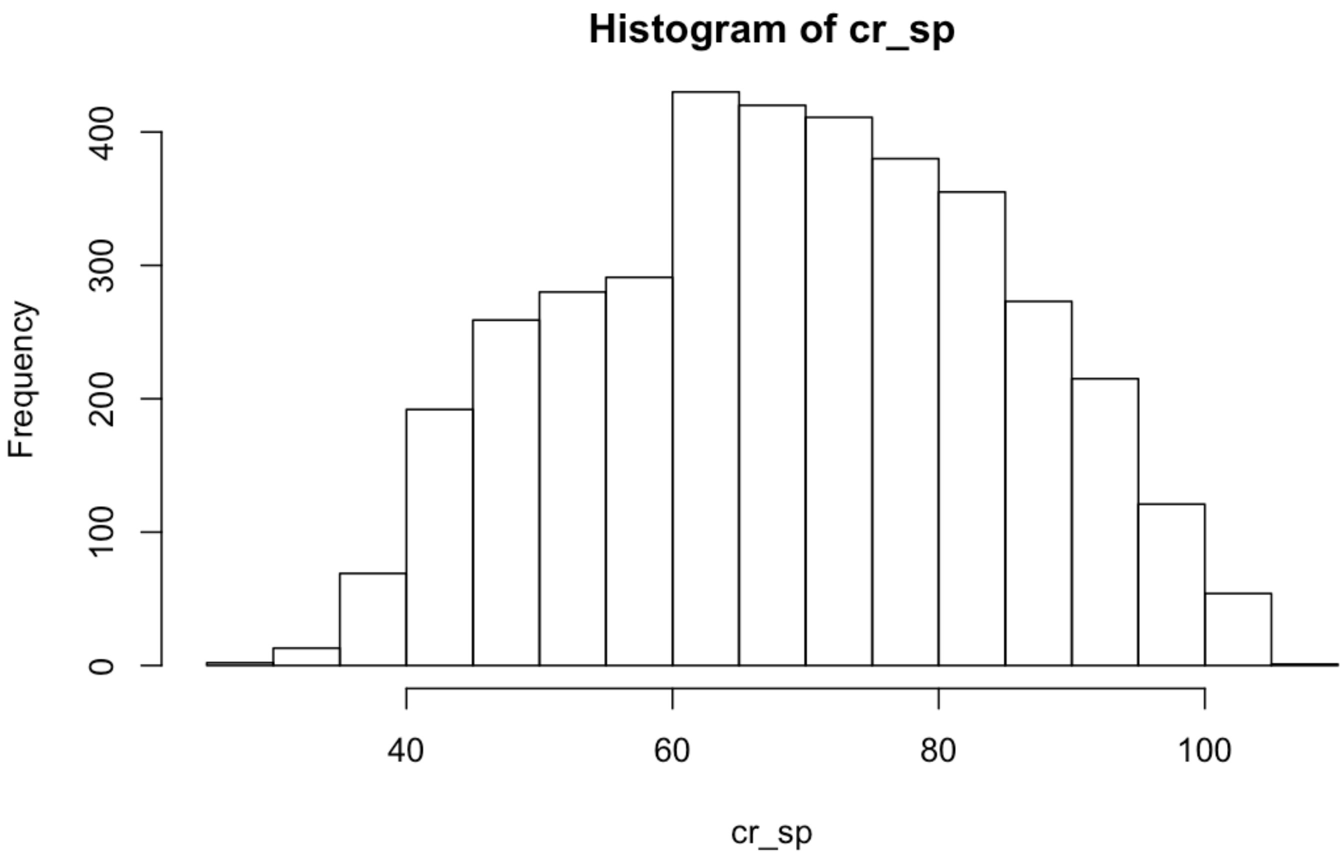
Hide

```
# Transform Data into Cube Root
cr_yr<-cars.no.df$year^(1/3)
cr_sp<-cars.no.df$selling_price^(1/3)
cr_kmD<-cars.no.df$km_driven^(1/3)

# Show histogram
hist(cr_yr)
```



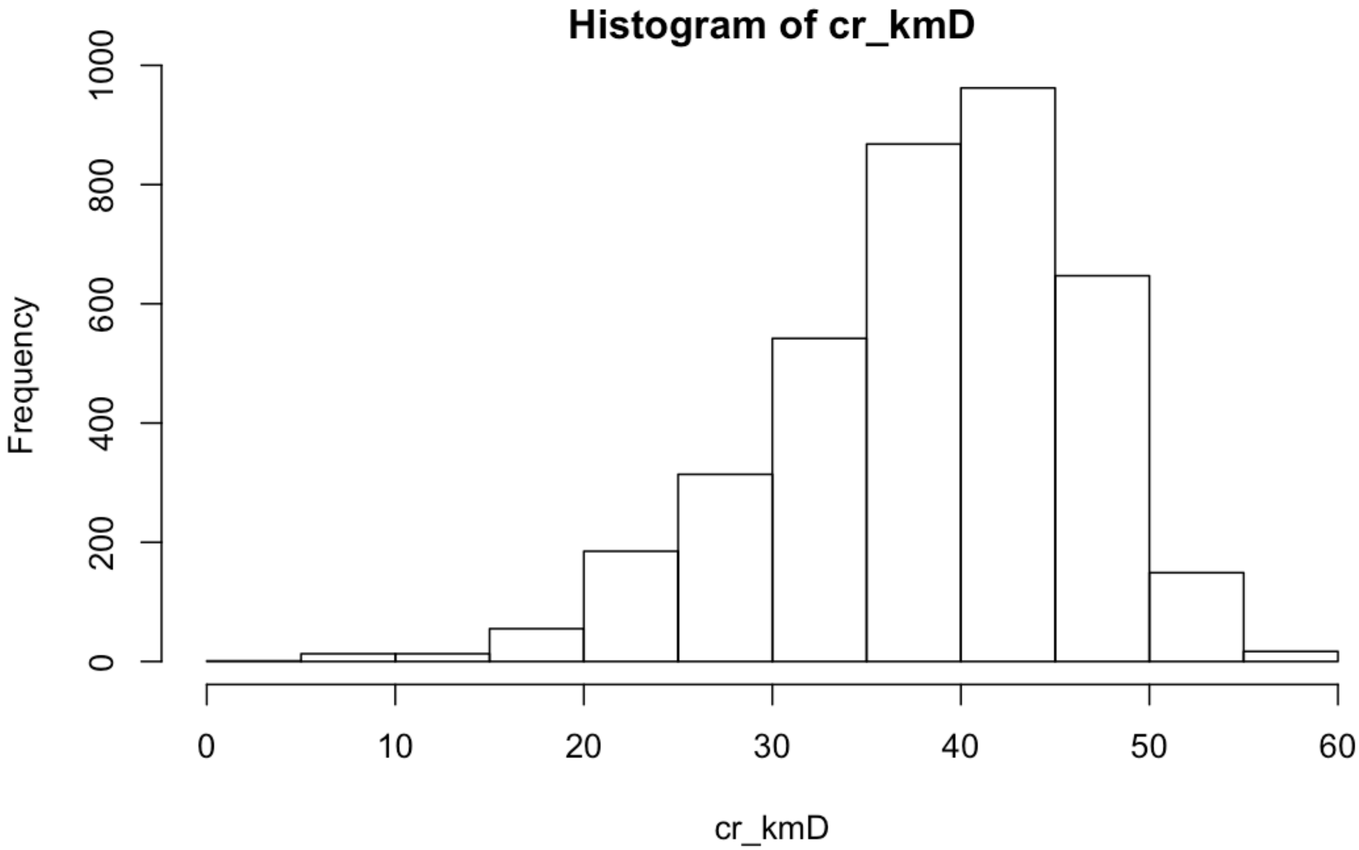
Hide

```
hist(cr_sp)
```



Hide

```
hist(cr_kmD)
```



Hide

```
# Shapiro Test for Cube Root
shapiro.test(cr_yr)
```

```

	Shapiro-Wilk normality test

data:  cr_yr
W = 0.93143, p-value < 0.00000000000000022
```

Hide

```
shapiro.test(cr_sp)
```

```

	Shapiro-Wilk normality test

data:  cr_sp
W = 0.98844, p-value < 0.00000000000000022
```

Hide

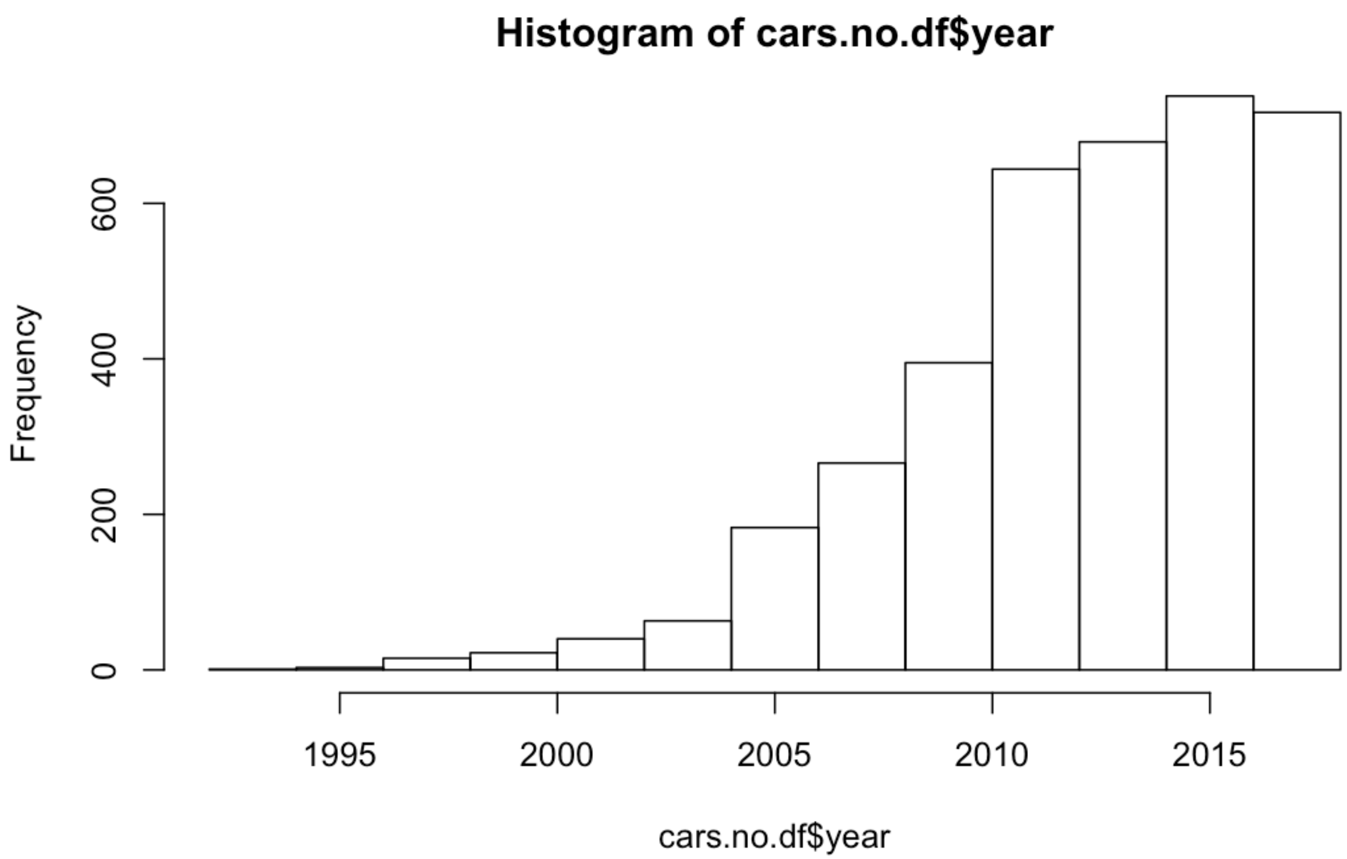
Hide

```
shapiro.test(cr_kmD)
```

```
	Shapiro-Wilk normality test

data:  cr_kmD
W = 0.97633, p-value < 0.00000000000000022
```

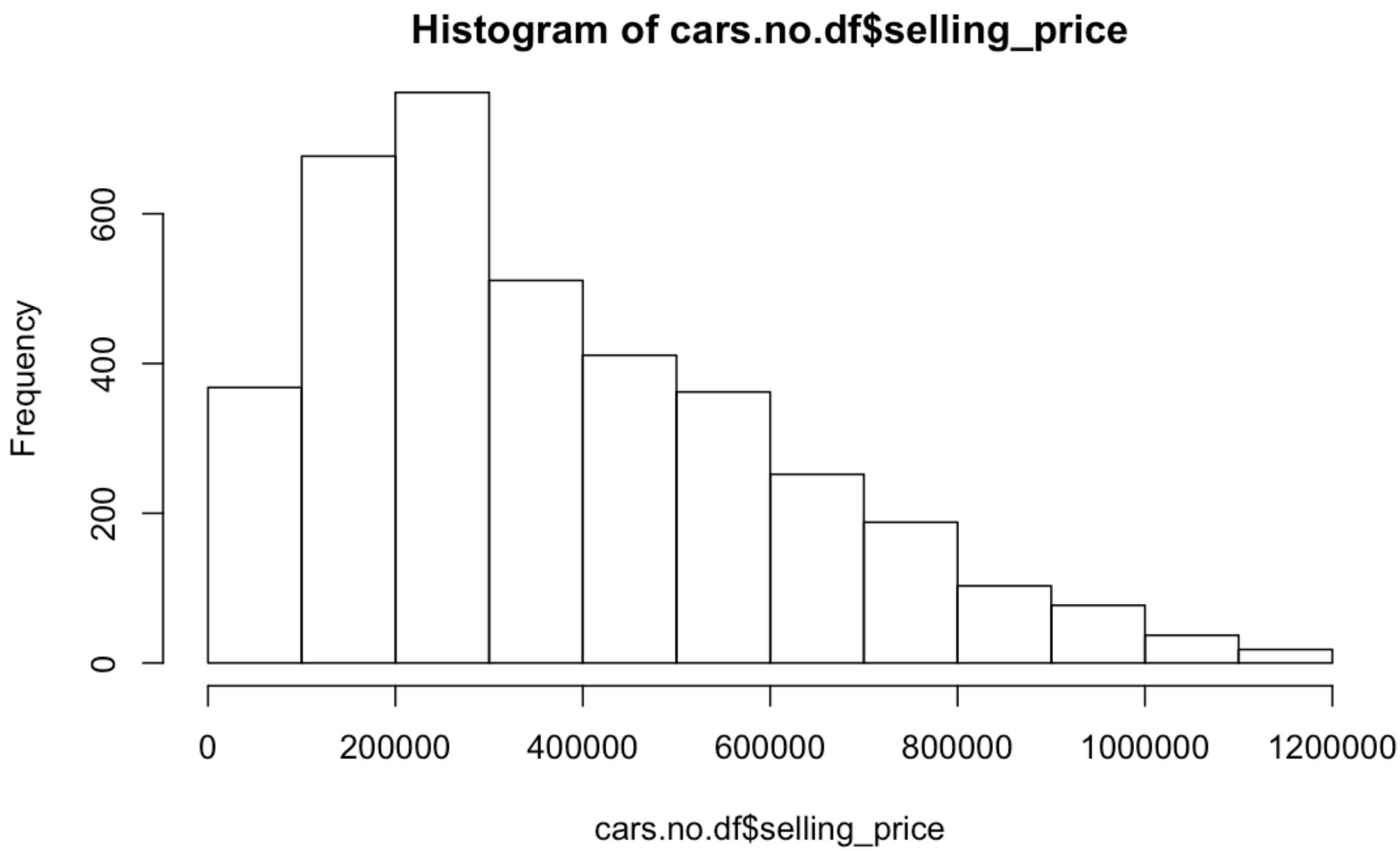
Hide

```
# Histogram distribution without the 3 transformations
hist(cars.no.df$year)
```



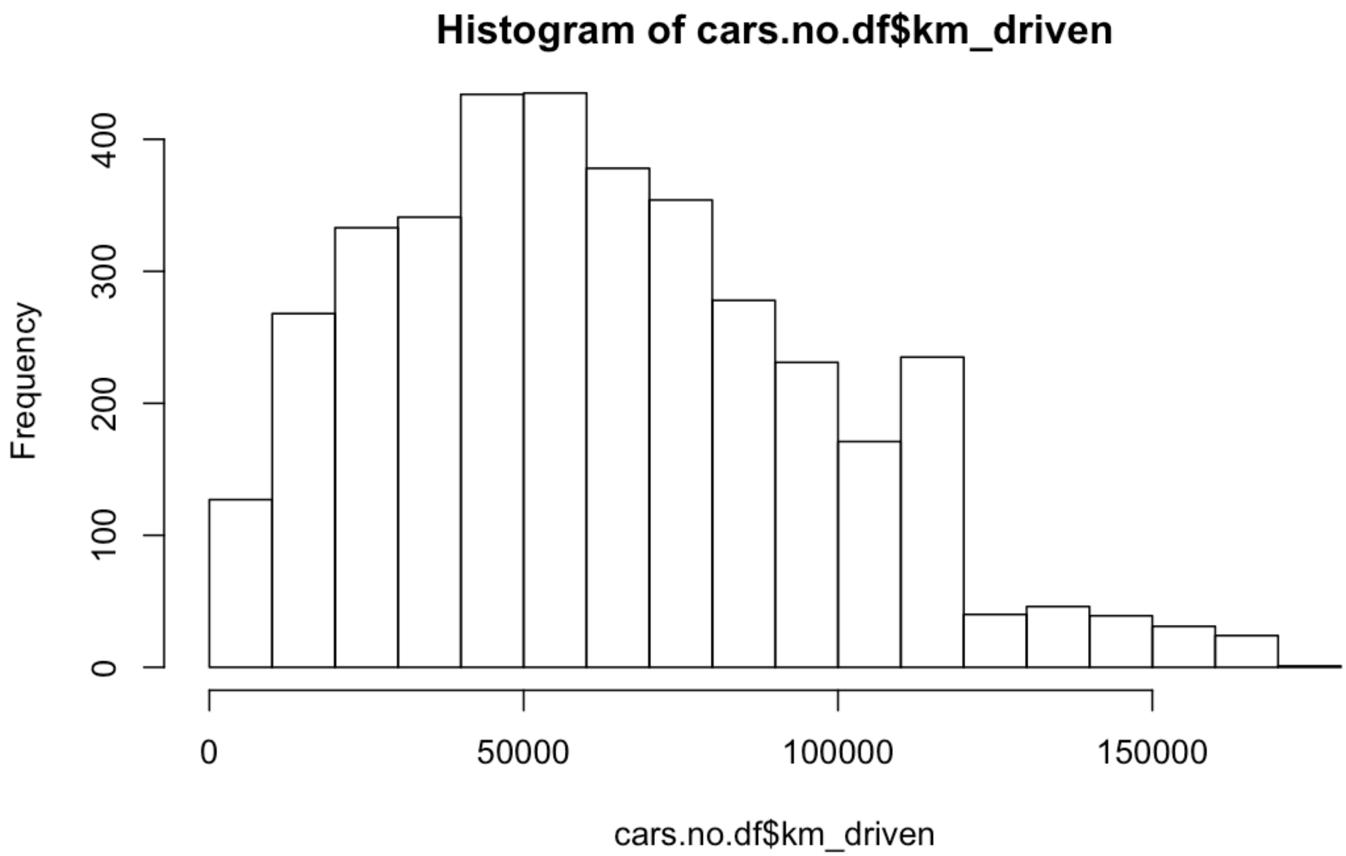
Hide

```
hist(cars.no.df$selling_price)
```



Hide

```
hist(cars.no.df$km_driven)
```



Hide

```
# Shapiro Test without transformations
shapiro.test(cars.no.df$year)
```

```

	Shapiro-Wilk normality test

data:  cars.no.df$year
W = 0.93177, p-value < 0.00000000000000022
```

Hide

```
shapiro.test(cars.no.df$selling_price)
```

```

	Shapiro-Wilk normality test

data:  cars.no.df$selling_price
W = 0.93457, p-value < 0.00000000000000022
```

Hide

Hide

```
shapiro.test(cars.no.df$km_driven)
```

```


	Shapiro-Wilk normality test

data:  cars.no.df$km_driven
W = 0.97522, p-value < 0.00000000000000022
```

Based from the 3 transformations, Square Root transformation on km_driven column seems to be more distributed compared to the rest of the transformations with other columns with a p-value of 0.000000000000003902.

Hide

```
# Load dplyr
library(dplyr)
```

Hide

```
# Apply Square Root transformation from km_drive to a new dataset: cars.tx
cars.tx<-cars.no.df %>%
  select(-km_driven)

# Add the transformed km_driven
cars.tx$km_driven<-sqrt_kmD

# Check cars.tx
summary(cars.tx)
```

```
      year      selling_price           fuel                 seller_type      transmis
sion
 Min.   :1992   Min.   :  20000   CNG     :  34   Dealer          : 832   Automatic:
232
 1st Qu.:2010   1st Qu.: 190000   Diesel  :1761   Individual      :2845   Manual   :3
534
 Median :2013   Median : 320000   Electric:   1   Trustmark Dealer:  89
 Mean   :2013   Mean   : 381072   LPG     :  23
 3rd Qu.:2016   3rd Qu.: 535000   Petrol  :1947
 Max.   :2018   Max.   :1165000
                  owner          km_driven
 First Owner         :2375   Min.   :  1.0
 Fourth & Above Owner:  75   1st Qu.:200.0
 Second Owner        :1047   Median :244.9
 Test Drive Car      :   2   Mean   :244.7
 Third Owner         : 267   3rd Qu.:300.0
                             Max.   :414.7
```

Hide

```
# Compare with previous dataset
summary(cars.no.df)
```

```
      year      selling_price       km_driven            fuel                 seller_t
ype
 Min.   :1992   Min.   :  20000   Min.   :     1   CNG     :  34   Dealer          :
832
 1st Qu.:2010   1st Qu.: 190000   1st Qu.: 40000   Diesel  :1761   Individual      :2
845
 Median :2013   Median : 320000   Median : 60000   Electric:   1   Trustmark Dealer:
89
 Mean   :2013   Mean   : 381072   Mean   : 65296   LPG     :  23
 3rd Qu.:2016   3rd Qu.: 535000   3rd Qu.: 90000   Petrol  :1947
 Max.   :2018   Max.   :1165000   Max.   :172000
   transmission                    owner
 Automatic: 232   First Owner         :2375
 Manual   :3534   Fourth & Above Owner:  75
                  Second Owner        :1047
                  Test Drive Car      :   2
                  Third Owner         : 267
```

4. What are the correlations to the response variable (km_driven) for cars.no.df? Are there collinearities? Build a full correlation matrix.

Hide

```
# Build correlation matrix for cars.no.df
corMat<-cars.no.df[,c(1, 2, 3)] # Separate the categorical variables with the numeric
al ones to be calculated
corCarsNoDf<-cor(corMat)
round(corCarsNoDf, 2)
```

```
               year selling_price km_driven
year           1.00          0.62     -0.41
selling_price  0.62          1.00     -0.25
km_driven     -0.41         -0.25      1.00
```

From the correlation matrix shown above, we obtained the data solely from cars.no.df and analyze the relationship between the variables. Both year and selling price depict inverse relationship(negative relationship) with km_driven. This could mean that when the car has higher km driven, the price will decrease.

However, since both correlation scores are considered to be weak, -0.41 and -0.25. There is no collinearities.

5. Split each of the three data sets, cars.no.df, cars.df, and cars.tx 75%/25% so you retain 25% for testing using random sampling without replacement. Call the data sets, cars.training and cars.testing, cars.no.training and cars.no.testing, and cars.tx.training and cars.tx.testing.

Hide

```
# Set seed
set.seed(123)

# Create training and testing for cars.no.df
indexCarsNo<-sort(sample(nrow(cars.no.df), nrow(cars.no.df)*.75))

cars.no.training<-cars.no.df[indexCarsNo,]
cars.no.testing<-cars.no.df[-indexCarsNo,]

# Create training and testing for cars.df
indexCarsDF<-sort(sample(nrow(cars.df), nrow(cars.df)*.75))

cars.training<-cars.df[indexCarsDF,]
cars.testing<-cars.df[-indexCarsDF,]

# Create training and testing for cars.tx
indexCarsTx<-sort(sample(nrow(cars.tx), nrow(cars.tx)*.75))

cars.tx.training<-cars.tx[indexCarsTx,]
cars.tx.testing<-cars.tx[-indexCarsTx,]
```

6. Build three ideal multiple regression models for cars.training, cars.no.training, and cars.tx.training using backward elimination based on p-value for predicting km_driven.

Hide

```
# Build Multiple Regression Model for cars.training
carsTrainingModel<-lm(km_driven ~., data = cars.training)
step(carsTrainingModel, direction = "backward")
```

```
Start:  AIC=68780.15
km_driven ~ year + selling_price + fuel + seller_type + transmission + 
    owner

                Df    Sum of Sq           RSS   AIC
- transmission   1    269246526 4850070600453 68778
<none>                          4849801353928 68780
- seller_type    2  26819967953 4876621321881 68794
- selling_price  1  42327999109 4892129353037 68806
- owner          4 103190608818 4952991962746 68841
- year           1 484926448992 5334727802920 69088
- fuel           4 784450210697 5634251564625 69260

Step:  AIC=68778.33
km_driven ~ year + selling_price + fuel + seller_type + owner

                Df    Sum of Sq           RSS   AIC
<none>                          4850070600453 68778
- seller_type    2  27631715909 4877702316362 68793
- selling_price  1  62696536752 4912767137205 68818
- owner          4 102922114481 4952992714934 68839
- year           1 486321063497 5336391663950 69087
- fuel           4 801336891942 5651407492395 69268

Call:
lm(formula = km_driven ~ year + selling_price + fuel + seller_type + 
    owner, data = cars.training)

Coefficients:
                (Intercept)                        year               selling_price  
             7189945.460285                -3544.389496                   -0.008705  
                 fuelDiesel                fuelElectric                     fuelLPG  
               22909.192508               -23385.889321                12657.880823  
                 fuelPetrol       seller_typeIndividual  seller_typeTrustmark Dealer  
              -10156.422161                 6778.394525                -3112.838873  
  ownerFourth & Above Owner            ownerSecond Owner          ownerTest Drive Car  
               16620.045919                 8184.308628               -19288.598076  
           ownerThird Owner  
               21284.777770  
```

Hide

Hide

```
# After backward elimination: cars.training
CTModel<-lm(formula = km_driven ~ year + selling_price + fuel + seller_type +
    owner, data = cars.training)
summary(CTModel)
```

```
Call:
lm(formula = km_driven ~ year + selling_price + fuel + seller_type + 
    owner, data = cars.training)

Residuals:
    Min      1Q  Median      3Q     Max 
-139083  -20580   -5682   15323  749665 

Coefficients:
                                  Estimate     Std. Error t value            Pr(>|t|
)
(Intercept)                 7189945.460285  396065.988268  18.153 < 0.000000000000000
2 ***
year                          -3544.389496     196.583664 -18.030 < 0.000000000000000
2 ***
selling_price                    -0.008705       0.001345  -6.474   0.000000000110054
8 ***
fuelDiesel                    22909.192508    7160.309801   3.199              0.0013
9 **
fuelElectric                 -23385.889321   39377.257488  -0.594              0.5526
2
fuelLPG                       12657.880823   11554.193914   1.096              0.2733
7
fuelPetrol                   -10156.422161    7151.423858  -1.420              0.1556
5
seller_typeIndividual          6778.394525    1718.787651   3.944   0.000081929188972
9 ***
seller_typeTrustmark Dealer   -3112.838873    4574.048500  -0.681              0.4962
1
ownerFourth & Above Owner     16620.045919    5194.680654   3.199              0.0013
9 **
ownerSecond Owner              8184.308628    1742.338587   4.697   0.000002745595372
1 ***
ownerTest Drive Car          -19288.598076   10492.523625  -1.838              0.0661
1 .
ownerThird Owner              21284.777770    2816.968842   7.556   0.000000000000053
8 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 38680 on 3242 degrees of freedom
Multiple R-squared:  0.3112,    Adjusted R-squared:  0.3087 
F-statistic: 122.1 on 12 and 3242 DF,  p-value: < 0.00000000000000022
```

Hide

```
# Build Multiple Regression Model for cars.no.training
carsNoTrainingModel<-lm(km_driven ~., data = cars.no.training)
step(carsNoTrainingModel, direction = "backward")
```

```
Start:  AIC=57866.02
km_driven ~ year + selling_price + fuel + seller_type + transmission +
    owner

                Df    Sum of Sq           RSS   AIC
- transmission   1    944209306 2217100395843 57865
<none>                          2216156186537 57866
- selling_price  1  26229466498 2242385653034 57897
- seller_type    2  36533532597 2252689719133 57908
- owner          4  55442492861 2271598679398 57928
- year           1 168179915982 2384336102519 58071
- fuel           4 492093078412 2708249264949 58424

Step:  AIC=57865.22
km_driven ~ year + selling_price + fuel + seller_type + owner

                Df    Sum of Sq           RSS   AIC
<none>                          2217100395843 57865
- selling_price  1  30856819344 2247957215187 57902
- seller_type    2  37806247045 2254906642888 57909
- owner          4  54890453340 2271990849183 57926
- year           1 167264826758 2384365222601 58069
- fuel           4 513656118550 2730756514393 58446

Call:
lm(formula = km_driven ~ year + selling_price + fuel + seller_type +
    owner, data = cars.no.training)

Coefficients:
                (Intercept)                         year                selling_price
              5247341.78631                  -2575.81243                     -0.01893
                 fuelDiesel                 fuelElectric                      fuelLPG
                14578.26539                 -21187.54920                   -876.22687
                 fuelPetrol        seller_typeIndividual  seller_typeTrustmark Dealer
               -14994.14165                   8689.66679                  -3261.32464
  ownerFourth & Above Owner             ownerSecond Owner        ownerTest Drive Car
                12574.96560                   8793.15688                 -13989.46675
           ownerThird Owner
                14778.89451
```

Hide

```
# After backward elimination: cars.no.training
CNOTModel<-lm(formula = km_driven ~ year + selling_price + fuel + seller_type +
    owner, data = cars.no.training)
summary(CNOTModel)
```

```
Call:
lm(formula = km_driven ~ year + selling_price + fuel + seller_type + 
    owner, data = cars.no.training)

Residuals:
    Min      1Q  Median      3Q     Max 
-125116  -18319   -3447   16869  117006 

Coefficients:
                                Estimate     Std. Error t value             Pr(>|t|
)    
(Intercept)                 5247341.786306  355817.310265  14.747 < 0.000000000000000
2 ***
year                          -2575.812429     176.878147 -14.563 < 0.000000000000000
2 ***
selling_price                    -0.018928       0.003026  -6.255      0.000000000458
6 ***
fuelDiesel                    14578.265392    6526.471702   2.234             0.02558
0 *  
fuelElectric                 -21187.549200   28863.238751  -0.734             0.46296
9    
fuelLPG                        -876.226871   10120.955034  -0.087             0.93101
5    
fuelPetrol                   -14994.141650    6499.599875  -2.307             0.02113
1 *  
seller_typeIndividual          8689.666792    1338.248350   6.493      0.000000000098
9 ***
seller_typeTrustmark Dealer   -3261.324637    3726.599229  -0.875             0.38156
8    
ownerFourth & Above Owner     12574.965597    3802.582908   3.307             0.00095
5 ***
ownerSecond Owner              8793.156882    1326.034276   6.631      0.000000000039
8 ***
ownerTest Drive Car          -13989.466751   19924.732729  -0.702             0.48266
5    
ownerThird Owner              14778.894511    2272.367864   6.504      0.000000000092
4 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 28080 on 2811 degrees of freedom
Multiple R-squared:  0.3681,    Adjusted R-squared:  0.3654 
F-statistic: 136.4 on 12 and 2811 DF,  p-value: < 0.00000000000000022
```

Hide

```
# Build Multiple Regression Model for cars.tx.training
carsTXTrainingModel<-lm(km_driven ~., data = cars.tx.training)
step(carsTXTrainingModel, direction = "backward")
```

```
Start:  AIC=23000.7
km_driven ~ year + selling_price + fuel + seller_type + transmission + 
    owner

                Df Sum of Sq      RSS   AIC
<none>                        9633110 23001
- transmission   1     13104  9646214 23002
- selling_price  1    118345  9751455 23033
- owner          4    172442  9805552 23043
- seller_type    2    167894  9801004 23046
- year           1    843939 10477048 23236
- fuel           4   1967265 11600375 23518

Call:
lm(formula = km_driven ~ year + selling_price + fuel + seller_type + 
    transmission + owner, data = cars.tx.training)

Coefficients:
                (Intercept)                         year                selling_price  
             11771.61205544                  -5.72820494                  -0.00003819  
                 fuelDiesel                 fuelElectric                      fuelLPG  
                18.26332108                 -36.28884786                  10.89979690  
                 fuelPetrol        seller_typeIndividual  seller_typeTrustmark Dealer  
               -40.31495778                  18.52545426                  -4.98779740  
         transmissionManual     ownerFourth & Above Owner            ownerSecond Owner  
                 9.16637949                  23.53188970                  16.29407663  
        ownerTest Drive Car             ownerThird Owner  
               -61.72044745                  23.50429069  
```

Hide

```
# After backward elimination: cars.tx.training
CTXTModel<-lm(formula = km_driven ~ year + selling_price + fuel + seller_type + 
    transmission + owner, data = cars.tx.training)
```

7. Build a Regression Tree model using rpart package for predicting km_driven: one with cars.training, one with cars.no.training, and one with cars.tx.training, i.e., regression models that contains all features.

Hide

```
# Build Regression Tree Model: cars.training
CTDecTree<-rpart(km_driven ~., data = cars.training)

# Check cars.training decision tree
CTDecTree
```

```
n= 3255

node), split, n, deviance, yval
      * denotes terminal node

 1) root 3255 7041348000000  65762.99
   2) year>=2015.5 1067 1345751000000  37183.36
     4) year>=2017.5 452  166801700000  23932.56 *
     5) year< 2017.5 615 1041257000000  46922.15
      10) fuel=Petrol 309  148447800000  33009.44 *
      11) fuel=CNG,Diesel 306  772600300000  60971.26 *
   3) year< 2015.5 2188 4399070000000  79700.13
     6) fuel=CNG,Electric,Petrol 1067 1641615000000  65938.54
      12) year>=2010.5 522  404300600000  54401.67 *
      13) year< 2010.5 545 1101291000000  76988.53 *
     7) fuel=Diesel,LPG 1121 2363050000000  92798.81
      14) seller_type=Dealer,Trustmark Dealer 269  406079500000  70472.18 *
      15) seller_type=Individual 852 1780544000000  99847.94
        30) year>=2012.5 356  580040700000  88540.83 *
        31) year< 2012.5 496 1122320000000 107963.50 *
```

Hide

```
# Build Regression Tree Model: cars.no.training
CNOTDecTree<-rpart(km_driven ~., data = cars.no.training)

# Check cars.training decision tree
CNOTDecTree
```

```
n= 2824

node), split, n, deviance, yval
      * denotes terminal node

 1) root 2824 3508555000000 65176.09
   2) year>=2014.5 1090  909307200000 45211.90
     4) fuel=Petrol 585  300685800000 33416.57 *
     5) fuel=CNG,Diesel,LPG 505  432946000000 58875.79
      10) year>=2016.5 229  133184100000 45946.30 *
      11) year< 2016.5 276  229716300000 69603.52 *
   3) year< 2014.5 1734 1891717000000 77725.66
     6) fuel=CNG,Electric,LPG,Petrol 904  775177600000 67746.11
      12) year>=2010.5 378  285376900000 57754.52 *
      13) year< 2010.5 526  424945700000 74926.37 *
     7) fuel=Diesel 830  928450700000 88594.96
      14) seller_type=Dealer 168  171942800000 69412.88 *
      15) seller_type=Individual 662  679004400000 93462.92 *
```

Hide

```
# Build Regression Tree Model: cars.tx.training
CTXDecTree<-rpart(km_driven ~., data = cars.tx.training)

# Check cars.no.training decision tree
CTXDecTree
```

```
n= 2824

node), split, n, deviance, yval
      * denotes terminal node

 1) root 2824 15125590.0 244.5504
   2) year>=2015.5 794  3193259.0 189.9084
     4) fuel=Petrol 440  1426102.0 165.8728 *
     5) fuel=CNG,Diesel 354  1197020.0 219.7831
      10) year>=2017.5 98   237970.1 180.9517 *
      11) year< 2017.5 256   754709.5 234.6482 *
   3) year< 2015.5 2030  8634400.0 265.9226
     6) fuel=Electric,Petrol 1033  3920280.0 247.0994
      12) year>=2010.5 520  1900599.0 226.5727 *
      13) year< 2010.5 513  1578492.0 267.9062 *
     7) fuel=CNG,Diesel,LPG 997  3968891.0 285.4256
      14) seller_type=Dealer,Trustmark Dealer 235   985672.1 252.2665 *
      15) seller_type=Individual 762  2645144.0 295.6518 *
```

8. Provide an analysis of all the 6 models (using their respective testing data sets), including Adjusted R-Squared and RMSE. Which of these models is the best? Why?

Evaluate Multiple Regression Models

Hide

```
# Evaluate cars.training model
CTPred<-predict(CTModel, cars.testing)

# Evaluate cars.no.training model
CNOTPred<-predict(CNOTModel, cars.no.testing)

# Evaluate cars.tx.training model
CTXPred<-predict(CTXTModel, cars.tx.testing)
```

Evaluate Decision Tree Models

Hide

```
# Evaluate cars.training model
CTDTPred<-predict(CTDecTree, cars.testing)

# Evaluate cars.no.training model
CNOTDTPred<-predict(CNOTDecTree, cars.no.testing)

# Evaluate cars.tx.training model
CTXDTPred<-predict(CTXDecTree, cars.tx.testing)
```

Calculate RMSE for Multiple Regression Models

Hide

```
# Build RMSE function

rmse <- function(actual, predicted)
{
    sqrt(mean(abs(actual - predicted)^2))
}
```

Hide

```
# RMSE for predicted multiple regression model: cars.training
rmse(cars.testing$km_driven, CTPred)
```

```
[1] 37505.58
```

Hide

```
# RMSE for predicted multiple regression model: cars.no.training
rmse(cars.no.testing$km_driven, CNOTPred)
```

```
[1] 28346.23
```

Hide

```
# RMSE for predicted multiple regression model: cars.tx.training
rmse(cars.tx.testing$km_driven, CTXPred)
```

```
[1] 57.06676
```

RMSE for Decision Tree Models:

Hide

```
# RMSE for predicted Decision Tree model: cars.training
rmse(cars.testing$km_driven, CTDTPred)
```

```
[1] 36730.25
```

Hide

```
# RMSE for predicted Decision Tree model: cars.not.training
rmse(cars.no.testing$km_driven, CNOTDTPred)
```

```
[1] 28414.98
```

Hide

```
# RMSE for predicted Decision Tree model: cars.tx.training
rmse(cars.no.testing$km_driven, CTXDTPred)
```

```
[1] 74531
```

ADJUSTED R-SQUARED:

Hide

```
# Adjusted R-Squared for Multiple Regression Model: cars.training
summary(CTModel)$adj.r.squared
```

```
[1] 0.3086519
```

Hide

```
# Adjusted R-Squared for Multiple Regression Model: cars.no.training
summary(CNOTModel)$adj.r.squared
```

```
[1] 0.3653896
```

Hide

```
# Adjusted R-Squared for Multiple Regression Model: cars.tx.training
summary(CTXTModel)$adj.r.squared
```

```
[1] 0.3601788
```

CHECK CORRELATION WITH ACTUAL VALUES

Decision Tree Models

Hide

```
# cars.training
cor(CTDTPred, cars.testing$km_driven)
```

```
[1] 0.6278282
```

Hide

```
# cars.no.training
cor(CNOTDTPred, cars.no.testing$km_driven)
```

```
[1] 0.6071065
```

Hide

```
# cars.tx.training
cor(CTXDTPred, cars.tx.testing$km_driven)
```

```
[1] 0.6616466
```

Based from the correlation evaluation above, cars.tx.training seems to have the strongest relationship with the actual values compared to the other training sets.

Multiple Regression Models

Hide

```
# cars.training
cor(CTPred, cars.testing$km_driven)
```

```
[1] 0.6091869
```

Hide

```
# cars.no.training
cor(CNOTPred, cars.no.testing$km_driven)
```

```
[1] 0.6088421
```

Hide

```
# cars.tx.training
cor(CTXPred, cars.tx.testing$km_driven)
```

```
[1] 0.6562801
```

Similar to the decision trees, multiple regression from cars.tx seems to have the highest correlation with the actual values. Thus far, the best model for cars.training goes to decision tree model with correlation score of 0.627; cars.no.training goes to multiple regression model with correlation score of 0.608 with the actual values; and lastly cars.tx.training obtained stronger correlation from the decision tree model with score of 0.661 from the actual values.

9. Using each of the regression models, what are the predicted odometer readings (km_driven) of a 2004 vehicle that was sold by a dealer for R87,000, has a Diesel engine, a manual transmission, and is the second owner? Why are the predictions different?

10. For each of the predictions, calculate the 95% prediction interval for the kilometers driven. (Exclude Regression Trees)

```
# cars.training: CTModel
predict(CTModel, cars.testing$km_driven, interval = "confidence")

# cars.training: CNOTModel
predict(CNOTModel, cars.no.testing$km_driven, interval = "confidence")


# cars.training: CTXTModel
predict(CTXTModel, cars.no.testing$km_driven, interval = "confidence")
```