FUNDAMENTALS OF DATA SCIENCE – ASSIGNMENT 1

Mrunali Vikas Patil

PROBLEM SET 1

```
getwd()

## [1] "C:/Users/Sanket Patil/Documents"

Mydata = read.csv("Bankdata.csv")

library(dplyr)

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

library(ggplot2)
```

Question A)

```
lapply(Mydata,class)

## $X
## [1] "integer"
##
## $cont1
## [1] "numeric"
##
## $cont2
## [1] "numeric"
##
## $cont3
## [1] "numeric"
##
## $bool1
## [1] "character"
##
## $bool2
## [1] "character"
```

```
##
## $cont4
## [1] "integer"
##
## $bool3
## [1] "character"
##
## $cont5
## [1] "integer"
##
## $cont6
## [1] "integer"
##
## $approval
## [1] "character"
##
## $credit.score
## [1] "numeric"
##
## $ages
## [1] "integer"
```

```
ls(Mydata)
```

```
##  [1] "ages"         "approval"     "bool1"        "bool2"        "bool3"
##  [6] "cont1"        "cont2"        "cont3"        "cont4"        "cont5"
## [11] "cont6"        "credit.score" "X"
```

Visualization for Numerical Variables

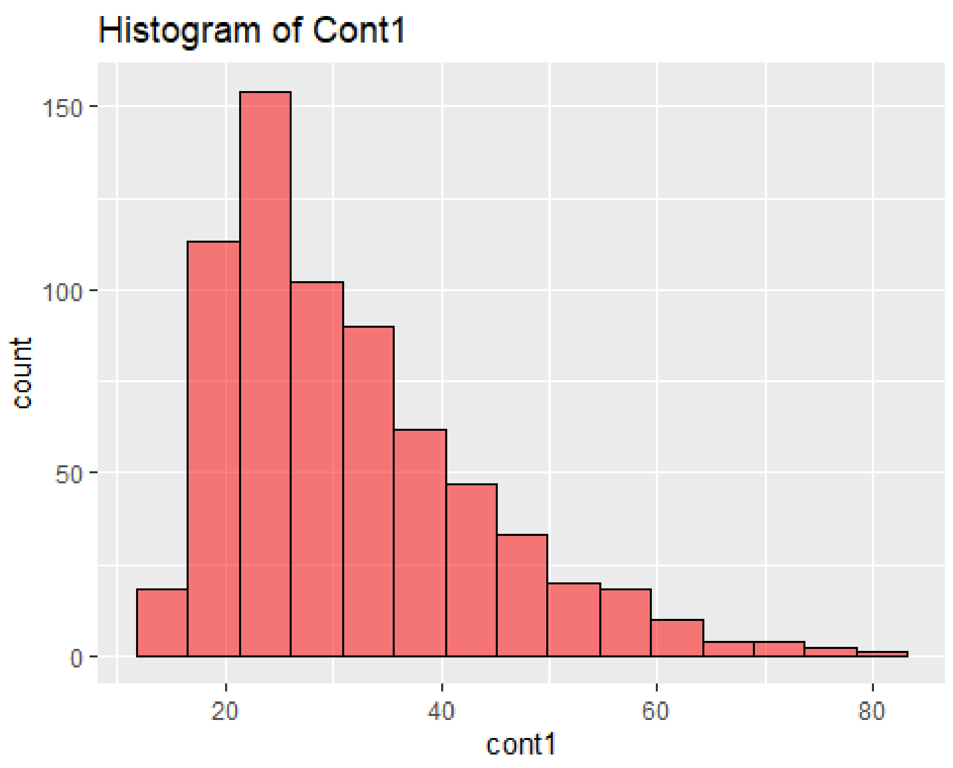
```
num_vars<- select_if(Mydata,is.numeric)
is.numeric(Mydata$cont1)
```

```
## [1] TRUE
```

Histogram for Numerical Data

```
Mydata %>% ggplot(aes(cont1)) +
  geom_histogram(fill = "red", bins = 15, color = "black", alpha = 0.5) +
  labs( x = "cont1", title = "Histogram of Cont1")
```

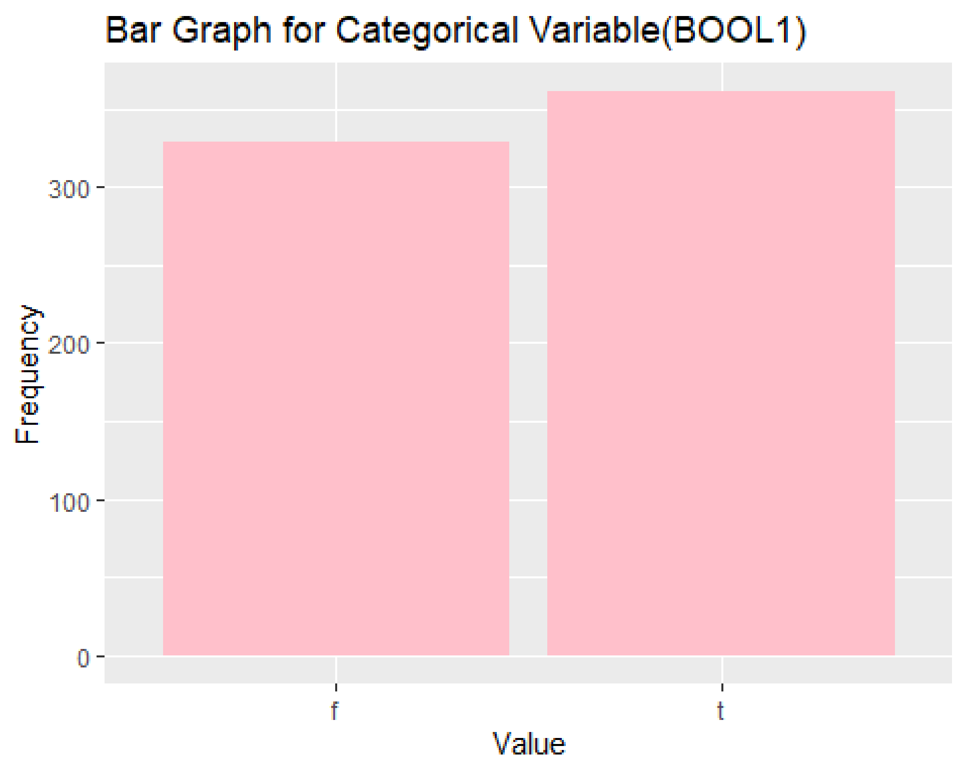
```
## Warning: Removed 12 rows containing non-finite values (`stat_bin()`).
```



We can conclude that the histogram is right skewed and the data is not normally distributed.

Create a bar graph for the boolean variable using ggplot2

```r
ggplot(Mydata, aes(x = factor(bool1))) +
  geom_bar(fill = "pink") +
  labs(
    title = "Bar Graph for Categorical Variable(BOOL1)",
    x = "Value",
    y = "Frequency")
```



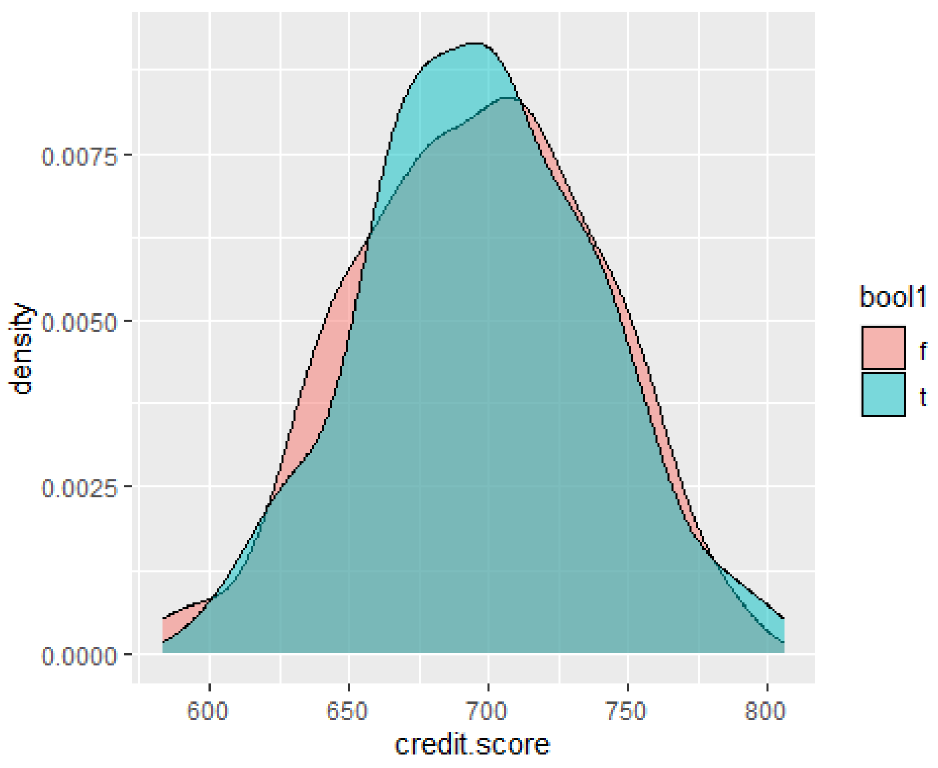
Both the bar does not have major difference so it derives a balanced distribution

Both the bool1 categorical values have similar frequency

Since both bars have similar heights, it suggests that the distribution of TRUE and FALSE values in the bool1 variable is approximately equal.

Create a density plot with two numerical

```r
Mydata %>% ggplot(aes(x = credit.score, fill = bool1)) +
geom_density(alpha=0.5)
```



With the help of density plot the data for bool1 = true and bool1 = false both looks normal hence we can conclude that the data is normally distributed

Calculate summary Statistics

```
summary(Mydata$cont1)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
##   13.75   22.60   28.46   31.57   38.23   80.25      12

print(summary)

## function (object, ...) 
## UseMethod("summary")
## <bytecode: 0x000001afa72f66e0>
## <environment: namespace:base>
```

Question B)

```
library(dplyr)
```

Applying normalisation to numerical distributions

- Z-score normalization on numerical variable Cont3

```
z_score_normalized_cont3 <- (Mydata$cont3 - mean(Mydata$cont3)) /
sd(Mydata$cont3)
head(z_score_normalized_cont3)
```

```
## [1] -0.29087163  0.24401343 -0.21616701  0.45617454 -0.15341513
0.08265146
```

- Min_Max normalization on numerical variable Cont2

```
Min_Max_Norm_cont2 <- (Mydata$cont2 - min(Mydata$cont2)) /
(max(Mydata$cont2) - min(Mydata$cont2))
head(Min_Max_Norm_cont2)

## [1] 0.00000000 0.15928571 0.01785714 0.05500000 0.20089286 0.14285714
```

- Decimal Scaling normalization on numerical variable ages

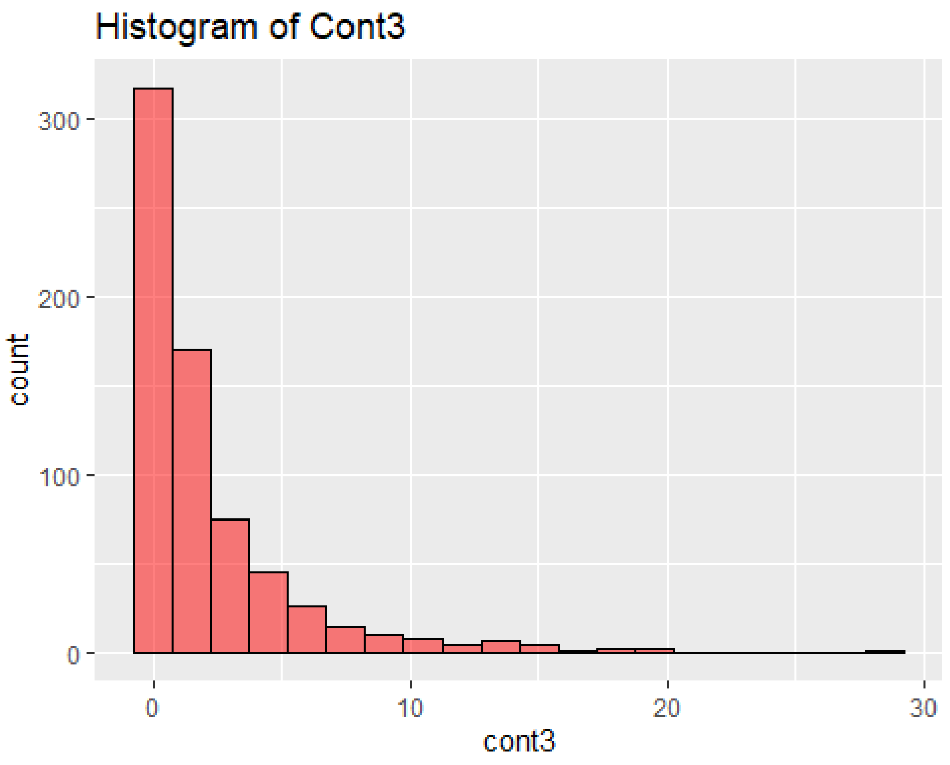
```
Decimal_Scaling_ages <- (Mydata$ages / 100)
head(Decimal_Scaling_ages)

## [1] 0.42 0.54 0.29 0.58 0.65 0.61
```

Question C) Visualizing the Distributions

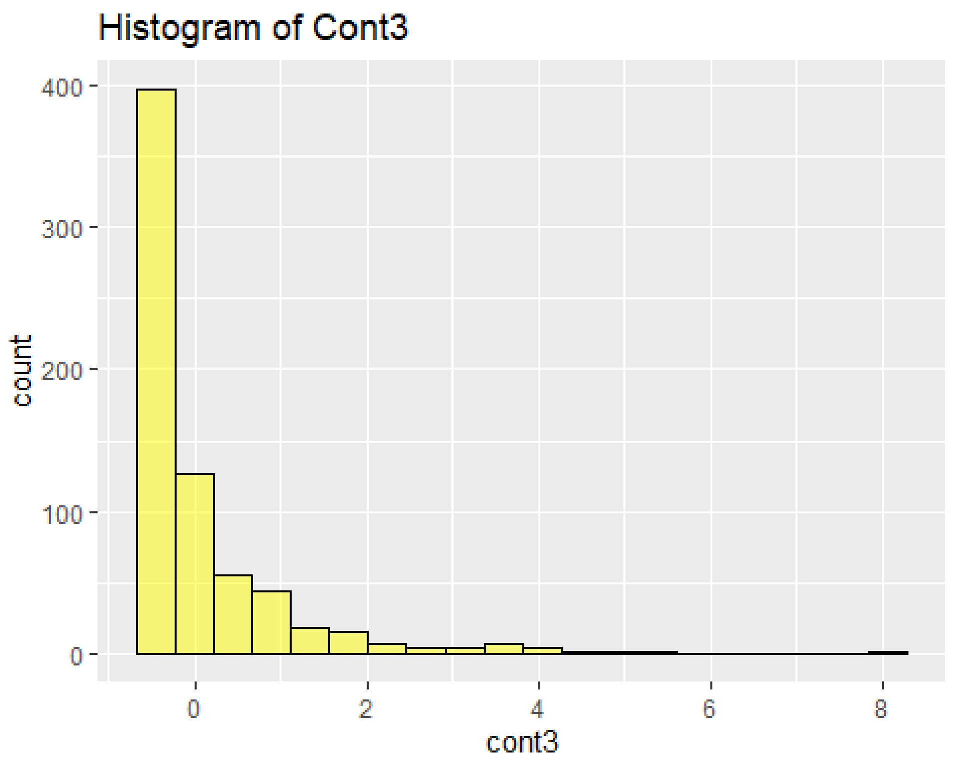
1)Data Visualization with old Variable

```
Mydata %>% ggplot(aes(cont3)) +
  geom_histogram(fill = "red", bins = 20, color = "black", alpha = 0.5) +
  labs( x = "cont3", title = "Histogram of Cont3")
```



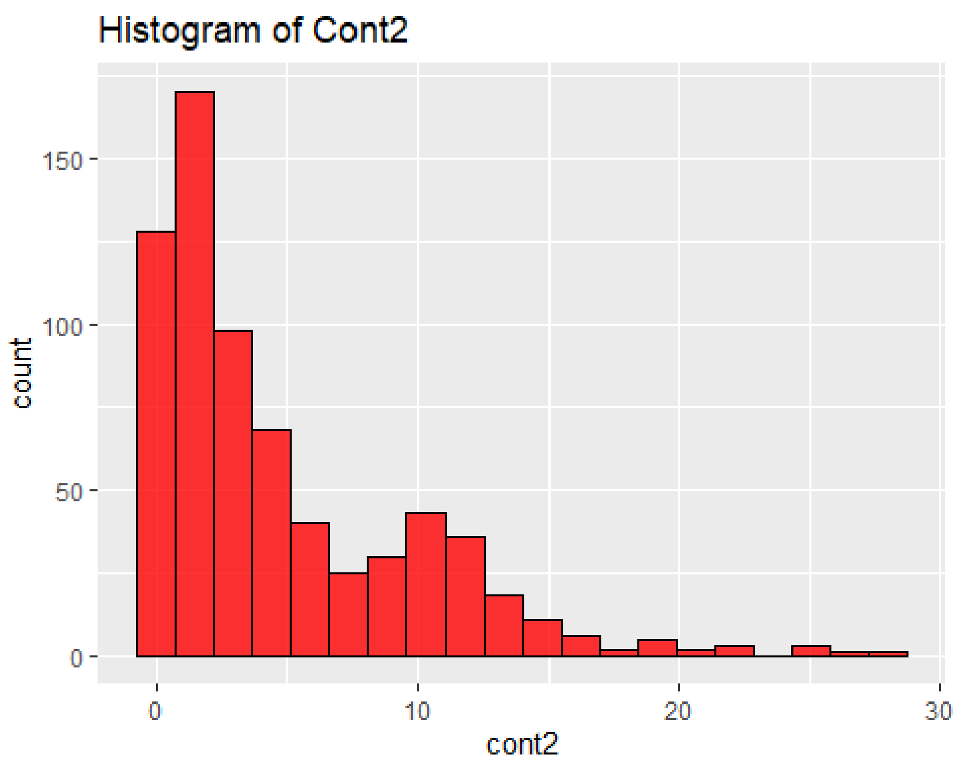
2)Data Visualization with new Variable

```
Mydata %>% ggplot(aes(z_score_normalized_cont3 )) +
  geom_histogram(fill = "yellow", bins = 20, color = "black", alpha = 0.5) +
  labs( x = "cont3", title = "Histogram of Cont3")
```

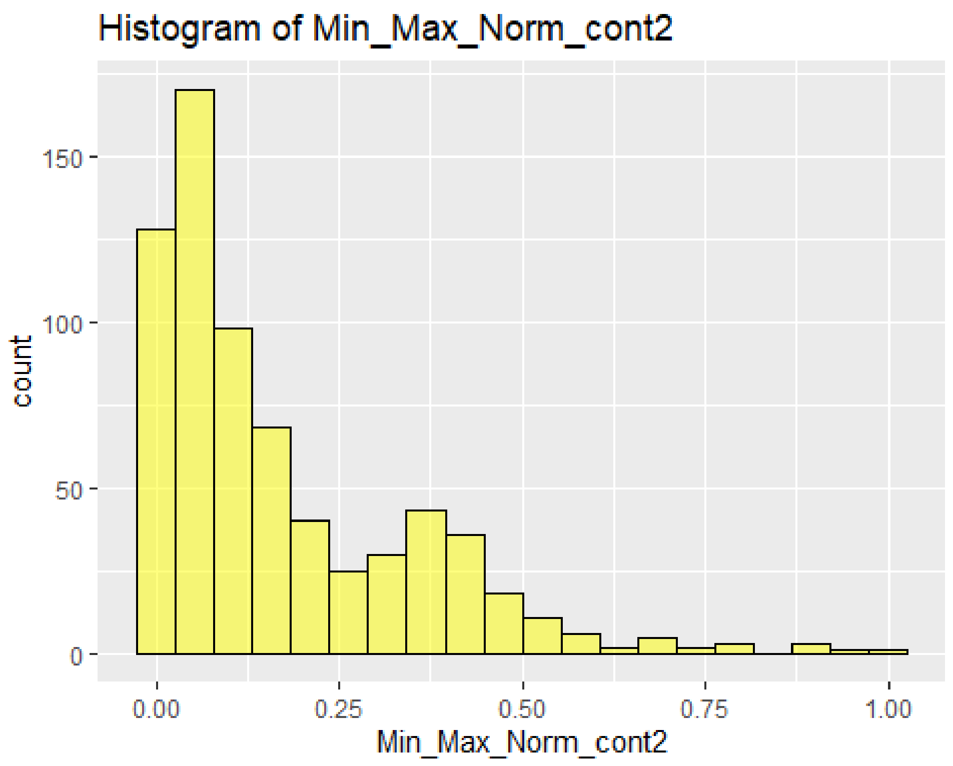


There is no change in the shape of histogram after normalization

```
Mydata %>% ggplot(aes(cont2)) +
  geom_histogram(fill = "red", bins = 20, color = "black", alpha = 0.8) +
  labs( x = "cont2", title = "Histogram of Cont2")
```

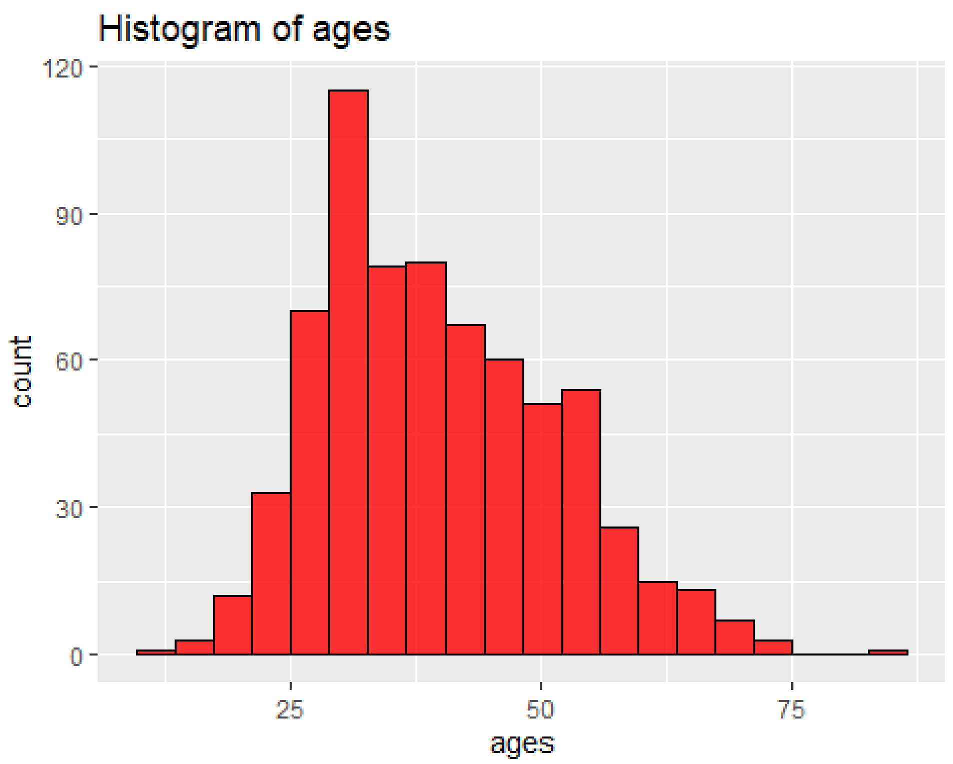


```r
Mydata %>% ggplot(aes(Min_Max_Norm_cont2)) +
  geom_histogram(fill = "yellow", bins = 20, color = "black", alpha = 0.5) +
  labs( x = "Min_Max_Norm_cont2", title = "Histogram of Min_Max_Norm_cont2")
```

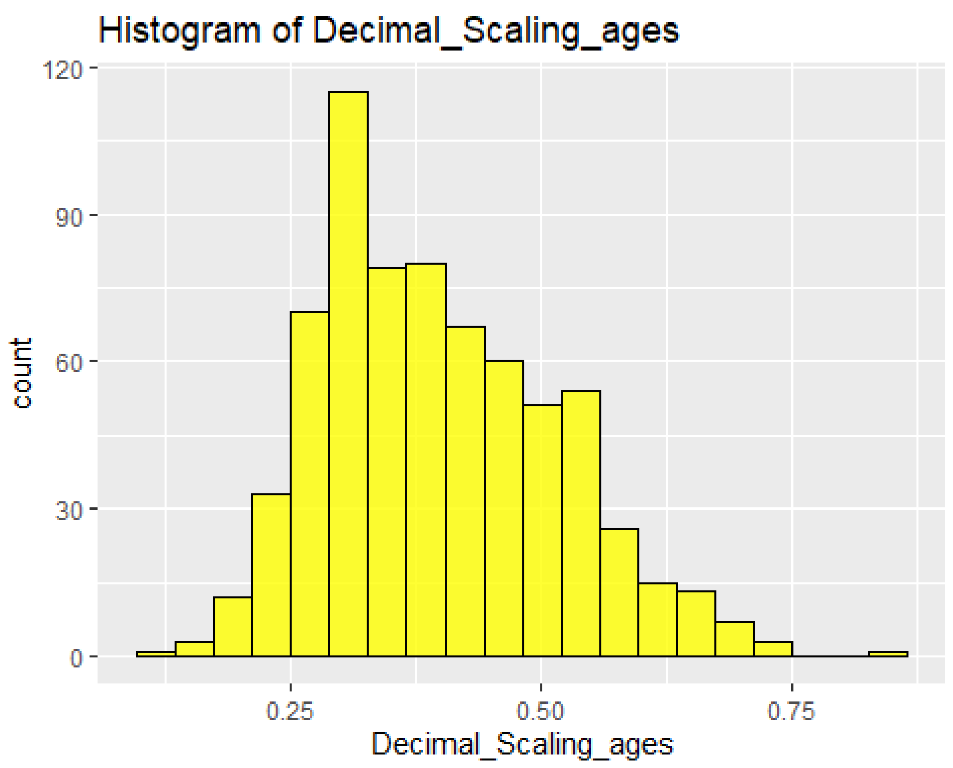


3)

```
Mydata %>% ggplot(aes(ages)) +
  geom_histogram(fill = "red", bins = 20, color = "black", alpha = 0.8) +
  labs( x = "ages", title = "Histogram of ages")
```



```
Mydata %>% ggplot(aes(Decimal_Scaling_ages)) +
  geom_histogram(fill = "yellow", bins = 20, color = "black", alpha = 0.8) +
  labs( x = "Decimal_Scaling_ages", title = "Histogram of
Decimal_Scaling_ages")
```

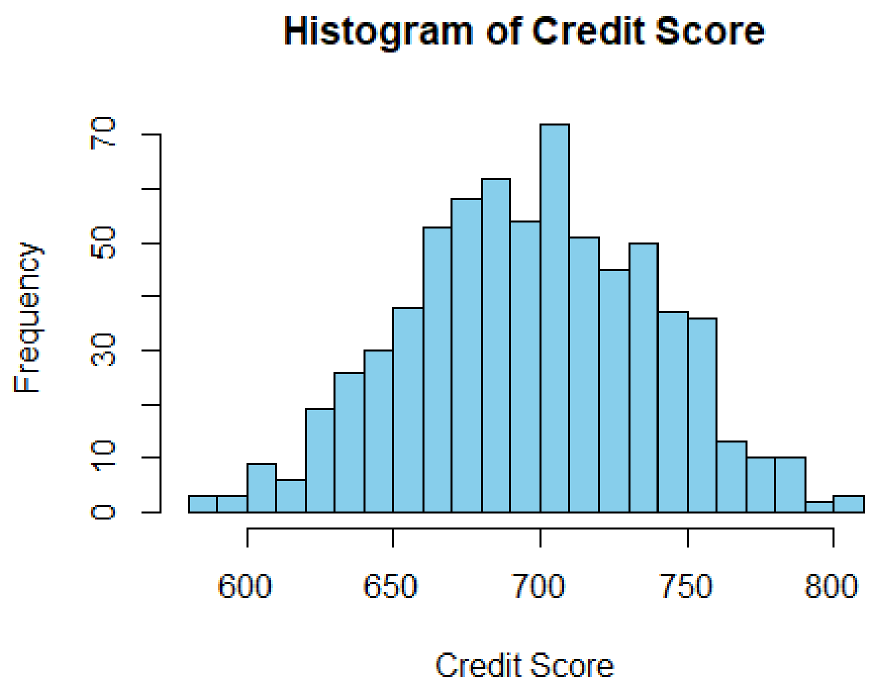


Question D)

BINNING-We are choosing “credit.score” column to get a sense of its distribution which will conclude the binning

Histogram of credit.score

```r
hist(Mydata$credit.score,
     main = "Histogram of Credit Score",
     xlab = "Credit Score",
     col = "skyblue",
     breaks = 30)
```



The distribution of the credit.score variable in histogram appears to be roughly normally distributed, with a slight skew to the right.

#Determining the bins :

- Equal Depth: This technique creates bins with roughly the same number of data points in the data. It's useful for skewed data.
- Equal Width: This technique creates equal-sized bins from the data range.
- Custom Ranges: Using this technique, we can define unique bin edges depending on domain expertise or predetermined standards.

By looking at the shape of histogram, we will use Equal depth method.

It appears that certain score ranges were more individuals fall, and using equal depth would prevent us from ending up with bins with very few data points.

Creating three bins : Low, Medium, High. With this option, credit scores are easily and clearly categorized, which is useful for analyses

Create bins using equal depth

```
Mydata$credit.score_bins <- cut(Mydata$credit.score,
breaks=quantile(Mydata$credit.score, probs=0:3/3),
                                labels=c("Low", "Medium", "High"),
include.lowest=TRUE)
```

To make the interpretation easy we are choosing the three bins (Low, Medium, High) which can easily determine where an individual's credit score stands relative to others.

Question E)

SMOOTHING -To Create a smoothed version of credit.score using the binned variable 'v-bins' we are replacing each value in v with the mean of the values within its respective bin. This is called as mean smoothing which is used to reduce the variability within each bin.

Create v_bins

```
Mydata$v_bins <- cut(Mydata$credit.score, breaks =
quantile(Mydata$credit.score, probs = 0:3/3),
                     labels = c("Low", "Medium", "High"), include.lowest =
TRUE)
```

Calculate mean for each bin

```
bin_means <- tapply(Mydata$credit.score, Mydata$v_bins, mean)
```

Replace original values with bin mean

```
Mydata$credit.score.smoothed <- as.numeric(bin_means[Mydata$v_bins])
```

View the first few rows

```
head(Mydata[, c("credit.score", "v_bins", "credit.score.smoothed")])

##   credit.score v_bins credit.score.smoothed
## 1       664.60    Low              650.1357
## 2       693.88 Medium              696.2524
## 3       621.82    Low              650.1357
## 4       653.97    Low              650.1357
## 5       670.26    Low              650.1357
## 6       672.16    Low              650.1357
```

PROBLEM SET 2

Load necessary libraries

```
library(e1071)
library(caret)

## Loading required package: lattice
```

#Question A)

SVM with 10-fold Cross Validation. Removing the rows with NA values

```
Bank_data <- na.omit(read.csv("BankData.csv"))
```

Splitting data into predictors and target

```
predictors <- Bank_data[, !(names(Bank_data) %in% c("approval"))]
target <- Bank_data$approval
```

SVM with 10-fold CV

```
set.seed(123)
train_control <- trainControl(method = "cv", number = 10)
svm_model<- train(approval ~ ., data = Bank_data, method =
"svmLinear", trControl = train_control)
print(svm_model)

## Support Vector Machines with Linear Kernel
##
## 666 samples
##  12 predictor
##   2 classes: '-', '+'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 599, 599, 599, 600, 599, 600, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.8633424  0.7286267
##
## Tuning parameter 'C' was held constant at a value of 1
```

#Question B)

Grid Search for optimal C

```
grid <- expand.grid(C = seq(0.1, 2, 0.1))
```

```r
set.seed(123)
svm_grid_model <- train(approval ~ ., data = Bank_data, method =
"svmLinear", trControl = train_control, tuneGrid = grid)
print(svm_grid_model)

## Support Vector Machines with Linear Kernel
##
## 666 samples
##  12 predictor
##   2 classes: '-', '+'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 599, 599, 599, 600, 599, 600, ...
## Resampling results across tuning parameters:
##
##   C    Accuracy   Kappa
##   0.1  0.8633424  0.7286267
##   0.2  0.8633424  0.7286267
##   0.3  0.8633424  0.7286267
##   0.4  0.8633424  0.7286267
##   0.5  0.8633424  0.7286267
##   0.6  0.8633424  0.7286267
##   0.7  0.8633424  0.7286267
##   0.8  0.8633424  0.7286267
##   0.9  0.8633424  0.7286267
##   1.0  0.8633424  0.7286267
##   1.1  0.8633424  0.7286267
##   1.2  0.8633424  0.7286267
##   1.3  0.8633424  0.7286267
##   1.4  0.8633424  0.7286267
##   1.5  0.8633424  0.7286267
##   1.6  0.8633424  0.7286267
##   1.7  0.8633424  0.7286267
##   1.8  0.8633424  0.7286267
##   1.9  0.8633424  0.7286267
##   2.0  0.8633424  0.7286267
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was C = 0.1.
```

The final value used for the model was C = 0.1.

Best accuracy of the model is 0.8633424

Question C)

Even if the grid search includes C = 1, there may still be a difference in accuracy between the default SVM model and the grid search because of the following reasons:

- Randomness in Cross Validation: The partitions or folds can be different between the two runs, even though cross-validation is used in both cases resulting in somewhat different training and validation sets.
- Model Fitting Variability: Multiple runs of the SVM might converge differently, especially in the presence of multiple local minima.
- Other Hyperparameters:While the focus is on C, other hyperparameters and default settings might interact differently across runs.

Hence, we are initializing the random number generator with a seed to guarantee consistent results bewteen runs which ensures the reproducibility of any process involving randomness, such as data splitting in CV.

PROBLEM SET 3

Question A) -Data Preprocessing

Loading dplyr and dataset

```
library(dplyr)
head(starwars)

## # A tibble: 6 × 14
##   name      height  mass hair_color skin_color eye_color birth_year sex
gender
##   <chr>      <int> <dbl> <chr>      <chr>      <chr>          <dbl> <chr>
<chr>
## 1 Luke Sky…    172    77 blond      fair       blue            19   male
mascu…
## 2 C-3PO        167    75 <NA>       gold       yellow         112   none
mascu…
## 3 R2-D2         96    32 <NA>       white, bl… red             33   none
mascu…
## 4 Darth Va…    202   136 none       white      yellow          41.9 male
mascu…
## 5 Leia Org…    150    49 brown      light      brown           19   fema…
femin…
## 6 Owen Lars    178   120 brown, gr… light      blue            52   male
mascu…
## # ℹ 5 more variables: homeworld <chr>, species <chr>, films <list>,
## #   vehicles <list>, starships <list>

colnames(starwars)

##  [1] "name"       "height"     "mass"       "hair_color" "skin_color"
##  [6] "eye_color"  "birth_year" "sex"        "gender"     "homeworld"
## [11] "species"    "films"      "vehicles"   "starships"
```

Remove unwanted columns and rows with missing values

```
sw_cleaned <- starwars %>%
  select(-films, -vehicles, -starships, -name) %>%
  na.omit()

colnames(sw_cleaned)

##  [1] "height"     "mass"       "hair_color" "skin_color" "eye_color"
##  [6] "birth_year" "sex"        "gender"     "homeworld"  "species"
```

Convert categorical variables to dummy variables, except 'gender'

```
sw_dummies <- model.matrix(~.+ 0 , data = sw_cleaned)

gender<-sw_cleaned$gender
```

```
sw_final <- cbind.data.frame(sw_dummies,gender)
```

Question B)-SVM Prediction on Gender

```
library(e1071)
```

Split data into training and test sets (e.g., 80-20% split)

```
set.seed(234)
train_index <- sample(1:nrow(sw_final), 0.8*nrow(sw_final))
train_data <- sw_final[train_index,]
test_data <- sw_final[-train_index,]

train_data$gender <- as.factor(train_data$gender)
test_data$gender <- as.factor(test_data$gender)
```

SVM Model

```
svm_model <- svm(gender ~ ., data = train_data, kernel = 'radial')

## Warning in svm.default(x, y, scale = scale, ..., na.action = na.action):
## Variable(s) 'X.hair_colorauburn..white.' and 'hair_colorgrey' and
## 'skin_colorred' and 'skin_coloryellow' and 'X.eye_colorblue.gray.' and
## 'homeworldDathomir' and 'homeworldMirial' and 'homeworldStewjon' and
## 'speciesGungan' and 'speciesMirialan' and 'speciesZabrak' constant. Cannot
## scale data.

svm_pred <- predict(svm_model, test_data)
```

Accuracy

```
mean(svm_pred == test_data$gender)

## [1] 0.6666667
```

Question C)- Principal Components Analysis

Remove gender before PCA

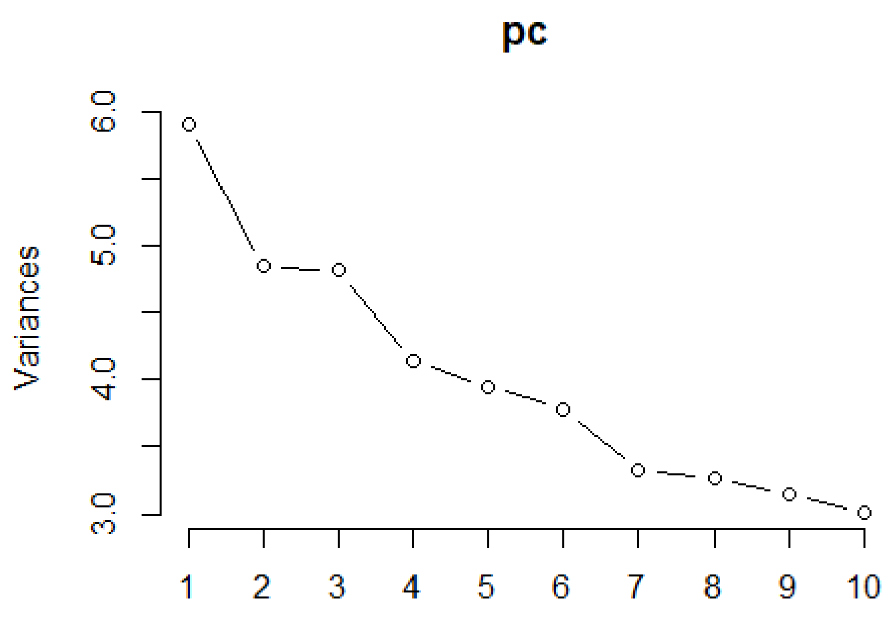
```
sw_no_gender <- sw_final %>% select(-gender)
```

Perform PCA

```
pc <- prcomp(sw_no_gender, center = TRUE, scale. = TRUE)
```

Plot to decide on number of components

```
plot(pc, type = "l")
```



A scree plot will help you decide the number of components.

Usually, we take the components up to the point where there is an

'elbow' in the graph, which indicates diminishing returns.

Add gender back

```
#reduced_data$gender <- sw_cleaned$gender
#head(reduced_data)

variance<-cumsum(pc$sdev^2)/sum(pc$sdev^2)
```

Let's assume the appropriate number of components is 'k'

```
k <- sum(variance >= 0.95)
```

Create a reduced version of the data

```
reduced_data <- data.frame(pc$x[, 1:k])

reduced_data$gender <- sw_final$gender
```

```r
reduced_data$gender <- as.factor(reduced_data$gender)
```

Question D) -Split data into training and test sets (e.g., 80-20% split)

```r
train_index_1 <- sample(1:nrow(reduced_data), nrow(reduced_data)*0.8)

train_data_pca <- reduced_data[train_index_1,]
test_data_pca <- reduced_data[-train_index_1,]
```

Grid search on the C parameter

```r
result <- tune.svm(gender ~ ., data = train_data_pca, kernel =
'radial', cost = 10^(-1:2))
```

Best model

```r
best_svm <- result$best.model
svm_pred_pca <- predict(best_svm, test_data_pca)
```

Confusion Matrix

```r
table(predicted = svm_pred_pca, actual = test_data_pca$gender)

##            actual
## predicted   feminine masculine
##   feminine         0         0
##   masculine        1         5

mean(svm_pred_pca == test_data_pca$gender)

## [1] 0.8333333
```

Since you asked for at least two partitioning methods, you can also

try k-fold cross-validation or stratified sampling.

These can be implemented using the caret package.

Question E) Implication of PCA on model complexity:

Yes, PCA has improved the model accuracy. Dimensionality Reduction: One of the primary utilities of PCA is the reduction in the number of features. It naturally lowers model complexity since there are fewer parameters to train

PCA primarily serves as a tool for dimensionality reduction and complexity management, under the right circumstances, its capacity to filter noise and highlight essential patterns can lead to an enhancement in model accuracy.

PROBLEM SET 4 [BONUS]

Load the Sacramento Housing Dataset

```
library(caret)
data(Sacramento)
head(Sacramento)
```

```
##         city    zip beds baths sqft        type price latitude longitude
## 1 SACRAMENTO z95838    2     1  836 Residential 59222 38.63191 -121.4349
## 2 SACRAMENTO z95823    3     1 1167 Residential 68212 38.47890 -121.4310
## 3 SACRAMENTO z95815    2     1  796 Residential 68880 38.61830 -121.4438
## 4 SACRAMENTO z95815    2     1  852 Residential 69307 38.61684 -121.4391
## 5 SACRAMENTO z95824    2     1  797 Residential 81900 38.51947 -121.4358
## 6 SACRAMENTO z95841    3     1 1122       Condo 89921 38.66260 -121.3278
```

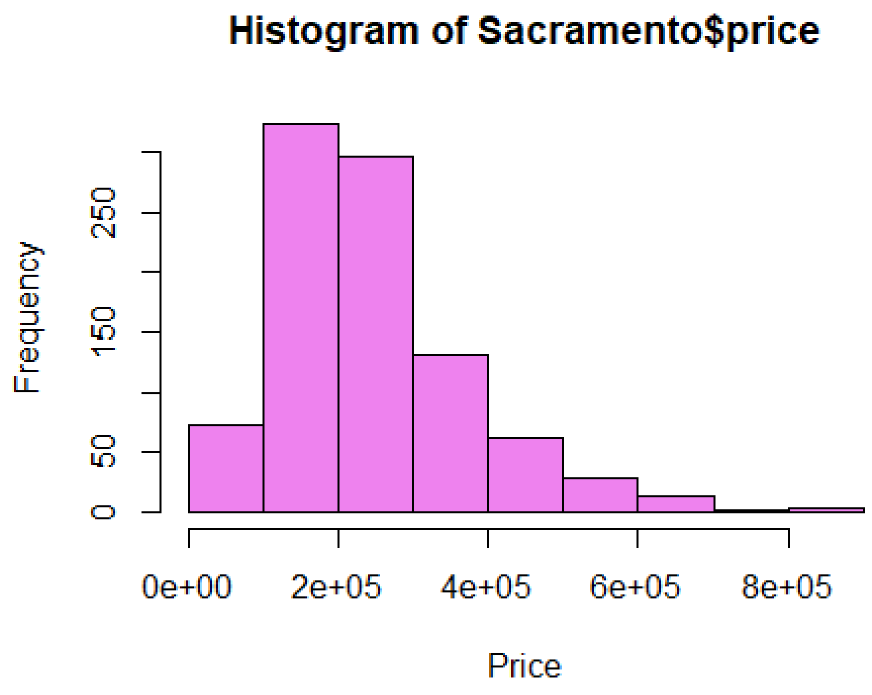
Question A)

After removing the zip and city variables, we need to explore the variables' distributions.

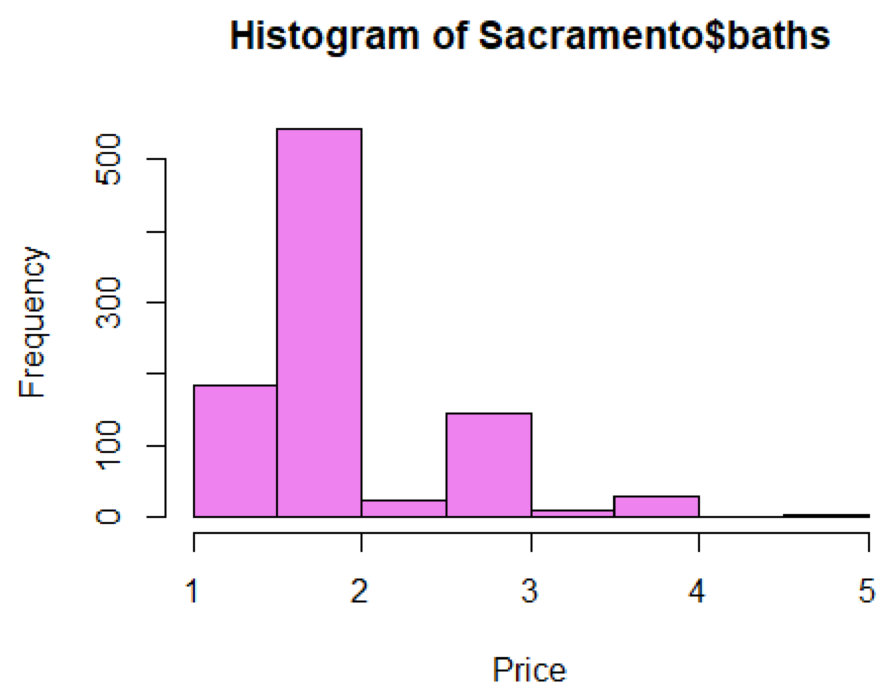
```
Sacramento$zip <- NULL
Sacramento$city <- NULL
```

Histogram for Sacramento dataset for Visualization

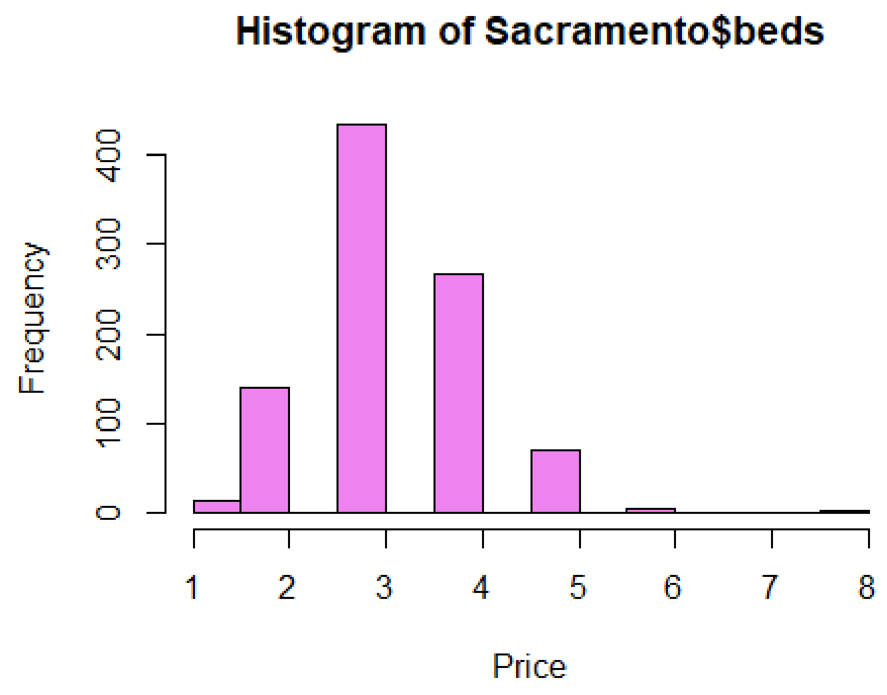
```
hist(Sacramento$price, xlab = "Price", col = "violet", border = "black")
```



```
hist(Sacramento$baths, xlab = "Price", col = "violet", border = "black")
```



```r
hist(Sacramento$beds, xlab = "Price", col = "violet", border = "black")
```



By looking at the histogram we can say that the data is skewed and not normally distributed

Summary of the data

```
summary(Sacramento)
```

```
##       beds           baths            sqft                   type    
##  Min.   :1.000   Min.   :1.000   Min.   : 484   Condo        : 53  
##  1st Qu.:3.000   1st Qu.:2.000   1st Qu.:1167   Multi_Family: 13  
##  Median :3.000   Median :2.000   Median :1470   Residential :866  
##  Mean   :3.276   Mean   :2.053   Mean   :1680                     
##  3rd Qu.:4.000   3rd Qu.:2.000   3rd Qu.:1954                     
##  Max.   :8.000   Max.   :5.000   Max.   :4878                     
##      price           latitude       longitude     
##  Min.   : 30000   Min.   :38.24   Min.   :-121.6  
##  1st Qu.:156000   1st Qu.:38.48   1st Qu.:-121.4  
##  Median :220000   Median :38.62   Median :-121.4  
##  Mean   :246662   Mean   :38.59   Mean   :-121.4  
##  3rd Qu.:305000   3rd Qu.:38.69   3rd Qu.:-121.3  
##  Max.   :884790   Max.   :39.02   Max.   :-120.6
```

Distribution of the 'type' variable

```
table(Sacramento$type)
```

```
## 
##        Condo Multi_Family  Residential 
##           53           13          866
```

Hence we can say that the class is imbalance as Residentials has more number of records than other two classes

Question B)

we wil choose to normalize the continuous variables to improve SVM performance.

```
num_vars <- sapply(Sacramento, is.numeric)
Sacramento[, num_vars] <- as.data.frame(lapply(Sacramento[, num_vars], 
scale))
```

#Question C) #Using SVM to predict type

```
set.seed(123)
trainIndex <- createDataPartition(Sacramento$type, p = .8, list = FALSE)
trainData <- Sacramento[ trainIndex,]
testData  <- Sacramento[-trainIndex,]
```

Grid search for SVM

```
svmGrid <- expand.grid(sigma = c(0.01, 0.05, 0.1, 0.5, 1),
                       C = 2^(-2:2))

ctrl <- trainControl(method = "repeatedcv",
                     repeats = 3,
                     classProbs = TRUE)

set.seed(123)
svmFit<- train(type ~ .,
                data = trainData,
                method = "svmRadial",
                metric = "Kappa",
                trControl = ctrl,
                tuneGrid = svmGrid)

## maximum number of iterations reached 0.000514077 -0.0005049466maximum
number of iterations reached 9.661635e-05 -9.628768e-05maximum number of
iterations reached 1.914973e-05 -1.912624e-05maximum number of iterations
reached 9.889054e-06 -9.895436e-06maximum number of iterations reached
3.155452e-05 -3.152959e-05maximum number of iterations reached 2.755035e-05 -
2.750956e-05maximum number of iterations reached 5.228284e-05 -5.212643e-05
```

Predict

```
predictions <- predict(svmFit, newdata = testData)
confusionMatrix(predictions, testData$type)

## Confusion Matrix and Statistics
##
##               Reference
## Prediction     Condo Multi_Family Residential
##   Condo            3            0           2
##   Multi_Family     0            1           0
##   Residential      7            1         171
##
## Overall Statistics
##
##                Accuracy : 0.9459
##                  95% CI : (0.9028, 0.9738)
##     No Information Rate : 0.9351
##     P-Value [Acc > NIR] : 0.3402
##
##                   Kappa : 0.423
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
```

```
##                          Class: Condo Class: Multi_Family Class: Residential
## Sensitivity                   0.30000            0.500000             0.9884
## Specificity                   0.98857            1.000000             0.3333
## Pos Pred Value                0.60000            1.000000             0.9553
## Neg Pred Value                0.96111            0.994565             0.6667
## Prevalence                    0.05405            0.010811             0.9351
## Detection Rate                0.01622            0.005405             0.9243
## Detection Prevalence          0.02703            0.005405             0.9676
## Balanced Accuracy             0.64429            0.750000             0.6609
```

Question D)

```
trainData$price <- log(trainData$price)

## Warning in log(trainData$price): NaNs produced

testData$price <- log(testData$price)

## Warning in log(testData$price): NaNs produced
```

Rerun SVM with the same grid and control

```
preProcValues <- preProcess(trainData, method = c("medianImpute"))
trainData <- predict(preProcValues, trainData)

set.seed(123)
svmFit2 <- train(type ~ .,
                 data = trainData,
                 method = "svmRadial",
                 metric = "Kappa",
                 trControl = ctrl,
                 tuneGrid = svmGrid)

## maximum number of iterations reached 0.00022114 -0.0002188617maximum
number of iterations reached 5.727031e-05 -5.712108e-05maximum number of
iterations reached 0.0004376835 -0.0004288011maximum number of iterations
reached 0.0005371391 -0.0005257595maximum number of iterations reached
3.885829e-05 -3.879001e-05maximum number of iterations reached 2.079603e-05 -
2.078118e-05maximum number of iterations reached 0.0001348999 -
0.0001341984maximum number of iterations reached 0.0009629987 -
0.0009366313maximum number of iterations reached 1.471911e-05 -1.470658e-
05maximum number of iterations reached 0.0002360849 -0.0002344039maximum
number of iterations reached 0.0007842599 -0.0007808549maximum number of
iterations reached 4.680926e-05 -4.669577e-05maximum number of iterations
reached 0.000168567 -0.0001680205maximum number of iterations reached
0.000581515 -0.0005667458maximum number of iterations reached 3.10351e-05 -
3.101702e-05maximum number of iterations reached 0.0003714756 -
0.0003659801maximum number of iterations reached 0.001622405 -
0.001590997maximum number of iterations reached 4.711169e-05 -4.698726e-
05maximum number of iterations reached 5.63516e-05 -5.616091e-05maximum
number of iterations reached 1.686072e-05 -1.687631e-05maximum number of
```

```
iterations reached 0.0002402774 -0.0002381366maximum number of iterations
reached 0.0001321113 -0.0001316357maximum number of iterations reached
0.0007423683 -0.0007273008maximum number of iterations reached 0.0001209395 -
0.0001206087maximum number of iterations reached 0.0005924003 -0.0005785523
```

Question E)

Balance the data with two smaller classes, then sample from the larger class.

```
table(Sacramento$type)

##
##        Condo Multi_Family  Residential
##           53           13          866
```

As residential has most frequent class and Condo and Multi_Family are less frequent:

```
set.seed(123)
residential_data <- subset(Sacramento, type == "Residential")
sampled_residential <-
residential_data[sample(1:nrow(residential_data), 200), ]
# Change 200 to the desired sample size

balanced_data <- rbind(sampled_residential, subset(Sacramento, type !=
"Residential"))
```

Summary

```
summary(balanced_data)

##       beds             baths              sqft                       type
##  Min.   :-2.5628   Min.   :-1.45739   Min.   :-1.6472   Condo        : 53
##  1st Qu.:-1.4367   1st Qu.:-0.07350   1st Qu.:-0.8197   Multi_Family: 13
##  Median :-0.3105   Median :-0.07350   Median :-0.3460   Residential :200
##  Mean   :-0.2005   Mean   :-0.06049   Mean   :-0.1403
##  3rd Qu.: 0.8156   3rd Qu.:-0.07350   3rd Qu.: 0.3152
##  Max.   : 5.3201   Max.   : 4.07818   Max.   : 3.3179
##      price            latitude            longitude
##  Min.   :-1.5760   Min.   :-2.530429   Min.   :-1.34658
##  1st Qu.:-0.8300   1st Qu.:-0.786893   1st Qu.:-0.61637
##  Median :-0.3106   Median : 0.194442   Median :-0.11920
##  Mean   :-0.1120   Mean   :-0.003266   Mean   : 0.01282
##  3rd Qu.: 0.2982   3rd Qu.: 0.624872   3rd Qu.: 0.35110
##  Max.   : 3.4496   Max.   : 2.585159   Max.   : 5.39610
```

Rerun SVM

```
trainIndex2 <- createDataPartition(balanced_data$type, p = .8, list = FALSE)
trainData2 <- balanced_data[ trainIndex2,]
testData2  <- balanced_data[-trainIndex2,]

set.seed(123)
```

```r
svmFit3 <- train(type ~ .,
                 data = trainData2,
                 method = "svmRadial",
                 metric = "Kappa",
                 trControl = ctrl,
                 tuneGrid = svmGrid)

predictions3 <- predict(svmFit3, newdata = testData2)
confusionMatrix(predictions3, testData2$type)
```

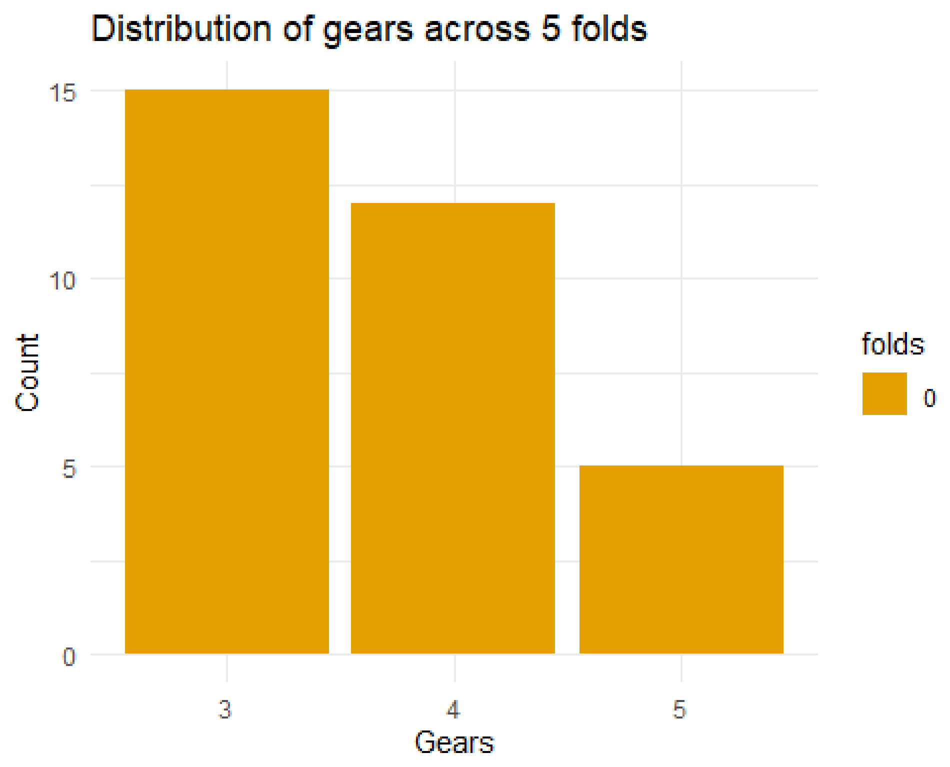
```
## Confusion Matrix and Statistics
##
##                Reference
## Prediction      Condo Multi_Family Residential
##   Condo             9            0           4
##   Multi_Family      0            1           0
##   Residential       1            1          36
##
## Overall Statistics
##
##                Accuracy : 0.8846
##                  95% CI : (0.7656, 0.9565)
##     No Information Rate : 0.7692
##     P-Value [Acc > NIR] : 0.02862
##
##                   Kappa : 0.7034
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Condo Class: Multi_Family Class: Residential
## Sensitivity                0.9000             0.50000             0.9000
## Specificity                0.9048             1.00000             0.8333
## Pos Pred Value             0.6923             1.00000             0.9474
## Neg Pred Value             0.9744             0.98039             0.7143
## Prevalence                 0.1923             0.03846             0.7692
## Detection Rate             0.1731             0.01923             0.6923
## Detection Prevalence       0.2500             0.01923             0.7308
## Balanced Accuracy          0.9024             0.75000             0.8667
```

PROBLEM SET 5 [BONUS]

```r
library(ggplot2)
mycars<-mtcars
mycars$folds = 0
```

Visualize the distribution of the gears variable across the folds:

```r
library(ggplot2)
ggplot(mycars, aes(x = as.factor(gear), fill = as.factor(folds))) +
  geom_bar(position = "dodge") +
  labs(title = "Distribution of gears across 5 folds",
       x = "Gears",
       y = "Count",
       fill = "folds") +
  theme_minimal() +
  scale_fill_manual(values = c("#E69F00", "#56B4E9", "#009E73",
"#F0E442", "#D55E00"))
```


Distribution of gears across 5 folds
Count
Gears
folds