FDS-2

Sadiya Amreen- 2079690

2022-10-06

## FUNDAMENTALS OF DATA SCIENCE – DSC 441

### HOMEWORK 2

### Problem 1 (25 points):

#### For this problem, you will load and perform some cleaning steps on a dataset in the provided BankData.csv, which is data about loan approvals from a bank in Japan (it has been modified from the original for our purposes in class, so use the provided version).

#### Specifically, you will use visualization to examine the variables and normalization, binning and smoothing to change them in particular ways.

##Code:  
  
# Getting the current directory  
print(getwd())

## [1] "C:/Sadiya Studies/Data Science/DS441-Fundamts DS/homework"

#Setting directory  
setwd("C:/Users/SADIYA/Downloads")  
#Reading Data from the .CSV file  
BnkDta <- read.csv("BankData.csv")  
  
#Loading Libraries  
library(tidyverse)

## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.2 ──  
## ✔ ggplot2 3.3.6 ✔ purrr 0.3.4   
## ✔ tibble 3.1.8 ✔ dplyr 1.0.10  
## ✔ tidyr 1.2.1 ✔ stringr 1.4.1   
## ✔ readr 2.1.2 ✔ forcats 0.5.2   
## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()

library(astsa)  
library(dplyr)  
library(ggplot2)  
library(caret)

## Loading required package: lattice  
##   
## Attaching package: 'caret'  
##   
## The following object is masked from 'package:purrr':  
##   
## lift

#calling the summary of the data\_file  
summary(BnkDta)

## X cont1 cont2 cont3   
## Min. : 1.0 Min. :13.75 Min. : 0.000 Min. : 0.000   
## 1st Qu.:173.2 1st Qu.:22.60 1st Qu.: 1.000 1st Qu.: 0.165   
## Median :345.5 Median :28.46 Median : 2.750 Median : 1.000   
## Mean :345.5 Mean :31.57 Mean : 4.759 Mean : 2.223   
## 3rd Qu.:517.8 3rd Qu.:38.23 3rd Qu.: 7.207 3rd Qu.: 2.625   
## Max. :690.0 Max. :80.25 Max. :28.000 Max. :28.500   
## NA's :12   
## bool1 bool2 cont4 bool3   
## Length:690 Length:690 Min. : 0.0 Length:690   
## Class :character Class :character 1st Qu.: 0.0 Class :character   
## Mode :character Mode :character Median : 0.0 Mode :character   
## Mean : 2.4   
## 3rd Qu.: 3.0   
## Max. :67.0   
##   
## cont5 cont6 approval credit.score   
## Min. : 0 Min. : 0.0 Length:690 Min. :583.7   
## 1st Qu.: 75 1st Qu.: 0.0 Class :character 1st Qu.:666.7   
## Median : 160 Median : 5.0 Mode :character Median :697.3   
## Mean : 184 Mean : 1017.4 Mean :696.4   
## 3rd Qu.: 276 3rd Qu.: 395.5 3rd Qu.:726.4   
## Max. :2000 Max. :100000.0 Max. :806.0   
## NA's :13   
## ages   
## Min. :11.00   
## 1st Qu.:31.00   
## Median :38.00   
## Mean :39.67   
## 3rd Qu.:48.00   
## Max. :84.00   
##

# Performing a check for NA values  
dim(BnkDta)

## [1] 690 13

colSums(is.na(BnkDta))

## X cont1 cont2 cont3 bool1 bool2   
## 0 12 0 0 0 0   
## cont4 bool3 cont5 cont6 approval credit.score   
## 0 0 13 0 0 0   
## ages   
## 0

### Inference:

Data Frame Dimensions (RxC) : 690 Rows and 13 columns.

Cont1 column has 12 NA values.

Cont5 column has 13 NA values

# Before removal  
summary(BnkDta)

## X cont1 cont2 cont3   
## Min. : 1.0 Min. :13.75 Min. : 0.000 Min. : 0.000   
## 1st Qu.:173.2 1st Qu.:22.60 1st Qu.: 1.000 1st Qu.: 0.165   
## Median :345.5 Median :28.46 Median : 2.750 Median : 1.000   
## Mean :345.5 Mean :31.57 Mean : 4.759 Mean : 2.223   
## 3rd Qu.:517.8 3rd Qu.:38.23 3rd Qu.: 7.207 3rd Qu.: 2.625   
## Max. :690.0 Max. :80.25 Max. :28.000 Max. :28.500   
## NA's :12   
## bool1 bool2 cont4 bool3   
## Length:690 Length:690 Min. : 0.0 Length:690   
## Class :character Class :character 1st Qu.: 0.0 Class :character   
## Mode :character Mode :character Median : 0.0 Mode :character   
## Mean : 2.4   
## 3rd Qu.: 3.0   
## Max. :67.0   
##   
## cont5 cont6 approval credit.score   
## Min. : 0 Min. : 0.0 Length:690 Min. :583.7   
## 1st Qu.: 75 1st Qu.: 0.0 Class :character 1st Qu.:666.7   
## Median : 160 Median : 5.0 Mode :character Median :697.3   
## Mean : 184 Mean : 1017.4 Mean :696.4   
## 3rd Qu.: 276 3rd Qu.: 395.5 3rd Qu.:726.4   
## Max. :2000 Max. :100000.0 Max. :806.0   
## NA's :13   
## ages   
## Min. :11.00   
## 1st Qu.:31.00   
## Median :38.00   
## Mean :39.67   
## 3rd Qu.:48.00   
## Max. :84.00   
##

#summary has NA values, hence we have to remove them  
BnkDta\_F<- BnkDta %>% drop\_na(cont1) %>% drop\_na(cont5)   
# After removal  
summary(BnkDta\_F)

## X cont1 cont2 cont3   
## Min. : 1.0 Min. :13.75 Min. : 0.000 Min. : 0.000   
## 1st Qu.:172.2 1st Qu.:22.60 1st Qu.: 1.010 1st Qu.: 0.165   
## Median :347.5 Median :28.50 Median : 2.750 Median : 1.000   
## Mean :345.8 Mean :31.57 Mean : 4.798 Mean : 2.222   
## 3rd Qu.:519.8 3rd Qu.:38.25 3rd Qu.: 7.207 3rd Qu.: 2.585   
## Max. :690.0 Max. :80.25 Max. :28.000 Max. :28.500   
## bool1 bool2 cont4 bool3   
## Length:666 Length:666 Min. : 0.000 Length:666   
## Class :character Class :character 1st Qu.: 0.000 Class :character   
## Mode :character Mode :character Median : 0.000 Mode :character   
## Mean : 2.459   
## 3rd Qu.: 3.000   
## Max. :67.000   
## cont5 cont6 approval credit.score   
## Min. : 0.00 Min. : 0.0 Length:666 Min. :585.1   
## 1st Qu.: 75.25 1st Qu.: 0.0 Class :character 1st Qu.:666.4   
## Median : 160.00 Median : 5.0 Mode :character Median :697.1   
## Mean : 182.12 Mean : 998.6 Mean :696.3   
## 3rd Qu.: 271.00 3rd Qu.: 399.0 3rd Qu.:726.4   
## Max. :2000.00 Max. :100000.0 Max. :806.0   
## ages   
## Min. :11.0   
## 1st Qu.:31.0   
## Median :38.0   
## Mean :39.7   
## 3rd Qu.:48.0   
## Max. :84.0

#### a. Visualize the distributions of the variables in this data. You can choose bar graphs, histograms and density plots.

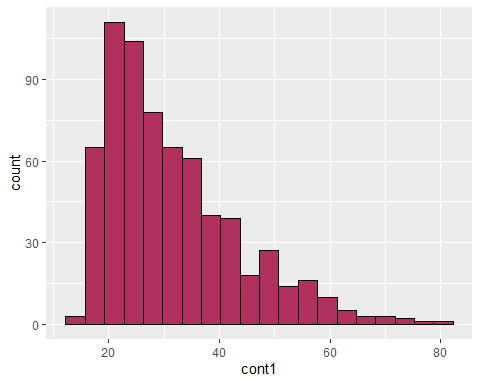
#### Make appropriate choices given each type of variables and be careful when selecting parameters like the number of bins for the histograms. Note there are some numerical variables and some categorical ones. The ones labeled as a ‘bool’ are Boolean variables, meaning they are only true or false and are thus a special type of categorical. Checking all the distributions with visualization and summary statistics is a typical step when beginning to work with new data.

### Visualization: Numerical data - cont1

summary(BnkDta\_F$cont1)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 13.75 22.60 28.50 31.57 38.25 80.25

ggplot (BnkDta\_F , aes (cont1)) + geom\_histogram( fill='maroon',color="black" , bins = 20)

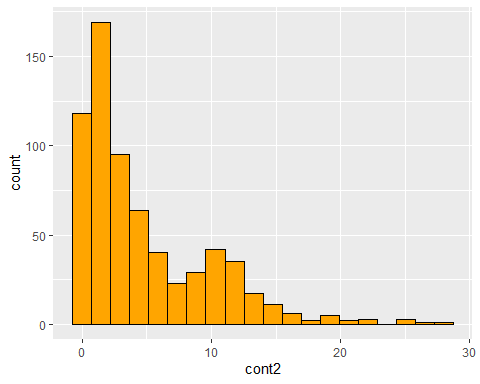


# Visualization: Numerical data - cont2

summary(BnkDta\_F$cont2)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.000 1.010 2.750 4.798 7.207 28.000

ggplot (BnkDta\_F , aes (cont2)) + geom\_histogram( fill='orange',color="black" , bins = 20)

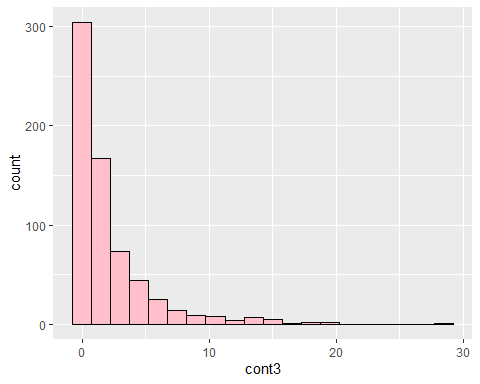


# Visualization: Numerical data - cont3

summary(BnkDta\_F$cont3)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.000 0.165 1.000 2.222 2.585 28.500

ggplot (BnkDta\_F , aes (cont3)) + geom\_histogram( fill='pink',color="black" , bins = 20)

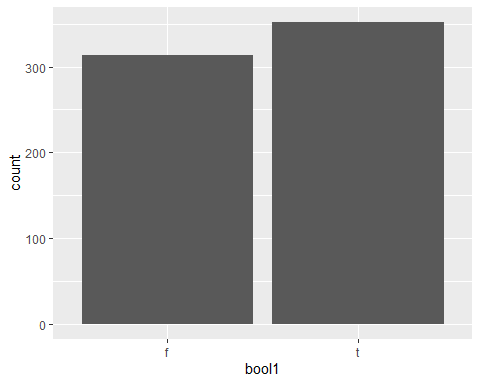


# Visualization: Categorical data : bool1 column

summary(BnkDta\_F$bool1)

## Length Class Mode   
## 666 character character

ggplot (BnkDta\_F , aes (bool1)) + geom\_bar()

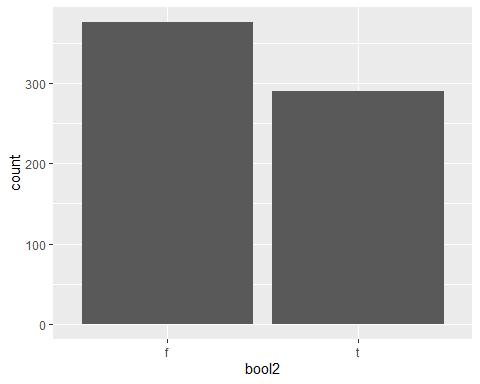


# Visualization: Categorical data : bool2 column

summary(BnkDta\_F$bool2)

## Length Class Mode   
## 666 character character

ggplot (BnkDta\_F , aes (bool2)) + geom\_bar()

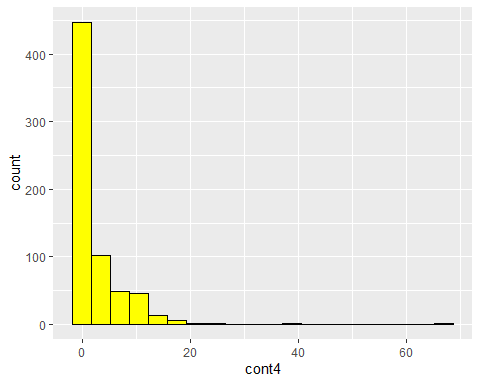


# Visualization: Numerical data - cont4

summary(BnkDta\_F$cont4)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.000 0.000 0.000 2.459 3.000 67.000

ggplot (BnkDta\_F , aes (cont4)) + geom\_histogram( fill='yellow',color="black" , bins = 20)

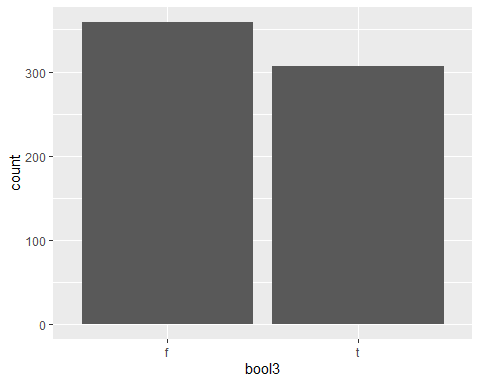


# Visualization: Categorical data : bool3 column

summary(BnkDta\_F$bool1)

## Length Class Mode   
## 666 character character

ggplot (BnkDta\_F , aes (bool3)) + geom\_bar()

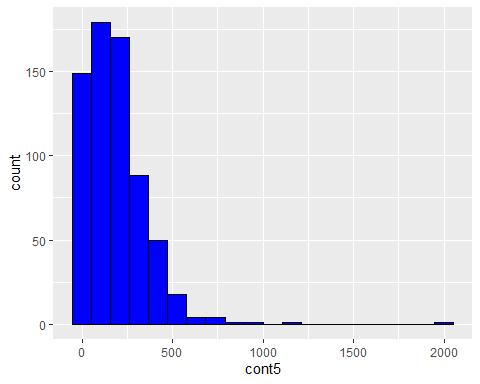


# Visualization: Numerical data - cont5

summary(BnkDta\_F$cont5)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.00 75.25 160.00 182.12 271.00 2000.00

ggplot (BnkDta\_F , aes (cont5)) + geom\_histogram( fill='blue',color="black" , bins = 20)

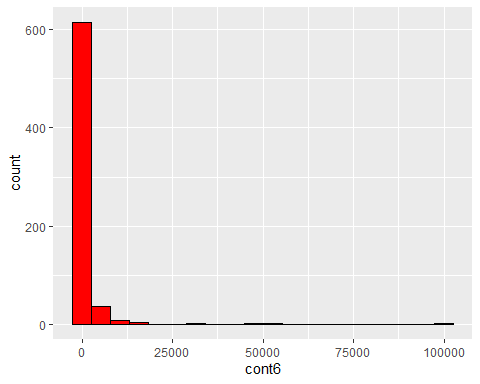


# Visualization: Numerical data - cont6

summary(BnkDta\_F$cont6)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.0 0.0 5.0 998.6 399.0 100000.0

ggplot (BnkDta\_F , aes (cont6)) + geom\_histogram( fill='red',color="black" , bins = 20)

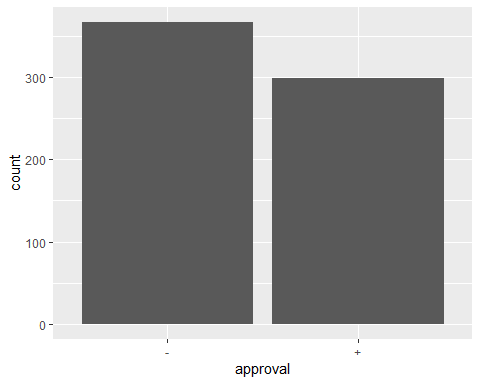


# Visualization: Categorical data : approval column

summary(BnkDta\_F$approval)

## Length Class Mode   
## 666 character character

ggplot (BnkDta\_F , aes (approval)) + geom\_bar()

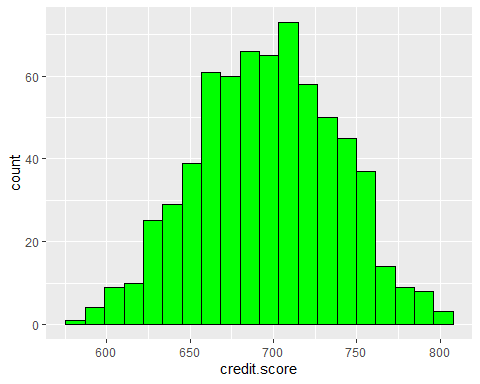


# Visualization: Numerical data - credit.score

summary(BnkDta\_F$credit.score)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 585.1 666.4 697.1 696.3 726.4 806.0

ggplot (BnkDta\_F , aes (credit.score)) + geom\_histogram( fill='green',color="black", bins = 20)

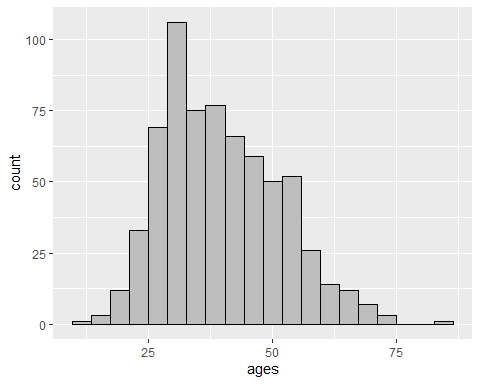


# Visualization: Numerical data - ages

summary(BnkDta\_F$ages)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 11.0 31.0 38.0 39.7 48.0 84.0

ggplot (BnkDta\_F , aes (ages)) + geom\_histogram( fill='grey',color="black" , bins = 20)



#### b. Now apply normalization to some of these numerical distributions. Specifically, choose to apply zscore to one, min-max to another, and decimal scaling to a third. Explain your choices of which normalization applies to which variable in terms of what the variable means, what distribution it starts with, and how the normalization will affect it.

#### min-max normalization on cont6

#### cont6 is a numeric column of the BAnkdata dataframe.It’s distribution starts with a min. of 0.00 to a max. of 100000.0. We only scale the data values between the ranges of 0 and 1 when using min-max scaling. As a result, the impact of outliers on the data values is somewhat suppressed. Additionally, it aids in lowering the data scale’s standard deviation.

install.packages(“caret”) library(caret)

#summary before normalization  
summary(BnkDta\_F$cont6)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.0 0.0 5.0 998.6 399.0 100000.0

#applying normalization  
file\_mm <- BnkDta\_F[c(10)]  
preproc\_mm <- preProcess(file\_mm, method=c("range"))  
norm\_mm <- predict(preproc\_mm, file\_mm)  
# the values range [0-1]  
#summary after normalization  
summary(norm\_mm)

## cont6   
## Min. :0.000000   
## 1st Qu.:0.000000   
## Median :0.000050   
## Mean :0.009986   
## 3rd Qu.:0.003990   
## Max. :1.000000

## z-score normalization on cont5

#### cont5 is a numeric column of the BAnkdata dataframe.It’s distribution starts with a min. of 0.00 to a max. of 2000.00 . Z-score By deducting each feature value from the sample mean and then dividing by the sample standard deviation, normalization changes x to x’. The standardised values’ final mean and standard deviation are, respectively, 0 and 1. This is done to ensure internal consistency of the data.

#summary before normalization  
summary(BnkDta\_F$cont5)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.00 75.25 160.00 182.12 271.00 2000.00

SD= sd(BnkDta\_F$cont5)  
SD

## [1] 171.4779

z\_normfile <- BnkDta\_F[c(9)]  
m\_z\_normfile<- mean(z\_normfile$cont5)  
sd\_z\_normfile <- sd(z\_normfile$cont5)  
final\_z <- (z\_normfile-m\_z\_normfile)/sd\_z\_normfile  
#summary after normalization  
summary(final\_z)

## cont5   
## Min. :-1.0620   
## 1st Qu.:-0.6232   
## Median :-0.1290   
## Mean : 0.0000   
## 3rd Qu.: 0.5183   
## Max. :10.6013

sd(final\_z$cont5)

## [1] 1

## decimal scaling normalization on ‘ages’

### ‘ages’ is a numeric column of the BAnkdata dataframe. It’s distribution starts with a min. of 11.00 to a max. of 84.00 . Decimal Scaling- Finding k such that the absolute value of the greatest value of each attribute divided by 10^k is = 1, causes decimal scaling to transform the data into [-1,1].

#summary before normalization  
summary(BnkDta\_F$ages)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 11.0 31.0 38.0 39.7 48.0 84.0

F\_dec1 <- BnkDta\_F[c(13)]  
norm\_dec1 <- F\_dec1/100  
#summary after normalization  
summary(norm\_dec1)

## ages   
## Min. :0.110   
## 1st Qu.:0.310   
## Median :0.380   
## Mean :0.397   
## 3rd Qu.:0.480   
## Max. :0.840

#### c. Visualize the new distributions for the variables that have been normalized. What has changed from the previous visualization?

# Normalization before and after decimal Scaling

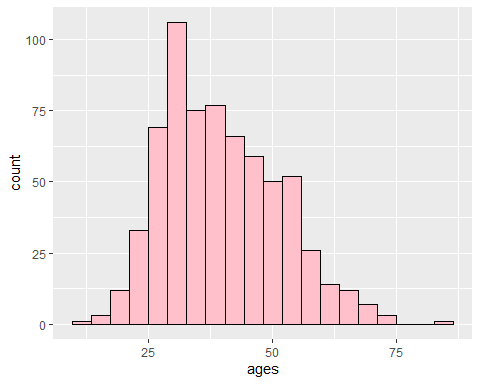
Before -decimal Scaling

Visualization: Numerical data – ages

summary(BnkDta\_F$ages)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 11.0 31.0 38.0 39.7 48.0 84.0

ggplot(BnkDta\_F ,aes (x = ages)) + geom\_histogram( fill='pink',color="black" , bins=20)



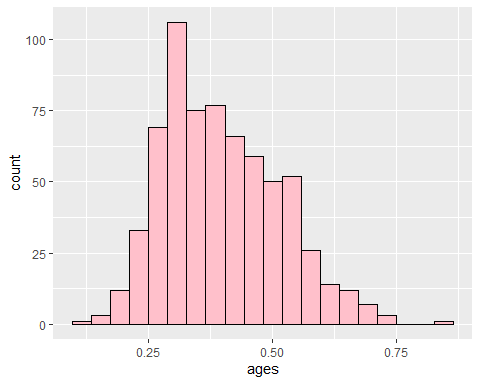
**After- decimal Scaling :**

**we observe that after decimal scaling the data is scaled in the range of [-1,1].**

#After  
library(ggplot2)  
summary(norm\_dec1)

## ages   
## Min. :0.110   
## 1st Qu.:0.310   
## Median :0.380   
## Mean :0.397   
## 3rd Qu.:0.480   
## Max. :0.840

ggplot( norm\_dec1, aes(ages)) + geom\_histogram( fill='pink',color="black" , bins=20)



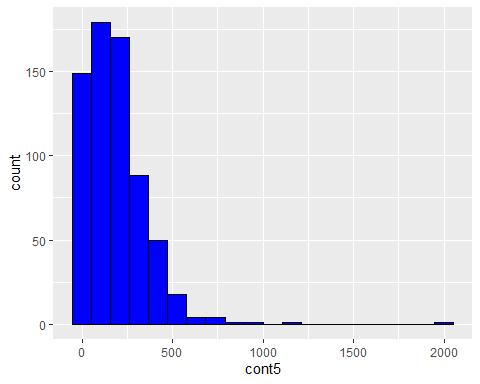
**Before- z-score:**

**Visualization: Numerical data - cont5**

summary(BnkDta\_F$cont5)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.00 75.25 160.00 182.12 271.00 2000.00

ggplot (BnkDta\_F , aes (cont5)) + geom\_histogram( fill='blue',color="black" ,bins=20)



**After- z-score**

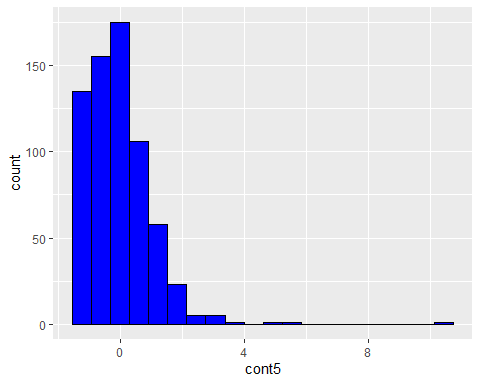
**Visualization: Numerical data - cont5**

**The standardised values of the data the final mean and standard deviation are, respectively, 0 and 1**

summary(final\_z)

## cont5   
## Min. :-1.0620   
## 1st Qu.:-0.6232   
## Median :-0.1290   
## Mean : 0.0000   
## 3rd Qu.: 0.5183   
## Max. :10.6013

ggplot (final\_z , aes (cont5)) + geom\_histogram( fill='blue',color="black" , bins=20)



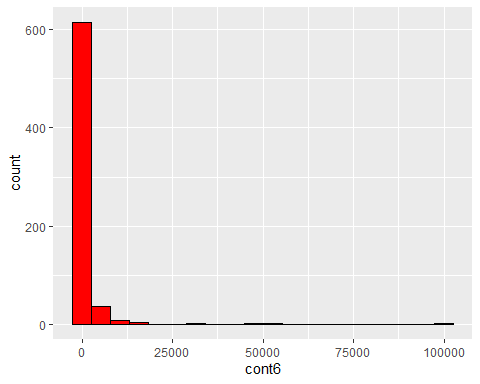
**Before - max-min**

**Visualization: Numerical data - cont6**

summary(BnkDta\_F$cont6)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.0 0.0 5.0 998.6 399.0 100000.0

ggplot (BnkDta\_F , aes (cont6)) + geom\_histogram( fill='red',color="black" , bins=20)



**After - max-min**

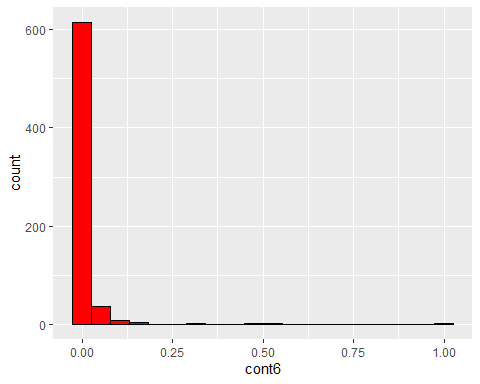
**Visualization: Numerical data - cont6**

**By using min-max scaling, the data values are scaled between the ranges of 0 and 1 only.**

summary(norm\_mm)

## cont6   
## Min. :0.000000   
## 1st Qu.:0.000000   
## Median :0.000050   
## Mean :0.009986   
## 3rd Qu.:0.003990   
## Max. :1.000000

ggplot (norm\_mm , aes (cont6)) + geom\_histogram( fill='red',color="black" , bins=20)



### d. Choose one of the numerical variables to work with for this problem. Let’s call it v. Create a new variable called v\_bins that is a binned version of that variable. This v\_bins will have a new set of values like low, medium, high. Choose the actual new values (you don’t need to use low, medium, high) and the ranges of v that they represent based on your understanding of v from your visualizations. You can use equal depth, equal width or custom ranges. Explain your choices: why did you choose to create that number of values and those particular ranges?

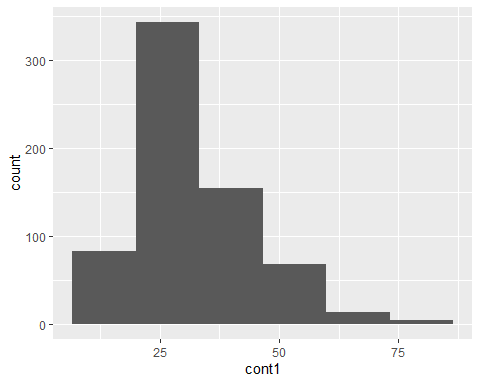
v <- BnkDta\_F[c(2)]  
v %>% mutate(v\_bins = cut(cont1,  
 breaks = c(13.75,35.91,58.01,80.25),  
 labels=c("low","medium","high")))

## cont1 v\_bins  
## 1 30.83 low  
## 2 58.67 high  
## 3 24.50 low  
## 4 27.83 low  
## 5 20.17 low  
## 6 32.08 low  
## 7 33.17 low  
## 8 22.92 low  
## 9 54.42 medium  
## 10 42.50 medium  
## 11 22.08 low  
## 12 29.92 low  
## 13 38.25 medium  
## 14 48.08 medium  
## 15 45.83 medium  
## 16 36.67 medium  
## 17 28.25 low  
## 18 23.25 low  
## 19 21.83 low  
## 20 19.17 low  
## 21 25.00 low  
## 22 23.25 low  
## 23 47.75 medium  
## 24 27.42 low  
## 25 41.17 medium  
## 26 15.83 low  
## 27 47.00 medium  
## 28 56.58 medium  
## 29 57.42 medium  
## 30 42.08 medium  
## 31 29.25 low  
## 32 42.00 medium  
## 33 49.50 medium  
## 34 36.75 medium  
## 35 22.58 low  
## 36 27.83 low  
## 37 27.25 low  
## 38 23.00 low  
## 39 27.75 low  
## 40 54.58 medium  
## 41 34.17 low  
## 42 28.92 low  
## 43 29.67 low  
## 44 39.58 medium  
## 45 56.42 medium  
## 46 54.33 medium  
## 47 41.00 medium  
## 48 31.92 low  
## 49 41.50 medium  
## 50 23.92 low  
## 51 25.75 low  
## 52 26.00 low  
## 53 37.42 medium  
## 54 34.92 low  
## 55 34.25 low  
## 56 23.33 low  
## 57 23.17 low  
## 58 44.33 medium  
## 59 35.17 low  
## 60 43.25 medium  
## 61 56.75 medium  
## 62 31.67 low  
## 63 23.42 low  
## 64 20.42 low  
## 65 26.67 low  
## 66 34.17 low  
## 67 36.00 medium  
## 68 25.50 low  
## 69 19.42 low  
## 70 35.17 low  
## 71 32.33 low  
## 72 38.58 medium  
## 73 44.25 medium  
## 74 44.83 medium  
## 75 20.67 low  
## 76 34.08 low  
## 77 19.17 low  
## 78 21.67 low  
## 79 21.50 low  
## 80 49.58 medium  
## 81 27.67 low  
## 82 39.83 medium  
## 83 27.25 low  
## 84 37.17 medium  
## 85 25.67 low  
## 86 34.00 low  
## 87 49.00 medium  
## 88 62.50 high  
## 89 31.42 low  
## 90 52.33 medium  
## 91 28.75 low  
## 92 28.58 low  
## 93 23.00 low  
## 94 22.50 low  
## 95 28.50 low  
## 96 37.50 medium  
## 97 35.25 low  
## 98 18.67 low  
## 99 25.00 low  
## 100 27.83 low  
## 101 54.83 medium  
## 102 28.75 low  
## 103 25.00 low  
## 104 40.92 medium  
## 105 19.75 low  
## 106 29.17 low  
## 107 24.50 low  
## 108 24.58 low  
## 109 33.75 low  
## 110 20.67 low  
## 111 25.42 low  
## 112 37.75 medium  
## 113 52.50 medium  
## 114 57.83 medium  
## 115 20.75 low  
## 116 39.92 medium  
## 117 25.67 low  
## 118 24.75 low  
## 119 44.17 medium  
## 120 23.50 low  
## 121 34.92 low  
## 122 47.67 medium  
## 123 22.75 low  
## 124 34.42 low  
## 125 28.42 low  
## 126 67.75 high  
## 127 20.42 low  
## 128 47.42 medium  
## 129 36.25 medium  
## 130 32.67 low  
## 131 48.58 medium  
## 132 39.92 medium  
## 133 33.58 low  
## 134 18.83 low  
## 135 26.92 low  
## 136 31.25 low  
## 137 56.50 medium  
## 138 43.00 medium  
## 139 22.33 low  
## 140 27.25 low  
## 141 32.83 low  
## 142 23.25 low  
## 143 40.33 medium  
## 144 30.50 low  
## 145 52.83 medium  
## 146 46.67 medium  
## 147 58.33 high  
## 148 37.33 medium  
## 149 23.08 low  
## 150 32.75 low  
## 151 21.67 low  
## 152 28.50 low  
## 153 68.67 high  
## 154 28.00 low  
## 155 34.08 low  
## 156 27.67 low  
## 157 44.00 medium  
## 158 25.08 low  
## 159 32.00 low  
## 160 60.58 high  
## 161 40.83 medium  
## 162 19.33 low  
## 163 32.33 low  
## 164 36.67 medium  
## 165 37.50 medium  
## 166 25.08 low  
## 167 41.33 medium  
## 168 56.00 medium  
## 169 49.83 medium  
## 170 22.67 low  
## 171 27.00 low  
## 172 25.00 low  
## 173 26.08 low  
## 174 18.42 low  
## 175 20.17 low  
## 176 47.67 medium  
## 177 21.25 low  
## 178 20.67 low  
## 179 57.08 medium  
## 180 22.42 low  
## 181 48.75 medium  
## 182 40.00 medium  
## 183 40.58 medium  
## 184 28.67 low  
## 185 33.08 low  
## 186 21.33 low  
## 187 42.00 medium  
## 188 41.75 medium  
## 189 22.67 low  
## 190 34.50 low  
## 191 28.25 low  
## 192 33.17 low  
## 193 48.17 medium  
## 194 27.58 low  
## 195 22.58 low  
## 196 24.08 low  
## 197 41.33 medium  
## 198 20.75 low  
## 199 36.33 medium  
## 200 35.42 low  
## 201 28.67 low  
## 202 35.17 low  
## 203 39.50 medium  
## 204 39.33 medium  
## 205 24.33 low  
## 206 60.08 high  
## 207 23.08 low  
## 208 26.67 low  
## 209 48.17 medium  
## 210 41.17 medium  
## 211 55.92 medium  
## 212 53.92 medium  
## 213 18.92 low  
## 214 50.08 medium  
## 215 65.42 high  
## 216 17.58 low  
## 217 18.83 low  
## 218 37.75 medium  
## 219 23.25 low  
## 220 18.08 low  
## 221 22.50 low  
## 222 19.67 low  
## 223 22.08 low  
## 224 25.17 low  
## 225 47.42 medium  
## 226 33.50 low  
## 227 27.67 low  
## 228 58.42 high  
## 229 20.67 low  
## 230 26.17 low  
## 231 21.33 low  
## 232 42.83 medium  
## 233 38.17 medium  
## 234 20.50 low  
## 235 48.25 medium  
## 236 28.33 low  
## 237 18.50 low  
## 238 33.17 low  
## 239 45.00 medium  
## 240 19.67 low  
## 241 24.50 low  
## 242 21.83 low  
## 243 40.25 medium  
## 244 41.42 medium  
## 245 17.83 low  
## 246 23.17 low  
## 247 18.17 low  
## 248 20.00 low  
## 249 20.00 low  
## 250 20.75 low  
## 251 24.50 low  
## 252 32.75 low  
## 253 52.17 medium  
## 254 48.17 medium  
## 255 20.42 low  
## 256 50.75 medium  
## 257 17.08 low  
## 258 18.33 low  
## 259 32.00 low  
## 260 59.67 high  
## 261 18.00 low  
## 262 32.33 low  
## 263 18.08 low  
## 264 38.25 medium  
## 265 30.67 low  
## 266 18.58 low  
## 267 19.17 low  
## 268 18.17 low  
## 269 16.25 low  
## 270 21.17 low  
## 271 23.92 low  
## 272 17.67 low  
## 273 16.50 low  
## 274 23.25 low  
## 275 17.58 low  
## 276 29.50 low  
## 277 18.83 low  
## 278 21.75 low  
## 279 23.00 low  
## 280 18.25 low  
## 281 25.42 low  
## 282 35.75 low  
## 283 16.08 low  
## 284 31.92 low  
## 285 69.17 high  
## 286 32.92 low  
## 287 16.33 low  
## 288 22.17 low  
## 289 57.58 medium  
## 290 18.25 low  
## 291 23.42 low  
## 292 15.92 low  
## 293 24.75 low  
## 294 48.75 medium  
## 295 23.50 low  
## 296 18.58 low  
## 297 27.75 low  
## 298 31.75 low  
## 299 24.83 low  
## 300 19.00 low  
## 301 16.33 low  
## 302 18.58 low  
## 303 16.25 low  
## 304 23.00 low  
## 305 21.17 low  
## 306 17.50 low  
## 307 19.17 low  
## 308 36.75 medium  
## 309 21.25 low  
## 310 18.08 low  
## 311 33.67 low  
## 312 48.58 medium  
## 313 33.67 low  
## 314 29.50 low  
## 315 30.17 low  
## 316 40.83 medium  
## 317 34.83 low  
## 318 33.25 low  
## 319 34.08 low  
## 320 25.25 low  
## 321 34.75 low  
## 322 27.67 low  
## 323 47.33 medium  
## 324 34.83 low  
## 325 33.25 low  
## 326 28.00 low  
## 327 39.08 medium  
## 328 42.75 medium  
## 329 26.92 low  
## 330 33.75 low  
## 331 38.92 medium  
## 332 62.75 high  
## 333 32.25 low  
## 334 26.75 low  
## 335 63.33 high  
## 336 27.83 low  
## 337 26.17 low  
## 338 22.17 low  
## 339 22.50 low  
## 340 30.75 low  
## 341 36.67 medium  
## 342 16.00 low  
## 343 41.17 medium  
## 344 19.50 low  
## 345 32.42 low  
## 346 36.75 medium  
## 347 30.25 low  
## 348 23.08 low  
## 349 26.83 low  
## 350 16.92 low  
## 351 24.42 low  
## 352 42.83 medium  
## 353 22.75 low  
## 354 39.42 medium  
## 355 23.58 low  
## 356 21.42 low  
## 357 33.00 low  
## 358 26.33 low  
## 359 45.00 medium  
## 360 26.25 low  
## 361 28.17 low  
## 362 20.83 low  
## 363 28.67 low  
## 364 20.67 low  
## 365 34.42 low  
## 366 33.58 low  
## 367 43.17 medium  
## 368 22.67 low  
## 369 24.33 low  
## 370 56.83 medium  
## 371 22.08 low  
## 372 34.00 low  
## 373 22.58 low  
## 374 21.17 low  
## 375 26.67 low  
## 376 22.92 low  
## 377 15.17 low  
## 378 39.92 medium  
## 379 27.42 low  
## 380 24.75 low  
## 381 41.17 medium  
## 382 33.08 low  
## 383 29.83 low  
## 384 23.58 low  
## 385 26.17 low  
## 386 31.00 low  
## 387 20.75 low  
## 388 28.92 low  
## 389 51.92 medium  
## 390 22.67 low  
## 391 34.00 low  
## 392 69.50 high  
## 393 19.58 low  
## 394 16.00 low  
## 395 17.08 low  
## 396 31.25 low  
## 397 25.17 low  
## 398 22.67 low  
## 399 40.58 medium  
## 400 22.25 low  
## 401 22.25 low  
## 402 22.50 low  
## 403 23.58 low  
## 404 38.42 medium  
## 405 26.58 low  
## 406 35.00 low  
## 407 20.42 low  
## 408 29.42 low  
## 409 26.17 low  
## 410 33.67 low  
## 411 24.58 low  
## 412 27.67 low  
## 413 37.50 medium  
## 414 49.17 medium  
## 415 33.58 low  
## 416 51.83 medium  
## 417 22.92 low  
## 418 21.83 low  
## 419 25.25 low  
## 420 58.58 high  
## 421 19.00 low  
## 422 19.58 low  
## 423 53.33 medium  
## 424 27.17 low  
## 425 25.92 low  
## 426 23.08 low  
## 427 39.58 medium  
## 428 30.58 low  
## 429 17.25 low  
## 430 17.67 low  
## 431 16.50 low  
## 432 27.33 low  
## 433 31.25 low  
## 434 20.00 low  
## 435 39.50 medium  
## 436 36.50 medium  
## 437 29.75 low  
## 438 52.42 medium  
## 439 36.17 medium  
## 440 29.67 low  
## 441 36.17 medium  
## 442 25.67 low  
## 443 24.50 low  
## 444 24.08 low  
## 445 21.92 low  
## 446 36.58 medium  
## 447 23.00 low  
## 448 27.58 low  
## 449 31.08 low  
## 450 30.42 low  
## 451 22.08 low  
## 452 16.33 low  
## 453 21.92 low  
## 454 21.08 low  
## 455 17.42 low  
## 456 19.17 low  
## 457 20.67 low  
## 458 26.75 low  
## 459 23.58 low  
## 460 39.17 medium  
## 461 22.75 low  
## 462 26.50 low  
## 463 16.92 low  
## 464 23.50 low  
## 465 17.33 low  
## 466 23.75 low  
## 467 34.67 low  
## 468 74.83 high  
## 469 28.17 low  
## 470 24.50 low  
## 471 18.83 low  
## 472 45.33 medium  
## 473 47.25 medium  
## 474 24.17 low  
## 475 39.25 medium  
## 476 20.50 low  
## 477 18.83 low  
## 478 19.17 low  
## 479 25.00 low  
## 480 20.17 low  
## 481 25.75 low  
## 482 20.42 low  
## 483 39.00 medium  
## 484 64.08 high  
## 485 28.25 low  
## 486 28.75 low  
## 487 31.33 low  
## 488 18.92 low  
## 489 24.75 low  
## 490 30.67 low  
## 491 21.00 low  
## 492 13.75 <NA>  
## 493 46.00 medium  
## 494 44.33 medium  
## 495 20.25 low  
## 496 22.67 low  
## 497 60.92 high  
## 498 16.08 low  
## 499 28.17 low  
## 500 39.17 medium  
## 501 20.42 low  
## 502 30.00 low  
## 503 22.83 low  
## 504 22.50 low  
## 505 28.58 low  
## 506 45.17 medium  
## 507 41.58 medium  
## 508 57.08 medium  
## 509 55.75 medium  
## 510 43.25 medium  
## 511 25.33 low  
## 512 24.58 low  
## 513 43.17 medium  
## 514 40.92 medium  
## 515 31.83 low  
## 516 33.92 low  
## 517 24.92 low  
## 518 35.25 low  
## 519 34.25 low  
## 520 80.25 high  
## 521 19.42 low  
## 522 42.75 medium  
## 523 19.67 low  
## 524 36.33 medium  
## 525 30.08 low  
## 526 44.25 medium  
## 527 23.58 low  
## 528 23.92 low  
## 529 33.17 low  
## 530 48.33 medium  
## 531 76.75 high  
## 532 51.33 medium  
## 533 34.75 low  
## 534 38.58 medium  
## 535 22.42 low  
## 536 41.92 medium  
## 537 29.58 low  
## 538 32.17 low  
## 539 51.42 medium  
## 540 22.83 low  
## 541 25.00 low  
## 542 26.75 low  
## 543 23.33 low  
## 544 24.42 low  
## 545 42.17 medium  
## 546 20.83 low  
## 547 23.08 low  
## 548 25.17 low  
## 549 43.08 medium  
## 550 35.75 low  
## 551 59.50 high  
## 552 21.00 low  
## 553 21.92 low  
## 554 65.17 high  
## 555 20.33 low  
## 556 32.25 low  
## 557 30.17 low  
## 558 25.17 low  
## 559 39.17 medium  
## 560 39.08 medium  
## 561 31.67 low  
## 562 41.00 medium  
## 563 48.50 medium  
## 564 32.67 low  
## 565 28.08 low  
## 566 73.42 high  
## 567 64.08 high  
## 568 51.58 medium  
## 569 26.67 low  
## 570 25.33 low  
## 571 30.17 low  
## 572 27.00 low  
## 573 34.17 low  
## 574 38.67 medium  
## 575 25.75 low  
## 576 46.08 medium  
## 577 21.50 low  
## 578 20.08 low  
## 579 20.50 low  
## 580 29.50 low  
## 581 42.25 medium  
## 582 29.83 low  
## 583 20.08 low  
## 584 23.42 low  
## 585 29.58 low  
## 586 16.17 low  
## 587 32.33 low  
## 588 47.83 medium  
## 589 20.00 low  
## 590 27.58 low  
## 591 22.00 low  
## 592 19.33 low  
## 593 38.33 medium  
## 594 29.42 low  
## 595 22.67 low  
## 596 32.25 low  
## 597 29.58 low  
## 598 18.42 low  
## 599 22.17 low  
## 600 22.67 low  
## 601 18.83 low  
## 602 21.58 low  
## 603 23.75 low  
## 604 36.08 medium  
## 605 29.25 low  
## 606 19.58 low  
## 607 22.92 low  
## 608 27.25 low  
## 609 38.75 medium  
## 610 32.42 low  
## 611 23.75 low  
## 612 18.17 low  
## 613 40.92 medium  
## 614 19.50 low  
## 615 28.58 low  
## 616 35.58 low  
## 617 34.17 low  
## 618 33.17 low  
## 619 31.58 low  
## 620 52.50 medium  
## 621 36.17 medium  
## 622 37.33 medium  
## 623 20.83 low  
## 624 24.08 low  
## 625 25.58 low  
## 626 35.17 low  
## 627 48.08 medium  
## 628 15.83 low  
## 629 22.50 low  
## 630 21.50 low  
## 631 23.58 low  
## 632 21.08 low  
## 633 25.67 low  
## 634 38.92 medium  
## 635 15.75 low  
## 636 28.58 low  
## 637 22.25 low  
## 638 29.83 low  
## 639 23.50 low  
## 640 32.08 low  
## 641 31.08 low  
## 642 31.83 low  
## 643 21.75 low  
## 644 17.92 low  
## 645 30.33 low  
## 646 51.83 medium  
## 647 47.17 medium  
## 648 25.83 low  
## 649 50.25 medium  
## 650 29.50 low  
## 651 37.33 medium  
## 652 41.58 medium  
## 653 30.58 low  
## 654 19.42 low  
## 655 17.92 low  
## 656 20.08 low  
## 657 19.50 low  
## 658 27.83 low  
## 659 17.08 low  
## 660 36.42 medium  
## 661 40.58 medium  
## 662 21.08 low  
## 663 22.67 low  
## 664 25.25 low  
## 665 17.92 low  
## 666 35.00 low

head(15)

## [1] 15

bin\_dplot <- ggplot(v,aes(x=cont1)) + geom\_histogram(bins = 6)  
bin\_dplot



### Inference:

#### for variable cont1, the minimum is 13.75 and maximum is 80.25. Moreover, difference of variable is 66.5. For this variable, using equi-width method to generate the range. Here, the bins are equally divided in 3 and the labels are low ,medium and high. So, now dividing the difference by 3 and then adding it to our minimum value. Therefore , for “Low” label range startes from 13.75 to 35.91. , “Medium” range starts from 35.91 to 58.01, “High” range starts from 58.01 to 80.25.

### e. Building on (d), use v\_bins to create a smoothed version of v. Choose a smoothing strategy to create a numerical version of the binned variable and explain your choices.

v1 <- BnkDta\_F[c(2)]  
v1 %>%  
 mutate(v\_bins\_smoothed = cut(cont1,  
 breaks = 3 ,  
 labels=c("25.21","44.26","64.44"))) %>%  
 head(15)

## cont1 v\_bins\_smoothed  
## 1 30.83 25.21  
## 2 58.67 64.44  
## 3 24.50 25.21  
## 4 27.83 25.21  
## 5 20.17 25.21  
## 6 32.08 25.21  
## 7 33.17 25.21  
## 8 22.92 25.21  
## 9 54.42 44.26  
## 10 42.50 44.26  
## 11 22.08 25.21  
## 12 29.92 25.21  
## 13 38.25 44.26  
## 14 48.08 44.26  
## 15 45.83 44.26

### For numeric binned version, will again use equi-width for bins and dividing in equal 3 bins and for the labeling using numeric labels using mean of the ranges from previous question the low, medium and high label.

### Problem 2 (15 points):

### This is the first homework problem using machine learning algorithms. You will perform a straightforward training and evaluation of a support vector machine on the bank data from Problem 1. Start with a fresh copy, but be sure to remove rows with missing values first.

## Apply SVM to the data from Problem 1 to predict approval and report the accuracy using 10-fold cross validation.

#taken the na omitted data set.

folds <- 10  
train\_control = trainControl(method = "cv", number = 10)#train\_control\_for\_universe  
train\_control\_cv <- trainControl(method = "cv" , number = 10)#train\_control\_for\_cross-validation  
svm\_cv <- train(approval ~., data = BnkDta\_F, method = "svmLinear", trControl = train\_control\_cv)  
svm\_cv

## 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, 601, 599, 599, 599, ...   
## Resampling results:  
##   
## Accuracy Kappa   
## 0.863269 0.7284136  
##   
## Tuning parameter 'C' was held constant at a value of 1

### Accuracy for predicting approval is 0.863269.

### 2.B) Next, use the grid search functionality when training to optimize the C parameter of the SVM. What parameter was chosen and what is the accuracy?

grid <- expand.grid(C = 10^seq(-5,2,0.5))#unsure seq numbers for the gird function  
svmgrid <- train(approval~., data = BnkDta\_F, method = "svmLinear", trControl = train\_control, tuneGrid = grid)  
svmgrid

## Support Vector Machines with Linear Kernel   
##   
## 666 samples  
## 12 predictor  
## 2 classes: '-', '+'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 600, 600, 600, 599, 600, 599, ...   
## Resampling results across tuning parameters:  
##   
## C Accuracy Kappa   
## 1.000000e-05 0.5510403 0.0000000  
## 3.162278e-05 0.5510403 0.0000000  
## 1.000000e-04 0.5510403 0.0000000  
## 3.162278e-04 0.5734735 0.0548418  
## 1.000000e-03 0.8377657 0.6697807  
## 3.162278e-03 0.8632972 0.7281373  
## 1.000000e-02 0.8648123 0.7312693  
## 3.162278e-02 0.8632972 0.7281373  
## 1.000000e-01 0.8632972 0.7281373  
## 3.162278e-01 0.8632972 0.7281373  
## 1.000000e+00 0.8632972 0.7281373  
## 3.162278e+00 0.8632972 0.7281373  
## 1.000000e+01 0.8632972 0.7281373  
## 3.162278e+01 0.8632972 0.7281373  
## 1.000000e+02 0.8632972 0.7281373

## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was C = 0.01.

The parameter C is 1.000000e+02

accuracy is 0.8632972

and

Kappa is 0.7281373

2.C) sometimes even if the grid of parameters in (b) includes the default value of C = 1 (used in (a)), the accuracy result will be different for this value of C. What could make that different?

Inference:

The C parameter tells SVM optimization how much you want to reduce the number of times each training example is erroneously categorized. If a certain hyperplane performs a better job of accurately categorizing all of the training points, the optimization will choose that hyperplane even if it has a narrower margin for error when applied to high values of C. If C is very low, the optimizer will look for a separating hyperplane with a wider margin, even if this results in a greater number of points being wrongly categorized by the hyperplane. You should obtain misclassified instances for very tiny values of C, and this should happen frequently even if your training data is linearly seperable.

### problem 3.

### A)Several variables are categorical. We will use dummy variables to make it possible for SVM to use these. Leave the gender category out of the dummy variable conversion to use as a categorical for prediction. Show the resulting head.

head(starwars)

## # A tibble: 6 × 14  
## name height mass hair\_…¹ skin\_…² eye\_c…³ birth…⁴ sex gender homew…⁵  
## <chr> <int> <dbl> <chr> <chr> <chr> <dbl> <chr> <chr> <chr>   
## 1 Luke Skywal… 172 77 blond fair blue 19 male mascu… Tatooi…  
## 2 C-3PO 167 75 <NA> gold yellow 112 none mascu… Tatooi…  
## 3 R2-D2 96 32 <NA> white,… red 33 none mascu… Naboo   
## 4 Darth Vader 202 136 none white yellow 41.9 male mascu… Tatooi…  
## 5 Leia Organa 150 49 brown light brown 19 fema… femin… Aldera…  
## 6 Owen Lars 178 120 brown,… light blue 52 male mascu… Tatooi…  
## # … with 4 more variables: species <chr>, films <list>, vehicles <list>,  
## # starships <list>, and abbreviated variable names ¹​hair\_color, ²​skin\_color,  
## # ³​eye\_color, ⁴​birth\_year, ⁵​homeworld

library(tidyverse)  
library(stats)  
library(datasets)  
library(caret)  
starwars <- as.data.frame(dplyr::starwars)  
starwars <- starwars %>% select(-c("name","films","vehicles","starships"))  
starwars <- na.omit(starwars)  
summary(starwars)

## height mass hair\_color skin\_color   
## Min. : 88 Min. : 20.00 Length:29 Length:29   
## 1st Qu.:170 1st Qu.: 75.00 Class :character Class :character   
## Median :180 Median : 79.00 Mode :character Mode :character   
## Mean :178 Mean : 77.77   
## 3rd Qu.:188 3rd Qu.: 83.00   
## Max. :228 Max. :136.00   
## eye\_color birth\_year sex gender   
## Length:29 Min. : 8.00 Length:29 Length:29   
## Class :character 1st Qu.: 31.00 Class :character Class :character   
## Mode :character Median : 46.00 Mode :character Mode :character   
## Mean : 51.29   
## 3rd Qu.: 57.00   
## Max. :200.00   
## homeworld species   
## Length:29 Length:29   
## Class :character Class :character   
## Mode :character Mode :character   
##   
##   
##

dummydata <- dummyVars(gender ~ ., data=starwars)  
dummies\_data<- as.data.frame(predict(dummydata, newdata = starwars))  
dummies\_data$gender <- starwars$gender  
head(dummies\_data)

## height mass hair\_colorauburn, white hair\_colorblack hair\_colorblond  
## 1 172 77 0 0 1  
## 4 202 136 0 0 0  
## 5 150 49 0 0 0  
## 6 178 120 0 0 0  
## 7 165 75 0 0 0  
## 9 183 84 0 1 0  
## hair\_colorbrown hair\_colorbrown, grey hair\_colorgrey hair\_colornone  
## 1 0 0 0 0  
## 4 0 0 0 1  
## 5 1 0 0 0  
## 6 0 1 0 0  
## 7 1 0 0 0  
## 9 0 0 0 0  
## hair\_colorwhite skin\_colorblue skin\_colorbrown skin\_colorbrown mottle  
## 1 0 0 0 0  
## 4 0 0 0 0  
## 5 0 0 0 0  
## 6 0 0 0 0  
## 7 0 0 0 0  
## 9 0 0 0 0  
## skin\_colordark skin\_colorfair skin\_colorgreen skin\_colorlight  
## 1 0 1 0 0  
## 4 0 0 0 0  
## 5 0 0 0 1  
## 6 0 0 0 1  
## 7 0 0 0 1  
## 9 0 0 0 1  
## skin\_colororange skin\_colorpale skin\_colorred skin\_colortan skin\_colorunknown  
## 1 0 0 0 0 0  
## 4 0 0 0 0 0  
## 5 0 0 0 0 0  
## 6 0 0 0 0 0  
## 7 0 0 0 0 0  
## 9 0 0 0 0 0  
## skin\_colorwhite skin\_coloryellow eye\_colorblack eye\_colorblue  
## 1 0 0 0 1  
## 4 1 0 0 0  
## 5 0 0 0 0  
## 6 0 0 0 1  
## 7 0 0 0 1  
## 9 0 0 0 0  
## eye\_colorblue-gray eye\_colorbrown eye\_colorhazel eye\_colororange eye\_colorred  
## 1 0 0 0 0 0  
## 4 0 0 0 0 0  
## 5 0 1 0 0 0  
## 6 0 0 0 0 0  
## 7 0 0 0 0 0  
## 9 0 1 0 0 0  
## eye\_coloryellow birth\_year sexfemale sexmale homeworldAlderaan  
## 1 0 19.0 0 1 0  
## 4 1 41.9 0 1 0  
## 5 0 19.0 1 0 1  
## 6 0 52.0 0 1 0  
## 7 0 47.0 1 0 0  
## 9 0 24.0 0 1 0  
## homeworldBespin homeworldCerea homeworldConcord Dawn homeworldCorellia  
## 1 0 0 0 0  
## 4 0 0 0 0  
## 5 0 0 0 0  
## 6 0 0 0 0  
## 7 0 0 0 0  
## 9 0 0 0 0  
## homeworldDathomir homeworldDorin homeworldEndor homeworldHaruun Kal  
## 1 0 0 0 0  
## 4 0 0 0 0  
## 5 0 0 0 0  
## 6 0 0 0 0  
## 7 0 0 0 0  
## 9 0 0 0 0  
## homeworldKamino homeworldKashyyyk homeworldMirial homeworldMon Cala  
## 1 0 0 0 0  
## 4 0 0 0 0  
## 5 0 0 0 0  
## 6 0 0 0 0  
## 7 0 0 0 0  
## 9 0 0 0 0  
## homeworldNaboo homeworldRyloth homeworldSerenno homeworldSocorro  
## 1 0 0 0 0  
## 4 0 0 0 0  
## 5 0 0 0 0  
## 6 0 0 0 0  
## 7 0 0 0 0  
## 9 0 0 0 0  
## homeworldStewjon homeworldTatooine homeworldTrandosha speciesCerean  
## 1 0 1 0 0  
## 4 0 1 0 0  
## 5 0 0 0 0  
## 6 0 1 0 0  
## 7 0 1 0 0  
## 9 0 1 0 0  
## speciesEwok speciesGungan speciesHuman speciesKel Dor speciesMirialan  
## 1 0 0 1 0 0  
## 4 0 0 1 0 0  
## 5 0 0 1 0 0  
## 6 0 0 1 0 0  
## 7 0 0 1 0 0  
## 9 0 0 1 0 0  
## speciesMon Calamari speciesTrandoshan speciesTwi'lek speciesWookiee  
## 1 0 0 0 0  
## 4 0 0 0 0  
## 5 0 0 0 0  
## 6 0 0 0 0  
## 7 0 0 0 0  
## 9 0 0 0 0  
## speciesZabrak gender  
## 1 0 masculine  
## 4 0 masculine  
## 5 0 feminine  
## 6 0 masculine  
## 7 0 feminine  
## 9 0 masculine

3.B) Use SVM to predict gender and report the accuracy.

library(caret)  
dummies\_data$gender <- starwars$gender  
dumdf <- data.frame(dummies\_data)  
svm1 <- train(gender ~., data = dummies\_data, method = "svmLinear")

## Warning in .local(x, ...): Variable(s) `' constant. Cannot scale data.  
  
## Warning in .local(x, ...): Variable(s) `' constant. Cannot scale data.  
  
## Warning in .local(x, ...): Variable(s) `' constant. Cannot scale data.  
  
## Warning in .local(x, ...): Variable(s) `' constant. Cannot scale data.  
  
## Warning in .local(x, ...): Variable(s) `' constant. Cannot scale data.  
  
## Warning in .local(x, ...): Variable(s) `' constant. Cannot scale data.  
  
## Warning in .local(x, ...): Variable(s) `' constant. Cannot scale data.  
  
## Warning in .local(x, ...): Variable(s) `' constant. Cannot scale data.  
  
## Warning in .local(x, ...): Variable(s) `' constant. Cannot scale data.  
  
## Warning in .local(x, ...): Variable(s) `' constant. Cannot scale data.  
  
## Warning in .local(x, ...): Variable(s) `' constant. Cannot scale data.  
  
## Warning in .local(x, ...): Variable(s) `' constant. Cannot scale data.  
  
## Warning in .local(x, ...): Variable(s) `' constant. Cannot scale data.  
  
## Warning in .local(x, ...): Variable(s) `' constant. Cannot scale data.  
  
## Warning in .local(x, ...): Variable(s) `' constant. Cannot scale data.  
  
## Warning in .local(x, ...): Variable(s) `' constant. Cannot scale data.  
  
## Warning in .local(x, ...): Variable(s) `' constant. Cannot scale data.  
  
## Warning in .local(x, ...): Variable(s) `' constant. Cannot scale data.  
  
## Warning in .local(x, ...): Variable(s) `' constant. Cannot scale data.  
  
## Warning in .local(x, ...): Variable(s) `' constant. Cannot scale data.  
  
## Warning in .local(x, ...): Variable(s) `' constant. Cannot scale data.  
  
## Warning in .local(x, ...): Variable(s) `' constant. Cannot scale data.  
  
## Warning in .local(x, ...): Variable(s) `' constant. Cannot scale data.  
  
## Warning in .local(x, ...): Variable(s) `' constant. Cannot scale data.  
  
## Warning in .local(x, ...): Variable(s) `' constant. Cannot scale data.

## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo, :  
## There were missing values in resampled performance measures.

svm1

## Support Vector Machines with Linear Kernel   
##   
## 29 samples  
## 66 predictors  
## 2 classes: 'feminine', 'masculine'   
##   
## No pre-processing  
## Resampling: Bootstrapped (25 reps)   
## Summary of sample sizes: 29, 29, 29, 29, 29, 29, ...   
## Resampling results:  
##   
## Accuracy Kappa   
## 0.8843656 0.6446393  
##   
## Tuning parameter 'C' was held constant at a value of 1

### 3.C Given that we have so many variables, it makes sense to consider using PCA. Run PCA on the data and determine an appropriate number of components to use. Document how you made the decision, including any graphs you used. Create a reduced version of the data with that number of principle components. Note: make sure to remove gender from the data before running PCA because it would be cheating if PCA had access to the label you will use. Add it back in after reducing the data and show the result.

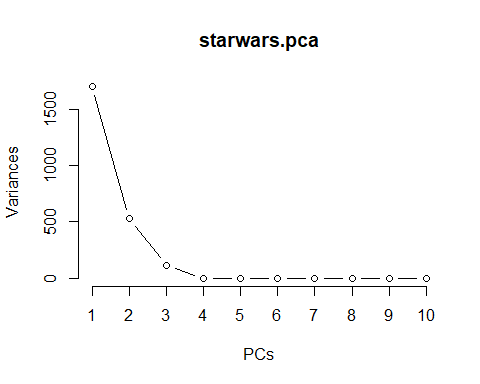
str(dummydata)

## List of 9  
## $ call : language dummyVars.default(formula = gender ~ ., data = starwars)  
## $ form :Class 'formula' language gender ~ .  
## .. ..- attr(\*, ".Environment")=<environment: R\_GlobalEnv>   
## $ vars : chr [1:10] "gender" "height" "mass" "hair\_color" ...  
## $ facVars : NULL  
## $ lvls : NULL  
## $ sep : chr "."  
## $ terms :Classes 'terms', 'formula' language gender ~ height + mass + hair\_color + skin\_color + eye\_color + birth\_year + sex + homeworld + species  
## .. ..- attr(\*, "variables")= language list(gender, height, mass, hair\_color, skin\_color, eye\_color, birth\_year, sex, homeworld, species)  
## .. ..- attr(\*, "factors")= int [1:10, 1:9] 0 1 0 0 0 0 0 0 0 0 ...  
## .. .. ..- attr(\*, "dimnames")=List of 2  
## .. .. .. ..$ : chr [1:10] "gender" "height" "mass" "hair\_color" ...  
## .. .. .. ..$ : chr [1:9] "height" "mass" "hair\_color" "skin\_color" ...  
## .. ..- attr(\*, "term.labels")= chr [1:9] "height" "mass" "hair\_color" "skin\_color" ...  
## .. ..- attr(\*, "order")= int [1:9] 1 1 1 1 1 1 1 1 1  
## .. ..- attr(\*, "intercept")= int 1  
## .. ..- attr(\*, "response")= int 1  
## .. ..- attr(\*, ".Environment")=<environment: R\_GlobalEnv>   
## .. ..- attr(\*, "predvars")= language list(gender, height, mass, hair\_color, skin\_color, eye\_color, birth\_year, sex, homeworld, species)  
## .. ..- attr(\*, "dataClasses")= Named chr [1:10] "character" "numeric" "numeric" "character" ...  
## .. .. ..- attr(\*, "names")= chr [1:10] "gender" "height" "mass" "hair\_color" ...  
## $ levelsOnly: logi FALSE  
## $ fullRank : logi FALSE  
## - attr(\*, "class")= chr "dummyVars"

dummies\_data$gender <- NULL  
nzv <- nearZeroVar(dummies\_data)  
starwars.pca <- prcomp(dummies\_data)  
summary(starwars.pca)

## Importance of components:  
## PC1 PC2 PC3 PC4 PC5 PC6 PC7  
## Standard deviation 41.2498 22.9472 10.69955 0.78075 0.73976 0.58724 0.57943  
## Proportion of Variance 0.7252 0.2244 0.04879 0.00026 0.00023 0.00015 0.00014  
## Cumulative Proportion 0.7252 0.9496 0.99838 0.99864 0.99887 0.99902 0.99916  
## PC8 PC9 PC10 PC11 PC12 PC13 PC14  
## Standard deviation 0.51580 0.4766 0.41954 0.38652 0.37506 0.37045 0.35266  
## Proportion of Variance 0.00011 0.0001 0.00008 0.00006 0.00006 0.00006 0.00005  
## Cumulative Proportion 0.99928 0.9994 0.99945 0.99951 0.99957 0.99963 0.99968  
## PC15 PC16 PC17 PC18 PC19 PC20 PC21  
## Standard deviation 0.34046 0.32506 0.31306 0.29384 0.27264 0.25462 0.24142  
## Proportion of Variance 0.00005 0.00005 0.00004 0.00004 0.00003 0.00003 0.00002  
## Cumulative Proportion 0.99973 0.99978 0.99982 0.99986 0.99989 0.99991 0.99994  
## PC22 PC23 PC24 PC25 PC26 PC27 PC28  
## Standard deviation 0.20136 0.18976 0.16411 0.14962 0.11579 0.0376 0.02181  
## Proportion of Variance 0.00002 0.00002 0.00001 0.00001 0.00001 0.0000 0.00000  
## Cumulative Proportion 0.99996 0.99997 0.99998 0.99999 1.00000 1.0000 1.00000  
## PC29  
## Standard deviation 3.822e-15  
## Proportion of Variance 0.000e+00  
## Cumulative Proportion 1.000e+00

screeplot(starwars.pca, type = "l") + title(xlab = "PCs")



## integer(0)

## integer(0)

target <- starwars %>% dplyr::select(gender)  
process <- preProcess(dummies\_data, method="pca", pcaComp=2)  
starwars.pc <- predict(process, dummies\_data)  
starwars.pc$gender <- starwars$gender  
head(starwars.pc)

## PC1 PC2 gender  
## 1 -0.1842293 1.7888683 masculine  
## 4 2.6287944 0.7247147 masculine  
## 5 -3.5412999 0.3049210 feminine  
## 6 0.4172034 1.7928384 masculine  
## 7 -2.1186809 0.6821707 feminine  
## 9 -0.7075658 1.4212027 masculine

3D) Use SVM to predict gender again, but this time use the data resulting from PCA. Evaluate the results with a confusion matrix and at least two partitioning methods, using grid search on the C parameter each time.

set.seed(128)  
new\_index = createDataPartition(y=starwars.pc$gender, p=0.7, list=FALSE)  
train\_set\_data = starwars.pc[new\_index,]  
test\_set\_data = starwars.pc[-new\_index,]  
train\_svm = trainControl(method = "cv", number = 100)  
svm\_split <- train(gender ~., data = starwars.pc, method = "svmLinear")

## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo, :  
## There were missing values in resampled performance measures.

pred\_split <- predict(svm\_split, starwars.pc)  
sum(pred\_split == starwars.pc$gender) / nrow(starwars.pc)

## [1] 0.8965517

starwars.pc$gender <- as.factor(starwars.pc$gender)   
confusionMatrix(starwars.pc$gender,pred\_split)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction feminine masculine  
## feminine 4 2  
## masculine 1 22  
##   
## Accuracy : 0.8966   
## 95% CI : (0.7265, 0.9781)  
## No Information Rate : 0.8276   
## P-Value [Acc > NIR] : 0.2387   
##   
## Kappa : 0.6641   
##   
## Mcnemar's Test P-Value : 1.0000   
##   
## Sensitivity : 0.8000   
## Specificity : 0.9167   
## Pos Pred Value : 0.6667   
## Neg Pred Value : 0.9565   
## Prevalence : 0.1724   
## Detection Rate : 0.1379   
## Detection Prevalence : 0.2069   
## Balanced Accuracy : 0.8583   
##   
## 'Positive' Class : feminine   
##

grid <- expand.grid(C = 10^seq(-5,2,0.5))  
svm\_grid <- train((gender) ~., data = starwars.pc, method = "svmLinear",  
 trControl = train\_svm, tuneGrid = grid)

## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo, :  
## There were missing values in resampled performance measures.

svm\_grid

## Support Vector Machines with Linear Kernel   
##   
## 29 samples  
## 2 predictor  
## 2 classes: 'feminine', 'masculine'   
##   
## No pre-processing  
## Resampling: Cross-Validated (100 fold)   
## Summary of sample sizes: 28, 28, 28, 28, 28, 28, ...   
## Resampling results across tuning parameters:  
##   
## C Accuracy Kappa  
## 1.000000e-05 0.7931034 0   
## 3.162278e-05 0.7931034 0   
## 1.000000e-04 0.7931034 0   
## 3.162278e-04 0.7931034 0   
## 1.000000e-03 0.7931034 0   
## 3.162278e-03 0.7931034 0   
## 1.000000e-02 0.7931034 0   
## 3.162278e-02 0.7931034 0   
## 1.000000e-01 0.7931034 0   
## 3.162278e-01 0.8965517 0   
## 1.000000e+00 0.8965517 0   
## 3.162278e+00 0.8965517 0   
## 1.000000e+01 0.8965517 0   
## 3.162278e+01 0.8620690 0   
## 1.000000e+02 0.8965517 0   
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was C = 0.3162278.

tc\_boot = trainControl(method = "boot", number = 100)  
svm5 <- train(gender ~., data = starwars.pc, method = "svmLinear",   
 trControl = tc\_boot)

## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo, :  
## There were missing values in resampled performance measures.

svm5

## Support Vector Machines with Linear Kernel   
##   
## 29 samples  
## 2 predictor  
## 2 classes: 'feminine', 'masculine'   
##   
## No pre-processing  
## Resampling: Bootstrapped (100 reps)   
## Summary of sample sizes: 29, 29, 29, 29, 29, 29, ...   
## Resampling results:  
##   
## Accuracy Kappa   
## 0.8860461 0.61841  
##   
## Tuning parameter 'C' was held constant at a value of 1

pred\_split <- predict(svm5, starwars.pc)  
confusionMatrix(starwars.pc$gender, pred\_split)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction feminine masculine  
## feminine 4 2  
## masculine 1 22  
##   
## Accuracy : 0.8966   
## 95% CI : (0.7265, 0.9781)  
## No Information Rate : 0.8276   
## P-Value [Acc > NIR] : 0.2387   
##   
## Kappa : 0.6641   
##   
## Mcnemar's Test P-Value : 1.0000   
##   
## Sensitivity : 0.8000   
## Specificity : 0.9167   
## Pos Pred Value : 0.6667   
## Neg Pred Value : 0.9565   
## Prevalence : 0.1724   
## Detection Rate : 0.1379   
## Detection Prevalence : 0.2069   
## Balanced Accuracy : 0.8583   
##   
## 'Positive' Class : feminine   
##

#### 3e) Whether or not it has improved the accuracy, what has PCA done for the complexity of the model?

**Inference**:

A statistical technique called Principal Component Analysis (PCA) reduces the dimension of the data by isolating the variables and leaving the variables with the least amount of information about what we projected. Hence PCA will reduce the overall complexity of the model.

### 5) Bonus Problem (5 points)

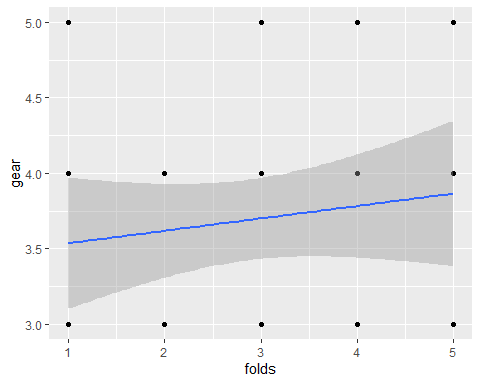
#### To understand just how much different subsets can differ, create a 5 fold partitioning of the cars data included in R (mtcars) and visualize the distribution of the gears variable across the folds. Rather than use the fancy trainControl methods for making the folds, create them directly so you actually can keep track of which data points are in which fold. This is not covered in the tutorial, but it is quick. Here is code to create 5 folds and a variable in the data frame that contains the fold index of each point. Use that resulting data frame to create your visualization.

#### mycars <- mtcars # make a copy to modify

#### mycars$folds = 0 # initialize new variable to hold fold indices #### Create 5 folds, get a list of lists of indices. #### Take a look at this result so you understand what is happening. #### Note we are not passing the data frame directly, but a list of its #### indices created by 1:nrow(mycars). If you don’t understand how #### that works, try the individual parts on their own first #### flds = createFolds(1:nrow(mycars), k=5, list=TRUE) #### This loop sets all the rows in a given fold to have that fold’s #### index in the folds variable. Take a look at the result and use it #### to create the visualization. #### for (i in 1:5) { mycars$folds[flds[[i]]] = i}

mycars <- mtcars   
mycars$folds = 0  
flds = createFolds(1:nrow(mycars), k=5, list=TRUE)  
for (i in 1:5) {   
 mycars$folds[flds[[i]]] = i  
 }  
ggplot(mycars, aes(folds, gear)) + geom\_point() + geom\_smooth(method = lm)

## `geom\_smooth()` using formula 'y ~ x'



## `geom\_smooth()` using formula 'y ~ x'