Corona VIrus and Wealth

Anita Adams, Randy Geszvain and Michael O’Keeefe

6/12/2020

knitr::opts\_current$get(c(  
 "cache",  
 "cache.path",  
 "cache.rebuild",  
 "dependson",  
 "autodep"  
))

## $cache  
## [1] 3  
##   
## $cache.path  
## [1] "Covid19ARA\_cache/docx/"  
##   
## $cache.rebuild  
## [1] FALSE  
##   
## $dependson  
## NULL  
##   
## $autodep  
## [1] FALSE

# Load Data

# OriginalACSData<- read\_csv("ACS\_full\_data\_wo\_over\_60.csv")  
OriginalACSData<- read.csv("C:/Users/anita/Documents/Syracuse/IST 707/Class Project/Data/ACS\_full\_data\_wo\_over\_60.csv")  
  
ACSData<- OriginalACSData

# Data Cleaning

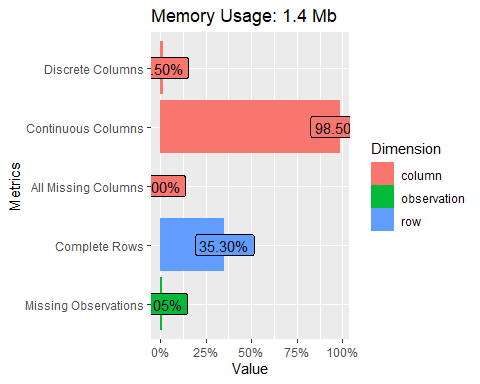
#ACSData %>% drop\_na()  
# ACSData <- na.omit(ACSData) dont omit NAs until we have whittled down the attributes.   
  
# ACSData$X<-factor(ACSData$X)  
ACSData$STATE<-factor(ACSData$STATE)  
ACSData$COUNTY<-factor(ACSData$COUNTY)

#displaying top 5 rows  
#head(ACSData)

#Variables

plot\_str(ACSData)

plot\_intro(ACSData)

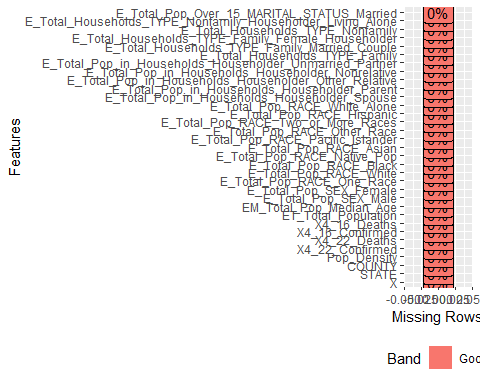


## Missing Data

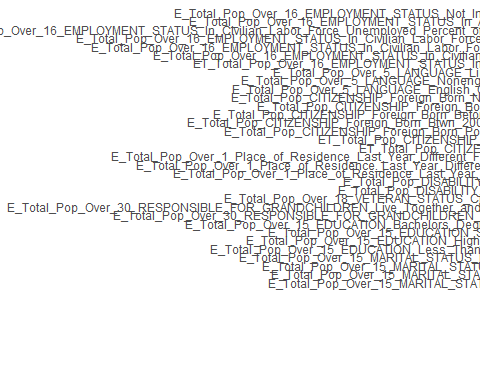
#Checking for any NA values  
sum(is.na(ACSData))

## [1] 2063

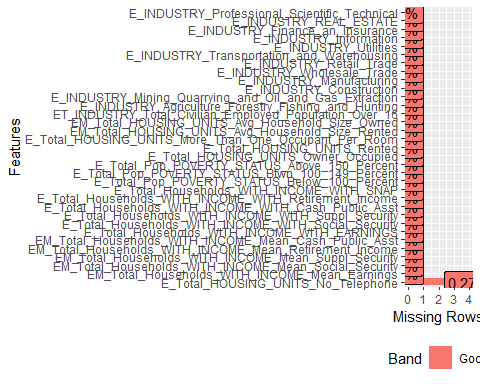
# plot\_missing(ACSData)  
  
plot\_missing(ACSData[,1:33])



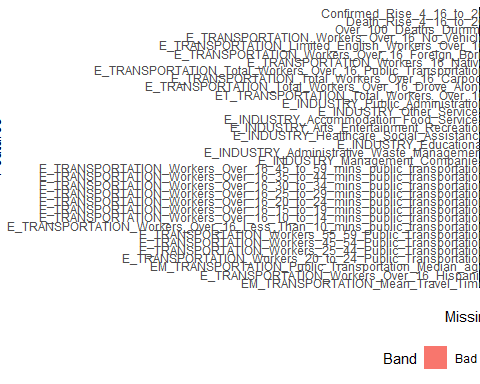
plot\_missing(ACSData[,34:66])



plot\_missing(ACSData[,67:99])



plot\_missing(ACSData[,100:133])



# MOst of the missing data is on the transportation data so we wont use those variables to preserve as many rows as possible  
#head(ACSData)

#EDA

introduce(ACSData)

## rows columns discrete\_columns continuous\_columns all\_missing\_columns  
## 1 1476 133 2 131 0  
## total\_missing\_values complete\_rows total\_observations memory\_usage  
## 1 2063 521 196308 1425880

#glimpse(ACSData)

#summary(ACSData)

#skim(ACSData)

# Select the subset of attributes we want to include in the analysis and transform / normalize them  
  
D <- data.frame (ACSData)  
#str(D)  
#colnames(D)  
Original\_D\_Col <- ncol(D)

D$Confirmed\_Per\_Capita\_4\_22 <- D$X4\_22\_Confirmed / D$ET\_Total\_Population  
D$Deaths\_Per\_Capita\_4\_22 <- D$X4\_22\_Deaths / D$ET\_Total\_Population  
D$Deaths\_Per\_Confirmed\_4\_22 <- D$Deaths\_Per\_Capita\_4\_22 / D$Confirmed\_Per\_Capita\_4\_22 # Per Capita is no longer meaningful  
  
D$Confirmed\_Per\_Capita\_4\_16 <- D$X4\_16\_Confirmed / D$ET\_Total\_Population  
D$Deaths\_Per\_Capita\_4\_16 <- D$X4\_16\_Deaths / D$ET\_Total\_Population  
D$Deaths\_Per\_Confirmed\_4\_16 <- D$Deaths\_Per\_Capita\_4\_16 / D$Confirmed\_Per\_Capita\_4\_16 # Per Capita is no longer meaningful  
  
  
D$Two\_Week\_Confirm\_Rate\_Per\_Capita <- (D$X4\_22\_Confirmed-D$X4\_16\_Confirmed)/D$ET\_Total\_Population  
D$Two\_Week\_Death\_Rate\_Per\_Capita <- (D$X4\_22\_Death-D$X4\_16\_Death)/D$ET\_Total\_Population  
  
D$Fraction\_Female <- D$E\_Total\_Pop\_SEX\_Female / D$ET\_Total\_Population  
  
D$Fraction\_White <- D$E\_Total\_Pop\_RACE\_White / D$ET\_Total\_Population  
D$Fraction\_Black <- D$E\_Total\_Pop\_RACE\_Black / D$ET\_Total\_Population  
D$Fraction\_Other <- (D$E\_Total\_Pop\_RACE\_Native\_Pop + D$E\_Total\_Pop\_RACE\_Asian +  
 D$E\_Total\_Pop\_RACE\_Pacific\_Islander + D$E\_Total\_Pop\_RACE\_Other\_Race +  
 D$E\_Total\_Pop\_RACE\_Two\_or\_More\_Races) / D$ET\_Total\_Population  
  
D$Hispanic <- D$E\_Total\_Pop\_RACE\_Hispanic / D$ET\_Total\_Population  
  
D$Population\_Over\_Age\_15 <- D$E\_Total\_Pop\_Over\_15\_EDUCATION\_Less\_Than\_High\_School +   
 D$E\_Total\_Pop\_Over\_15\_EDUCATION\_High\_School\_Grad +   
 D$E\_Total\_Pop\_Over\_15\_EDUCATION\_Some\_College+   
 D$E\_Total\_Pop\_Over\_15\_EDUCATION\_Bachelors\_Degree\_or\_Higher  
D$Fraction\_Less\_Than\_HS <- D$E\_Total\_Pop\_Over\_15\_EDUCATION\_Less\_Than\_High\_School / D$Population\_Over\_Age\_15  
D$Fraction\_High\_School\_Grad <- D$E\_Total\_Pop\_Over\_15\_EDUCATION\_High\_School\_Grad / D$Population\_Over\_Age\_15  
D$Fraction\_Some\_College <- D$E\_Total\_Pop\_Over\_15\_EDUCATION\_Some\_College / D$Population\_Over\_Age\_15  
D$Fraction\_Bachelors\_or\_Higher <- D$E\_Total\_Pop\_Over\_15\_EDUCATION\_Bachelors\_Degree\_or\_Higher / D$Population\_Over\_Age\_15  
  
D$Fraction\_Disabled <- D$E\_Total\_Pop\_DISABILITY\_STATUS\_Yes / D$ET\_Total\_Population  
  
D$Fraction\_\_Limited\_English <- D$E\_Total\_Pop\_Over\_5\_LANGUAGE\_Limited\_English / D$ET\_Total\_Population  
  
D$Fraction\_Unemployed <- D$E\_Total\_Pop\_Over\_16\_EMPLOYMENT\_STATUS\_In\_Civilian\_Labor\_Force\_Unemployed /   
 D$ET\_Total\_Pop\_Over\_16\_EMPLOYMENT\_STATUS\_In\_Labor\_Force  
  
D$Fraction\_Below\_Poverty\_Line <- D$E\_Total\_Pop\_POVERTY\_STATUS\_Below\_100\_Percent / D$ET\_Total\_Population  
D$Fraction\_Btwn\_100\_149\_Above\_Poverty\_Line <- D$E\_Total\_Pop\_POVERTY\_STATUS\_Btwn\_100\_149\_Percent / D$ET\_Total\_Population  
D$Fraction\_Not\_Citizen <- D$E\_Total\_Pop\_CITIZENSHIP\_Foreign\_Born\_Not\_US\_Citizen / D$ET\_Total\_Population

# Save the orignial attributes we still want (first 10) to a dataframe (E) and clean up the naming  
  
E<- D[,1:10]  
#str(E)  
cnames <- colnames(E)  
cnames[1] <- c("ID")  
cnames[5] <- c("Confirmed\_4\_22")  
cnames[6] <- c("Deaths\_4\_22")  
cnames[7] <- c("Confirmed\_4\_16")  
cnames[8] <- c("Deaths\_4\_16")  
cnames

## [1] "ID" "STATE"   
## [3] "COUNTY" "Pop\_Density"   
## [5] "Confirmed\_4\_22" "Deaths\_4\_22"   
## [7] "Confirmed\_4\_16" "Deaths\_4\_16"   
## [9] "ET\_Total\_Population" "EM\_Total\_Pop\_Median\_Age"

colnames(E) <- cnames

# Now add the newly computed attributes to E  
  
Original\_D\_Col

## [1] 133

New\_D\_Col <- ncol(D)  
New\_D\_Col

## [1] 157

D\_Col\_to\_Add <- Original\_D\_Col+1  
  
E <- data.frame(E,D[,D\_Col\_to\_Add:New\_D\_Col])  
colnames(E)

## [1] "ID"   
## [2] "STATE"   
## [3] "COUNTY"   
## [4] "Pop\_Density"   
## [5] "Confirmed\_4\_22"   
## [6] "Deaths\_4\_22"   
## [7] "Confirmed\_4\_16"   
## [8] "Deaths\_4\_16"   
## [9] "ET\_Total\_Population"   
## [10] "EM\_Total\_Pop\_Median\_Age"   
## [11] "Confirmed\_Per\_Capita\_4\_22"   
## [12] "Deaths\_Per\_Capita\_4\_22"   
## [13] "Deaths\_Per\_Confirmed\_4\_22"   
## [14] "Confirmed\_Per\_Capita\_4\_16"   
## [15] "Deaths\_Per\_Capita\_4\_16"   
## [16] "Deaths\_Per\_Confirmed\_4\_16"   
## [17] "Two\_Week\_Confirm\_Rate\_Per\_Capita"   
## [18] "Two\_Week\_Death\_Rate\_Per\_Capita"   
## [19] "Fraction\_Female"   
## [20] "Fraction\_White"   
## [21] "Fraction\_Black"   
## [22] "Fraction\_Other"   
## [23] "Hispanic"   
## [24] "Population\_Over\_Age\_15"   
## [25] "Fraction\_Less\_Than\_HS"   
## [26] "Fraction\_High\_School\_Grad"   
## [27] "Fraction\_Some\_College"   
## [28] "Fraction\_Bachelors\_or\_Higher"   
## [29] "Fraction\_Disabled"   
## [30] "Fraction\_\_Limited\_English"   
## [31] "Fraction\_Unemployed"   
## [32] "Fraction\_Below\_Poverty\_Line"   
## [33] "Fraction\_Btwn\_100\_149\_Above\_Poverty\_Line"  
## [34] "Fraction\_Not\_Citizen"

#head(E)  
#str(E)

E$Confirmed\_4\_16 <- as.numeric(E$Confirmed\_4\_16)  
E$Deaths\_4\_16 <- as.numeric(E$Deaths\_4\_16)  
E$Confirmed\_4\_22 <- as.numeric(E$Confirmed\_4\_22)  
E$Deaths\_4\_22 <- as.numeric(E$Deaths\_4\_22)  
E$ET\_Total\_Population <- as.numeric(E$ET\_Total\_Population)

New\_Data <- data.frame(E)  
str(New\_Data)

## 'data.frame': 1476 obs. of 34 variables:  
## $ ID : int 1 2 3 4 5 6 7 8 9 10 ...  
## $ STATE : Factor w/ 51 levels "ALABAMA","ALASKA",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ COUNTY : Factor w/ 990 levels "Abbeville","Acadia",..: 45 47 84 128 157 171 175 201 202 221 ...  
## $ Pop\_Density : num 92.9 130.9 89.4 190 56.7 ...  
## $ Confirmed\_4\_22 : num 32 132 29 85 270 13 46 73 20 27 ...  
## $ Deaths\_4\_22 : num 2 3 0 3 16 0 1 1 1 0 ...  
## $ Confirmed\_4\_16 : num 26 101 18 62 231 11 37 47 12 18 ...  
## $ Deaths\_4\_16 : num 1 2 0 0 11 0 0 0 1 0 ...  
## $ ET\_Total\_Population : num 55200 208107 57645 115098 33826 ...  
## $ EM\_Total\_Pop\_Median\_Age : num 37.8 42.8 40.8 39.7 43 45.9 38.6 39.3 42.7 43.9 ...  
## $ Confirmed\_Per\_Capita\_4\_22 : num 0.00058 0.000634 0.000503 0.000739 0.007982 ...  
## $ Deaths\_Per\_Capita\_4\_22 : num 3.62e-05 1.44e-05 0.00 2.61e-05 4.73e-04 ...  
## $ Deaths\_Per\_Confirmed\_4\_22 : num 0.0625 0.0227 0 0.0353 0.0593 ...  
## $ Confirmed\_Per\_Capita\_4\_16 : num 0.000471 0.000485 0.000312 0.000539 0.006829 ...  
## $ Deaths\_Per\_Capita\_4\_16 : num 1.81e-05 9.61e-06 0.00 0.00 3.25e-04 ...  
## $ Deaths\_Per\_Confirmed\_4\_16 : num 0.0385 0.0198 0 0 0.0476 ...  
## $ Two\_Week\_Confirm\_Rate\_Per\_Capita : num 0.000109 0.000149 0.000191 0.0002 0.001153 ...  
## $ Two\_Week\_Death\_Rate\_Per\_Capita : num 1.81e-05 4.81e-06 0.00 2.61e-05 1.48e-04 ...  
## $ Fraction\_Female : num 0.513 0.514 0.507 0.519 0.519 0.504 0.507 0.506 0.522 0.513 ...  
## $ Fraction\_White : num 0.769 0.863 0.955 0.743 0.576 0.928 0.833 0.759 0.794 0.847 ...  
## $ Fraction\_Black : num 0.191 0.095 0.015 0.206 0.394 0.049 0.094 0.172 0.158 0.129 ...  
## $ Fraction\_Other : num 0.04 0.042 0.029 0.051 0.03 0.022 0.072 0.069 0.048 0.024 ...  
## $ Hispanic : num 0.028 0.045 0.091 0.037 0.023 0.015 0.078 0.069 0.025 0.016 ...  
## $ Population\_Over\_Age\_15 : num 37166 146842 39587 79135 23540 ...  
## $ Fraction\_Less\_Than\_HS : num 0.113 0.0971 0.1982 0.159 0.1862 ...  
## $ Fraction\_High\_School\_Grad : num 0.326 0.276 0.34 0.324 0.384 ...  
## $ Fraction\_Some\_College : num 0.284 0.313 0.335 0.337 0.297 ...  
## $ Fraction\_Bachelors\_or\_Higher : num 0.277 0.313 0.126 0.18 0.132 ...  
## $ Fraction\_Disabled : num 0.19 0.138 0.141 0.205 0.165 ...  
## $ Fraction\_\_Limited\_English : num 0.01223 0.01605 0.031 0.01509 0.00754 ...  
## $ Fraction\_Unemployed : num 0.0422 0.0444 0.0412 0.0876 0.0497 ...  
## $ Fraction\_Below\_Poverty\_Line : num 0.153 0.104 0.143 0.181 0.163 ...  
## $ Fraction\_Btwn\_100\_149\_Above\_Poverty\_Line: num 0.0853 0.0778 0.1089 0.1119 0.1446 ...  
## $ Fraction\_Not\_Citizen : num 0.00969 0.0185 0.02838 0.01358 0.01171 ...

# Look for any NAs  
sum(is.na(New\_Data))

## [1] 2

New\_Data <- na.omit(New\_Data)  
sum(is.na(New\_Data))

## [1] 0

# Look at descriptive statistics  
  
summary(New\_Data)

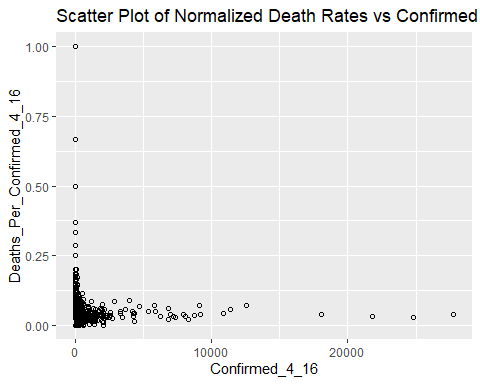
## ID STATE COUNTY Pop\_Density   
## Min. : 1.0 TEXAS : 96 Washington: 17 Min. : 2.404   
## 1st Qu.: 369.2 OHIO : 75 Franklin : 16 1st Qu.: 63.419   
## Median : 738.5 NORTH CAROLINA: 74 Jefferson : 16 Median : 119.569   
## Mean : 738.3 VIRGINIA : 62 Jackson : 12 Mean : 380.267   
## 3rd Qu.:1106.8 GEORGIA : 60 Lincoln : 12 3rd Qu.: 310.787   
## Max. :1476.0 MICHIGAN : 60 Montgomery: 12 Max. :18565.536   
## (Other) :1047 (Other) :1389   
## Confirmed\_4\_22 Deaths\_4\_22 Confirmed\_4\_16 Deaths\_4\_16   
## Min. : 1.0 Min. : 0.00 Min. : 1.0 Min. : 0.00   
## 1st Qu.: 18.0 1st Qu.: 0.00 1st Qu.: 14.0 1st Qu.: 0.00   
## Median : 53.0 Median : 2.00 Median : 41.0 Median : 1.00   
## Mean : 470.9 Mean : 21.08 Mean : 370.3 Mean : 14.39   
## 3rd Qu.: 195.0 3rd Qu.: 7.00 3rd Qu.: 146.0 3rd Qu.: 5.00   
## Max. :31555.0 Max. :1431.00 Max. :27772.0 Max. :1109.00   
##   
## ET\_Total\_Population EM\_Total\_Pop\_Median\_Age Confirmed\_Per\_Capita\_4\_22  
## Min. : 18699 Min. :24.60 Min. :1.578e-05   
## 1st Qu.: 43016 1st Qu.:37.10 1st Qu.:3.263e-04   
## Median : 75242 Median :40.10 Median :6.378e-04   
## Mean : 204250 Mean :40.17 Mean :1.327e-03   
## 3rd Qu.: 178056 3rd Qu.:43.10 3rd Qu.:1.311e-03   
## Max. :10098052 Max. :67.00 Max. :3.270e-02   
##   
## Deaths\_Per\_Capita\_4\_22 Deaths\_Per\_Confirmed\_4\_22 Confirmed\_Per\_Capita\_4\_16  
## Min. :0.000e+00 Min. :0.00000 Min. :1.578e-05   
## 1st Qu.:0.000e+00 1st Qu.:0.00000 1st Qu.:2.538e-04   
## Median :2.057e-05 Median :0.02737 Median :4.963e-04   
## Mean :5.629e-05 Mean :0.04008 Mean :1.001e-03   
## 3rd Qu.:5.611e-05 3rd Qu.:0.05620 3rd Qu.:9.851e-04   
## Max. :1.312e-03 Max. :1.00000 Max. :2.704e-02   
##   
## Deaths\_Per\_Capita\_4\_16 Deaths\_Per\_Confirmed\_4\_16  
## Min. :0.000e+00 Min. :0.00000   
## 1st Qu.:0.000e+00 1st Qu.:0.00000   
## Median :1.345e-05 Median :0.02255   
## Mean :3.843e-05 Mean :0.03500   
## 3rd Qu.:3.737e-05 3rd Qu.:0.04928   
## Max. :1.082e-03 Max. :1.00000   
##   
## Two\_Week\_Confirm\_Rate\_Per\_Capita Two\_Week\_Death\_Rate\_Per\_Capita  
## Min. :-4.155e-05 Min. :-1.621e-04   
## 1st Qu.: 3.945e-05 1st Qu.: 0.000e+00   
## Median : 1.225e-04 Median : 0.000e+00   
## Mean : 3.259e-04 Mean : 1.785e-05   
## 3rd Qu.: 3.145e-04 3rd Qu.: 2.041e-05   
## Max. : 2.848e-02 Max. : 4.367e-04   
##   
## Fraction\_Female Fraction\_White Fraction\_Black Fraction\_Other   
## Min. :0.3880 Min. :0.1500 Min. :0.00100 Min. :0.00700   
## 1st Qu.:0.5010 1st Qu.:0.7522 1st Qu.:0.01400 1st Qu.:0.03800   
## Median :0.5070 Median :0.8740 Median :0.04200 Median :0.05900   
## Mean :0.5053 Mean :0.8209 Mean :0.09599 Mean :0.08312   
## 3rd Qu.:0.5130 3rd Qu.:0.9310 3rd Qu.:0.12000 3rd Qu.:0.09500   
## Max. :0.5390 Max. :0.9840 Max. :0.72300 Max. :0.84200   
##   
## Hispanic Population\_Over\_Age\_15 Fraction\_Less\_Than\_HS  
## Min. :0.00500 Min. : 14430 Min. :0.0200   
## 1st Qu.:0.02800 1st Qu.: 29481 1st Qu.:0.0850   
## Median :0.05300 Median : 50996 Median :0.1120   
## Mean :0.09587 Mean : 138100 Mean :0.1204   
## 3rd Qu.:0.10775 3rd Qu.: 119942 3rd Qu.:0.1460   
## Max. :0.99100 Max. :6845489 Max. :0.4850   
##   
## Fraction\_High\_School\_Grad Fraction\_Some\_College Fraction\_Bachelors\_or\_Higher  
## Min. :0.0810 Min. :0.1140 Min. :0.0820   
## 1st Qu.:0.2720 1st Qu.:0.2767 1st Qu.:0.1770   
## Median :0.3210 Median :0.3060 Median :0.2300   
## Mean :0.3194 Mean :0.3056 Mean :0.2546   
## 3rd Qu.:0.3709 3rd Qu.:0.3336 3rd Qu.:0.3120   
## Max. :0.5180 Max. :0.4520 Max. :0.7460   
##   
## Fraction\_Disabled Fraction\_\_Limited\_English Fraction\_Unemployed  
## Min. :0.04292 Min. :0.0009462 Min. :0.01613   
## 1st Qu.:0.12023 1st Qu.:0.0112994 1st Qu.:0.04485   
## Median :0.14374 Median :0.0223996 Median :0.05565   
## Mean :0.14534 Mean :0.0364552 Mean :0.05797   
## 3rd Qu.:0.16729 3rd Qu.:0.0439510 3rd Qu.:0.06852   
## Max. :0.28615 Max. :0.4287729 Max. :0.15769   
##   
## Fraction\_Below\_Poverty\_Line Fraction\_Btwn\_100\_149\_Above\_Poverty\_Line  
## Min. :0.03486 Min. :0.01992   
## 1st Qu.:0.10496 1st Qu.:0.07445   
## Median :0.13750 Median :0.09189   
## Mean :0.14121 Mean :0.09206   
## 3rd Qu.:0.17029 3rd Qu.:0.10885   
## Max. :0.35692 Max. :0.17622   
##   
## Fraction\_Not\_Citizen  
## Min. :0.0004721   
## 1st Qu.:0.0110260   
## Median :0.0225659   
## Mean :0.0332756   
## 3rd Qu.:0.0417633   
## Max. :0.2295249   
##

summary(New\_Data$Deaths\_Per\_Confirmed\_4\_16)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.00000 0.00000 0.02255 0.03500 0.04928 1.00000

# Observation: The max death rate is 1 but the third quartile is only 5%.

# Plot deaths per confirmed vs confirmed to examine the values of "1"   
  
g <- ggplot(New\_Data, aes(x=Confirmed\_4\_16, y=Deaths\_Per\_Confirmed\_4\_16))  
g <- g+ geom\_point(shape=1) + ggtitle("Scatter Plot of Normalized Death Rates vs Confirmed Cases")  
g

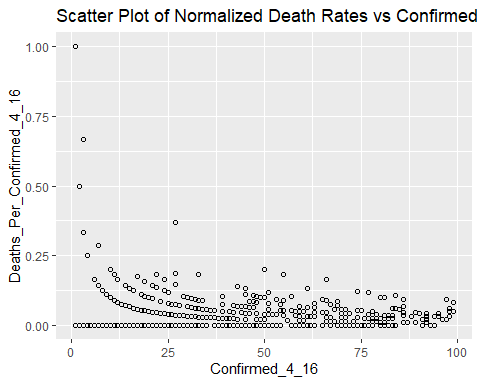


# Observation: The high death rates are concentrated where the number of confirmed case is very small. Zoom in on the data

DataZoomed<- New\_Data[New\_Data$Confirmed\_4\_16<100,]  
  
nrow(DataZoomed)

## [1] 1004

g <- ggplot(DataZoomed, aes(x=Confirmed\_4\_16, y=Deaths\_Per\_Confirmed\_4\_16))  
g <- g+ geom\_point(shape=1) + ggtitle("Scatter Plot of Normalized Death Rates vs Confirmed Cases Zoomed into Comfirmed<100")  
g



# Observation : There seem to be very few with death rates>.25

BigRatesRow <- which(New\_Data$Deaths\_Per\_Confirmed\_4\_16>.25)  
BigRatesRow

## [1] 122 278 408 560 593 728 876 917 1422

# Observation: There are 9/ 1476 data points with Death Rate > .25

BigRatesZScore <- (New\_Data$Deaths\_Per\_Confirmed\_4\_16[BigRatesRow]-mean(New\_Data$Deaths\_Per\_Confirmed\_4\_16[]))/  
 sd(New\_Data$Deaths\_Per\_Confirmed\_4\_16)  
BigRatesZScore

## [1] 16.106724 5.597661 16.106724 10.543102 4.979480 4.184677 7.761291  
## [8] 4.979480 4.979480

# Observation - all have Zscore > 3 so can be treated as outliers.

# It would be more systematic for us to see ALL points outside 3 SD  
Threshold <- 3\*sd(New\_Data$Deaths\_Per\_Confirmed\_4\_16) +mean(New\_Data$Deaths\_Per\_Confirmed\_4\_16)  
Threshold

## [1] 0.2147367

# Observation: Greater than 3 SD away means Death Rate >.2147. See how many points that is and how many

# confirmed cases they are associated with  
  
BigRatesRow <- which(New\_Data$Deaths\_Per\_Confirmed\_4\_16>Threshold)  
BigRatesRow

## [1] 122 278 290 408 560 593 728 876 907 917 1152 1422 1444 1465

New\_Data$Confirmed\_4\_16[BigRatesRow]

## [1] 1 27 4 1 3 3 7 2 4 3 4 3 4 4

# Observation: THere are 14 points and they all were associated with fewer than 7 confirmed cases in the   
# entire county except one where there were 27 cases. Remove these rows from the data set as outliers.

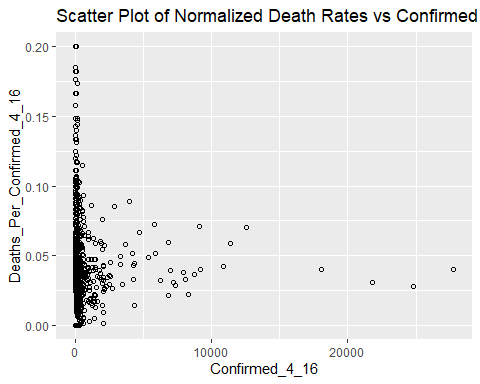
No\_Outliers <- New\_Data[New\_Data$Deaths\_Per\_Confirmed\_4\_16<Threshold,]  
nrow(New\_Data)

## [1] 1474

nrow(No\_Outliers)

## [1] 1460

# Look at the data now that outliers are removed  
# Plot deaths per confirmed vs confirmed to examine the values of "1"   
  
g <- ggplot(No\_Outliers, aes(x=Confirmed\_4\_16, y=Deaths\_Per\_Confirmed\_4\_16))  
g <- g+ geom\_point(shape=1) + ggtitle("Scatter Plot of Normalized Death Rates vs Confirmed Cases w/ Outliers Removed")  
g



# Observation: That looks better.

str(No\_Outliers)

## 'data.frame': 1460 obs. of 34 variables:  
## $ ID : int 1 2 3 4 5 6 7 8 9 10 ...  
## $ STATE : Factor w/ 51 levels "ALABAMA","ALASKA",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ COUNTY : Factor w/ 990 levels "Abbeville","Acadia",..: 45 47 84 128 157 171 175 201 202 221 ...  
## $ Pop\_Density : num 92.9 130.9 89.4 190 56.7 ...  
## $ Confirmed\_4\_22 : num 32 132 29 85 270 13 46 73 20 27 ...  
## $ Deaths\_4\_22 : num 2 3 0 3 16 0 1 1 1 0 ...  
## $ Confirmed\_4\_16 : num 26 101 18 62 231 11 37 47 12 18 ...  
## $ Deaths\_4\_16 : num 1 2 0 0 11 0 0 0 1 0 ...  
## $ ET\_Total\_Population : num 55200 208107 57645 115098 33826 ...  
## $ EM\_Total\_Pop\_Median\_Age : num 37.8 42.8 40.8 39.7 43 45.9 38.6 39.3 42.7 43.9 ...  
## $ Confirmed\_Per\_Capita\_4\_22 : num 0.00058 0.000634 0.000503 0.000739 0.007982 ...  
## $ Deaths\_Per\_Capita\_4\_22 : num 3.62e-05 1.44e-05 0.00 2.61e-05 4.73e-04 ...  
## $ Deaths\_Per\_Confirmed\_4\_22 : num 0.0625 0.0227 0 0.0353 0.0593 ...  
## $ Confirmed\_Per\_Capita\_4\_16 : num 0.000471 0.000485 0.000312 0.000539 0.006829 ...  
## $ Deaths\_Per\_Capita\_4\_16 : num 1.81e-05 9.61e-06 0.00 0.00 3.25e-04 ...  
## $ Deaths\_Per\_Confirmed\_4\_16 : num 0.0385 0.0198 0 0 0.0476 ...  
## $ Two\_Week\_Confirm\_Rate\_Per\_Capita : num 0.000109 0.000149 0.000191 0.0002 0.001153 ...  
## $ Two\_Week\_Death\_Rate\_Per\_Capita : num 1.81e-05 4.81e-06 0.00 2.61e-05 1.48e-04 ...  
## $ Fraction\_Female : num 0.513 0.514 0.507 0.519 0.519 0.504 0.507 0.506 0.522 0.513 ...  
## $ Fraction\_White : num 0.769 0.863 0.955 0.743 0.576 0.928 0.833 0.759 0.794 0.847 ...  
## $ Fraction\_Black : num 0.191 0.095 0.015 0.206 0.394 0.049 0.094 0.172 0.158 0.129 ...  
## $ Fraction\_Other : num 0.04 0.042 0.029 0.051 0.03 0.022 0.072 0.069 0.048 0.024 ...  
## $ Hispanic : num 0.028 0.045 0.091 0.037 0.023 0.015 0.078 0.069 0.025 0.016 ...  
## $ Population\_Over\_Age\_15 : num 37166 146842 39587 79135 23540 ...  
## $ Fraction\_Less\_Than\_HS : num 0.113 0.0971 0.1982 0.159 0.1862 ...  
## $ Fraction\_High\_School\_Grad : num 0.326 0.276 0.34 0.324 0.384 ...  
## $ Fraction\_Some\_College : num 0.284 0.313 0.335 0.337 0.297 ...  
## $ Fraction\_Bachelors\_or\_Higher : num 0.277 0.313 0.126 0.18 0.132 ...  
## $ Fraction\_Disabled : num 0.19 0.138 0.141 0.205 0.165 ...  
## $ Fraction\_\_Limited\_English : num 0.01223 0.01605 0.031 0.01509 0.00754 ...  
## $ Fraction\_Unemployed : num 0.0422 0.0444 0.0412 0.0876 0.0497 ...  
## $ Fraction\_Below\_Poverty\_Line : num 0.153 0.104 0.143 0.181 0.163 ...  
## $ Fraction\_Btwn\_100\_149\_Above\_Poverty\_Line: num 0.0853 0.0778 0.1089 0.1119 0.1446 ...  
## $ Fraction\_Not\_Citizen : num 0.00969 0.0185 0.02838 0.01358 0.01171 ...  
## - attr(\*, "na.action")= 'omit' Named int [1:2] 557 1186  
## ..- attr(\*, "names")= chr [1:2] "557" "1186"

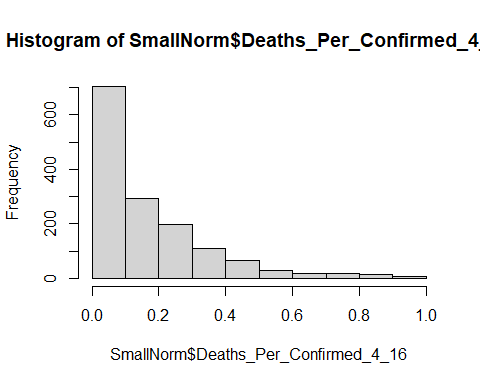
# Now remove the death and confirmed columns that we won't be using.   
SmallData <- data.frame(No\_Outliers[,-c(5,6,7,8,11,12,14,15,17,18)])  
#str(SmallData)

# Now normalize all of the numeric variables.  
  
  
Min\_Max\_function <- function(x) {  
 return( (x-min(x)) / (max(x) - min(x)))  
}  
  
# Test it  
# Min\_Max\_function(c(1,2,3))  
  
# Apply to the DF  
  
SmallDataNumeric <- SmallData[,-1:-3]  
#str(SmallDataNumeric)  
SmallNorm <- as.data.frame(lapply(SmallDataNumeric,Min\_Max\_function))  
#str(SmallNorm)  
summary(SmallNorm)

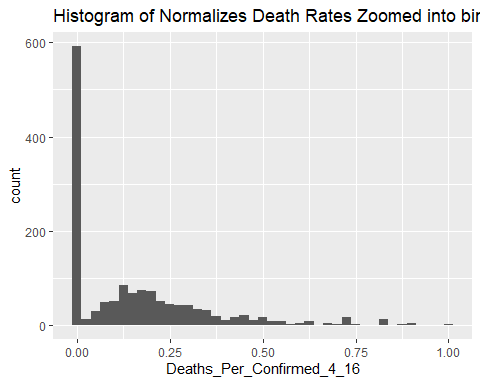
## Pop\_Density ET\_Total\_Population EM\_Total\_Pop\_Median\_Age  
## Min. :0.000000 Min. :0.000000 Min. :0.0000   
## 1st Qu.:0.003306 1st Qu.:0.002449 1st Qu.:0.2948   
## Median :0.006385 Median :0.005662 Median :0.3644   
## Mean :0.020519 Mean :0.018563 Mean :0.3667   
## 3rd Qu.:0.016976 3rd Qu.:0.015938 3rd Qu.:0.4363   
## Max. :1.000000 Max. :1.000000 Max. :1.0000   
## Deaths\_Per\_Confirmed\_4\_22 Deaths\_Per\_Confirmed\_4\_16 Fraction\_Female   
## Min. :0.00000 Min. :0.0000 Min. :0.0000   
## 1st Qu.:0.00000 1st Qu.:0.0000 1st Qu.:0.7483   
## Median :0.05405 Median :0.1111 Median :0.7881   
## Mean :0.07358 Mean :0.1559 Mean :0.7774   
## 3rd Qu.:0.11111 3rd Qu.:0.2381 3rd Qu.:0.8278   
## Max. :1.00000 Max. :1.0000 Max. :1.0000   
## Fraction\_White Fraction\_Black Fraction\_Other Hispanic   
## Min. :0.0000 Min. :0.00000 Min. :0.00000 Min. :0.00000   
## 1st Qu.:0.7203 1st Qu.:0.01801 1st Qu.:0.03713 1st Qu.:0.02333   
## Median :0.8669 Median :0.05817 Median :0.06228 Median :0.04868   
## Mean :0.8032 Mean :0.13263 Mean :0.09148 Mean :0.09241   
## 3rd Qu.:0.9353 3rd Qu.:0.16898 3rd Qu.:0.10659 3rd Qu.:0.10370   
## Max. :1.0000 Max. :1.00000 Max. :1.00000 Max. :1.00000   
## Population\_Over\_Age\_15 Fraction\_Less\_Than\_HS Fraction\_High\_School\_Grad  
## Min. :0.000000 Min. :0.0000 Min. :0.0000   
## 1st Qu.:0.002224 1st Qu.:0.1398 1st Qu.:0.4352   
## Median :0.005476 Median :0.1978 Median :0.5492   
## Mean :0.018256 Mean :0.2161 Mean :0.5448   
## 3rd Qu.:0.015643 3rd Qu.:0.2710 3rd Qu.:0.6630   
## Max. :1.000000 Max. :1.0000 Max. :1.0000   
## Fraction\_Some\_College Fraction\_Bachelors\_or\_Higher Fraction\_Disabled  
## Min. :0.0000 Min. :0.0000 Min. :0.0000   
## 1st Qu.:0.4811 1st Qu.:0.1440 1st Qu.:0.3168   
## Median :0.5671 Median :0.2236 Median :0.4125   
## Mean :0.5660 Mean :0.2608 Mean :0.4201   
## 3rd Qu.:0.6489 3rd Qu.:0.3485 3rd Qu.:0.5100   
## Max. :1.0000 Max. :1.0000 Max. :1.0000   
## Fraction\_\_Limited\_English Fraction\_Unemployed Fraction\_Below\_Poverty\_Line  
## Min. :0.00000 Min. :0.0000 Min. :0.0000   
## 1st Qu.:0.02421 1st Qu.:0.2029 1st Qu.:0.2173   
## Median :0.05021 Median :0.2797 Median :0.3177   
## Mean :0.08329 Mean :0.2958 Mean :0.3296   
## 3rd Qu.:0.10068 3rd Qu.:0.3701 3rd Qu.:0.4202   
## Max. :1.00000 Max. :1.0000 Max. :1.0000   
## Fraction\_Btwn\_100\_149\_Above\_Poverty\_Line Fraction\_Not\_Citizen  
## Min. :0.0000 Min. :0.00000   
## 1st Qu.:0.3474 1st Qu.:0.04609   
## Median :0.4600 Median :0.09672   
## Mean :0.4608 Mean :0.14366   
## 3rd Qu.:0.5684 3rd Qu.:0.18049   
## Max. :1.0000 Max. :1.00000

# Now get rid of 4-22 data for now  
  
SmallNorm <- SmallNorm[,-4]

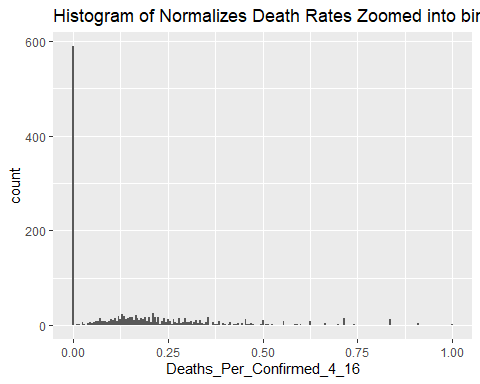
# Now discretize the normalized death rate  
  
# First look at how it is distributed  
  
hist(SmallNorm$Deaths\_Per\_Confirmed\_4\_16)



# Observation: There are a lot of points in the <.1 range - over 700. Zoom in on that area  
g <- ggplot(SmallNorm, aes(x=Deaths\_Per\_Confirmed\_4\_16))  
g <- g+ geom\_histogram(binwidth=.025) + ggtitle("Histogram of Normalizes Death Rates Zoomed into binwidth = .025")  
g



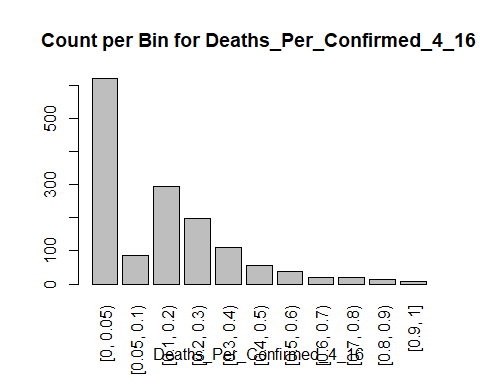
# Observation: ~ 600 of the 700 are <.025. Zoom a little more  
g <- ggplot(SmallNorm, aes(x=Deaths\_Per\_Confirmed\_4\_16))  
g <- g+ geom\_histogram(binwidth=.005) + ggtitle("Histogram of Normalizes Death Rates Zoomed into binwidth = .005")  
g



# Observation - almost all of the 600 are <.005. Clearly some counties have a very low death rate.  
# Don't want to use .05 as the bucket size becasue that will needlessly increase dimensionality.  
# Use 11 bins. The first will be normalized death rate (NDR) <= .05, the second will be NDR >.05 but  
# less than .1 and the rest will be remaining 9 tenths.

### Preprocess the data

Preprocessed\_Data <- SmallNorm  
  
Preprocessed\_Data[, "Pop\_Density"] <- bin\_data(Preprocessed\_Data$Pop\_Density, bins=10, binType = "explicit")  
  
Preprocessed\_Data[, "ET\_Total\_Population"] <- bin\_data(Preprocessed\_Data$ET\_Total\_Population, bins=10, binType = "explicit")  
  
Preprocessed\_Data[, "EM\_Total\_Pop\_Median\_Age"] <- bin\_data(Preprocessed\_Data$EM\_Total\_Pop\_Median\_Age, bins=10, binType = "explicit")  
  
Preprocessed\_Data[, "Deaths\_Per\_Confirmed\_4\_16"] <- bin\_data(Preprocessed\_Data$Deaths\_Per\_Confirmed\_4\_16, bins=c(0,0.05,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0), binType = "explicit")  
  
Preprocessed\_Data[, "Fraction\_Female"] <- bin\_data(Preprocessed\_Data$Fraction\_Female, bins=10, binType = "explicit")  
  
Preprocessed\_Data[, "Fraction\_White"] <- bin\_data(Preprocessed\_Data$Fraction\_White, bins=10, binType = "explicit")  
  
Preprocessed\_Data[, "Fraction\_Black"] <- bin\_data(Preprocessed\_Data$Fraction\_Black, bins=10, binType = "explicit")  
  
Preprocessed\_Data[, "Fraction\_Other"] <- bin\_data(Preprocessed\_Data$Fraction\_Other, bins=10, binType = "explicit")  
  
Preprocessed\_Data[, "Hispanic"] <- bin\_data(Preprocessed\_Data$Hispanic, bins=10, binType = "explicit")  
  
Preprocessed\_Data[, "Fraction\_Less\_Than\_HS"] <- bin\_data(Preprocessed\_Data$Fraction\_Less\_Than\_HS, bins=10, binType = "explicit")  
  
Preprocessed\_Data[, "Fraction\_High\_School\_Grad"] <- bin\_data(Preprocessed\_Data$Fraction\_High\_School\_Grad, bins=10, binType = "explicit")  
  
Preprocessed\_Data[, "Fraction\_Some\_College"] <- bin\_data(Preprocessed\_Data$Fraction\_Some\_College, bins=10, binType = "explicit")  
  
Preprocessed\_Data[, "Fraction\_Bachelors\_or\_Higher"] <- bin\_data(Preprocessed\_Data$Fraction\_Bachelors\_or\_Higher, bins=10, binType = "explicit")  
  
Preprocessed\_Data[, "Fraction\_Disabled"] <- bin\_data(Preprocessed\_Data$Fraction\_Disabled, bins=10, binType = "explicit")  
  
Preprocessed\_Data[, "Fraction\_\_Limited\_English"] <- bin\_data(Preprocessed\_Data$Fraction\_\_Limited\_English, bins=10, binType = "explicit")  
  
Preprocessed\_Data[, "Fraction\_Unemployed"] <- bin\_data(Preprocessed\_Data$Fraction\_Unemployed, bins=10, binType = "explicit")  
  
Preprocessed\_Data[, "Fraction\_Below\_Poverty\_Line"] <- bin\_data(Preprocessed\_Data$Fraction\_Below\_Poverty\_Line, bins=10, binType = "explicit")  
  
Preprocessed\_Data[, "Fraction\_Btwn\_100\_149\_Above\_Poverty\_Line"] <- bin\_data(Preprocessed\_Data$Fraction\_Btwn\_100\_149\_Above\_Poverty\_Line, bins=10, binType = "explicit")  
  
Preprocessed\_Data[, "Fraction\_Not\_Citizen"] <- bin\_data(Preprocessed\_Data$Fraction\_Not\_Citizen, bins=10, binType = "explicit")  
  
counts <- table(Preprocessed\_Data$Deaths\_Per\_Confirmed\_4\_16)  
barplot(counts, main="Count per Bin for Deaths\_Per\_Confirmed\_4\_16",   
 xlab="Deaths\_Per\_Confirmed\_4\_16",las=3)



# Classification

## The fully disretized data is for ARM. Only need to discretize the predicted variable for classification so go back to that,

ClassData<- SmallNorm  
#str(ClassData)  
ClassData$Bin<-Preprocessed\_Data$Deaths\_Per\_Confirmed\_4\_16  
  
# Get rid of the non-discretized Deaths Per Confirmed  
  
ClassData <- ClassData[,-4]  
str(ClassData)

## 'data.frame': 1460 obs. of 20 variables:  
## $ Pop\_Density : num 0.00487 0.00692 0.00469 0.0101 0.00293 ...  
## $ ET\_Total\_Population : num 0.00362 0.01879 0.00386 0.00956 0.0015 ...  
## $ EM\_Total\_Pop\_Median\_Age : num 0.311 0.429 0.382 0.356 0.434 ...  
## $ Fraction\_Female : num 0.828 0.834 0.788 0.868 0.868 ...  
## $ Fraction\_White : num 0.742 0.855 0.965 0.711 0.511 ...  
## $ Fraction\_Black : num 0.2632 0.1302 0.0194 0.2839 0.5443 ...  
## $ Fraction\_Other : num 0.0395 0.0419 0.0263 0.0527 0.0275 ...  
## $ Hispanic : num 0.0233 0.0406 0.0872 0.0325 0.0183 ...  
## $ Population\_Over\_Age\_15 : num 0.00333 0.01938 0.00368 0.00947 0.00133 ...  
## $ Fraction\_Less\_Than\_HS : num 0.2 0.166 0.383 0.299 0.357 ...  
## $ Fraction\_High\_School\_Grad : num 0.561 0.447 0.593 0.556 0.694 ...  
## $ Fraction\_Some\_College : num 0.503 0.59 0.655 0.66 0.542 ...  
## $ Fraction\_Bachelors\_or\_Higher : num 0.2937 0.3484 0.0665 0.1476 0.0755 ...  
## $ Fraction\_Disabled : num 0.604 0.392 0.403 0.667 0.502 ...  
## $ Fraction\_\_Limited\_English : num 0.0264 0.0353 0.0703 0.0331 0.0154 ...  
## $ Fraction\_Unemployed : num 0.184 0.2 0.177 0.505 0.237 ...  
## $ Fraction\_Below\_Poverty\_Line : num 0.366 0.216 0.335 0.454 0.399 ...  
## $ Fraction\_Btwn\_100\_149\_Above\_Poverty\_Line: num 0.418 0.37 0.569 0.588 0.798 ...  
## $ Fraction\_Not\_Citizen : num 0.0402 0.0787 0.1218 0.0572 0.0491 ...  
## $ Bin : Ord.factor w/ 11 levels "[0, 0.05)"<"[0.05, 0.1)"<..: 3 2 1 1 4 1 1 1 6 1 ...

## Naive Bayes With Cross-Fold Validation

#Building a model  
#split data into training and test data sets  
data<-ClassData  
set.seed(300)  
indxTrain <- createDataPartition(y = data$Bin,p = 0.75,list = FALSE)  
training <- data[indxTrain,]  
testing <- data[-indxTrain,]   
  
#Check dimensions of the split   
  
#prop.table(table(training$Bin)) \* 100  
  
#prop.table(table(testing$Bin)) \* 100  
  
# Create variables with and without the bin  
  
x <- training[,-20]  
y <- training$Bin  
  
# Naive Bayesian  
  
model <- train(x,y, 'nb',trControl=trainControl(method='cv',number=5))  
model

## Naive Bayes   
##   
## 1102 samples  
## 19 predictor  
## 11 classes: '[0, 0.05)', '[0.05, 0.1)', '[0.1, 0.2)', '[0.2, 0.3)', '[0.3, 0.4)', '[0.4, 0.5)', '[0.5, 0.6)', '[0.6, 0.7)', '[0.7, 0.8)', '[0.8, 0.9)', '[0.9, 1]'   
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold)   
## Summary of sample sizes: 881, 882, 882, 881, 882   
## Resampling results across tuning parameters:  
##   
## usekernel Accuracy Kappa   
## FALSE 0.2105348 0.05494234  
## TRUE 0.3910654 0.17029269  
##   
## Tuning parameter 'fL' was held constant at a value of 0  
## Tuning  
## parameter 'adjust' was held constant at a value of 1  
## Accuracy was used to select the optimal model using the largest value.  
## The final values used for the model were fL = 0, usekernel = TRUE and adjust  
## = 1.

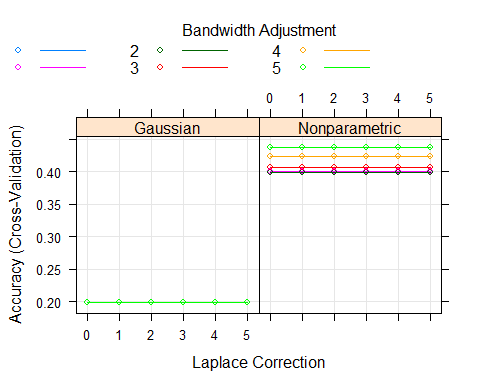
confusionMatrix(model)

## Cross-Validated (5 fold) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction [0, 0.05) [0.05, 0.1) [0.1, 0.2) [0.2, 0.3) [0.3, 0.4) [0.4, 0.5)  
## [0, 0.05) 29.0 0.8 5.3 4.7 3.9 2.1  
## [0.05, 0.1) 1.9 1.7 4.3 2.5 1.1 0.2  
## [0.1, 0.2) 2.5 2.3 6.4 3.2 1.0 0.6  
## [0.2, 0.3) 1.7 0.7 1.4 1.5 0.4 0.1  
## [0.3, 0.4) 1.4 0.1 0.8 0.5 0.0 0.5  
## [0.4, 0.5) 0.8 0.0 0.6 0.5 0.6 0.1  
## [0.5, 0.6) 0.3 0.2 0.3 0.1 0.0 0.0  
## [0.6, 0.7) 0.7 0.1 0.1 0.2 0.0 0.0  
## [0.7, 0.8) 0.1 0.0 0.1 0.0 0.0 0.0  
## [0.8, 0.9) 3.4 0.0 0.7 0.5 0.5 0.3  
## [0.9, 1] 0.5 0.0 0.2 0.0 0.0 0.0  
## Reference  
## Prediction [0.5, 0.6) [0.6, 0.7) [0.7, 0.8) [0.8, 0.9) [0.9, 1]  
## [0, 0.05) 1.9 0.9 1.0 0.8 0.3  
## [0.05, 0.1) 0.2 0.2 0.1 0.0 0.1  
## [0.1, 0.2) 0.1 0.0 0.0 0.0 0.0  
## [0.2, 0.3) 0.1 0.0 0.1 0.0 0.0  
## [0.3, 0.4) 0.0 0.1 0.0 0.0 0.0  
## [0.4, 0.5) 0.0 0.0 0.0 0.0 0.0  
## [0.5, 0.6) 0.1 0.0 0.0 0.0 0.0  
## [0.6, 0.7) 0.0 0.1 0.0 0.0 0.0  
## [0.7, 0.8) 0.0 0.0 0.0 0.0 0.1  
## [0.8, 0.9) 0.2 0.1 0.1 0.2 0.1  
## [0.9, 1] 0.0 0.0 0.1 0.0 0.0  
##   
## Accuracy (average) : 0.3911

# Observation: Model accuracy is 39%. See if it can be tuned  
# Great reference: https://uc-r.github.io/naive\_bayes  
  
# set up tuning grid  
search\_grid <- expand.grid(  
 usekernel = c(TRUE, FALSE),  
 fL = 0:5,  
 adjust = seq(0, 5, by = 1)  
)  
  
# train model  
  
model2 <- train(x,y, 'nb',trControl=trainControl(method='cv',number=5),  
 tuneGrid = search\_grid)  
  
  
# top 5 modesl  
model2$results %>%   
 top\_n(5, wt = Accuracy) %>%  
 arrange(desc(Accuracy))

## usekernel fL adjust Accuracy Kappa AccuracySD KappaSD  
## 1 TRUE 0 5 0.4373001 0.1416737 0.03392749 0.03281815  
## 2 TRUE 1 5 0.4373001 0.1416737 0.03392749 0.03281815  
## 3 TRUE 2 5 0.4373001 0.1416737 0.03392749 0.03281815  
## 4 TRUE 3 5 0.4373001 0.1416737 0.03392749 0.03281815  
## 5 TRUE 4 5 0.4373001 0.1416737 0.03392749 0.03281815  
## 6 TRUE 5 5 0.4373001 0.1416737 0.03392749 0.03281815

plot(model2)



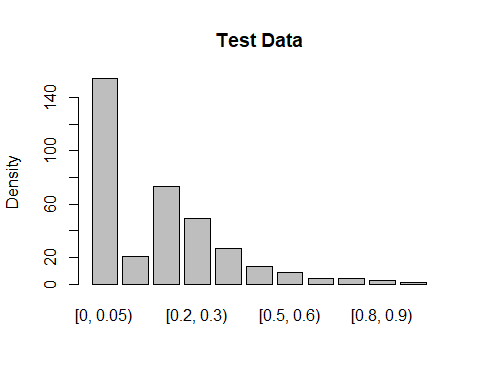
confusionMatrix(model2)

## Cross-Validated (5 fold) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction [0, 0.05) [0.05, 0.1) [0.1, 0.2) [0.2, 0.3) [0.3, 0.4) [0.4, 0.5)  
## [0, 0.05) 36.9 2.8 11.3 8.3 5.6 2.9  
## [0.05, 0.1) 0.2 0.3 1.0 0.3 0.2 0.0  
## [0.1, 0.2) 1.1 1.9 6.2 3.8 0.8 0.5  
## [0.2, 0.3) 0.1 0.1 0.3 0.2 0.2 0.0  
## [0.3, 0.4) 0.6 0.4 0.9 0.5 0.1 0.0  
## [0.4, 0.5) 0.0 0.2 0.0 0.0 0.0 0.1  
## [0.5, 0.6) 0.0 0.1 0.2 0.0 0.0 0.0  
## [0.6, 0.7) 0.0 0.1 0.0 0.0 0.0 0.0  
## [0.7, 0.8) 0.0 0.0 0.0 0.0 0.0 0.0  
## [0.8, 0.9) 3.3 0.1 0.4 0.5 0.6 0.4  
## [0.9, 1] 0.0 0.0 0.0 0.0 0.0 0.0  
## Reference  
## Prediction [0.5, 0.6) [0.6, 0.7) [0.7, 0.8) [0.8, 0.9) [0.9, 1]  
## [0, 0.05) 1.8 1.4 1.0 1.0 0.5  
## [0.05, 0.1) 0.0 0.0 0.0 0.0 0.0  
## [0.1, 0.2) 0.4 0.0 0.0 0.0 0.0  
## [0.2, 0.3) 0.0 0.0 0.0 0.0 0.0  
## [0.3, 0.4) 0.0 0.0 0.0 0.0 0.0  
## [0.4, 0.5) 0.0 0.0 0.0 0.0 0.0  
## [0.5, 0.6) 0.0 0.0 0.0 0.0 0.0  
## [0.6, 0.7) 0.0 0.0 0.0 0.0 0.0  
## [0.7, 0.8) 0.0 0.0 0.0 0.0 0.0  
## [0.8, 0.9) 0.4 0.0 0.3 0.0 0.0  
## [0.9, 1] 0.0 0.0 0.1 0.0 0.0  
##   
## Accuracy (average) : 0.4374

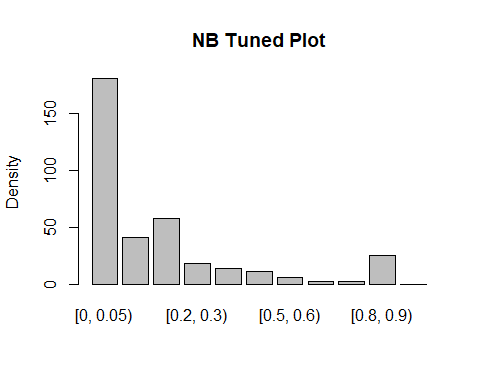
# Look at predicted results  
  
Prediction <- predict(model, newdata=testing)  
#Results <- table(unlist(Prediction),unlist(testing$Bin))  
#Results <-as.data.frame.matrix(Results)  
#Results  
confusionMatrix(Prediction, testing$Bin)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction [0, 0.05) [0.05, 0.1) [0.1, 0.2) [0.2, 0.3) [0.3, 0.4) [0.4, 0.5)  
## [0, 0.05) 111 5 11 21 12 5  
## [0.05, 0.1) 10 6 15 6 2 2  
## [0.1, 0.2) 2 8 29 15 3 1  
## [0.2, 0.3) 5 2 5 1 3 2  
## [0.3, 0.4) 4 0 5 3 0 1  
## [0.4, 0.5) 5 0 3 0 2 0  
## [0.5, 0.6) 4 0 0 1 1 0  
## [0.6, 0.7) 1 0 1 0 0 0  
## [0.7, 0.8) 1 0 1 0 0 0  
## [0.8, 0.9) 11 0 3 2 4 2  
## [0.9, 1] 0 0 0 0 0 0  
## Reference  
## Prediction [0.5, 0.6) [0.6, 0.7) [0.7, 0.8) [0.8, 0.9) [0.9, 1]  
## [0, 0.05) 6 4 3 2 1  
## [0.05, 0.1) 0 0 0 0 0  
## [0.1, 0.2) 0 0 0 0 0  
## [0.2, 0.3) 0 0 0 0 0  
## [0.3, 0.4) 1 0 0 0 0  
## [0.4, 0.5) 1 0 0 0 0  
## [0.5, 0.6) 0 0 0 0 0  
## [0.6, 0.7) 0 0 0 0 0  
## [0.7, 0.8) 0 0 0 0 0  
## [0.8, 0.9) 1 0 1 1 0  
## [0.9, 1] 0 0 0 0 0  
##   
## Overall Statistics  
##   
## Accuracy : 0.4134   
## 95% CI : (0.3619, 0.4664)  
## No Information Rate : 0.4302   
## P-Value [Acc > NIR] : 0.7557   
##   
## Kappa : 0.1972   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: [0, 0.05) Class: [0.05, 0.1) Class: [0.1, 0.2)  
## Sensitivity 0.7208 0.28571 0.39726  
## Specificity 0.6569 0.89614 0.89825  
## Pos Pred Value 0.6133 0.14634 0.50000  
## Neg Pred Value 0.7571 0.95268 0.85333  
## Prevalence 0.4302 0.05866 0.20391  
## Detection Rate 0.3101 0.01676 0.08101  
## Detection Prevalence 0.5056 0.11453 0.16201  
## Balanced Accuracy 0.6888 0.59093 0.64775  
## Class: [0.2, 0.3) Class: [0.3, 0.4) Class: [0.4, 0.5)  
## Sensitivity 0.020408 0.00000 0.00000  
## Specificity 0.944984 0.95770 0.96812  
## Pos Pred Value 0.055556 0.00000 0.00000  
## Neg Pred Value 0.858824 0.92151 0.96254  
## Prevalence 0.136872 0.07542 0.03631  
## Detection Rate 0.002793 0.00000 0.00000  
## Detection Prevalence 0.050279 0.03911 0.03073  
## Balanced Accuracy 0.482696 0.47885 0.48406  
## Class: [0.5, 0.6) Class: [0.6, 0.7) Class: [0.7, 0.8)  
## Sensitivity 0.00000 0.000000 0.000000  
## Specificity 0.98281 0.994350 0.994350  
## Pos Pred Value 0.00000 0.000000 0.000000  
## Neg Pred Value 0.97443 0.988764 0.988764  
## Prevalence 0.02514 0.011173 0.011173  
## Detection Rate 0.00000 0.000000 0.000000  
## Detection Prevalence 0.01676 0.005587 0.005587  
## Balanced Accuracy 0.49140 0.497175 0.497175  
## Class: [0.8, 0.9) Class: [0.9, 1]  
## Sensitivity 0.333333 0.000000  
## Specificity 0.932394 1.000000  
## Pos Pred Value 0.040000 NaN  
## Neg Pred Value 0.993994 0.997207  
## Prevalence 0.008380 0.002793  
## Detection Rate 0.002793 0.000000  
## Detection Prevalence 0.069832 0.000000  
## Balanced Accuracy 0.632864 0.500000

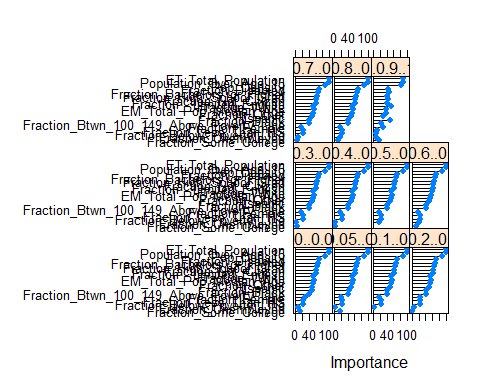
plot(testing$Bin, ylab="Density",main="Test Data")



plot( Prediction, ylab = "Density", main = "NB Tuned Plot")



# Plot variable performance  
  
VarImport <- varImp(model2)  
plot(VarImport)



### Observation: Traiing model accuracy up to 44% with kappa .14 and testing to 41%. The model is predicting vales for most bins. The driving variables are population totals, pop over 15 and density - not wealth / poverty indicatros at all. The variables are not independent so that my be a good reason for the poor results.

## KNN with Cross Fold Validation

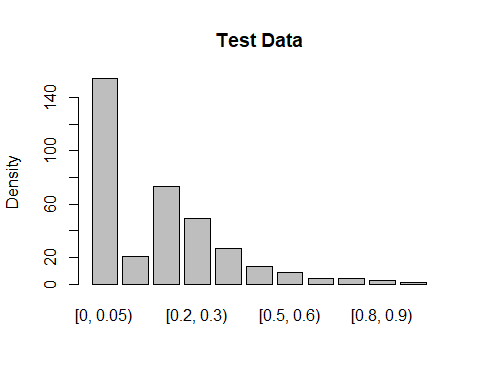
#str(training)  
model <- train(x,y, 'knn',trControl=trainControl(method='cv',number=5), tuneLength=20)  
model

## k-Nearest Neighbors   
##   
## 1102 samples  
## 19 predictor  
## 11 classes: '[0, 0.05)', '[0.05, 0.1)', '[0.1, 0.2)', '[0.2, 0.3)', '[0.3, 0.4)', '[0.4, 0.5)', '[0.5, 0.6)', '[0.6, 0.7)', '[0.7, 0.8)', '[0.8, 0.9)', '[0.9, 1]'   
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold)   
## Summary of sample sizes: 881, 882, 881, 882, 882   
## Resampling results across tuning parameters:  
##   
## k Accuracy Kappa   
## 5 0.4038462 0.1517833  
## 7 0.4228795 0.1601589  
## 9 0.4410202 0.1749647  
## 11 0.4437680 0.1747425  
## 13 0.4519498 0.1779542  
## 15 0.4564541 0.1772839  
## 17 0.4664459 0.1875238  
## 19 0.4564624 0.1719570  
## 21 0.4610119 0.1754983  
## 23 0.4619210 0.1730707  
## 25 0.4601234 0.1693179  
## 27 0.4637392 0.1714888  
## 29 0.4673632 0.1760936  
## 31 0.4682682 0.1755987  
## 33 0.4709790 0.1756571  
## 35 0.4664418 0.1683490  
## 37 0.4727766 0.1767141  
## 39 0.4664212 0.1642377  
## 41 0.4691567 0.1706356  
## 43 0.4746072 0.1767629  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was k = 43.

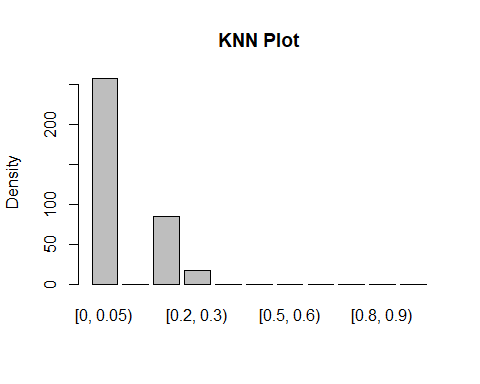
# Look at predicted results  
  
Prediction <- predict(model, newdata=testing)  
#Results <- table(unlist(Prediction),unlist(testing$Bin))  
#Results <-as.data.frame.matrix(Results)  
#Results  
confusionMatrix(Prediction, testing$Bin)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction [0, 0.05) [0.05, 0.1) [0.1, 0.2) [0.2, 0.3) [0.3, 0.4) [0.4, 0.5)  
## [0, 0.05) 137 9 32 30 19 10  
## [0.05, 0.1) 0 0 0 0 0 0  
## [0.1, 0.2) 11 12 35 16 8 2  
## [0.2, 0.3) 6 0 6 3 0 1  
## [0.3, 0.4) 0 0 0 0 0 0  
## [0.4, 0.5) 0 0 0 0 0 0  
## [0.5, 0.6) 0 0 0 0 0 0  
## [0.6, 0.7) 0 0 0 0 0 0  
## [0.7, 0.8) 0 0 0 0 0 0  
## [0.8, 0.9) 0 0 0 0 0 0  
## [0.9, 1] 0 0 0 0 0 0  
## Reference  
## Prediction [0.5, 0.6) [0.6, 0.7) [0.7, 0.8) [0.8, 0.9) [0.9, 1]  
## [0, 0.05) 8 4 4 3 1  
## [0.05, 0.1) 0 0 0 0 0  
## [0.1, 0.2) 0 0 0 0 0  
## [0.2, 0.3) 1 0 0 0 0  
## [0.3, 0.4) 0 0 0 0 0  
## [0.4, 0.5) 0 0 0 0 0  
## [0.5, 0.6) 0 0 0 0 0  
## [0.6, 0.7) 0 0 0 0 0  
## [0.7, 0.8) 0 0 0 0 0  
## [0.8, 0.9) 0 0 0 0 0  
## [0.9, 1] 0 0 0 0 0  
##   
## Overall Statistics  
##   
## Accuracy : 0.4888   
## 95% CI : (0.4359, 0.5419)  
## No Information Rate : 0.4302   
## P-Value [Acc > NIR] : 0.01459   
##   
## Kappa : 0.1973   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: [0, 0.05) Class: [0.05, 0.1) Class: [0.1, 0.2)  
## Sensitivity 0.8896 0.00000 0.47945  
## Specificity 0.4118 1.00000 0.82807  
## Pos Pred Value 0.5331 NaN 0.41667  
## Neg Pred Value 0.8317 0.94134 0.86131  
## Prevalence 0.4302 0.05866 0.20391  
## Detection Rate 0.3827 0.00000 0.09777  
## Detection Prevalence 0.7179 0.00000 0.23464  
## Balanced Accuracy 0.6507 0.50000 0.65376  
## Class: [0.2, 0.3) Class: [0.3, 0.4) Class: [0.4, 0.5)  
## Sensitivity 0.06122 0.00000 0.00000  
## Specificity 0.95469 1.00000 1.00000  
## Pos Pred Value 0.17647 NaN NaN  
## Neg Pred Value 0.86510 0.92458 0.96369  
## Prevalence 0.13687 0.07542 0.03631  
## Detection Rate 0.00838 0.00000 0.00000  
## Detection Prevalence 0.04749 0.00000 0.00000  
## Balanced Accuracy 0.50796 0.50000 0.50000  
## Class: [0.5, 0.6) Class: [0.6, 0.7) Class: [0.7, 0.8)  
## Sensitivity 0.00000 0.00000 0.00000  
## Specificity 1.00000 1.00000 1.00000  
## Pos Pred Value NaN NaN NaN  
## Neg Pred Value 0.97486 0.98883 0.98883  
## Prevalence 0.02514 0.01117 0.01117  
## Detection Rate 0.00000 0.00000 0.00000  
## Detection Prevalence 0.00000 0.00000 0.00000  
## Balanced Accuracy 0.50000 0.50000 0.50000  
## Class: [0.8, 0.9) Class: [0.9, 1]  
## Sensitivity 0.00000 0.000000  
## Specificity 1.00000 1.000000  
## Pos Pred Value NaN NaN  
## Neg Pred Value 0.99162 0.997207  
## Prevalence 0.00838 0.002793  
## Detection Rate 0.00000 0.000000  
## Detection Prevalence 0.00000 0.000000  
## Balanced Accuracy 0.50000 0.500000

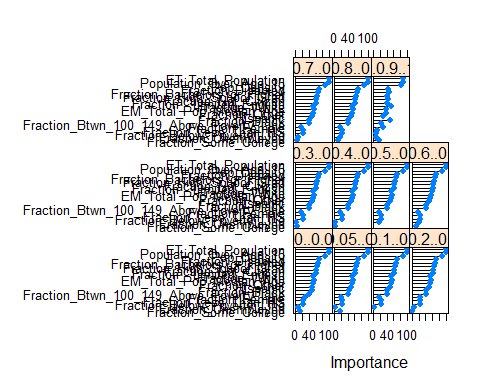
plot(testing$Bin, ylab="Density",main="Test Data")



plot( Prediction, ylab = "Density", main = "KNN Plot")



# Plot variable performance  
  
VarImport <- varImp(model)  
plot(VarImport)



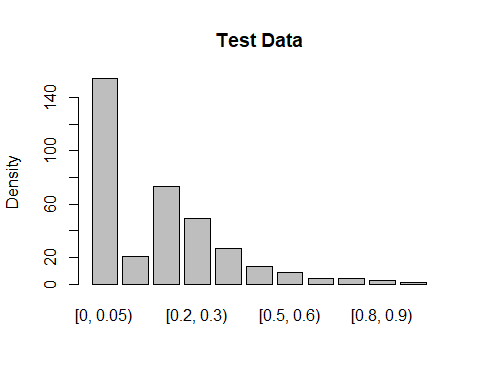
### Observation: 20 values of K were attempted. K=43 was best with an training accuracy of 47% and kappa .18 The testing accuracy is 49% This model is putting most of the predictions in the first bin and a few in the 3rd and 4th and none elsewhere

## RF with Cross Fold Validation

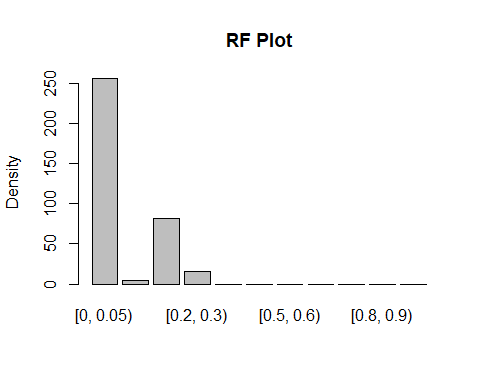
# split data into training and test data sets  
data<-ClassData  
set.seed(300)  
indxTrain <- createDataPartition(y = data$Bin,p = 0.75,list = FALSE)  
training <- data[indxTrain,]  
testing <- data[-indxTrain,]   
  
#Check dimensions of the split   
  
#prop.table(table(training$Bin)) \* 100  
  
#prop.table(table(testing$Bin)) \* 100  
  
# Create variables with and without the bin  
  
x <- training[,-20]  
y <- training$Bin  
  
# Reference : https://machinelearningmastery.com/tune-machine-learning-algorithms-in-r/  
# Create model with default paramters  
  
# mtry: Number of variables randomly sampled as candidates at each split.  
# ntree: Number of trees to grow.(default is 500)  
control <- trainControl(method="repeatedcv", number=10, repeats=3)  
seed <- 7  
metric <- "Accuracy"  
set.seed(seed)  
mtry <- sqrt(ncol(x))  
tunegrid <- expand.grid(.mtry=mtry)  
model <- train(Bin~., data=training, method="rf", metric=metric, tuneGrid=tunegrid, trControl=control)  
print(model)

## Random Forest   
##   
## 1102 samples  
## 19 predictor  
## 11 classes: '[0, 0.05)', '[0.05, 0.1)', '[0.1, 0.2)', '[0.2, 0.3)', '[0.3, 0.4)', '[0.4, 0.5)', '[0.5, 0.6)', '[0.6, 0.7)', '[0.7, 0.8)', '[0.8, 0.9)', '[0.9, 1]'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold, repeated 3 times)   
## Summary of sample sizes: 990, 993, 989, 991, 992, 992, ...   
## Resampling results:  
##   
## Accuracy Kappa   
## 0.4921092 0.2277347  
##   
## Tuning parameter 'mtry' was held constant at a value of 4.358899

# Observation: That yielded about 49% training accuracy.   
  
Prediction <- predict(model, newdata=testing)  
plot(testing$Bin, ylab="Density",main="Test Data")



plot( Prediction, ylab = "Density", main = "RF Plot")



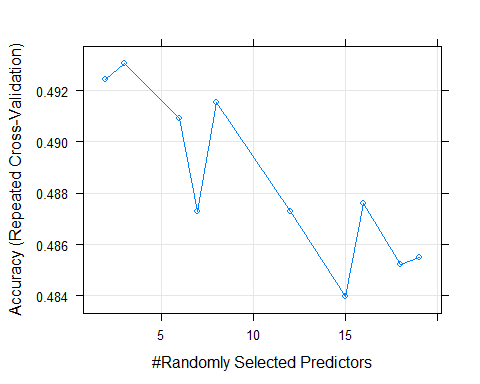
# Results <- table(unlist(Prediction),unlist(testing$Bin))  
# Results <-as.data.frame.matrix(Results)  
# Results  
confusionMatrix(Prediction, testing$Bin)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction [0, 0.05) [0.05, 0.1) [0.1, 0.2) [0.2, 0.3) [0.3, 0.4) [0.4, 0.5)  
## [0, 0.05) 144 9 23 31 19 10  
## [0.05, 0.1) 2 0 1 1 1 0  
## [0.1, 0.2) 7 10 44 13 6 1  
## [0.2, 0.3) 1 2 5 4 1 2  
## [0.3, 0.4) 0 0 0 0 0 0  
## [0.4, 0.5) 0 0 0 0 0 0  
## [0.5, 0.6) 0 0 0 0 0 0  
## [0.6, 0.7) 0 0 0 0 0 0  
## [0.7, 0.8) 0 0 0 0 0 0  
## [0.8, 0.9) 0 0 0 0 0 0  
## [0.9, 1] 0 0 0 0 0 0  
## Reference  
## Prediction [0.5, 0.6) [0.6, 0.7) [0.7, 0.8) [0.8, 0.9) [0.9, 1]  
## [0, 0.05) 8 4 4 3 1  
## [0.05, 0.1) 0 0 0 0 0  
## [0.1, 0.2) 1 0 0 0 0  
## [0.2, 0.3) 0 0 0 0 0  
## [0.3, 0.4) 0 0 0 0 0  
## [0.4, 0.5) 0 0 0 0 0  
## [0.5, 0.6) 0 0 0 0 0  
## [0.6, 0.7) 0 0 0 0 0  
## [0.7, 0.8) 0 0 0 0 0  
## [0.8, 0.9) 0 0 0 0 0  
## [0.9, 1] 0 0 0 0 0  
##   
## Overall Statistics  
##   
## Accuracy : 0.5363   
## 95% CI : (0.4831, 0.5889)  
## No Information Rate : 0.4302   
## P-Value [Acc > NIR] : 3.445e-05   
##   
## Kappa : 0.2745   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: [0, 0.05) Class: [0.05, 0.1) Class: [0.1, 0.2)  
## Sensitivity 0.9351 0.00000 0.6027  
## Specificity 0.4510 0.98516 0.8667  
## Pos Pred Value 0.5625 0.00000 0.5366  
## Neg Pred Value 0.9020 0.94051 0.8949  
## Prevalence 0.4302 0.05866 0.2039  
## Detection Rate 0.4022 0.00000 0.1229  
## Detection Prevalence 0.7151 0.01397 0.2291  
## Balanced Accuracy 0.6930 0.49258 0.7347  
## Class: [0.2, 0.3) Class: [0.3, 0.4) Class: [0.4, 0.5)  
## Sensitivity 0.08163 0.00000 0.00000  
## Specificity 0.96440 1.00000 1.00000  
## Pos Pred Value 0.26667 NaN NaN  
## Neg Pred Value 0.86880 0.92458 0.96369  
## Prevalence 0.13687 0.07542 0.03631  
## Detection Rate 0.01117 0.00000 0.00000  
## Detection Prevalence 0.04190 0.00000 0.00000  
## Balanced Accuracy 0.52302 0.50000 0.50000  
## Class: [0.5, 0.6) Class: [0.6, 0.7) Class: [0.7, 0.8)  
## Sensitivity 0.00000 0.00000 0.00000  
## Specificity 1.00000 1.00000 1.00000  
## Pos Pred Value NaN NaN NaN  
## Neg Pred Value 0.97486 0.98883 0.98883  
## Prevalence 0.02514 0.01117 0.01117  
## Detection Rate 0.00000 0.00000 0.00000  
## Detection Prevalence 0.00000 0.00000 0.00000  
## Balanced Accuracy 0.50000 0.50000 0.50000  
## Class: [0.8, 0.9) Class: [0.9, 1]  
## Sensitivity 0.00000 0.000000  
## Specificity 1.00000 1.000000  
## Pos Pred Value NaN NaN  
## Neg Pred Value 0.99162 0.997207  
## Prevalence 0.00838 0.002793  
## Detection Rate 0.00000 0.000000  
## Detection Prevalence 0.00000 0.000000  
## Balanced Accuracy 0.50000 0.500000

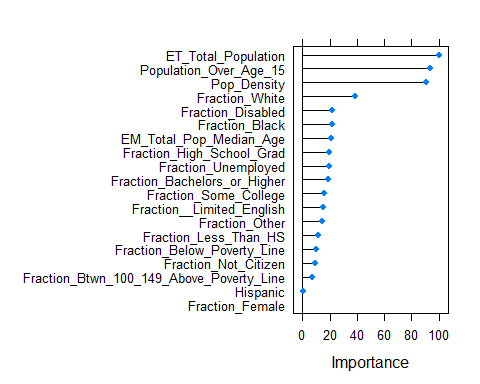
# Can we tune it? Tune on mytry using random search  
  
# Random Search  
control <- trainControl(method="repeatedcv", number=10, repeats=3, search="random")  
set.seed(seed)  
mtry <- sqrt(ncol(x))  
model2 <- train(Bin~., data=training, method="rf", metric=metric, tuneLength=15, trControl=control)  
print(model2)

## Random Forest   
##   
## 1102 samples  
## 19 predictor  
## 11 classes: '[0, 0.05)', '[0.05, 0.1)', '[0.1, 0.2)', '[0.2, 0.3)', '[0.3, 0.4)', '[0.4, 0.5)', '[0.5, 0.6)', '[0.6, 0.7)', '[0.7, 0.8)', '[0.8, 0.9)', '[0.9, 1]'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold, repeated 3 times)   
## Summary of sample sizes: 990, 993, 989, 991, 992, 992, ...   
## Resampling results across tuning parameters:  
##   
## mtry Accuracy Kappa   
## 2 0.4924485 0.2210049  
## 3 0.4930705 0.2265736  
## 6 0.4909163 0.2268757  
## 7 0.4873012 0.2224749  
## 8 0.4915278 0.2286648  
## 12 0.4873015 0.2241460  
## 15 0.4839627 0.2201392  
## 16 0.4876185 0.2261520  
## 18 0.4852242 0.2217693  
## 19 0.4854862 0.2225165  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was mtry = 3.

# Random selection of mtry esssentially same results - ACcuracy = 0.49 Kappa = 0.23  
plot(model2)



# Plot variable performance  
  
VarImport <- varImp(model)  
plot(VarImport)



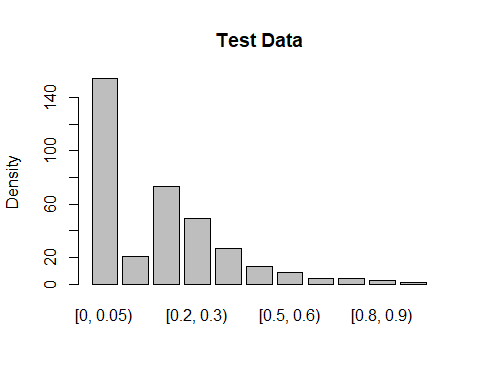
### Observation: This produced 49% Testing accuracy and Kappa = .23 but even more that KNN it is putting all the predictions into the first bin. Testing accuracy was 54% with Kappa .27

## Try SVM - Linear

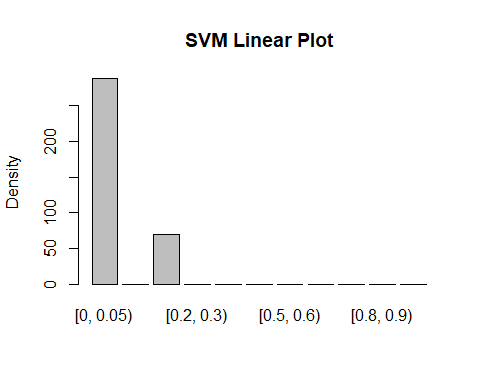
trctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3)  
set.seed(3233)  
   
model <- train(Bin ~., data = training, method = "svmLinear",  
 trControl=trctrl,  
 preProcess = c("center", "scale"),  
 tuneLength = 10)  
model

## Support Vector Machines with Linear Kernel   
##   
## 1102 samples  
## 19 predictor  
## 11 classes: '[0, 0.05)', '[0.05, 0.1)', '[0.1, 0.2)', '[0.2, 0.3)', '[0.3, 0.4)', '[0.4, 0.5)', '[0.5, 0.6)', '[0.6, 0.7)', '[0.7, 0.8)', '[0.8, 0.9)', '[0.9, 1]'   
##   
## Pre-processing: centered (19), scaled (19)   
## Resampling: Cross-Validated (10 fold, repeated 3 times)   
## Summary of sample sizes: 992, 992, 990, 993, 991, 990, ...   
## Resampling results:  
##   
## Accuracy Kappa   
## 0.4900147 0.1824628  
##   
## Tuning parameter 'C' was held constant at a value of 1

# Training accuracy is 49% and Kappa is 18%  
  
Prediction <- predict(model, newdata=testing)  
plot(testing$Bin, ylab="Density",main="Test Data")



plot( Prediction, ylab = "Density", main = "SVM Linear Plot")



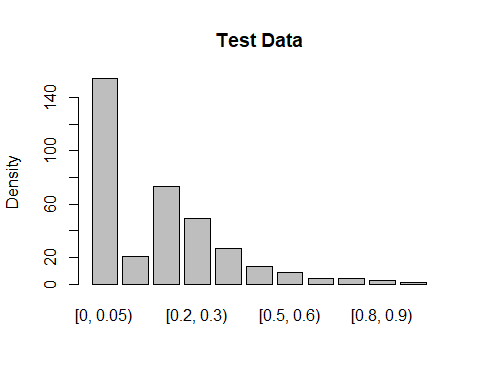
# Results <- table(unlist(Prediction),unlist(testing$Bin))  
# Results <-as.data.frame.matrix(Results)  
# Results  
confusionMatrix(Prediction, testing$Bin)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction [0, 0.05) [0.05, 0.1) [0.1, 0.2) [0.2, 0.3) [0.3, 0.4) [0.4, 0.5)  
## [0, 0.05) 148 15 33 37 23 11  
## [0.05, 0.1) 0 0 0 0 0 0  
## [0.1, 0.2) 6 6 40 12 4 2  
## [0.2, 0.3) 0 0 0 0 0 0  
## [0.3, 0.4) 0 0 0 0 0 0  
## [0.4, 0.5) 0 0 0 0 0 0  
## [0.5, 0.6) 0 0 0 0 0 0  
## [0.6, 0.7) 0 0 0 0 0 0  
## [0.7, 0.8) 0 0 0 0 0 0  
## [0.8, 0.9) 0 0 0 0 0 0  
## [0.9, 1] 0 0 0 0 0 0  
## Reference  
## Prediction [0.5, 0.6) [0.6, 0.7) [0.7, 0.8) [0.8, 0.9) [0.9, 1]  
## [0, 0.05) 9 4 4 3 1  
## [0.05, 0.1) 0 0 0 0 0  
## [0.1, 0.2) 0 0 0 0 0  
## [0.2, 0.3) 0 0 0 0 0  
## [0.3, 0.4) 0 0 0 0 0  
## [0.4, 0.5) 0 0 0 0 0  
## [0.5, 0.6) 0 0 0 0 0  
## [0.6, 0.7) 0 0 0 0 0  
## [0.7, 0.8) 0 0 0 0 0  
## [0.8, 0.9) 0 0 0 0 0  
## [0.9, 1] 0 0 0 0 0  
##   
## Overall Statistics  
##   
## Accuracy : 0.5251   
## 95% CI : (0.472, 0.5779)  
## No Information Rate : 0.4302   
## P-Value [Acc > NIR] : 0.0001876   
##   
## Kappa : 0.2267   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: [0, 0.05) Class: [0.05, 0.1) Class: [0.1, 0.2)  
## Sensitivity 0.9610 0.00000 0.5479  
## Specificity 0.3137 1.00000 0.8947  
## Pos Pred Value 0.5139 NaN 0.5714  
## Neg Pred Value 0.9143 0.94134 0.8854  
## Prevalence 0.4302 0.05866 0.2039  
## Detection Rate 0.4134 0.00000 0.1117  
## Detection Prevalence 0.8045 0.00000 0.1955  
## Balanced Accuracy 0.6374 0.50000 0.7213  
## Class: [0.2, 0.3) Class: [0.3, 0.4) Class: [0.4, 0.5)  
## Sensitivity 0.0000 0.00000 0.00000  
## Specificity 1.0000 1.00000 1.00000  
## Pos Pred Value NaN NaN NaN  
## Neg Pred Value 0.8631 0.92458 0.96369  
## Prevalence 0.1369 0.07542 0.03631  
## Detection Rate 0.0000 0.00000 0.00000  
## Detection Prevalence 0.0000 0.00000 0.00000  
## Balanced Accuracy 0.5000 0.50000 0.50000  
## Class: [0.5, 0.6) Class: [0.6, 0.7) Class: [0.7, 0.8)  
## Sensitivity 0.00000 0.00000 0.00000  
## Specificity 1.00000 1.00000 1.00000  
## Pos Pred Value NaN NaN NaN  
## Neg Pred Value 0.97486 0.98883 0.98883  
## Prevalence 0.02514 0.01117 0.01117  
## Detection Rate 0.00000 0.00000 0.00000  
## Detection Prevalence 0.00000 0.00000 0.00000  
## Balanced Accuracy 0.50000 0.50000 0.50000  
## Class: [0.8, 0.9) Class: [0.9, 1]  
## Sensitivity 0.00000 0.000000  
## Specificity 1.00000 1.000000  
## Pos Pred Value NaN NaN  
## Neg Pred Value 0.99162 0.997207  
## Prevalence 0.00838 0.002793  
## Detection Rate 0.00000 0.000000  
## Detection Prevalence 0.00000 0.000000  
## Balanced Accuracy 0.50000 0.500000

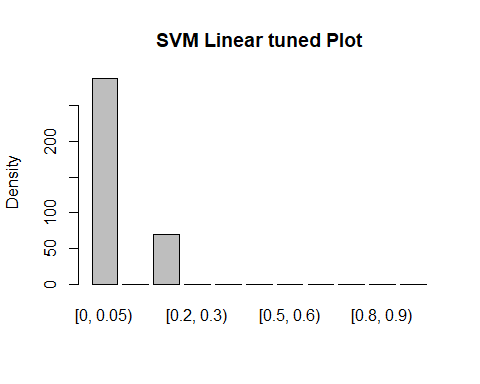
# 49% accuracy and kappa ..18 for SVM Linear training and C=1. Test data accuracy = 53% and Kapp = .23  
  
# Tune  
grid <- expand.grid(C = c(0,0.01, 0.05, 0.1, 0.25, 0.5, 0.75, 1, 1.25, 1.5, 1.75, 2,5))  
model <- train(Bin ~., data = training, method = "svmLinear",  
 trControl=trctrl,  
 preProcess = c("center", "scale"),  
 tuneGrid = grid,  
 tuneLength = 10)  
model

## Support Vector Machines with Linear Kernel   
##   
## 1102 samples  
## 19 predictor  
## 11 classes: '[0, 0.05)', '[0.05, 0.1)', '[0.1, 0.2)', '[0.2, 0.3)', '[0.3, 0.4)', '[0.4, 0.5)', '[0.5, 0.6)', '[0.6, 0.7)', '[0.7, 0.8)', '[0.8, 0.9)', '[0.9, 1]'   
##   
## Pre-processing: centered (19), scaled (19)   
## Resampling: Cross-Validated (10 fold, repeated 3 times)   
## Summary of sample sizes: 989, 993, 995, 990, 990, 992, ...   
## Resampling results across tuning parameters:  
##   
## C Accuracy Kappa   
## 0.00 NaN NaN  
## 0.01 0.4743139 0.1285578  
## 0.05 0.4855237 0.1619015  
## 0.10 0.4870472 0.1696547  
## 0.25 0.4891332 0.1772002  
## 0.50 0.4894143 0.1801692  
## 0.75 0.4906376 0.1830753  
## 1.00 0.4906376 0.1832692  
## 1.25 0.4912549 0.1846984  
## 1.50 0.4909519 0.1845389  
## 1.75 0.4912522 0.1849560  
## 2.00 0.4909492 0.1847587  
## 5.00 0.4894254 0.1825105  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was C = 1.25.

Prediction <- predict(model, newdata=testing)  
plot(testing$Bin, ylab="Density",main="Test Data")



plot( Prediction, ylab = "Density", main = "SVM Linear tuned Plot")



# Results <- table(unlist(Prediction),unlist(testing$Bin))  
# Results <-as.data.frame.matrix(Results)  
# Results  
confusionMatrix(Prediction, testing$Bin)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction [0, 0.05) [0.05, 0.1) [0.1, 0.2) [0.2, 0.3) [0.3, 0.4) [0.4, 0.5)  
## [0, 0.05) 148 15 33 37 23 11  
## [0.05, 0.1) 0 0 0 0 0 0  
## [0.1, 0.2) 6 6 40 12 4 2  
## [0.2, 0.3) 0 0 0 0 0 0  
## [0.3, 0.4) 0 0 0 0 0 0  
## [0.4, 0.5) 0 0 0 0 0 0  
## [0.5, 0.6) 0 0 0 0 0 0  
## [0.6, 0.7) 0 0 0 0 0 0  
## [0.7, 0.8) 0 0 0 0 0 0  
## [0.8, 0.9) 0 0 0 0 0 0  
## [0.9, 1] 0 0 0 0 0 0  
## Reference  
## Prediction [0.5, 0.6) [0.6, 0.7) [0.7, 0.8) [0.8, 0.9) [0.9, 1]  
## [0, 0.05) 9 4 4 3 1  
## [0.05, 0.1) 0 0 0 0 0  
## [0.1, 0.2) 0 0 0 0 0  
## [0.2, 0.3) 0 0 0 0 0  
## [0.3, 0.4) 0 0 0 0 0  
## [0.4, 0.5) 0 0 0 0 0  
## [0.5, 0.6) 0 0 0 0 0  
## [0.6, 0.7) 0 0 0 0 0  
## [0.7, 0.8) 0 0 0 0 0  
## [0.8, 0.9) 0 0 0 0 0  
## [0.9, 1] 0 0 0 0 0  
##   
## Overall Statistics  
##   
## Accuracy : 0.5251   
## 95% CI : (0.472, 0.5779)  
## No Information Rate : 0.4302   
## P-Value [Acc > NIR] : 0.0001876   
##   
## Kappa : 0.2267   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: [0, 0.05) Class: [0.05, 0.1) Class: [0.1, 0.2)  
## Sensitivity 0.9610 0.00000 0.5479  
## Specificity 0.3137 1.00000 0.8947  
## Pos Pred Value 0.5139 NaN 0.5714  
## Neg Pred Value 0.9143 0.94134 0.8854  
## Prevalence 0.4302 0.05866 0.2039  
## Detection Rate 0.4134 0.00000 0.1117  
## Detection Prevalence 0.8045 0.00000 0.1955  
## Balanced Accuracy 0.6374 0.50000 0.7213  
## Class: [0.2, 0.3) Class: [0.3, 0.4) Class: [0.4, 0.5)  
## Sensitivity 0.0000 0.00000 0.00000  
## Specificity 1.0000 1.00000 1.00000  
## Pos Pred Value NaN NaN NaN  
## Neg Pred Value 0.8631 0.92458 0.96369  
## Prevalence 0.1369 0.07542 0.03631  
## Detection Rate 0.0000 0.00000 0.00000  
## Detection Prevalence 0.0000 0.00000 0.00000  
## Balanced Accuracy 0.5000 0.50000 0.50000  
## Class: [0.5, 0.6) Class: [0.6, 0.7) Class: [0.7, 0.8)  
## Sensitivity 0.00000 0.00000 0.00000  
## Specificity 1.00000 1.00000 1.00000  
## Pos Pred Value NaN NaN NaN  
## Neg Pred Value 0.97486 0.98883 0.98883  
## Prevalence 0.02514 0.01117 0.01117  
## Detection Rate 0.00000 0.00000 0.00000  
## Detection Prevalence 0.00000 0.00000 0.00000  
## Balanced Accuracy 0.50000 0.50000 0.50000  
## Class: [0.8, 0.9) Class: [0.9, 1]  
## Sensitivity 0.00000 0.000000  
## Specificity 1.00000 1.000000  
## Pos Pred Value NaN NaN  
## Neg Pred Value 0.99162 0.997207  
## Prevalence 0.00838 0.002793  
## Detection Rate 0.00000 0.000000  
## Detection Prevalence 0.00000 0.000000  
## Balanced Accuracy 0.50000 0.500000

### Observation: 49% accuracy and kappa .18 for SVM Linear training and C=1. Test data accuracy = 53% and Kapp = .23

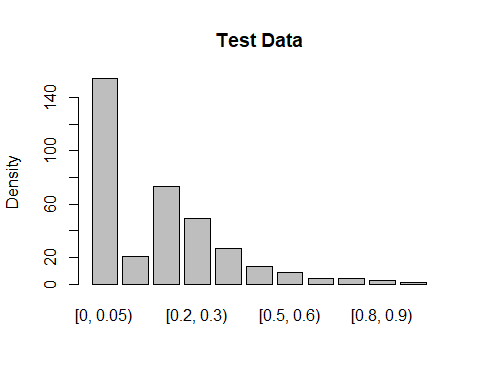
### Tuned model - 49% accuracy and kapps .18 for training SVM Linear and selected C=1.25 - no real change. This model also predicts only 2 bins

## Try SVM - Radial Basis

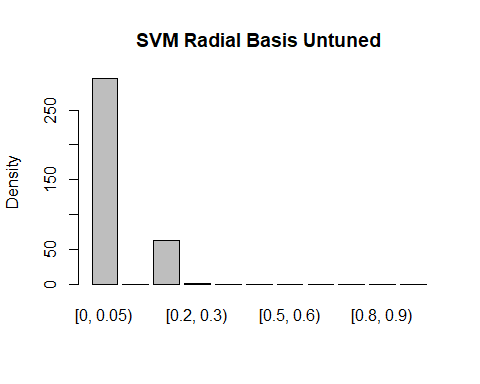
trctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3)  
set.seed(3233)  
   
model <- train(Bin ~., data = training, method = "svmRadial",  
 trControl=trctrl,  
 preProcess = c("center", "scale"),  
 tuneLength = 10)  
model

## Support Vector Machines with Radial Basis Function Kernel   
##   
## 1102 samples  
## 19 predictor  
## 11 classes: '[0, 0.05)', '[0.05, 0.1)', '[0.1, 0.2)', '[0.2, 0.3)', '[0.3, 0.4)', '[0.4, 0.5)', '[0.5, 0.6)', '[0.6, 0.7)', '[0.7, 0.8)', '[0.8, 0.9)', '[0.9, 1]'   
##   
## Pre-processing: centered (19), scaled (19)   
## Resampling: Cross-Validated (10 fold, repeated 3 times)   
## Summary of sample sizes: 992, 992, 990, 993, 991, 990, ...   
## Resampling results across tuning parameters:  
##   
## C Accuracy Kappa   
## 0.25 0.4746163 0.1396689  
## 0.50 0.4821876 0.1617583  
## 1.00 0.4797493 0.1688831  
## 2.00 0.4776799 0.1816230  
## 4.00 0.4737023 0.1953858  
## 8.00 0.4594494 0.1877133  
## 16.00 0.4455576 0.1837291  
## 32.00 0.4216999 0.1668104  
## 64.00 0.3926823 0.1420304  
## 128.00 0.3814467 0.1367549  
##   
## Tuning parameter 'sigma' was held constant at a value of 0.07292807  
## Accuracy was used to select the optimal model using the largest value.  
## The final values used for the model were sigma = 0.07292807 and C = 0.5.

Prediction <- predict(model, newdata=testing)  
plot(testing$Bin, ylab="Density",main="Test Data")



plot( Prediction, ylab = "Density", main = "SVM Radial Basis Untuned")



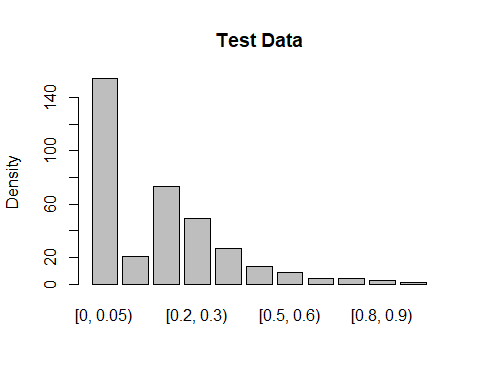
# Results <- table(unlist(Prediction),unlist(testing$Bin))  
# Results <-as.data.frame.matrix(Results)  
# Results  
confusionMatrix(Prediction, testing$Bin)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction [0, 0.05) [0.05, 0.1) [0.1, 0.2) [0.2, 0.3) [0.3, 0.4) [0.4, 0.5)  
## [0, 0.05) 150 14 38 37 24 11  
## [0.05, 0.1) 0 0 0 0 0 0  
## [0.1, 0.2) 4 7 34 12 3 2  
## [0.2, 0.3) 0 0 1 0 0 0  
## [0.3, 0.4) 0 0 0 0 0 0  
## [0.4, 0.5) 0 0 0 0 0 0  
## [0.5, 0.6) 0 0 0 0 0 0  
## [0.6, 0.7) 0 0 0 0 0 0  
## [0.7, 0.8) 0 0 0 0 0 0  
## [0.8, 0.9) 0 0 0 0 0 0  
## [0.9, 1] 0 0 0 0 0 0  
## Reference  
## Prediction [0.5, 0.6) [0.6, 0.7) [0.7, 0.8) [0.8, 0.9) [0.9, 1]  
## [0, 0.05) 9 4 4 3 1  
## [0.05, 0.1) 0 0 0 0 0  
## [0.1, 0.2) 0 0 0 0 0  
## [0.2, 0.3) 0 0 0 0 0  
## [0.3, 0.4) 0 0 0 0 0  
## [0.4, 0.5) 0 0 0 0 0  
## [0.5, 0.6) 0 0 0 0 0  
## [0.6, 0.7) 0 0 0 0 0  
## [0.7, 0.8) 0 0 0 0 0  
## [0.8, 0.9) 0 0 0 0 0  
## [0.9, 1] 0 0 0 0 0  
##   
## Overall Statistics  
##   
## Accuracy : 0.514   
## 95% CI : (0.4609, 0.5668)  
## No Information Rate : 0.4302   
## P-Value [Acc > NIR] : 0.0008631   
##   
## Kappa : 0.203   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: [0, 0.05) Class: [0.05, 0.1) Class: [0.1, 0.2)  
## Sensitivity 0.9740 0.00000 0.46575  
## Specificity 0.2892 1.00000 0.90175  
## Pos Pred Value 0.5085 NaN 0.54839  
## Neg Pred Value 0.9365 0.94134 0.86824  
## Prevalence 0.4302 0.05866 0.20391  
## Detection Rate 0.4190 0.00000 0.09497  
## Detection Prevalence 0.8240 0.00000 0.17318  
## Balanced Accuracy 0.6316 0.50000 0.68375  
## Class: [0.2, 0.3) Class: [0.3, 0.4) Class: [0.4, 0.5)  
## Sensitivity 0.000000 0.00000 0.00000  
## Specificity 0.996764 1.00000 1.00000  
## Pos Pred Value 0.000000 NaN NaN  
## Neg Pred Value 0.862745 0.92458 0.96369  
## Prevalence 0.136872 0.07542 0.03631  
## Detection Rate 0.000000 0.00000 0.00000  
## Detection Prevalence 0.002793 0.00000 0.00000  
## Balanced Accuracy 0.498382 0.50000 0.50000  
## Class: [0.5, 0.6) Class: [0.6, 0.7) Class: [0.7, 0.8)  
## Sensitivity 0.00000 0.00000 0.00000  
## Specificity 1.00000 1.00000 1.00000  
## Pos Pred Value NaN NaN NaN  
## Neg Pred Value 0.97486 0.98883 0.98883  
## Prevalence 0.02514 0.01117 0.01117  
## Detection Rate 0.00000 0.00000 0.00000  
## Detection Prevalence 0.00000 0.00000 0.00000  
## Balanced Accuracy 0.50000 0.50000 0.50000  
## Class: [0.8, 0.9) Class: [0.9, 1]  
## Sensitivity 0.00000 0.000000  
## Specificity 1.00000 1.000000  
## Pos Pred Value NaN NaN  
## Neg Pred Value 0.99162 0.997207  
## Prevalence 0.00838 0.002793  
## Detection Rate 0.00000 0.000000  
## Detection Prevalence 0.00000 0.000000  
## Balanced Accuracy 0.50000 0.500000

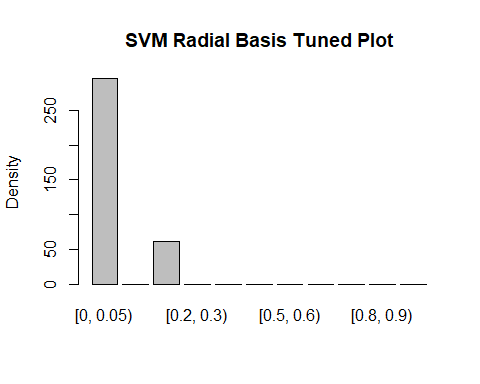
# Tune  
grid\_radial <- expand.grid(sigma = c(0,0.01, 0.02, 0.025, 0.03, 0.04,  
 0.05, 0.06, 0.07,0.08, 0.09, 0.1, 0.25, 0.5, 0.75,0.9),  
 C = c(0,0.01, 0.05, 0.1, 0.25, 0.5, 0.75,  
 1, 1.5, 2,5))  
  
model <- train(Bin ~., data = training, method = "svmRadial",  
 trControl=trctrl,  
 preProcess = c("center", "scale"),  
 tuneGrid = grid\_radial,  
 tuneLength = 10)  
model

## Support Vector Machines with Radial Basis Function Kernel   
##   
## 1102 samples  
## 19 predictor  
## 11 classes: '[0, 0.05)', '[0.05, 0.1)', '[0.1, 0.2)', '[0.2, 0.3)', '[0.3, 0.4)', '[0.4, 0.5)', '[0.5, 0.6)', '[0.6, 0.7)', '[0.7, 0.8)', '[0.8, 0.9)', '[0.9, 1]'   
##   
## Pre-processing: centered (19), scaled (19)   
## Resampling: Cross-Validated (10 fold, repeated 3 times)   
## Summary of sample sizes: 990, 991, 991, 992, 992, 993, ...   
## Resampling results across tuning parameters:  
##   
## sigma C Accuracy Kappa   
## 0.000 0.00 NaN NaN  
## 0.000 0.01 0.4220148 0.0000000000  
## 0.000 0.05 0.4220148 0.0000000000  
## 0.000 0.10 0.4220148 0.0000000000  
## 0.000 0.25 0.4220148 0.0000000000  
## 0.000 0.50 0.4220148 0.0000000000  
## 0.000 0.75 0.4220148 0.0000000000  
## 0.000 1.00 0.4220148 0.0000000000  
## 0.000 1.50 0.4220148 0.0000000000  
## 0.000 2.00 0.4220148 0.0000000000  
## 0.000 5.00 0.4220148 0.0000000000  
## 0.010 0.00 NaN NaN  
## 0.010 0.01 0.4220148 0.0000000000  
## 0.010 0.05 0.4223206 0.0007845234  
## 0.010 0.10 0.4579945 0.0874499306  
## 0.010 0.25 0.4649028 0.1110617186  
## 0.010 0.50 0.4745680 0.1332959650  
## 0.010 0.75 0.4790901 0.1449247974  
## 0.010 1.00 0.4818149 0.1527874593  
## 0.010 1.50 0.4884598 0.1654066891  
## 0.010 2.00 0.4893499 0.1698693501  
## 0.010 5.00 0.4911659 0.1825686030  
## 0.020 0.00 NaN NaN  
## 0.020 0.01 0.4220148 0.0000000000  
## 0.020 0.05 0.4259439 0.0100882101  
## 0.020 0.10 0.4567772 0.0914346552  
## 0.020 0.25 0.4691649 0.1234283840  
## 0.020 0.50 0.4848804 0.1581645876  
## 0.020 0.75 0.4884684 0.1673904106  
## 0.020 1.00 0.4920804 0.1755314709  
## 0.020 1.50 0.4914737 0.1781659413  
## 0.020 2.00 0.4884731 0.1761335874  
## 0.020 5.00 0.4875806 0.1867048909  
## 0.025 0.00 NaN NaN  
## 0.025 0.01 0.4220148 0.0000000000  
## 0.025 0.05 0.4244341 0.0072667449  
## 0.025 0.10 0.4576569 0.0953730504  
## 0.025 0.25 0.4731133 0.1324655159  
## 0.025 0.50 0.4845854 0.1594675610  
## 0.025 0.75 0.4920806 0.1752302356  
## 0.025 1.00 0.4920832 0.1774848329  
## 0.025 1.50 0.4878802 0.1748365811  
## 0.025 2.00 0.4894149 0.1799056402  
## 0.025 5.00 0.4885007 0.1943357361  
## 0.030 0.00 NaN NaN  
## 0.030 0.01 0.4220148 0.0000000000  
## 0.030 0.05 0.4232326 0.0045417391  
## 0.030 0.10 0.4594806 0.0994061646  
## 0.030 0.25 0.4737137 0.1354706992  
## 0.030 0.50 0.4845557 0.1609143906  
## 0.030 0.75 0.4920881 0.1768518804  
## 0.030 1.00 0.4917796 0.1786224525  
## 0.030 1.50 0.4900126 0.1796049534  
## 0.030 2.00 0.4870037 0.1786480157  
## 0.030 5.00 0.4836730 0.1921455234  
## 0.040 0.00 NaN NaN  
## 0.040 0.01 0.4220148 0.0000000000  
## 0.040 0.05 0.4223234 0.0008968973  
## 0.040 0.10 0.4576974 0.0952139489  
## 0.040 0.25 0.4752235 0.1397236509  
## 0.040 0.50 0.4842384 0.1617749911  
## 0.040 0.75 0.4887757 0.1725614393  
## 0.040 1.00 0.4908889 0.1792496911  
## 0.040 1.50 0.4870174 0.1783535726  
## 0.040 2.00 0.4864006 0.1822228333  
## 0.040 5.00 0.4780283 0.1919922182  
## 0.050 0.00 NaN NaN  
## 0.050 0.01 0.4220148 0.0000000000  
## 0.050 0.05 0.4220148 0.0000000000  
## 0.050 0.10 0.4525094 0.0820867352  
## 0.050 0.25 0.4761076 0.1424881091  
## 0.050 0.50 0.4842652 0.1625211295  
## 0.050 0.75 0.4854740 0.1681569413  
## 0.050 1.00 0.4879207 0.1761573201  
## 0.050 1.50 0.4855017 0.1789889626  
## 0.050 2.00 0.4849091 0.1854490363  
## 0.050 5.00 0.4810591 0.2021734619  
## 0.060 0.00 NaN NaN  
## 0.060 0.01 0.4220148 0.0000000000  
## 0.060 0.05 0.4220148 0.0000000000  
## 0.060 0.10 0.4444049 0.0641074750  
## 0.060 0.25 0.4740184 0.1390606128  
## 0.060 0.50 0.4845681 0.1637591253  
## 0.060 0.75 0.4854876 0.1693784098  
## 0.060 1.00 0.4864087 0.1761455170  
## 0.060 1.50 0.4824980 0.1774147732  
## 0.060 2.00 0.4789024 0.1796147672  
## 0.060 5.00 0.4701896 0.1912431833  
## 0.070 0.00 NaN NaN  
## 0.070 0.01 0.4220148 0.0000000000  
## 0.070 0.05 0.4220148 0.0000000000  
## 0.070 0.10 0.4374562 0.0475972216  
## 0.070 0.25 0.4715803 0.1343547641  
## 0.070 0.50 0.4812289 0.1582949878  
## 0.070 0.75 0.4872903 0.1744761309  
## 0.070 1.00 0.4840134 0.1733822022  
## 0.070 1.50 0.4816022 0.1786112946  
## 0.070 2.00 0.4798288 0.1849909527  
## 0.070 5.00 0.4711201 0.1957921315  
## 0.080 0.00 NaN NaN  
## 0.080 0.01 0.4220148 0.0000000000  
## 0.080 0.05 0.4220148 0.0000000000  
## 0.080 0.10 0.4347585 0.0394807826  
## 0.080 0.25 0.4712608 0.1324047421  
## 0.080 0.50 0.4791296 0.1554173224  
## 0.080 0.75 0.4885163 0.1772879748  
## 0.080 1.00 0.4849197 0.1765014953  
## 0.080 1.50 0.4810208 0.1816115779  
## 0.080 2.00 0.4813307 0.1901985340  
## 0.080 5.00 0.4713506 0.1981034107  
## 0.090 0.00 NaN NaN  
## 0.090 0.01 0.4220148 0.0000000000  
## 0.090 0.05 0.4220148 0.0000000000  
## 0.090 0.10 0.4292836 0.0257827352  
## 0.090 0.25 0.4706569 0.1294956798  
## 0.090 0.50 0.4779417 0.1529532860  
## 0.090 0.75 0.4842950 0.1703948083  
## 0.090 1.00 0.4840298 0.1761385180  
## 0.090 1.50 0.4831618 0.1873937796  
## 0.090 2.00 0.4795330 0.1911911514  
## 0.090 5.00 0.4734205 0.2049281123  
## 0.100 0.00 NaN NaN  
## 0.100 0.01 0.4220148 0.0000000000  
## 0.100 0.05 0.4220148 0.0000000000  
## 0.100 0.10 0.4277546 0.0193198143  
## 0.100 0.25 0.4688662 0.1244844179  
## 0.100 0.50 0.4770213 0.1512646072  
## 0.100 0.75 0.4794431 0.1619180912  
## 0.100 1.00 0.4831231 0.1754319101  
## 0.100 1.50 0.4816410 0.1866278392  
## 0.100 2.00 0.4789270 0.1925791563  
## 0.100 5.00 0.4728420 0.2062901450  
## 0.250 0.00 NaN NaN  
## 0.250 0.01 0.4220148 0.0000000000  
## 0.250 0.05 0.4220148 0.0000000000  
## 0.250 0.10 0.4220148 0.0000000000  
## 0.250 0.25 0.4428532 0.0550352709  
## 0.250 0.50 0.4598117 0.1036655367  
## 0.250 0.75 0.4689066 0.1350237795  
## 0.250 1.00 0.4743674 0.1591407806  
## 0.250 1.50 0.4746540 0.1768596965  
## 0.250 2.00 0.4631673 0.1689728646  
## 0.250 5.00 0.4371871 0.1617371770  
## 0.500 0.00 NaN NaN  
## 0.500 0.01 0.4220148 0.0000000000  
## 0.500 0.05 0.4220148 0.0000000000  
## 0.500 0.10 0.4220148 0.0000000000  
## 0.500 0.25 0.4229103 0.0024657351  
## 0.500 0.50 0.4458816 0.0568595500  
## 0.500 0.75 0.4516069 0.0796669229  
## 0.500 1.00 0.4555987 0.1030827148  
## 0.500 1.50 0.4589218 0.1284609290  
## 0.500 2.00 0.4553337 0.1350186012  
## 0.500 5.00 0.4341998 0.1261937987  
## 0.750 0.00 NaN NaN  
## 0.750 0.01 0.4220148 0.0000000000  
## 0.750 0.05 0.4220148 0.0000000000  
## 0.750 0.10 0.4220148 0.0000000000  
## 0.750 0.25 0.4220148 0.0000000000  
## 0.750 0.50 0.4310636 0.0199231468  
## 0.750 0.75 0.4437734 0.0510197588  
## 0.750 1.00 0.4482873 0.0680700176  
## 0.750 1.50 0.4446426 0.0829925169  
## 0.750 2.00 0.4383164 0.0820056611  
## 0.750 5.00 0.4262530 0.0724590390  
## 0.900 0.00 NaN NaN  
## 0.900 0.01 0.4220148 0.0000000000  
## 0.900 0.05 0.4220148 0.0000000000  
## 0.900 0.10 0.4220148 0.0000000000  
## 0.900 0.25 0.4220148 0.0000000000  
## 0.900 0.50 0.4301489 0.0174411864  
## 0.900 0.75 0.4362185 0.0325960156  
## 0.900 1.00 0.4449481 0.0564056820  
## 0.900 1.50 0.4419171 0.0688015667  
## 0.900 2.00 0.4382941 0.0710808307  
## 0.900 5.00 0.4358615 0.0713913024  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final values used for the model were sigma = 0.03 and C = 0.75.

Prediction <- predict(model, newdata=testing)  
plot(testing$Bin, ylab="Density",main="Test Data")



plot( Prediction, ylab = "Density", main = "SVM Radial Basis Tuned Plot")



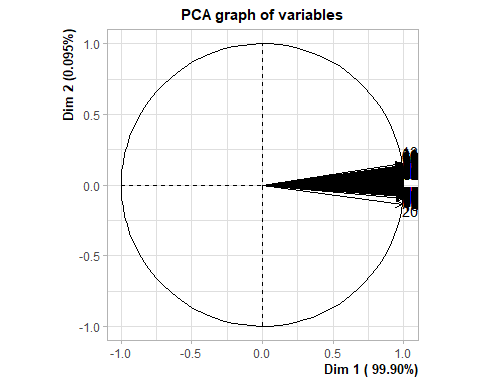
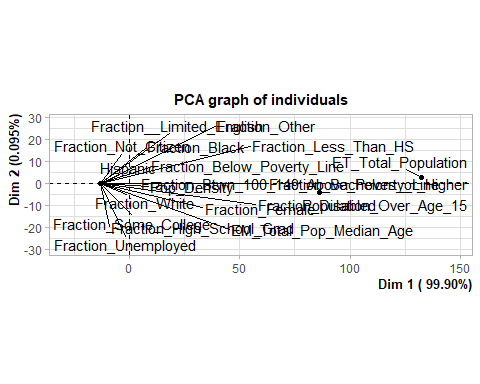
# Tuning sigma had no impact  
  
# Results <- table(unlist(Prediction),unlist(testing$Bin))  
# Results <-as.data.frame.matrix(Results)  
# Results  
confusionMatrix(Prediction, testing$Bin)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction [0, 0.05) [0.05, 0.1) [0.1, 0.2) [0.2, 0.3) [0.3, 0.4) [0.4, 0.5)  
## [0, 0.05) 150 14 39 38 23 11  
## [0.05, 0.1) 0 0 0 0 0 0  
## [0.1, 0.2) 4 7 34 11 4 2  
## [0.2, 0.3) 0 0 0 0 0 0  
## [0.3, 0.4) 0 0 0 0 0 0  
## [0.4, 0.5) 0 0 0 0 0 0  
## [0.5, 0.6) 0 0 0 0 0 0  
## [0.6, 0.7) 0 0 0 0 0 0  
## [0.7, 0.8) 0 0 0 0 0 0  
## [0.8, 0.9) 0 0 0 0 0 0  
## [0.9, 1] 0 0 0 0 0 0  
## Reference  
## Prediction [0.5, 0.6) [0.6, 0.7) [0.7, 0.8) [0.8, 0.9) [0.9, 1]  
## [0, 0.05) 9 4 4 3 1  
## [0.05, 0.1) 0 0 0 0 0  
## [0.1, 0.2) 0 0 0 0 0  
## [0.2, 0.3) 0 0 0 0 0  
## [0.3, 0.4) 0 0 0 0 0  
## [0.4, 0.5) 0 0 0 0 0  
## [0.5, 0.6) 0 0 0 0 0  
## [0.6, 0.7) 0 0 0 0 0  
## [0.7, 0.8) 0 0 0 0 0  
## [0.8, 0.9) 0 0 0 0 0  
## [0.9, 1] 0 0 0 0 0  
##   
## Overall Statistics  
##   
## Accuracy : 0.514   
## 95% CI : (0.4609, 0.5668)  
## No Information Rate : 0.4302   
## P-Value [Acc > NIR] : 0.0008631   
##   
## Kappa : 0.2019   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: [0, 0.05) Class: [0.05, 0.1) Class: [0.1, 0.2)  
## Sensitivity 0.9740 0.00000 0.46575  
## Specificity 0.2843 1.00000 0.90175  
## Pos Pred Value 0.5068 NaN 0.54839  
## Neg Pred Value 0.9355 0.94134 0.86824  
## Prevalence 0.4302 0.05866 0.20391  
## Detection Rate 0.4190 0.00000 0.09497  
## Detection Prevalence 0.8268 0.00000 0.17318  
## Balanced Accuracy 0.6292 0.50000 0.68375  
## Class: [0.2, 0.3) Class: [0.3, 0.4) Class: [0.4, 0.5)  
## Sensitivity 0.0000 0.00000 0.00000  
## Specificity 1.0000 1.00000 1.00000  
## Pos Pred Value NaN NaN NaN  
## Neg Pred Value 0.8631 0.92458 0.96369  
## Prevalence 0.1369 0.07542 0.03631  
## Detection Rate 0.0000 0.00000 0.00000  
## Detection Prevalence 0.0000 0.00000 0.00000  
## Balanced Accuracy 0.5000 0.50000 0.50000  
## Class: [0.5, 0.6) Class: [0.6, 0.7) Class: [0.7, 0.8)  
## Sensitivity 0.00000 0.00000 0.00000  
## Specificity 1.00000 1.00000 1.00000  
## Pos Pred Value NaN NaN NaN  
## Neg Pred Value 0.97486 0.98883 0.98883  
## Prevalence 0.02514 0.01117 0.01117  
## Detection Rate 0.00000 0.00000 0.00000  
## Detection Prevalence 0.00000 0.00000 0.00000  
## Balanced Accuracy 0.50000 0.50000 0.50000  
## Class: [0.8, 0.9) Class: [0.9, 1]  
## Sensitivity 0.00000 0.000000  
## Specificity 1.00000 1.000000  
## Pos Pred Value NaN NaN  
## Neg Pred Value 0.99162 0.997207  
## Prevalence 0.00838 0.002793  
## Detection Rate 0.00000 0.000000  
## Detection Prevalence 0.00000 0.000000  
## Balanced Accuracy 0.50000 0.500000

### Observation: 48% training accuracy with Kappa .16. 51% test accuracy with Kappa .2 SVM Radial Basis, no tuning. No change with tuning

## Try Dimensionality Reduction and RF

# go back to unnormalized and undiscretized data set and get rid of April 22  
  
MyDF <- SmallDataNumeric[,-4]  
# dim(MyDF)  
# str(MyDF)  
  
# Hold out the variable we are trying to predict  
  
Temp <- MyDF[,-4]  
Temp <- t(Temp)  
pca = PCA(Temp)



DigitDF = data.frame(MyDF$Deaths\_Per\_Confirmed\_4\_16,pca$var$coord)  
#dim(DigitDF)  
#head(DigitDF)  
#unique(DigitDF$MyDF.label)  
cnames <- colnames(DigitDF)  
cnames[1]<- c("Deaths\_Per\_Confirmed\_4\_16")  
colnames(DigitDF)<-cnames  
  
# Normalize then Discretize the Death Rate  
  
  
NormDeathRate <- (DigitDF$Deaths\_Per\_Confirmed\_4\_16 - min(DigitDF$Deaths\_Per\_Confirmed\_4\_16)) /  
(max(DigitDF$Deaths\_Per\_Confirmed\_4\_16)-min(DigitDF$Deaths\_Per\_Confirmed\_4\_16))  
 NormDeathRate <- as.data.frame(NormDeathRate)  
 str(NormDeathRate)

## 'data.frame': 1460 obs. of 1 variable:  
## $ NormDeathRate: num 0.192 0.099 0 0 0.238 ...

summary(NormDeathRate)

## NormDeathRate   
## Min. :0.0000   
## 1st Qu.:0.0000   
## Median :0.1111   
## Mean :0.1559   
## 3rd Qu.:0.2381   
## Max. :1.0000

colnames(NormDeathRate)

## [1] "NormDeathRate"

NormDeathRate[, "NormDeathRate"] <- bin\_data(NormDeathRate$NormDeathRate, bins=c(0,0.05,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0), binType = "explicit")  
  
  
NormDeathRatePCA <- data.frame(NormDeathRate, DigitDF)  
  
NormDeathRatePCA <- NormDeathRatePCA[,-2]  
#summary(NormDeathRatePCA)  
cname<- colnames(NormDeathRatePCA)  
cname

## [1] "NormDeathRate" "Dim.1" "Dim.2" "Dim.3"   
## [5] "Dim.4" "Dim.5"

cname[1]<-c("Bin")  
colnames(NormDeathRatePCA)<-cname  
str(NormDeathRatePCA)

## 'data.frame': 1460 obs. of 6 variables:  
## $ Bin : Ord.factor w/ 11 levels "[0, 0.05)"<"[0.05, 0.1)"<..: 3 2 1 1 4 1 1 1 6 1 ...  
## $ Dim.1: num 1 1 1 1 1 ...  
## $ Dim.2: num 0.00614 -0.01694 -0.00355 -0.00412 -0.01008 ...  
## $ Dim.3: num 0.000201 0.001121 0.000344 0.000236 0.000298 ...  
## $ Dim.4: num 8.28e-05 -4.60e-04 4.55e-05 -2.60e-04 4.80e-04 ...  
## $ Dim.5: num 1.21e-06 -1.93e-07 -1.76e-06 1.08e-06 1.12e-05 ...

#Try RF again  
  
# Set up the data  
  
#split data into training and test data sets  
data<-NormDeathRatePCA  
set.seed(300)  
indxTrain <- createDataPartition(y = data$Bin,p = 0.75,list = FALSE)  
training <- data[indxTrain,]  
testing <- data[-indxTrain,]   
  
#Check dimensions of the split   
  
prop.table(table(training$Bin)) \* 100

##   
## [0, 0.05) [0.05, 0.1) [0.1, 0.2) [0.2, 0.3) [0.3, 0.4) [0.4, 0.5)   
## 42.1960073 5.8983666 20.1451906 13.6116152 7.5317604 3.8112523   
## [0.5, 0.6) [0.6, 0.7) [0.7, 0.8) [0.8, 0.9) [0.9, 1]   
## 2.5408348 1.3611615 1.3611615 0.9981851 0.5444646

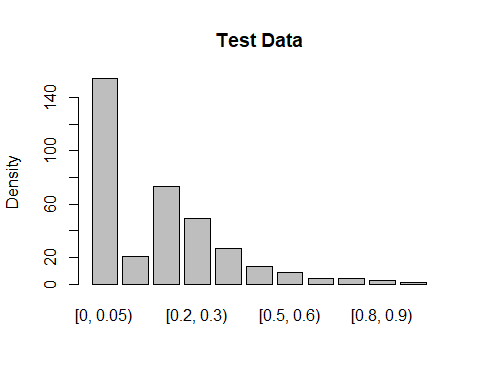
prop.table(table(testing$Bin)) \* 100

##   
## [0, 0.05) [0.05, 0.1) [0.1, 0.2) [0.2, 0.3) [0.3, 0.4) [0.4, 0.5)   
## 43.0167598 5.8659218 20.3910615 13.6871508 7.5418994 3.6312849   
## [0.5, 0.6) [0.6, 0.7) [0.7, 0.8) [0.8, 0.9) [0.9, 1]   
## 2.5139665 1.1173184 1.1173184 0.8379888 0.2793296

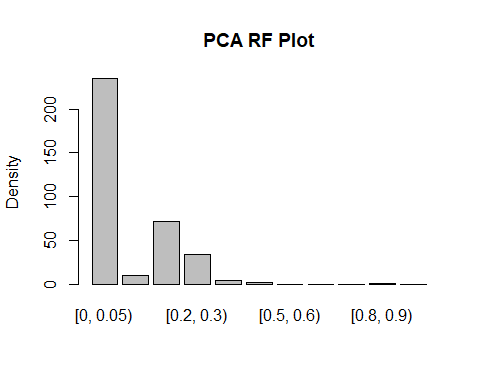
# Create variables with and without the bin  
  
x <- training[,-1]  
y <- training$Bin  
  
control <- trainControl(method="repeatedcv", number=10, repeats=3)  
seed <- 7  
metric <- "Accuracy"  
set.seed(seed)  
mtry <- sqrt(ncol(x))  
tunegrid <- expand.grid(.mtry=mtry)  
model <- train(Bin~., data=training, method="rf", metric=metric, tuneGrid=tunegrid, trControl=control)  
print(model)

## Random Forest   
##   
## 1102 samples  
## 5 predictor  
## 11 classes: '[0, 0.05)', '[0.05, 0.1)', '[0.1, 0.2)', '[0.2, 0.3)', '[0.3, 0.4)', '[0.4, 0.5)', '[0.5, 0.6)', '[0.6, 0.7)', '[0.7, 0.8)', '[0.8, 0.9)', '[0.9, 1]'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold, repeated 3 times)   
## Summary of sample sizes: 990, 993, 989, 991, 992, 992, ...   
## Resampling results:  
##   
## Accuracy Kappa   
## 0.4750062 0.2229138  
##   
## Tuning parameter 'mtry' was held constant at a value of 2.236068

#Observation: That yielded about 47% accuracy.   
  
Prediction <- predict(model, newdata=testing)  
plot(testing$Bin, ylab="Density",main="Test Data")



plot( Prediction, ylab = "Density", main = "PCA RF Plot")



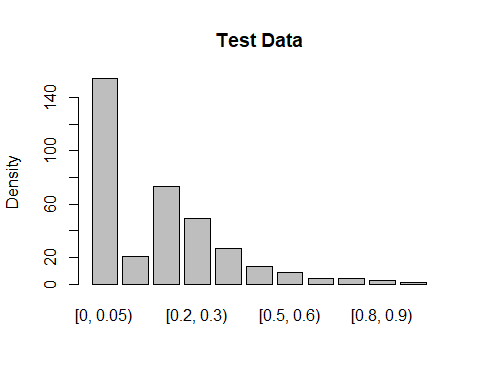
# Results <- table(unlist(Prediction),unlist(testing$Bin))  
# Results <-as.data.frame.matrix(Results)  
# Results  
confusionMatrix(Prediction, testing$Bin)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction [0, 0.05) [0.05, 0.1) [0.1, 0.2) [0.2, 0.3) [0.3, 0.4) [0.4, 0.5)  
## [0, 0.05) 134 5 25 28 18 7  
## [0.05, 0.1) 1 4 3 2 0 0  
## [0.1, 0.2) 8 9 33 13 6 3  
## [0.2, 0.3) 8 2 11 5 2 3  
## [0.3, 0.4) 1 1 1 0 1 0  
## [0.4, 0.5) 1 0 0 1 0 0  
## [0.5, 0.6) 0 0 0 0 0 0  
## [0.6, 0.7) 0 0 0 0 0 0  
## [0.7, 0.8) 0 0 0 0 0 0  
## [0.8, 0.9) 1 0 0 0 0 0  
## [0.9, 1] 0 0 0 0 0 0  
## Reference  
## Prediction [0.5, 0.6) [0.6, 0.7) [0.7, 0.8) [0.8, 0.9) [0.9, 1]  
## [0, 0.05) 7 3 4 3 1  
## [0.05, 0.1) 0 0 0 0 0  
## [0.1, 0.2) 0 0 0 0 0  
## [0.2, 0.3) 2 1 0 0 0  
## [0.3, 0.4) 0 0 0 0 0  
## [0.4, 0.5) 0 0 0 0 0  
## [0.5, 0.6) 0 0 0 0 0  
## [0.6, 0.7) 0 0 0 0 0  
## [0.7, 0.8) 0 0 0 0 0  
## [0.8, 0.9) 0 0 0 0 0  
## [0.9, 1] 0 0 0 0 0  
##   
## Overall Statistics  
##   
## Accuracy : 0.4944   
## 95% CI : (0.4415, 0.5475)  
## No Information Rate : 0.4302   
## P-Value [Acc > NIR] : 0.00836   
##   
## Kappa : 0.235   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: [0, 0.05) Class: [0.05, 0.1) Class: [0.1, 0.2)  
## Sensitivity 0.8701 0.19048 0.45205  
## Specificity 0.5049 0.98220 0.86316  
## Pos Pred Value 0.5702 0.40000 0.45833  
## Neg Pred Value 0.8374 0.95115 0.86014  
## Prevalence 0.4302 0.05866 0.20391  
## Detection Rate 0.3743 0.01117 0.09218  
## Detection Prevalence 0.6564 0.02793 0.20112  
## Balanced Accuracy 0.6875 0.58634 0.65761  
## Class: [0.2, 0.3) Class: [0.3, 0.4) Class: [0.4, 0.5)  
## Sensitivity 0.10204 0.037037 0.000000  
## Specificity 0.90615 0.990937 0.994203  
## Pos Pred Value 0.14706 0.250000 0.000000  
## Neg Pred Value 0.86420 0.926554 0.963483  
## Prevalence 0.13687 0.075419 0.036313  
## Detection Rate 0.01397 0.002793 0.000000  
## Detection Prevalence 0.09497 0.011173 0.005587  
## Balanced Accuracy 0.50409 0.513987 0.497101  
## Class: [0.5, 0.6) Class: [0.6, 0.7) Class: [0.7, 0.8)  
## Sensitivity 0.00000 0.00000 0.00000  
## Specificity 1.00000 1.00000 1.00000  
## Pos Pred Value NaN NaN NaN  
## Neg Pred Value 0.97486 0.98883 0.98883  
## Prevalence 0.02514 0.01117 0.01117  
## Detection Rate 0.00000 0.00000 0.00000  
## Detection Prevalence 0.00000 0.00000 0.00000  
## Balanced Accuracy 0.50000 0.50000 0.50000  
## Class: [0.8, 0.9) Class: [0.9, 1]  
## Sensitivity 0.000000 0.000000  
## Specificity 0.997183 1.000000  
## Pos Pred Value 0.000000 NaN  
## Neg Pred Value 0.991597 0.997207  
## Prevalence 0.008380 0.002793  
## Detection Rate 0.000000 0.000000  
## Detection Prevalence 0.002793 0.000000  
## Balanced Accuracy 0.498592 0.500000

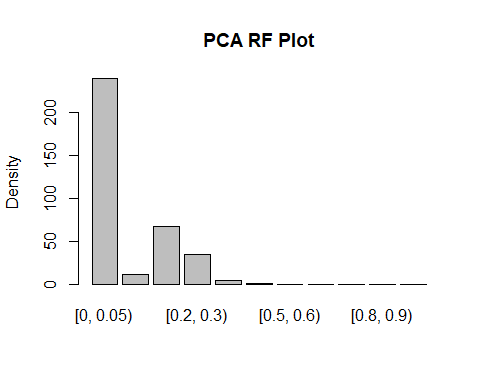
# Can we tune it? Tune on mytry using random search  
  
# Random Search  
control <- trainControl(method="repeatedcv", number=10, repeats=3, search="random")  
set.seed(seed)  
mtry <- sqrt(ncol(x))  
model2 <- train(Bin~., data=training, method="rf", metric=metric, tuneLength=15, trControl=control)  
print(model2)

## Random Forest   
##   
## 1102 samples  
## 5 predictor  
## 11 classes: '[0, 0.05)', '[0.05, 0.1)', '[0.1, 0.2)', '[0.2, 0.3)', '[0.3, 0.4)', '[0.4, 0.5)', '[0.5, 0.6)', '[0.6, 0.7)', '[0.7, 0.8)', '[0.8, 0.9)', '[0.9, 1]'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold, repeated 3 times)   
## Summary of sample sizes: 990, 993, 989, 991, 992, 992, ...   
## Resampling results across tuning parameters:  
##   
## mtry Accuracy Kappa   
## 2 0.4771249 0.2265593  
## 3 0.4655699 0.2136751  
## 4 0.4661733 0.2167138  
## 5 0.4649908 0.2157292  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was mtry = 2.

# That did not improve anything.  
#plot(model2)  
  
Prediction <- predict(model2, newdata=testing)  
plot(testing$Bin, ylab="Density",main="Test Data")



plot( Prediction, ylab = "Density", main = "PCA RF Plot")



# Results <- table(unlist(Prediction),unlist(testing$Bin))  
# Results <-as.data.frame.matrix(Results)  
# Results  
confusionMatrix(Prediction, testing$Bin)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction [0, 0.05) [0.05, 0.1) [0.1, 0.2) [0.2, 0.3) [0.3, 0.4) [0.4, 0.5)  
## [0, 0.05) 135 5 27 29 19 7  
## [0.05, 0.1) 3 4 2 2 0 0  
## [0.1, 0.2) 6 9 31 13 5 3  
## [0.2, 0.3) 8 2 12 5 2 3  
## [0.3, 0.4) 1 1 1 0 1 0  
## [0.4, 0.5) 1 0 0 0 0 0  
## [0.5, 0.6) 0 0 0 0 0 0  
## [0.6, 0.7) 0 0 0 0 0 0  
## [0.7, 0.8) 0 0 0 0 0 0  
## [0.8, 0.9) 0 0 0 0 0 0  
## [0.9, 1] 0 0 0 0 0 0  
## Reference  
## Prediction [0.5, 0.6) [0.6, 0.7) [0.7, 0.8) [0.8, 0.9) [0.9, 1]  
## [0, 0.05) 7 3 4 3 1  
## [0.05, 0.1) 0 0 0 0 0  
## [0.1, 0.2) 0 0 0 0 0  
## [0.2, 0.3) 2 1 0 0 0  
## [0.3, 0.4) 0 0 0 0 0  
## [0.4, 0.5) 0 0 0 0 0  
## [0.5, 0.6) 0 0 0 0 0  
## [0.6, 0.7) 0 0 0 0 0  
## [0.7, 0.8) 0 0 0 0 0  
## [0.8, 0.9) 0 0 0 0 0  
## [0.9, 1] 0 0 0 0 0  
##   
## Overall Statistics  
##   
## Accuracy : 0.4916   
## 95% CI : (0.4387, 0.5447)  
## No Information Rate : 0.4302   
## P-Value [Acc > NIR] : 0.0111   
##   
## Kappa : 0.2266   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: [0, 0.05) Class: [0.05, 0.1) Class: [0.1, 0.2)  
## Sensitivity 0.8766 0.19048 0.42466  
## Specificity 0.4853 0.97923 0.87368  
## Pos Pred Value 0.5625 0.36364 0.46269  
## Neg Pred Value 0.8390 0.95101 0.85567  
## Prevalence 0.4302 0.05866 0.20391  
## Detection Rate 0.3771 0.01117 0.08659  
## Detection Prevalence 0.6704 0.03073 0.18715  
## Balanced Accuracy 0.6810 0.58485 0.64917  
## Class: [0.2, 0.3) Class: [0.3, 0.4) Class: [0.4, 0.5)  
## Sensitivity 0.10204 0.037037 0.000000  
## Specificity 0.90291 0.990937 0.997101  
## Pos Pred Value 0.14286 0.250000 0.000000  
## Neg Pred Value 0.86378 0.926554 0.963585  
## Prevalence 0.13687 0.075419 0.036313  
## Detection Rate 0.01397 0.002793 0.000000  
## Detection Prevalence 0.09777 0.011173 0.002793  
## Balanced Accuracy 0.50248 0.513987 0.498551  
## Class: [0.5, 0.6) Class: [0.6, 0.7) Class: [0.7, 0.8)  
## Sensitivity 0.00000 0.00000 0.00000  
## Specificity 1.00000 1.00000 1.00000  
## Pos Pred Value NaN NaN NaN  
## Neg Pred Value 0.97486 0.98883 0.98883  
## Prevalence 0.02514 0.01117 0.01117  
## Detection Rate 0.00000 0.00000 0.00000  
## Detection Prevalence 0.00000 0.00000 0.00000  
## Balanced Accuracy 0.50000 0.50000 0.50000  
## Class: [0.8, 0.9) Class: [0.9, 1]  
## Sensitivity 0.00000 0.000000  
## Specificity 1.00000 1.000000  
## Pos Pred Value NaN NaN  
## Neg Pred Value 0.99162 0.997207  
## Prevalence 0.00838 0.002793  
## Detection Rate 0.00000 0.000000  
## Detection Prevalence 0.00000 0.000000  
## Balanced Accuracy 0.50000 0.500000

## Try to predict everything but the very low death rate and high # cases with RF

#- go back to class data  
  
data<-ClassData  
#str(ClassData)  
BiggerCases <- ClassData[ClassData$Bin != "[0, 0.05)",]  
  
# Get rid of that factor ("[0, 0.05)")  
BiggerCases$Bin<-factor(BiggerCases$Bin)  
#str(BiggerCases$Bin)  
  
  
# Try Random Forest  
  
# Set up the data  
  
#split data into training and test data sets  
data<-BiggerCases  
set.seed(300)  
indxTrain <- createDataPartition(y = data$Bin,p = 0.75,list = FALSE)  
training <- data[indxTrain,]  
testing <- data[-indxTrain,]   
  
#Check dimensions of the split   
  
prop.table(table(training$Bin)) \* 100

##   
## [0.05, 0.1) [0.1, 0.2) [0.2, 0.3) [0.3, 0.4) [0.4, 0.5) [0.5, 0.6)   
## 10.2040816 34.8508634 23.5478807 13.0298273 6.5934066 4.3956044   
## [0.6, 0.7) [0.7, 0.8) [0.8, 0.9) [0.9, 1]   
## 2.3547881 2.3547881 1.7268446 0.9419152

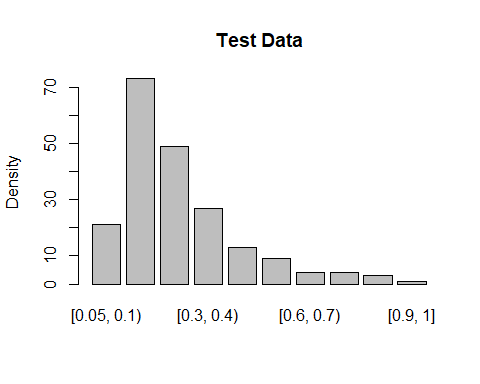
prop.table(table(testing$Bin)) \* 100

##   
## [0.05, 0.1) [0.1, 0.2) [0.2, 0.3) [0.3, 0.4) [0.4, 0.5) [0.5, 0.6)   
## 10.2941176 35.7843137 24.0196078 13.2352941 6.3725490 4.4117647   
## [0.6, 0.7) [0.7, 0.8) [0.8, 0.9) [0.9, 1]   
## 1.9607843 1.9607843 1.4705882 0.4901961

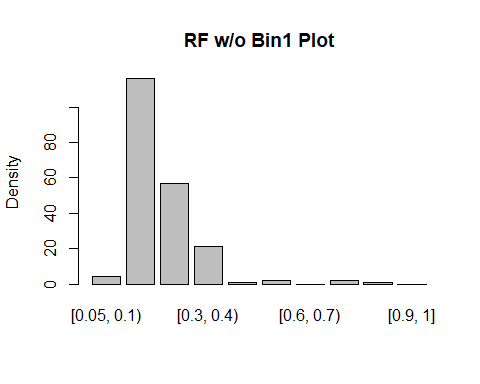
# Create variables with and without the bin  
  
x <- training[,-20]  
y <- training$Bin  
  
control <- trainControl(method="repeatedcv", number=10, repeats=3)  
seed <- 7  
metric <- "Accuracy"  
set.seed(seed)  
mtry <- sqrt(ncol(x))  
tunegrid <- expand.grid(.mtry=mtry)  
model <- train(Bin~., data=training, method="rf", metric=metric, tuneGrid=tunegrid, trControl=control)  
print(model)

## Random Forest   
##   
## 637 samples  
## 19 predictor  
## 10 classes: '[0.05, 0.1)', '[0.1, 0.2)', '[0.2, 0.3)', '[0.3, 0.4)', '[0.4, 0.5)', '[0.5, 0.6)', '[0.6, 0.7)', '[0.7, 0.8)', '[0.8, 0.9)', '[0.9, 1]'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold, repeated 3 times)   
## Summary of sample sizes: 572, 572, 572, 575, 574, 575, ...   
## Resampling results:  
##   
## Accuracy Kappa   
## 0.3032923 0.01901402  
##   
## Tuning parameter 'mtry' was held constant at a value of 4.358899

#Observation: That yielded about 30% accuracy. So the abilty to predict the cases where we have high death rates but fewer cases is very limited.  
  
Prediction <- predict(model, newdata=testing)  
plot(testing$Bin, ylab="Density",main="Test Data")



plot( Prediction, ylab = "Density", main = "RF w/o Bin1 Plot")



# Results <- table(unlist(Prediction),unlist(testing$Bin))  
# Results <-as.data.frame.matrix(Results)  
# Results  
confusionMatrix(Prediction, testing$Bin)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction [0.05, 0.1) [0.1, 0.2) [0.2, 0.3) [0.3, 0.4) [0.4, 0.5)  
## [0.05, 0.1) 0 1 2 1 0  
## [0.1, 0.2) 15 53 24 12 6  
## [0.2, 0.3) 6 10 19 9 3  
## [0.3, 0.4) 0 8 3 5 2  
## [0.4, 0.5) 0 0 0 0 0  
## [0.5, 0.6) 0 1 1 0 0  
## [0.6, 0.7) 0 0 0 0 0  
## [0.7, 0.8) 0 0 0 0 1  
## [0.8, 0.9) 0 0 0 0 1  
## [0.9, 1] 0 0 0 0 0  
## Reference  
## Prediction [0.5, 0.6) [0.6, 0.7) [0.7, 0.8) [0.8, 0.9) [0.9, 1]  
## [0.05, 0.1) 0 0 0 0 0  
## [0.1, 0.2) 2 2 1 1 0  
## [0.2, 0.3) 7 0 2 1 0  
## [0.3, 0.4) 0 1 1 0 1  
## [0.4, 0.5) 0 1 0 0 0  
## [0.5, 0.6) 0 0 0 0 0  
## [0.6, 0.7) 0 0 0 0 0  
## [0.7, 0.8) 0 0 0 1 0  
## [0.8, 0.9) 0 0 0 0 0  
## [0.9, 1] 0 0 0 0 0  
##   
## Overall Statistics  
##   
## Accuracy : 0.3775   
## 95% CI : (0.3107, 0.4478)  
## No Information Rate : 0.3578   
## P-Value [Acc > NIR] : 0.3029   
##   
## Kappa : 0.1266   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: [0.05, 0.1) Class: [0.1, 0.2) Class: [0.2, 0.3)  
## Sensitivity 0.00000 0.7260 0.38776  
## Specificity 0.97814 0.5191 0.75484  
## Pos Pred Value 0.00000 0.4569 0.33333  
## Neg Pred Value 0.89500 0.7727 0.79592  
## Prevalence 0.10294 0.3578 0.24020  
## Detection Rate 0.00000 0.2598 0.09314  
## Detection Prevalence 0.01961 0.5686 0.27941  
## Balanced Accuracy 0.48907 0.6226 0.57130  
## Class: [0.3, 0.4) Class: [0.4, 0.5) Class: [0.5, 0.6)  
## Sensitivity 0.18519 0.000000 0.000000  
## Specificity 0.90960 0.994764 0.989744  
## Pos Pred Value 0.23810 0.000000 0.000000  
## Neg Pred Value 0.87978 0.935961 0.955446  
## Prevalence 0.13235 0.063725 0.044118  
## Detection Rate 0.02451 0.000000 0.000000  
## Detection Prevalence 0.10294 0.004902 0.009804  
## Balanced Accuracy 0.54739 0.497382 0.494872  
## Class: [0.6, 0.7) Class: [0.7, 0.8) Class: [0.8, 0.9)  
## Sensitivity 0.00000 0.000000 0.000000  
## Specificity 1.00000 0.990000 0.995025  
## Pos Pred Value NaN 0.000000 0.000000  
## Neg Pred Value 0.98039 0.980198 0.985222  
## Prevalence 0.01961 0.019608 0.014706  
## Detection Rate 0.00000 0.000000 0.000000  
## Detection Prevalence 0.00000 0.009804 0.004902  
## Balanced Accuracy 0.50000 0.495000 0.497512  
## Class: [0.9, 1]  
## Sensitivity 0.000000  
## Specificity 1.000000  
## Pos Pred Value NaN  
## Neg Pred Value 0.995098  
## Prevalence 0.004902  
## Detection Rate 0.000000  
## Detection Prevalence 0.000000  
## Balanced Accuracy 0.500000

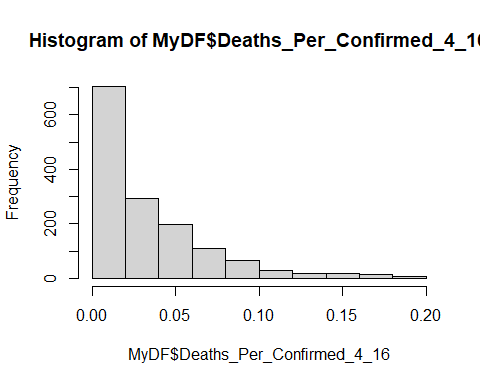
### Observation: That yielded about 30% accuracy. So the abilty to predict the cases where we have high death rates but fewer cases is very limited.

## Try without normalizing the fractional data (Everything but Pop Density, total pop and age)

# go back to unnormalized and undiscretized data set and get rid of April 22  
  
MyDF <- SmallDataNumeric[,-4]  
#dim(MyDF)  
#str(MyDF)  
  
summary(MyDF$Deaths\_Per\_Confirmed\_4\_16)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.00000 0.00000 0.02222 0.03117 0.04763 0.20000

ToNorm <- MyDF[,1:3]  
  
ToNorm <- as.data.frame(lapply(ToNorm,Min\_Max\_function))  
  
MyDF<-MyDF[,-1:-3]  
  
# Save this for running without those variables later  
  
OnlyPovertyDataNotNorm <- MyDF  
  
MyDF<- data.frame(ToNorm, MyDF)  
  
hist(MyDF$Deaths\_Per\_Confirmed\_4\_16)



# Discretize the Death Rate - since not normalized the bins change  
  
 MyDF[, "Deaths\_Per\_Confirmed\_4\_16"] <- bin\_data(MyDF$Deaths\_Per\_Confirmed\_4\_16, bins=c(0,0.01,0.02,0.04, .05,.06,.08,.10,.12,.14,.16,.18,.20) , binType = "explicit")  
  
cname<- colnames(MyDF)  
cname

## [1] "Pop\_Density"   
## [2] "ET\_Total\_Population"   
## [3] "EM\_Total\_Pop\_Median\_Age"   
## [4] "Deaths\_Per\_Confirmed\_4\_16"   
## [5] "Fraction\_Female"   
## [6] "Fraction\_White"   
## [7] "Fraction\_Black"   
## [8] "Fraction\_Other"   
## [9] "Hispanic"   
## [10] "Population\_Over\_Age\_15"   
## [11] "Fraction\_Less\_Than\_HS"   
## [12] "Fraction\_High\_School\_Grad"   
## [13] "Fraction\_Some\_College"   
## [14] "Fraction\_Bachelors\_or\_Higher"   
## [15] "Fraction\_Disabled"   
## [16] "Fraction\_\_Limited\_English"   
## [17] "Fraction\_Unemployed"   
## [18] "Fraction\_Below\_Poverty\_Line"   
## [19] "Fraction\_Btwn\_100\_149\_Above\_Poverty\_Line"  
## [20] "Fraction\_Not\_Citizen"

cname[4]<-c("Bin")  
colnames(MyDF)<-cname  
table(MyDF$Bin)

##   
## [0, 0.01) [0.01, 0.02) [0.02, 0.04) [0.04, 0.05) [0.05, 0.06) [0.06, 0.08)   
## 619 84 294 115 87 108   
## [0.08, 0.1) [0.1, 0.12) [0.12, 0.14) [0.14, 0.16) [0.16, 0.18) [0.18, 0.2]   
## 57 37 19 19 14 7

# Try RF  
#split data into training and test data sets  
data<-MyDF  
set.seed(300)  
indxTrain <- createDataPartition(y = data$Bin,p = 0.75,list = FALSE)  
training <- data[indxTrain,]  
testing <- data[-indxTrain,]   
  
#Check dimensions of the split   
  
prop.table(table(training$Bin)) \* 100

##   
## [0, 0.01) [0.01, 0.02) [0.02, 0.04) [0.04, 0.05) [0.05, 0.06) [0.06, 0.08)   
## 42.2343324 5.7220708 20.0726612 7.9019074 5.9945504 7.3569482   
## [0.08, 0.1) [0.1, 0.12) [0.12, 0.14) [0.14, 0.16) [0.16, 0.18) [0.18, 0.2]   
## 3.9055404 2.5431426 1.3623978 1.3623978 0.9990917 0.5449591

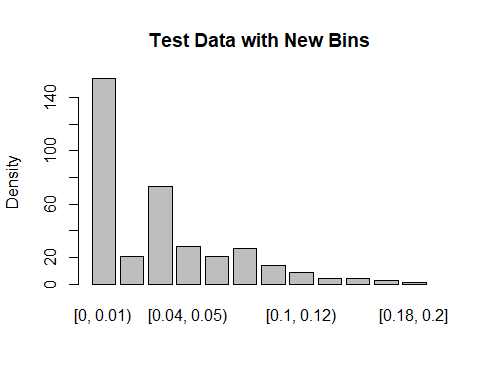
prop.table(table(testing$Bin)) \* 100

##   
## [0, 0.01) [0.01, 0.02) [0.02, 0.04) [0.04, 0.05) [0.05, 0.06) [0.06, 0.08)   
## 42.8969359 5.8495822 20.3342618 7.7994429 5.8495822 7.5208914   
## [0.08, 0.1) [0.1, 0.12) [0.12, 0.14) [0.14, 0.16) [0.16, 0.18) [0.18, 0.2]   
## 3.8997214 2.5069638 1.1142061 1.1142061 0.8356546 0.2785515

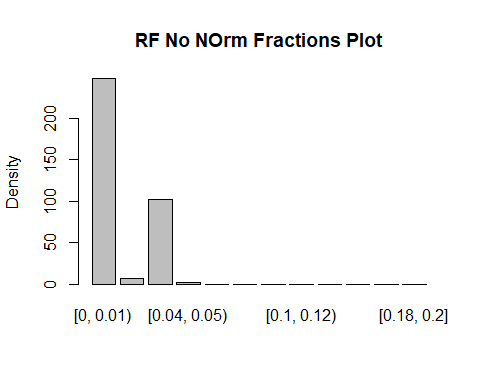
# Create variables with and without the bin  
  
x <- training[,-20]  
y <- training$Bin  
  
control <- trainControl(method="repeatedcv", number=10, repeats=3)  
seed <- 7  
metric <- "Accuracy"  
set.seed(seed)  
mtry <- sqrt(ncol(x))  
tunegrid <- expand.grid(.mtry=mtry)  
model <- train(Bin~., data=training, method="rf", metric=metric, tuneGrid=tunegrid, trControl=control)  
print(model)

## Random Forest   
##   
## 1101 samples  
## 19 predictor  
## 12 classes: '[0, 0.01)', '[0.01, 0.02)', '[0.02, 0.04)', '[0.04, 0.05)', '[0.05, 0.06)', '[0.06, 0.08)', '[0.08, 0.1)', '[0.1, 0.12)', '[0.12, 0.14)', '[0.14, 0.16)', '[0.16, 0.18)', '[0.18, 0.2]'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold, repeated 3 times)   
## Summary of sample sizes: 990, 990, 989, 992, 989, 992, ...   
## Resampling results:  
##   
## Accuracy Kappa   
## 0.4861285 0.2127572  
##   
## Tuning parameter 'mtry' was held constant at a value of 4.358899

Prediction <- predict(model, newdata=testing)  
plot(testing$Bin, ylab="Density",main="Test Data with New Bins")



plot( Prediction, ylab = "Density", main = "RF No NOrm Fractions Plot")

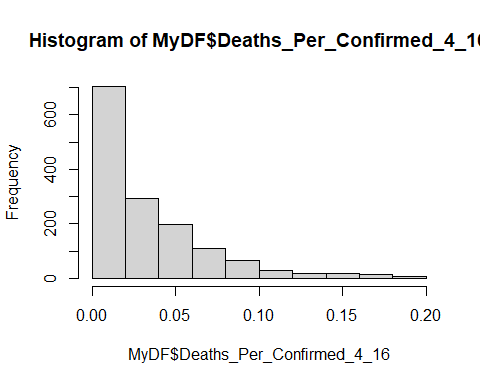


# Results <- table(unlist(Prediction),unlist(testing$Bin))  
# Results <-as.data.frame.matrix(Results)  
# Results  
confusionMatrix(Prediction, testing$Bin)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction [0, 0.01) [0.01, 0.02) [0.02, 0.04) [0.04, 0.05) [0.05, 0.06)  
## [0, 0.01) 140 10 23 12 13  
## [0.01, 0.02) 2 0 4 0 0  
## [0.02, 0.04) 12 11 45 15 8  
## [0.04, 0.05) 0 0 1 1 0  
## [0.05, 0.06) 0 0 0 0 0  
## [0.06, 0.08) 0 0 0 0 0  
## [0.08, 0.1) 0 0 0 0 0  
## [0.1, 0.12) 0 0 0 0 0  
## [0.12, 0.14) 0 0 0 0 0  
## [0.14, 0.16) 0 0 0 0 0  
## [0.16, 0.18) 0 0 0 0 0  
## [0.18, 0.2] 0 0 0 0 0  
## Reference  
## Prediction [0.06, 0.08) [0.08, 0.1) [0.1, 0.12) [0.12, 0.14) [0.14, 0.16)  
## [0, 0.01) 19 12 8 4 4  
## [0.01, 0.02) 1 0 0 0 0  
## [0.02, 0.04) 7 2 1 0 0  
## [0.04, 0.05) 0 0 0 0 0  
## [0.05, 0.06) 0 0 0 0 0  
## [0.06, 0.08) 0 0 0 0 0  
## [0.08, 0.1) 0 0 0 0 0  
## [0.1, 0.12) 0 0 0 0 0  
## [0.12, 0.14) 0 0 0 0 0  
## [0.14, 0.16) 0 0 0 0 0  
## [0.16, 0.18) 0 0 0 0 0  
## [0.18, 0.2] 0 0 0 0 0  
## Reference  
## Prediction [0.16, 0.18) [0.18, 0.2]  
## [0, 0.01) 2 1  
## [0.01, 0.02) 0 0  
## [0.02, 0.04) 1 0  
## [0.04, 0.05) 0 0  
## [0.05, 0.06) 0 0  
## [0.06, 0.08) 0 0  
## [0.08, 0.1) 0 0  
## [0.1, 0.12) 0 0  
## [0.12, 0.14) 0 0  
## [0.14, 0.16) 0 0  
## [0.16, 0.18) 0 0  
## [0.18, 0.2] 0 0  
##   
## Overall Statistics  
##   
## Accuracy : 0.5181   
## 95% CI : (0.4651, 0.5709)  
## No Information Rate : 0.429   
## P-Value [Acc > NIR] : 0.0004168   
##   
## Kappa : 0.2521   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: [0, 0.01) Class: [0.01, 0.02) Class: [0.02, 0.04)  
## Sensitivity 0.9091 0.0000 0.6164  
## Specificity 0.4732 0.9793 0.8007  
## Pos Pred Value 0.5645 0.0000 0.4412  
## Neg Pred Value 0.8739 0.9403 0.8911  
## Prevalence 0.4290 0.0585 0.2033  
## Detection Rate 0.3900 0.0000 0.1253  
## Detection Prevalence 0.6908 0.0195 0.2841  
## Balanced Accuracy 0.6911 0.4896 0.7086  
## Class: [0.04, 0.05) Class: [0.05, 0.06)  
## Sensitivity 0.035714 0.0000  
## Specificity 0.996979 1.0000  
## Pos Pred Value 0.500000 NaN  
## Neg Pred Value 0.924370 0.9415  
## Prevalence 0.077994 0.0585  
## Detection Rate 0.002786 0.0000  
## Detection Prevalence 0.005571 0.0000  
## Balanced Accuracy 0.516347 0.5000  
## Class: [0.06, 0.08) Class: [0.08, 0.1) Class: [0.1, 0.12)  
## Sensitivity 0.00000 0.000 0.00000  
## Specificity 1.00000 1.000 1.00000  
## Pos Pred Value NaN NaN NaN  
## Neg Pred Value 0.92479 0.961 0.97493  
## Prevalence 0.07521 0.039 0.02507  
## Detection Rate 0.00000 0.000 0.00000  
## Detection Prevalence 0.00000 0.000 0.00000  
## Balanced Accuracy 0.50000 0.500 0.50000  
## Class: [0.12, 0.14) Class: [0.14, 0.16)  
## Sensitivity 0.00000 0.00000  
## Specificity 1.00000 1.00000  
## Pos Pred Value NaN NaN  
## Neg Pred Value 0.98886 0.98886  
## Prevalence 0.01114 0.01114  
## Detection Rate 0.00000 0.00000  
## Detection Prevalence 0.00000 0.00000  
## Balanced Accuracy 0.50000 0.50000  
## Class: [0.16, 0.18) Class: [0.18, 0.2]  
## Sensitivity 0.000000 0.000000  
## Specificity 1.000000 1.000000  
## Pos Pred Value NaN NaN  
## Neg Pred Value 0.991643 0.997214  
## Prevalence 0.008357 0.002786  
## Detection Rate 0.000000 0.000000  
## Detection Prevalence 0.000000 0.000000  
## Balanced Accuracy 0.500000 0.500000

## Remove non-wealth variables then use RF

MyDF <- OnlyPovertyDataNotNorm   
  
hist(MyDF$Deaths\_Per\_Confirmed\_4\_16)



# Discretize the Death Rate - since not normalized the bins change  
  
 MyDF[, "Deaths\_Per\_Confirmed\_4\_16"] <- bin\_data(MyDF$Deaths\_Per\_Confirmed\_4\_16, bins=c(0,0.01,0.02,0.04, .05,.06,.08,.10,.12,.14,.16,.18,.20) , binType = "explicit")  
  
cname<- colnames(MyDF)  
cname

## [1] "Deaths\_Per\_Confirmed\_4\_16"   
## [2] "Fraction\_Female"   
## [3] "Fraction\_White"   
## [4] "Fraction\_Black"   
## [5] "Fraction\_Other"   
## [6] "Hispanic"   
## [7] "Population\_Over\_Age\_15"   
## [8] "Fraction\_Less\_Than\_HS"   
## [9] "Fraction\_High\_School\_Grad"   
## [10] "Fraction\_Some\_College"   
## [11] "Fraction\_Bachelors\_or\_Higher"   
## [12] "Fraction\_Disabled"   
## [13] "Fraction\_\_Limited\_English"   
## [14] "Fraction\_Unemployed"   
## [15] "Fraction\_Below\_Poverty\_Line"   
## [16] "Fraction\_Btwn\_100\_149\_Above\_Poverty\_Line"  
## [17] "Fraction\_Not\_Citizen"

cname[1]<-c("Bin")  
colnames(MyDF)<-cname  
table(MyDF$Bin)

##   
## [0, 0.01) [0.01, 0.02) [0.02, 0.04) [0.04, 0.05) [0.05, 0.06) [0.06, 0.08)   
## 619 84 294 115 87 108   
## [0.08, 0.1) [0.1, 0.12) [0.12, 0.14) [0.14, 0.16) [0.16, 0.18) [0.18, 0.2]   
## 57 37 19 19 14 7

# Try RF  
#split data into training and test data sets  
data<-MyDF  
set.seed(300)  
indxTrain <- createDataPartition(y = data$Bin,p = 0.75,list = FALSE)  
training <- data[indxTrain,]  
testing <- data[-indxTrain,]   
  
#Check dimensions of the split   
  
prop.table(table(training$Bin)) \* 100

##   
## [0, 0.01) [0.01, 0.02) [0.02, 0.04) [0.04, 0.05) [0.05, 0.06) [0.06, 0.08)   
## 42.2343324 5.7220708 20.0726612 7.9019074 5.9945504 7.3569482   
## [0.08, 0.1) [0.1, 0.12) [0.12, 0.14) [0.14, 0.16) [0.16, 0.18) [0.18, 0.2]   
## 3.9055404 2.5431426 1.3623978 1.3623978 0.9990917 0.5449591

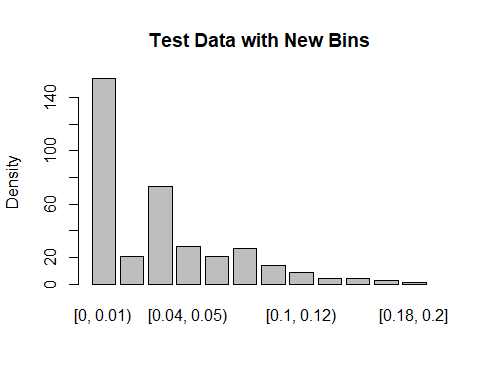
prop.table(table(testing$Bin)) \* 100

##   
## [0, 0.01) [0.01, 0.02) [0.02, 0.04) [0.04, 0.05) [0.05, 0.06) [0.06, 0.08)   
## 42.8969359 5.8495822 20.3342618 7.7994429 5.8495822 7.5208914   
## [0.08, 0.1) [0.1, 0.12) [0.12, 0.14) [0.14, 0.16) [0.16, 0.18) [0.18, 0.2]   
## 3.8997214 2.5069638 1.1142061 1.1142061 0.8356546 0.2785515

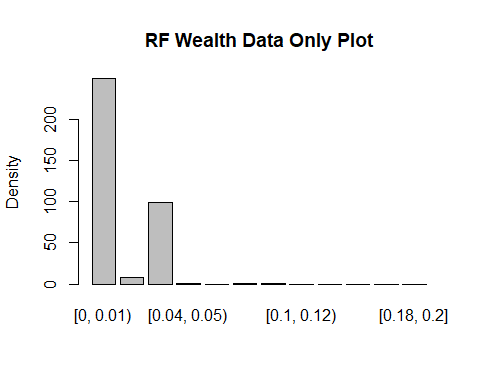
# Create variables with and without the bin  
  
x <- training[,-20]  
y <- training$Bin  
  
control <- trainControl(method="repeatedcv", number=10, repeats=3)  
seed <- 7  
metric <- "Accuracy"  
set.seed(seed)  
mtry <- sqrt(ncol(x))  
tunegrid <- expand.grid(.mtry=mtry)  
model <- train(Bin~., data=training, method="rf", metric=metric, tuneGrid=tunegrid, trControl=control)  
print(model)

## Random Forest   
##   
## 1101 samples  
## 16 predictor  
## 12 classes: '[0, 0.01)', '[0.01, 0.02)', '[0.02, 0.04)', '[0.04, 0.05)', '[0.05, 0.06)', '[0.06, 0.08)', '[0.08, 0.1)', '[0.1, 0.12)', '[0.12, 0.14)', '[0.14, 0.16)', '[0.16, 0.18)', '[0.18, 0.2]'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold, repeated 3 times)   
## Summary of sample sizes: 990, 990, 989, 992, 989, 992, ...   
## Resampling results:  
##   
## Accuracy Kappa   
## 0.4836676 0.2055558  
##   
## Tuning parameter 'mtry' was held constant at a value of 4.123106

Prediction <- predict(model, newdata=testing)  
plot(testing$Bin, ylab="Density",main="Test Data with New Bins")



plot( Prediction, ylab = "Density", main = "RF Wealth Data Only Plot")



# Results <- table(unlist(Prediction),unlist(testing$Bin))  
# Results <-as.data.frame.matrix(Results)  
# Results  
confusionMatrix(Prediction, testing$Bin)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction [0, 0.01) [0.01, 0.02) [0.02, 0.04) [0.04, 0.05) [0.05, 0.06)  
## [0, 0.01) 142 9 24 11 12  
## [0.01, 0.02) 2 3 3 0 0  
## [0.02, 0.04) 10 8 46 16 9  
## [0.04, 0.05) 0 0 0 1 0  
## [0.05, 0.06) 0 0 0 0 0  
## [0.06, 0.08) 0 1 0 0 0  
## [0.08, 0.1) 0 0 0 0 0  
## [0.1, 0.12) 0 0 0 0 0  
## [0.12, 0.14) 0 0 0 0 0  
## [0.14, 0.16) 0 0 0 0 0  
## [0.16, 0.18) 0 0 0 0 0  
## [0.18, 0.2] 0 0 0 0 0  
## Reference  
## Prediction [0.06, 0.08) [0.08, 0.1) [0.1, 0.12) [0.12, 0.14) [0.14, 0.16)  
## [0, 0.01) 19 11 9 4 4  
## [0.01, 0.02) 0 0 0 0 0  
## [0.02, 0.04) 7 3 0 0 0  
## [0.04, 0.05) 0 0 0 0 0  
## [0.05, 0.06) 0 0 0 0 0  
## [0.06, 0.08) 0 0 0 0 0  
## [0.08, 0.1) 1 0 0 0 0  
## [0.1, 0.12) 0 0 0 0 0  
## [0.12, 0.14) 0 0 0 0 0  
## [0.14, 0.16) 0 0 0 0 0  
## [0.16, 0.18) 0 0 0 0 0  
## [0.18, 0.2] 0 0 0 0 0  
## Reference  
## Prediction [0.16, 0.18) [0.18, 0.2]  
## [0, 0.01) 3 1  
## [0.01, 0.02) 0 0  
## [0.02, 0.04) 0 0  
## [0.04, 0.05) 0 0  
## [0.05, 0.06) 0 0  
## [0.06, 0.08) 0 0  
## [0.08, 0.1) 0 0  
## [0.1, 0.12) 0 0  
## [0.12, 0.14) 0 0  
## [0.14, 0.16) 0 0  
## [0.16, 0.18) 0 0  
## [0.18, 0.2] 0 0  
##   
## Overall Statistics  
##   
## Accuracy : 0.5348   
## 95% CI : (0.4817, 0.5873)  
## No Information Rate : 0.429   
## P-Value [Acc > NIR] : 3.515e-05   
##   
## Kappa : 0.2783   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: [0, 0.01) Class: [0.01, 0.02) Class: [0.02, 0.04)  
## Sensitivity 0.9221 0.142857 0.6301  
## Specificity 0.4780 0.985207 0.8147  
## Pos Pred Value 0.5703 0.375000 0.4646  
## Neg Pred Value 0.8909 0.948718 0.8962  
## Prevalence 0.4290 0.058496 0.2033  
## Detection Rate 0.3955 0.008357 0.1281  
## Detection Prevalence 0.6936 0.022284 0.2758  
## Balanced Accuracy 0.7001 0.564032 0.7224  
## Class: [0.04, 0.05) Class: [0.05, 0.06)  
## Sensitivity 0.035714 0.0000  
## Specificity 1.000000 1.0000  
## Pos Pred Value 1.000000 NaN  
## Neg Pred Value 0.924581 0.9415  
## Prevalence 0.077994 0.0585  
## Detection Rate 0.002786 0.0000  
## Detection Prevalence 0.002786 0.0000  
## Balanced Accuracy 0.517857 0.5000  
## Class: [0.06, 0.08) Class: [0.08, 0.1) Class: [0.1, 0.12)  
## Sensitivity 0.000000 0.000000 0.00000  
## Specificity 0.996988 0.997101 1.00000  
## Pos Pred Value 0.000000 0.000000 NaN  
## Neg Pred Value 0.924581 0.960894 0.97493  
## Prevalence 0.075209 0.038997 0.02507  
## Detection Rate 0.000000 0.000000 0.00000  
## Detection Prevalence 0.002786 0.002786 0.00000  
## Balanced Accuracy 0.498494 0.498551 0.50000  
## Class: [0.12, 0.14) Class: [0.14, 0.16)  
## Sensitivity 0.00000 0.00000  
## Specificity 1.00000 1.00000  
## Pos Pred Value NaN NaN  
## Neg Pred Value 0.98886 0.98886  
## Prevalence 0.01114 0.01114  
## Detection Rate 0.00000 0.00000  
## Detection Prevalence 0.00000 0.00000  
## Balanced Accuracy 0.50000 0.50000  
## Class: [0.16, 0.18) Class: [0.18, 0.2]  
## Sensitivity 0.000000 0.000000  
## Specificity 1.000000 1.000000  
## Pos Pred Value NaN NaN  
## Neg Pred Value 0.991643 0.997214  
## Prevalence 0.008357 0.002786  
## Detection Rate 0.000000 0.000000  
## Detection Prevalence 0.000000 0.000000  
## Balanced Accuracy 0.500000 0.500000

# Plot variable performance  
  
VarImport <- varImp(model)  
plot(VarImport)

