ADS 503 Project

Hunter Blum, Brianne Bell, Eva Chow

2022-06-21

### Libraries

library(haven)  
library(caret)  
library(gridExtra)  
library(corrplot)  
library(e1071)  
library(car)  
library(lattice)  
library(doParallel)  
library(RANN)  
library(rpart)  
library(party)  
library(partykit)  
library(rpart.plot)  
library(randomForest)  
library(RWeka)  
library(gbm)  
library(Cubist)  
  
library(tidyverse)

### Set up Parallelization

cl <- makeCluster(6)  
registerDoParallel(cl)

# Creating Dataset

### Read in data

Demographic <- read\_xpt("P\_DEMO.XPT")  
BodySize <- read\_xpt("P\_BMX.XPT")  
Chol\_ldl <- read\_xpt("P\_TRIGLY.XPT")

### Chol-ldl

#Select Variables of interest  
Chol\_ldl <- Chol\_ldl %>% select(SEQN, LBDLDL)  
#NA in target feature won't be useful  
Chol\_ldl <- Chol\_ldl %>% drop\_na()

### Demographic

#Get rid of variables we don't need  
Drop\_col <- c('SDDSRVYR', 'RIDSTATR', 'RIDEXMON', 'SIAPROXY', 'SIAINTRP', 'FIAPROXY', 'FIAINTRP', 'MIAPROXY', 'MIAINTRP', 'WTINTPRP', 'WTMECPRP', 'SDMVPSU', 'SDMVSTRA')  
Demographic <- Demographic %>% select(-one\_of(Drop\_col))

### BodySize

Drop\_col <- c('BMIWT', 'BMIRECUM', 'BMIHEAD', 'BMIHT', 'BMILEG', 'BMIARML', 'BMIARMC', 'BMIWAIST', 'BMIHIP', 'BMDSTATS')  
BodySize <- BodySize %>% select(-one\_of(Drop\_col))

### Join

J1 <- Chol\_ldl %>% left\_join(Demographic, by = "SEQN")  
Chol <- J1 %>% left\_join(BodySize, by = "SEQN")  
Chol <- Chol %>% select(!SEQN)

# Cleaning

## Changing factors for EDA

### Changing Variables to the Correct Type

Chol\_2 <- Chol  
factors <- c("RIAGENDR", "RIDRETH1", "RIDRETH3", "DMDBORN4", "DMDEDUC2", "DMDMARTZ", "RIDEXPRG", "SIALANG", "FIALANG", "MIALANG", "AIALANGA")  
Chol\_2[,factors] <- lapply(Chol\_2[,factors], factor)

### Change factor levels to be more interpretable

levels(Chol\_2$RIAGENDR) <- c("Male", "Female")  
levels(Chol\_2$RIDRETH1) <- c("Mex", "OHis", "White", "Black", "Oth")  
levels(Chol\_2$RIDRETH3) <- c("Mex", "OHis", "White", "Black", "Asian", "Oth")  
levels(Chol\_2$DMDBORN4) <- c("USA", "Oth", "Ref", "DK")  
levels(Chol\_2$DMDYRUSZ) <- c("<5", "5-15", "15-30", ">30", "Ref", "DK")  
levels(Chol\_2$DMDEDUC2) <- c("<9", "9-11", "HS", "AA", "BS+", "Ref", "DK")  
levels(Chol\_2$DMDMARTZ) <- c("Mar", "Sep", "Nev", "Ref", "DK")  
levels(Chol\_2$RIDEXPRG) <- c("Yes", "No", "DK")  
levels(Chol\_2$SIALANG) <- c("English", "Spanish")  
levels(Chol\_2$FIALANG) <- c("English", "Spanish")  
levels(Chol\_2$MIALANG) <- c("English", "Spanish")  
levels(Chol\_2$AIALANGA) <- c("English", "Spanish", "Asian")

# EDA

## NAs

### By Variable - Removed variables with over 3000 observations missing

Variable\_na <- Chol\_2 %>% select(everything()) %>% summarise\_all(funs(sum(is.na(.)))) %>% pivot\_longer(cols = c(colnames(Chol\_2[,1:ncol(Chol\_2)])), names\_to = "Variable", values\_to = "Missing") %>% arrange(desc(Missing))  
Drop\_col <- c("RIDAGEMN", "BMXRECUM", "BMXHEAD", "BMDBMIC", "RIDEXPRG", "DMDYRUSZ")  
Chol\_2 <- Chol\_2 %>% select(-one\_of(Drop\_col))

### By Row

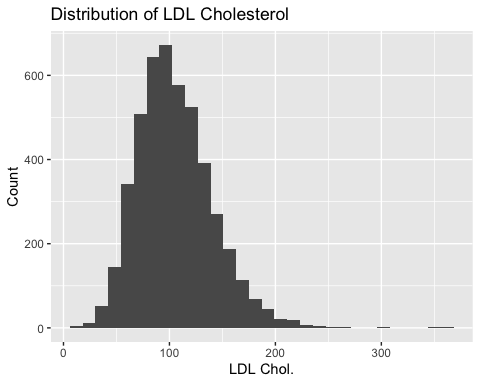
row\_na <- rowSums(is.na(Chol\_2))  
row\_na <- data.frame(row\_na, Row = c(1:length(row\_na)))  
row\_na <- row\_na %>% arrange(desc(row\_na))  
#Most missing values in a row is 12, not bad

## Distributions

### Response - LDL Cholesterol

#Looks like a fairly normal distribution, maybe a little skewed to the right.   
ggplot(Chol\_2, aes(x = LBDLDL)) + geom\_histogram() + ggtitle("Distribution of LDL Cholesterol") + xlab("LDL Chol.") + ylab("Count")

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



skewness(Chol\_2$LBDLDL)

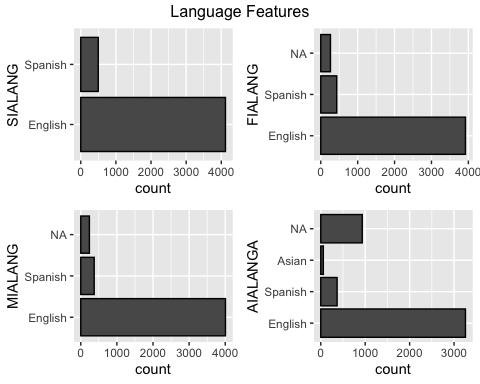
## [1] 0.7886403

#skewness value .7886403 confirms very mild skewness to the right

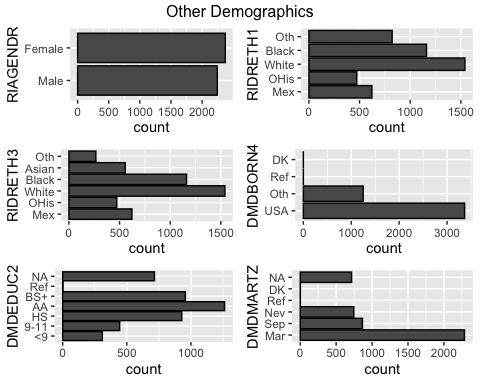
### Predictors

Factors

Chol\_fact <- Chol\_2 %>% select\_if(is.factor)  
Chol.bar <- function(xvar){  
 ggplot(Chol\_fact, aes\_(x = as.name(xvar))) +  
 geom\_bar(color = "black") + coord\_flip()  
}  
Lang\_barplots <- lapply(names(Chol\_fact[,7:10]), Chol.bar)  
Oth\_barplots <- lapply(names(Chol\_fact[,1:6]), Chol.bar)  
grid.arrange(grobs = Lang\_barplots, top = "Language Features")

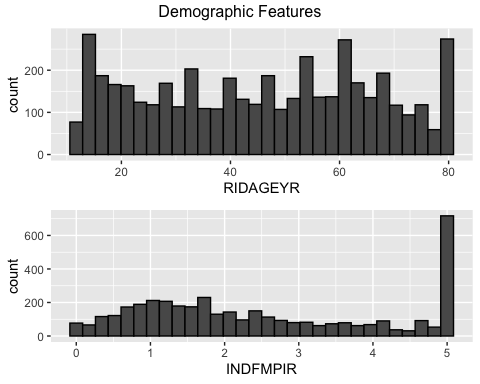


grid.arrange(grobs = Oth\_barplots, top = "Other Demographics")

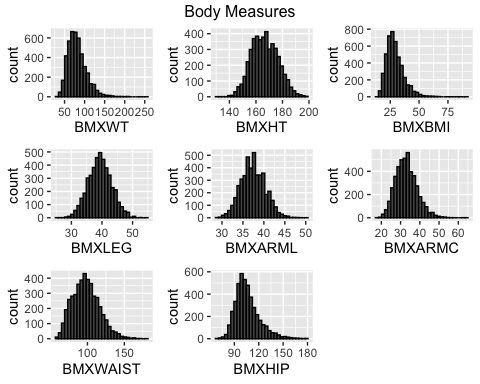


Numeric

Chol\_num <- Chol\_2 %>% select\_if(is.numeric) %>% select(!LBDLDL)  
Chol.hist <- function(xvar){  
 ggplot(Chol\_num, aes\_(x = as.name(xvar))) +  
 geom\_histogram(color = "black")   
}  
Dem\_hist <- lapply(names(Chol\_num[,1:2]), Chol.hist)  
Body\_hist <- lapply(names(Chol\_num[,3:10]), Chol.hist)  
grid.arrange(grobs = Dem\_hist, top = "Demographic Features")



grid.arrange(grobs = Body\_hist, top = "Body Measures")



## Correlations

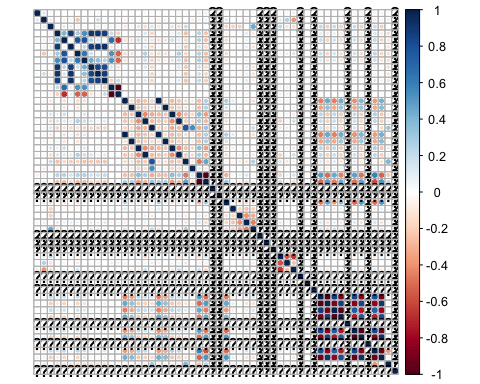
Heatmap

Chol\_dummy <- fastDummies::dummy\_cols(Chol\_2)  
Chol\_dummy <- Chol\_dummy %>% select\_if(~!is.factor(.))  
Chol\_dummy[] <- lapply(Chol\_dummy, as.numeric)  
Chol\_cor <- cor(Chol\_dummy, use = "complete.obs")

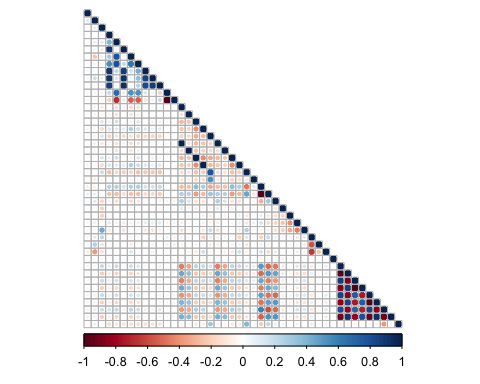
## Warning in cor(Chol\_dummy, use = "complete.obs"): the standard deviation is zero

Chol\_corplot <- corrplot(cor(Chol\_dummy, use = "complete.obs"), tl.pos = 'n')

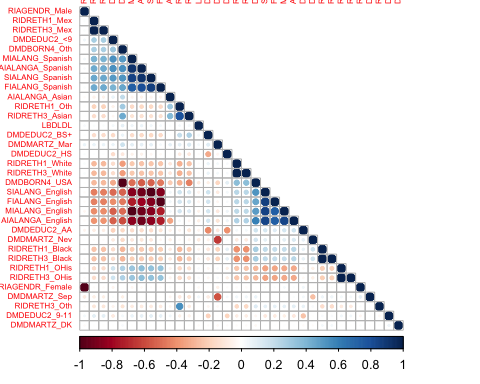
## Warning in cor(Chol\_dummy, use = "complete.obs"): the standard deviation is zero



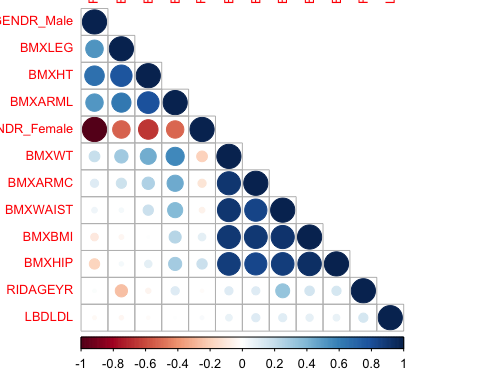
#Looks like some dummy variables that are refusal could be messing up correlations  
Drop\_col <- c("DMDBORN4\_Ref", "DMDBORN4\_DK", "DMDEDUC2\_Ref", "DMDEDUC2\_DK", "DMDEDUC2\_NA", "DMDMARTZ\_Ref", "DMDMARTZ\_NA", "FIALANG\_NA", "MIALANG\_NA", "AIALANGA\_NA")  
Chol\_dummy\_2 <- Chol\_dummy %>% select(-one\_of(Drop\_col))  
invisible(cor(Chol\_dummy\_2, use = "complete.obs")) # using invisible() to reduce extensive output  
corrplot(cor(Chol\_dummy\_2, use = "complete.obs"), tl.pos = 'n', type = 'lower')

 ### Smaller plots for easier interpretation.

# Mini Correlations: Sociological Measures:  
socio <- Chol\_dummy\_2[,c("RIAGENDR\_Male","RIAGENDR\_Female",  
 "RIDRETH1\_Mex", "RIDRETH1\_OHis", "RIDRETH1\_White",   
 "RIDRETH1\_Black", "RIDRETH1\_Oth", "RIDRETH3\_Mex", "RIDRETH3\_OHis",  
 "RIDRETH3\_White", "RIDRETH3\_Black", "RIDRETH3\_Asian", "RIDRETH3\_Oth",  
 "DMDBORN4\_USA", "DMDBORN4\_Oth", "DMDEDUC2\_<9", "DMDEDUC2\_9-11",  
 "DMDEDUC2\_HS", "DMDEDUC2\_AA", "DMDEDUC2\_BS+","DMDMARTZ\_Mar",  
 "DMDMARTZ\_Sep", "DMDMARTZ\_Nev", "DMDMARTZ\_DK", "SIALANG\_English",   
 "SIALANG\_Spanish", "FIALANG\_English", "FIALANG\_Spanish",  
 "MIALANG\_English","MIALANG\_Spanish", "AIALANGA\_English",  
 "AIALANGA\_Spanish", "AIALANGA\_Asian", "LBDLDL")]  
  
invisible(cor(socio, use = "complete.obs")) # using invisible() to reduce extensive output  
corrplot(cor(socio, use = "complete.obs"), tl.pos = 'y', type = 'lower',   
 order = "hclust", tl.cex = 0.5)



# Mini Correlations: Biological Measures:  
biologic <- Chol\_dummy\_2[,c("RIDAGEYR", "BMXWT", "BMXHT", "BMXBMI",   
 "BMXLEG", "BMXARML", "BMXARMC", "BMXWAIST",  
 "BMXHIP", "RIAGENDR\_Male", "RIAGENDR\_Female","LBDLDL")]  
invisible(cor(biologic, use = "complete.obs")) # using invisible() to reduce extensive output  
corrplot(cor(biologic, use = "complete.obs"), tl.pos = 'y', type = 'lower',   
 order = "hclust", tl.cex = 0.8)



# correlations between certain biological measures make sense. BMI is derived from the MASS and height of an individual, so it makes sense that many of the BMI measurements correlate with each other. (i.e. hip, waist, and weight measurements correlate with a higher BMI. Being female correlates negatively with leg and arm length as well as height)

### Highly-correlated variables in cholesterol

Note: Highly correlated variables were removed in the model training process - this was just for EDA

# let's check for highly correlated predictors  
# we'll do this on our non factor transformed dataset  
dim(Chol)

## [1] 4617 27

# 27 variables. let's find correlations greater than 0.80 and see how the data looks if removed  
corr\_Chol <- cor(Chol)  
  
# if removed, how many variables are left  
high\_corr\_Chol <- findCorrelation(corr\_Chol, cutoff = 0.80)  
no\_corr\_Chol <- Chol[, -high\_corr\_Chol]  
dim(no\_corr\_Chol)

## [1] 4617 26

### Looking at a simple ols model to get an idea of important predictors.

# looking at a base linear model, to see significant variables   
model0 <- lm(LBDLDL~., Chol\_2)  
summary(model0)

##   
## Call:  
## lm(formula = LBDLDL ~ ., data = Chol\_2)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -102.103 -23.607 -2.937 19.941 248.656   
##   
## Coefficients: (4 not defined because of singularities)  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 112.86475 55.86920 2.020 0.043473 \*   
## RIAGENDRFemale -1.91563 2.57466 -0.744 0.456926   
## RIDAGEYR 0.24155 0.06308 3.829 0.000132 \*\*\*  
## RIDRETH1OHis 0.32070 2.94340 0.109 0.913247   
## RIDRETH1White 2.35185 2.76857 0.849 0.395694   
## RIDRETH1Black -0.73860 2.94082 -0.251 0.801716   
## RIDRETH1Oth 4.71480 3.77829 1.248 0.212196   
## RIDRETH3OHis NA NA NA NA   
## RIDRETH3White NA NA NA NA   
## RIDRETH3Black NA NA NA NA   
## RIDRETH3Asian -2.84491 4.23201 -0.672 0.501495   
## RIDRETH3Oth NA NA NA NA   
## DMDBORN4Oth 4.36638 2.46324 1.773 0.076414 .   
## DMDEDUC29-11 4.10693 3.99431 1.028 0.303959   
## DMDEDUC2HS 2.20049 3.78025 0.582 0.560550   
## DMDEDUC2AA 1.20340 3.78500 0.318 0.750558   
## DMDEDUC2BS+ 3.46252 3.97469 0.871 0.383761   
## DMDMARTZSep 1.97727 1.98256 0.997 0.318702   
## DMDMARTZNev -1.53988 1.94945 -0.790 0.429657   
## DMDMARTZDK -41.40880 35.15436 -1.178 0.238943   
## SIALANGSpanish -7.99794 6.62103 -1.208 0.227177   
## FIALANGSpanish 0.40544 4.87296 0.083 0.933697   
## MIALANGSpanish -5.35299 6.25014 -0.856 0.391825   
## AIALANGASpanish 15.53960 7.63044 2.037 0.041803 \*   
## AIALANGAAsian 1.27914 5.50446 0.232 0.816261   
## INDFMPIR 0.52626 0.52568 1.001 0.316879   
## BMXWT -0.01949 0.30794 -0.063 0.949549   
## BMXHT -0.38550 0.36123 -1.067 0.286000   
## BMXBMI -0.88512 0.96788 -0.914 0.360544   
## BMXLEG 0.69926 0.32788 2.133 0.033051 \*   
## BMXARML -0.24752 0.46888 -0.528 0.597616   
## BMXARMC 0.98602 0.34684 2.843 0.004507 \*\*   
## BMXWAIST 0.24437 0.13572 1.801 0.071888 .   
## BMXHIP -0.01965 0.18110 -0.109 0.913595   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 34.91 on 2498 degrees of freedom  
## (2089 observations deleted due to missingness)  
## Multiple R-squared: 0.03362, Adjusted R-squared: 0.0224   
## F-statistic: 2.997 on 29 and 2498 DF, p-value: 1.543e-07

# significant contributors: variable (Pr(>|t|))   
 # RIDAGEYR (0.000132)  
 # (Intercept) (0.043474)  
 # MDBORN4Oth (0.076414)   
 # AIALANGASpanish (0.041803)  
 # BMXLEG (0.033051)  
 # BMXARMC (0.004507)   
 # BMXWAIST (0.071888)

Looking at VIF of simple model showed aliased coefficients. Removed them in the model training process.

# looking at VIF for baseline linear:  
#vif(model0)  
# highly/perfectly correlated factors, we might need to drop some

### Degenerate Predictors in the non-dummy dataset

# Let's check for degenerate predictors from the original dataset  
nearZeroVar(Chol, saveMetrics = FALSE)

## [1] 4 11 16 18 19

deg\_chol <- subset(Chol, select=c(4,11,16,18,19))  
colnames(deg\_chol)

## [1] "RIDAGEMN" "RIDEXPRG" "INDFMPIR" "BMXRECUM" "BMXHEAD"

# Do it again on factor dataset  
nearZeroVar(Chol\_2, saveMetrics = FALSE)

## [1] 13

deg\_chol2 <- subset(Chol\_2, select=c(13))  
colnames(deg\_chol2)

## [1] "INDFMPIR"

# we may have to consider removing depending on the data used for modeling

# Preparing data for modeling

### Splitting dummy-variable data set and resampling

# set the seed and split the data. We'll do an 80/20 split  
set.seed(123)  
Chol\_split <- createDataPartition(Chol$LBDLDL, p=0.80, list=FALSE)  
  
# split into train and test  
Chol\_train <- Chol\_dummy[Chol\_split,]  
Chol\_test <- Chol\_dummy[-Chol\_split,]  
  
# split predictors from the target  
Chol\_train\_X <- as.data.frame(subset(Chol\_train, select=-c(LBDLDL)))   
Chol\_train\_y <- Chol\_train$LBDLDL  
  
Chol\_test\_X <- as.data.frame(subset(Chol\_test, select=-c(LBDLDL)))  
Chol\_test\_y <- Chol\_test$LBDLDL  
  
# Creating imputed data sets  
Chol\_imp <- preProcess(Chol\_train\_X, method = c("center", "scale", "knnImpute"))  
Chol\_train\_X\_imp <- predict(Chol\_imp, Chol\_train\_X)  
Chol\_test\_X\_imp <- predict(Chol\_imp, Chol\_test\_X)  
  
# Adding Resampling/Validation Set and Control   
set.seed(123)  
Chol\_folds <- createFolds(y = Chol\_train\_X, k = 10, returnTrain = T)  
Chol\_control <- trainControl(method = "cv", index = Chol\_folds)

### Numeric training data set with just the numeric variables and gender (Just going off a hunch that having too many dummy variables is hurting linear model performance).

Drop\_col <- c('RIDRETH1', 'RIDRETH3', 'DMDBORN4', 'DMDEDUC2', 'DMDMARTZ', 'SIALANG', 'FIALANG', 'MIALANG', 'AIALANGA', 'LBDLDL')  
Chol\_num <- Chol\_2 %>% select(-one\_of(Drop\_col))  
Chol\_dummy <- fastDummies::dummy\_cols(Chol\_num)  
Chol\_num <- Chol\_dummy %>% select\_if(~!is.factor(.))  
Chol\_num[] <- lapply(Chol\_num, as.numeric)  
  
Chol\_num\_tr\_X <- as.data.frame(Chol\_num[Chol\_split, ])  
Chol\_num\_test\_X <- as.data.frame(Chol\_num[-Chol\_split, ])  
  
#Preprocess  
Chol\_imp <- preProcess(Chol\_num\_tr\_X, method = c("center", "scale", "knnImpute"))  
Chol\_num\_tr\_X <- predict(Chol\_imp, Chol\_num\_tr\_X)  
Chol\_num\_test\_X <- predict(Chol\_imp, Chol\_num\_test\_X)  
  
# Adding Resampling/Validation Set and Control   
set.seed(123)  
Chol\_folds\_num <- createFolds(y = Chol\_num\_tr\_X, k = 10, returnTrain = T)  
Chol\_control\_num <- trainControl(method = "cv", index = Chol\_folds\_num)

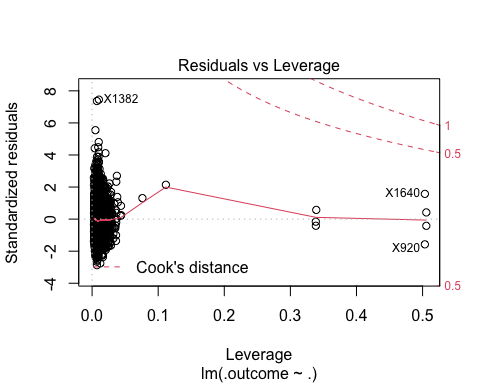
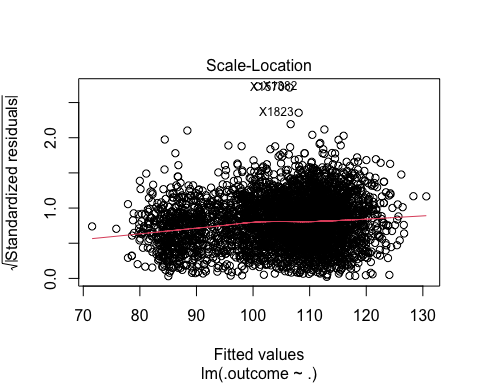
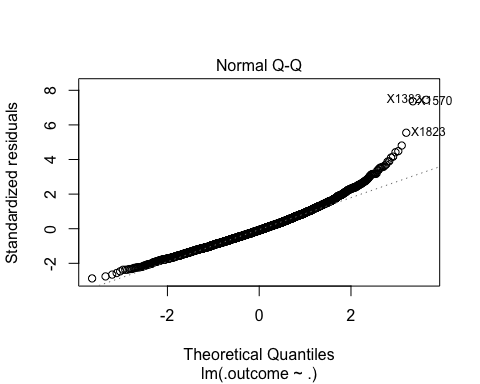
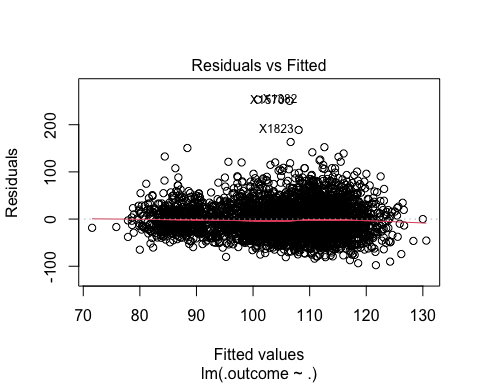
# Linear Models - Hunter

## OLS

### Create Initial Model

Chol\_ols\_tune <- train(x = Chol\_train\_X\_imp, y = Chol\_train\_y, method = "lm", trControl = Chol\_control)  
plot(Chol\_ols\_tune$finalModel)

## Warning: not plotting observations with leverage one:  
## 704, 1663

 ### FIX TRAINING SET

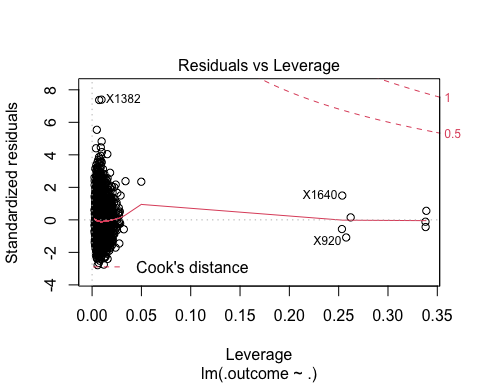
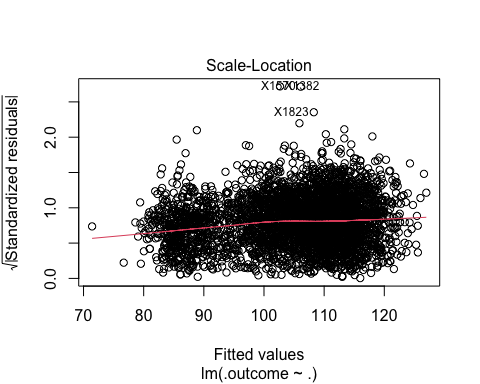
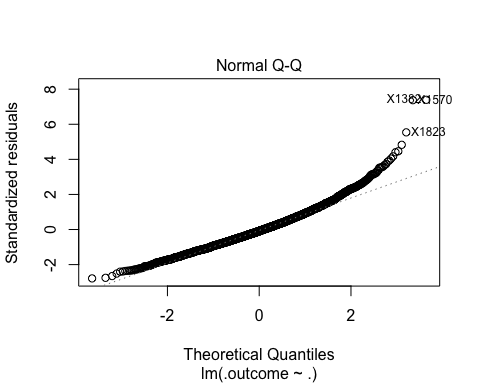
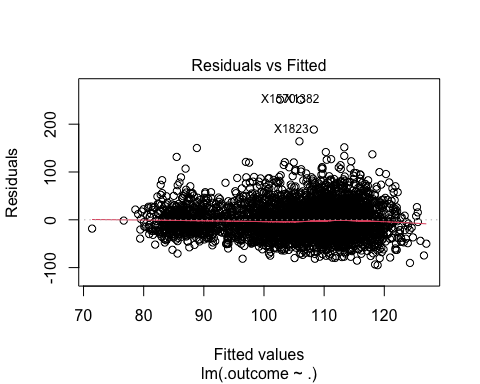
# VIF shows aliased coefficients, need to get rid of those by removing high cor predictors  
test <- cor(Chol\_train\_X\_imp)

## Warning in cor(Chol\_train\_X\_imp): the standard deviation is zero

# Also have an issue with DMDEDUC2\_DK all being zero so get rid of high var predictors  
Chol\_tr\_x\_imp\_vr <- Chol\_train\_X\_imp[, -nearZeroVar(Chol\_train\_X\_imp)]  
Chol\_tr\_X\_imp\_fin <- Chol\_tr\_x\_imp\_vr[, -findCorrelation(cor(Chol\_tr\_x\_imp\_vr), cutoff = 0.9)]

### Tune Another Model

Chol\_ols\_tune2 <- train(x = Chol\_tr\_X\_imp\_fin, y = Chol\_train\_y, method = "lm", trControl = Chol\_control)  
plot(Chol\_ols\_tune2$finalModel)

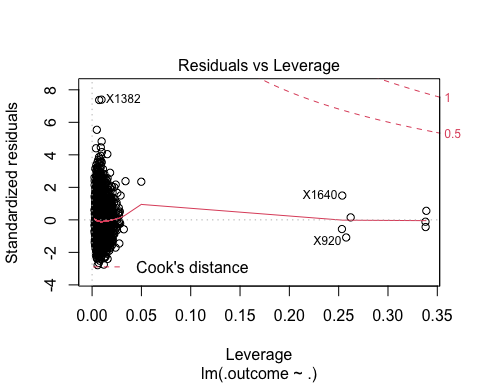
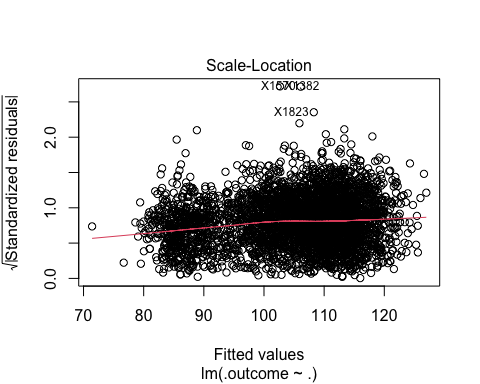
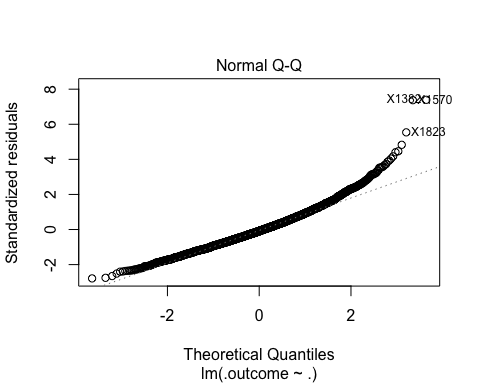
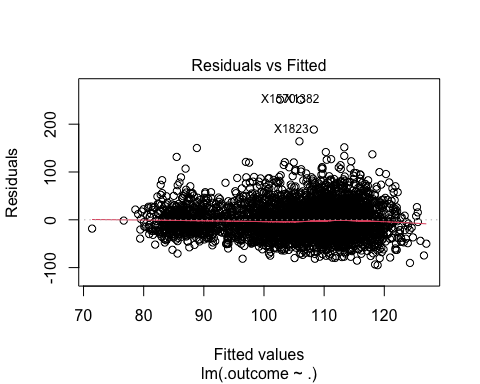


summary(Chol\_ols\_tune2$finalModel)

##   
## Call:  
## lm(formula = .outcome ~ ., data = dat)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -94.895 -23.065 -3.065 19.439 251.335   
##   
## Coefficients: (2 not defined because of singularities)  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 105.31270 0.57066 184.546 < 2e-16 \*\*\*  
## RIDAGEYR 4.52387 1.05609 4.284 1.89e-05 \*\*\*  
## INDFMPIR 0.25641 0.71302 0.360 0.71915   
## BMXHT -3.83360 1.33252 -2.877 0.00404 \*\*   
## BMXBMI -4.35512 1.45346 -2.996 0.00275 \*\*   
## BMXLEG 2.97807 1.01656 2.930 0.00342 \*\*   
## BMXARML 0.27463 1.09490 0.251 0.80196   
## BMXARMC 5.96990 1.52805 3.907 9.52e-05 \*\*\*  
## RIAGENDR\_Male -0.86654 0.80467 -1.077 0.28160   
## RIDRETH1\_OHis 0.31552 0.75090 0.420 0.67437   
## RIDRETH1\_Oth 1.30082 1.01537 1.281 0.20023   
## RIDRETH3\_Mex 1.02780 0.79135 1.299 0.19410   
## RIDRETH3\_White 0.89147 0.75600 1.179 0.23840   
## RIDRETH3\_Black NA NA NA NA   
## RIDRETH3\_Asian -0.42976 1.10977 -0.387 0.69859   
## RIDRETH3\_Oth NA NA NA NA   
## DMDBORN4\_Oth 1.91695 0.92813 2.065 0.03896 \*   
## `DMDEDUC2\_<9` -6.24744 5.32698 -1.173 0.24096   
## `DMDEDUC2\_9-11` -5.93924 6.31711 -0.940 0.34719   
## DMDEDUC2\_HS -8.35928 8.53591 -0.979 0.32749   
## DMDEDUC2\_AA -9.09136 9.35132 -0.972 0.33101   
## `DMDEDUC2\_BS+` -7.28858 8.56310 -0.851 0.39474   
## DMDEDUC2\_NA -5.14541 0.80362 -6.403 1.72e-10 \*\*\*  
## DMDMARTZ\_Mar 4.16243 8.48632 0.490 0.62382   
## DMDMARTZ\_Sep 4.08979 7.16973 0.570 0.56842   
## DMDMARTZ\_Nev 2.60283 6.84793 0.380 0.70390   
## SIALANG\_Spanish -0.59672 1.37337 -0.434 0.66395   
## FIALANG\_Spanish 0.39180 1.11501 0.351 0.72532   
## FIALANG\_NA -0.24256 0.56982 -0.426 0.67037   
## MIALANG\_Spanish -0.08927 1.15483 -0.077 0.93838   
## MIALANG\_NA 2.96009 0.68244 4.338 1.48e-05 \*\*\*  
## AIALANGA\_English -1.19479 1.19890 -0.997 0.31904   
## AIALANGA\_NA -6.21091 0.82160 -7.560 5.08e-14 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 34.15 on 3665 degrees of freedom  
## Multiple R-squared: 0.08048, Adjusted R-squared: 0.07296   
## F-statistic: 10.69 on 30 and 3665 DF, p-value: < 2.2e-16

### Tune model with BoxCox to see if it will help normality issues - Didn’t do much, we’ll just stick with the non-transformed data. Also tried transformin LDL, didn’t work either.

Chol\_bct <- preProcess(Chol\_tr\_X\_imp\_fin, method = "BoxCox")  
Chol\_tr\_boxcox <- predict(Chol\_bct, Chol\_tr\_X\_imp\_fin)  
  
Chol\_ols\_tune3 <- train(x = Chol\_tr\_boxcox, y = Chol\_train\_y, method = "lm", trControl = Chol\_control)  
plot(Chol\_ols\_tune3$finalModel)

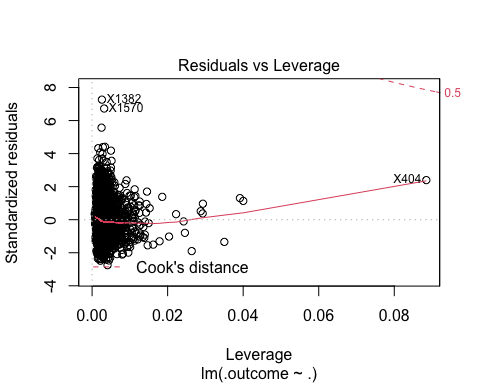
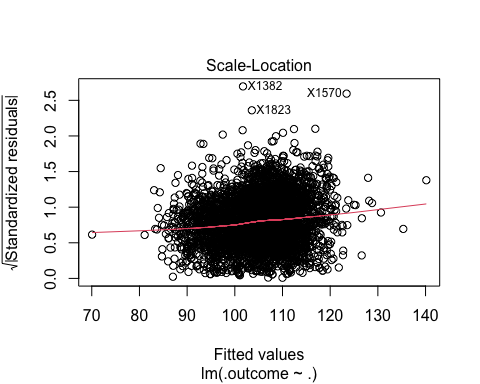
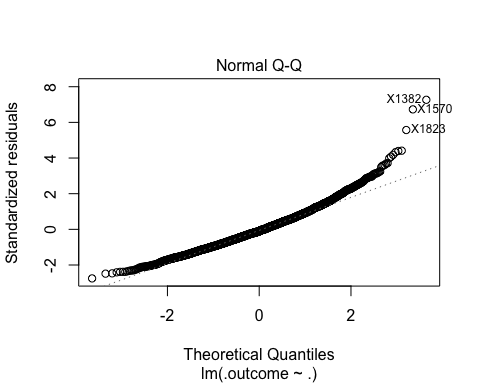
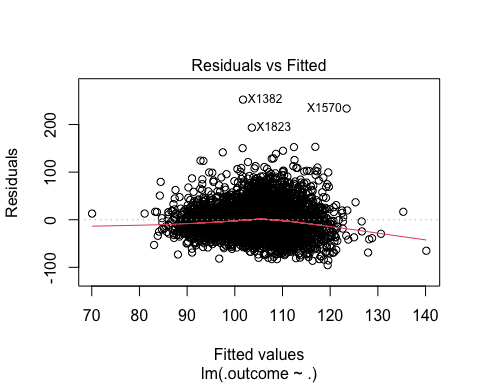


summary(Chol\_ols\_tune3$finalModel)

##   
## Call:  
## lm(formula = .outcome ~ ., data = dat)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -94.895 -23.065 -3.065 19.439 251.335   
##   
## Coefficients: (2 not defined because of singularities)  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 105.31270 0.57066 184.546 < 2e-16 \*\*\*  
## RIDAGEYR 4.52387 1.05609 4.284 1.89e-05 \*\*\*  
## INDFMPIR 0.25641 0.71302 0.360 0.71915   
## BMXHT -3.83360 1.33252 -2.877 0.00404 \*\*   
## BMXBMI -4.35512 1.45346 -2.996 0.00275 \*\*   
## BMXLEG 2.97807 1.01656 2.930 0.00342 \*\*   
## BMXARML 0.27463 1.09490 0.251 0.80196   
## BMXARMC 5.96990 1.52805 3.907 9.52e-05 \*\*\*  
## RIAGENDR\_Male -0.86654 0.80467 -1.077 0.28160   
## RIDRETH1\_OHis 0.31552 0.75090 0.420 0.67437   
## RIDRETH1\_Oth 1.30082 1.01537 1.281 0.20023   
## RIDRETH3\_Mex 1.02780 0.79135 1.299 0.19410   
## RIDRETH3\_White 0.89147 0.75600 1.179 0.23840   
## RIDRETH3\_Black NA NA NA NA   
## RIDRETH3\_Asian -0.42976 1.10977 -0.387 0.69859   
## RIDRETH3\_Oth NA NA NA NA   
## DMDBORN4\_Oth 1.91695 0.92813 2.065 0.03896 \*   
## `DMDEDUC2\_<9` -6.24744 5.32698 -1.173 0.24096   
## `DMDEDUC2\_9-11` -5.93924 6.31711 -0.940 0.34719   
## DMDEDUC2\_HS -8.35928 8.53591 -0.979 0.32749   
## DMDEDUC2\_AA -9.09136 9.35132 -0.972 0.33101   
## `DMDEDUC2\_BS+` -7.28858 8.56310 -0.851 0.39474   
## DMDEDUC2\_NA -5.14541 0.80362 -6.403 1.72e-10 \*\*\*  
## DMDMARTZ\_Mar 4.16243 8.48632 0.490 0.62382   
## DMDMARTZ\_Sep 4.08979 7.16973 0.570 0.56842   
## DMDMARTZ\_Nev 2.60283 6.84793 0.380 0.70390   
## SIALANG\_Spanish -0.59672 1.37337 -0.434 0.66395   
## FIALANG\_Spanish 0.39180 1.11501 0.351 0.72532   
## FIALANG\_NA -0.24256 0.56982 -0.426 0.67037   
## MIALANG\_Spanish -0.08927 1.15483 -0.077 0.93838   
## MIALANG\_NA 2.96009 0.68244 4.338 1.48e-05 \*\*\*  
## AIALANGA\_English -1.19479 1.19890 -0.997 0.31904   
## AIALANGA\_NA -6.21091 0.82160 -7.560 5.08e-14 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 34.15 on 3665 degrees of freedom  
## Multiple R-squared: 0.08048, Adjusted R-squared: 0.07296   
## F-statistic: 10.69 on 30 and 3665 DF, p-value: < 2.2e-16

### Try reduced data - Didn’t really help our diagnostic plot, so we’ll go with the regular dummy data

Chol\_ols\_tune\_num <- train(x = Chol\_num\_tr\_X, y = Chol\_train\_y, method = "lm", trControl = Chol\_control\_num)  
plot(Chol\_ols\_tune\_num$finalModel)

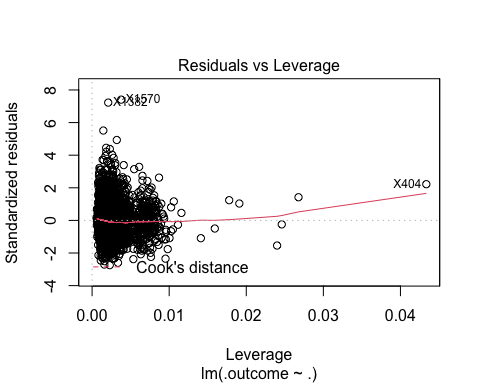
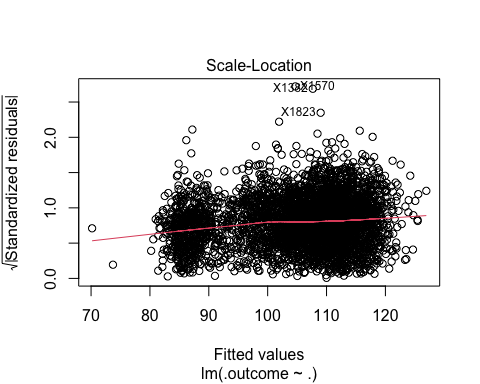
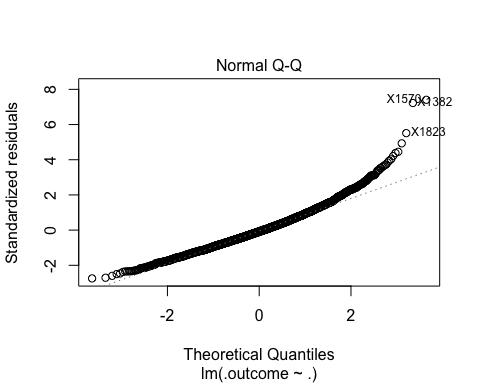
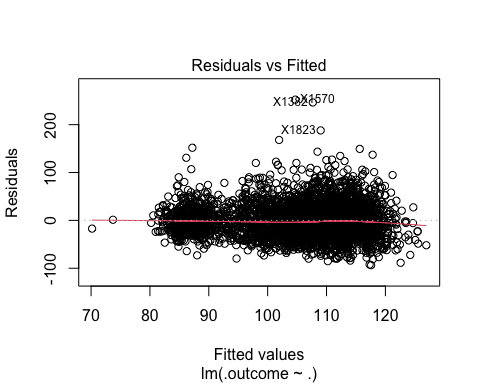


summary(Chol\_ols\_tune\_num$finalModel)

##   
## Call:  
## lm(formula = .outcome ~ ., data = dat)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -95.467 -23.705 -3.484 19.616 252.285   
##   
## Coefficients: (1 not defined because of singularities)  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 105.2332 0.5724 183.830 < 2e-16 \*\*\*  
## RIDAGEYR 4.7111 0.7542 6.247 4.67e-10 \*\*\*  
## INDFMPIR 1.2124 0.6242 1.942 0.052179 .   
## BMXWT -3.0812 5.6647 -0.544 0.586519   
## BMXHT 0.6167 2.7006 0.228 0.819391   
## BMXBMI -2.1456 5.7513 -0.373 0.709124   
## BMXLEG 1.9531 1.0102 1.933 0.053275 .   
## BMXARML -2.3916 1.0924 -2.189 0.028638 \*   
## BMXARMC 11.0858 1.5514 7.146 1.08e-12 \*\*\*  
## BMXWAIST 4.0671 1.9632 2.072 0.038369 \*   
## BMXHIP -6.6430 2.2078 -3.009 0.002639 \*\*   
## RIAGENDR\_Male -3.3691 0.9501 -3.546 0.000396 \*\*\*  
## RIAGENDR\_Female NA NA NA NA   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 34.76 on 3684 degrees of freedom  
## Multiple R-squared: 0.04226, Adjusted R-squared: 0.0394   
## F-statistic: 14.78 on 11 and 3684 DF, p-value: < 2.2e-16

### Final OLS Model

Chol\_sig\_tr <- Chol\_tr\_X\_imp\_fin %>% select(RIDAGEYR, BMXHT, BMXBMI, BMXLEG, BMXARMC, DMDBORN4\_Oth, DMDEDUC2\_NA, MIALANG\_NA, AIALANGA\_NA)  
Chol\_ols <- train(x = Chol\_sig\_tr, y = Chol\_train\_y, method = "lm", trControl = Chol\_control)  
plot(Chol\_ols$finalModel)



summary(Chol\_ols$finalModel)

##   
## Call:  
## lm(formula = .outcome ~ ., data = dat)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -93.570 -23.022 -2.777 19.504 252.299   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 105.2477 0.5615 187.436 < 2e-16 \*\*\*  
## RIDAGEYR 4.8758 0.9228 5.284 1.34e-07 \*\*\*  
## BMXHT -3.6994 0.9837 -3.761 0.000172 \*\*\*  
## BMXBMI -4.3529 1.4249 -3.055 0.002268 \*\*   
## BMXLEG 2.3397 0.9676 2.418 0.015656 \*   
## BMXARMC 5.8276 1.4811 3.935 8.49e-05 \*\*\*  
## DMDBORN4\_Oth 2.3726 0.6055 3.918 9.07e-05 \*\*\*  
## DMDEDUC2\_NA -5.1963 0.7824 -6.641 3.57e-11 \*\*\*  
## MIALANG\_NA 2.9351 0.6732 4.360 1.34e-05 \*\*\*  
## AIALANGA\_NA -6.3145 0.8050 -7.844 5.65e-15 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 34.13 on 3686 degrees of freedom  
## Multiple R-squared: 0.07647, Adjusted R-squared: 0.07421   
## F-statistic: 33.91 on 9 and 3686 DF, p-value: < 2.2e-16

#Predict on test data  
Chol\_ols\_res <- predict(Chol\_ols, Chol\_test\_X)

## PCR and PLS

### PCR

set.seed(123)  
Chol\_pcr <- train(x = Chol\_tr\_X\_imp\_fin, y = Chol\_train\_y, method = "pcr", tuneGrid = expand.grid(ncomp=1:32), trControl = Chol\_control)  
Chol\_pcr

## Principal Component Analysis   
##   
## 3696 samples  
## 32 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 51, 50, 48, 50, 46, 46, ...   
## Resampling results across tuning parameters:  
##   
## ncomp RMSE Rsquared MAE   
## 1 3.607365e+01 0.0009472784 2.824159e+01  
## 2 3.582222e+01 0.0056720037 2.796978e+01  
## 3 3.605529e+01 0.0065705319 2.821513e+01  
## 4 3.595069e+01 0.0093855966 2.810244e+01  
## 5 3.585736e+01 0.0118360813 2.800256e+01  
## 6 3.587428e+01 0.0116831830 2.801287e+01  
## 7 3.595720e+01 0.0111553267 2.808494e+01  
## 8 3.597211e+01 0.0119099822 2.813118e+01  
## 9 3.610565e+01 0.0112405860 2.826877e+01  
## 10 3.628901e+01 0.0112251080 2.844356e+01  
## 11 3.654682e+01 0.0108373263 2.862576e+01  
## 12 3.659482e+01 0.0109789513 2.866332e+01  
## 13 3.674159e+01 0.0104754941 2.877617e+01  
## 14 3.675477e+01 0.0107158572 2.877915e+01  
## 15 3.675366e+01 0.0118199483 2.876276e+01  
## 16 3.658668e+01 0.0153222354 2.858466e+01  
## 17 3.674689e+01 0.0141178136 2.871621e+01  
## 18 3.725190e+01 0.0115914314 2.914393e+01  
## 19 3.828365e+01 0.0072457628 2.991566e+01  
## 20 3.966385e+01 0.0019887547 3.106541e+01  
## 21 4.006473e+01 0.0022332617 3.139574e+01  
## 22 4.062474e+01 0.0019996999 3.184595e+01  
## 23 4.373689e+01 0.0027565697 3.423241e+01  
## 24 4.841546e+01 0.0034073520 3.790139e+01  
## 25 4.990853e+01 0.0029138702 3.895776e+01  
## 26 2.340062e+13 0.0022746643 7.300666e+12  
## 27 2.422551e+14 0.0010076512 5.085548e+13  
## 28 2.395890e+15 0.0002912437 5.821270e+14  
## 29 2.445967e+15 0.0001811593 5.327135e+14  
## 30 2.699350e+15 0.0003335516 5.065370e+14  
## 31 5.277874e+15 0.0003205010 1.019668e+15  
## 32 9.774703e+15 0.0006010439 2.113406e+15  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final value used for the model was ncomp = 2.

set.seed(123)  
Chol\_pcr\_box <- train(x = Chol\_tr\_boxcox, y = Chol\_train\_y, method = "pcr", tuneGrid = expand.grid(ncomp=1:32), trControl = Chol\_control)  
Chol\_pcr\_box

## Principal Component Analysis   
##   
## 3696 samples  
## 32 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 51, 50, 48, 50, 46, 46, ...   
## Resampling results across tuning parameters:  
##   
## ncomp RMSE Rsquared MAE   
## 1 3.607365e+01 0.0009472784 2.824159e+01  
## 2 3.582222e+01 0.0056720037 2.796978e+01  
## 3 3.605529e+01 0.0065705319 2.821513e+01  
## 4 3.595069e+01 0.0093855966 2.810244e+01  
## 5 3.585736e+01 0.0118360813 2.800256e+01  
## 6 3.587428e+01 0.0116831830 2.801287e+01  
## 7 3.595720e+01 0.0111553267 2.808494e+01  
## 8 3.597211e+01 0.0119099822 2.813118e+01  
## 9 3.610565e+01 0.0112405860 2.826877e+01  
## 10 3.628901e+01 0.0112251080 2.844356e+01  
## 11 3.654682e+01 0.0108373263 2.862576e+01  
## 12 3.659482e+01 0.0109789513 2.866332e+01  
## 13 3.674159e+01 0.0104754941 2.877617e+01  
## 14 3.675477e+01 0.0107158572 2.877915e+01  
## 15 3.675366e+01 0.0118199483 2.876276e+01  
## 16 3.658668e+01 0.0153222354 2.858466e+01  
## 17 3.674689e+01 0.0141178136 2.871621e+01  
## 18 3.725190e+01 0.0115914314 2.914393e+01  
## 19 3.828365e+01 0.0072457628 2.991566e+01  
## 20 3.966385e+01 0.0019887547 3.106541e+01  
## 21 4.006473e+01 0.0022332617 3.139574e+01  
## 22 4.062474e+01 0.0019996999 3.184595e+01  
## 23 4.373689e+01 0.0027565697 3.423241e+01  
## 24 4.841546e+01 0.0034073520 3.790139e+01  
## 25 4.990853e+01 0.0029138702 3.895776e+01  
## 26 2.340062e+13 0.0022746643 7.300666e+12  
## 27 2.422551e+14 0.0010076512 5.085548e+13  
## 28 2.395890e+15 0.0002912437 5.821270e+14  
## 29 2.445967e+15 0.0001811593 5.327135e+14  
## 30 2.699350e+15 0.0003335516 5.065370e+14  
## 31 5.277874e+15 0.0003205010 1.019668e+15  
## 32 9.774703e+15 0.0006010439 2.113406e+15  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final value used for the model was ncomp = 2.

set.seed(123)  
Chol\_pcr\_num <- train(x = Chol\_num\_tr\_X, y = Chol\_train\_y, method = "pcr", tuneGrid = expand.grid(ncomp=1:8), trControl = Chol\_control\_num)  
Chol\_pcr\_num

## Principal Component Analysis   
##   
## 3696 samples  
## 12 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 11, 9, 11, 10, 10, 11, ...   
## Resampling results across tuning parameters:  
##   
## ncomp RMSE Rsquared MAE   
## 1 40.19584 0.002466163 32.06866  
## 2 40.48634 0.016438265 31.67889  
## 3 40.25297 0.017418707 31.46126  
## 4 41.03485 0.016028277 32.22306  
## 5 43.03356 0.012647497 33.83330  
## 6 47.53816 0.008249975 37.24345  
## 7 59.70175 0.003126153 47.00137  
## 8 67.61538 0.001586872 53.25255  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final value used for the model was ncomp = 1.

pcr\_resamp <- Chol\_pcr$results  
pcr\_resamp$Model <- "PCR"  
  
box\_pcr\_resamp <- Chol\_pcr\_box$results  
box\_pcr\_resamp$Model <- "BPCR"  
  
num\_pcr\_resamp <- Chol\_pcr\_num$results  
num\_pcr\_resamp$Model <- "PCR"

### PLS

set.seed(123)  
Chol\_pls <- train(x = Chol\_tr\_X\_imp\_fin, y = Chol\_train\_y, method = "pls", tuneGrid = expand.grid(ncomp = 1:32), trControl = Chol\_control)  
Chol\_pls

## Partial Least Squares   
##   
## 3696 samples  
## 32 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 51, 50, 48, 50, 46, 46, ...   
## Resampling results across tuning parameters:  
##   
## ncomp RMSE Rsquared MAE   
## 1 3.619222e+01 0.0095702569 2.828385e+01  
## 2 3.658160e+01 0.0141540807 2.861634e+01  
## 3 3.789116e+01 0.0076494661 2.962868e+01  
## 4 3.930606e+01 0.0065656458 3.066259e+01  
## 5 4.142046e+01 0.0028587406 3.234501e+01  
## 6 4.326723e+01 0.0024752947 3.385570e+01  
## 7 4.509748e+01 0.0013633354 3.541446e+01  
## 8 4.674500e+01 0.0014246065 3.678726e+01  
## 9 4.856265e+01 0.0014258901 3.818240e+01  
## 10 4.974819e+01 0.0016474825 3.904299e+01  
## 11 5.048636e+01 0.0018405363 3.955767e+01  
## 12 5.091217e+01 0.0018258953 3.986031e+01  
## 13 5.127850e+01 0.0017590056 4.009306e+01  
## 14 5.162712e+01 0.0016162040 4.029139e+01  
## 15 5.196320e+01 0.0013723077 4.049353e+01  
## 16 5.225482e+01 0.0012547820 4.067150e+01  
## 17 5.256880e+01 0.0011885845 4.084985e+01  
## 18 5.300055e+01 0.0010501900 4.114137e+01  
## 19 5.380775e+01 0.0009692891 4.158623e+01  
## 20 5.483917e+01 0.0009223097 4.209385e+01  
## 21 5.522431e+01 0.0009068171 4.225622e+01  
## 22 5.558012e+01 0.0009193425 4.240227e+01  
## 23 5.591282e+01 0.0009267561 4.253250e+01  
## 24 5.608452e+01 0.0009462771 4.258795e+01  
## 25 5.620958e+01 0.0009714210 4.264772e+01  
## 26 2.156421e+15 0.0012692598 1.732930e+15  
## 27 7.511462e+15 0.0010289876 6.034184e+15  
## 28 2.831204e+16 0.0025298567 2.263006e+16  
## 29 4.282583e+16 0.0043797169 3.386933e+16  
## 30 5.653343e+16 0.0041461779 4.482876e+16  
## 31 6.816377e+16 0.0036365074 5.386664e+16  
## 32 7.930837e+16 0.0040278438 6.245948e+16  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final value used for the model was ncomp = 1.

set.seed(123)  
Chol\_pls\_box <- train(x = Chol\_tr\_boxcox, y = Chol\_train\_y, method = "pls", tuneGrid = expand.grid(ncomp = 1:32), trControl = Chol\_control)  
Chol\_pls\_box

## Partial Least Squares   
##   
## 3696 samples  
## 32 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 51, 50, 48, 50, 46, 46, ...   
## Resampling results across tuning parameters:  
##   
## ncomp RMSE Rsquared MAE   
## 1 3.619222e+01 0.0095702569 2.828385e+01  
## 2 3.658160e+01 0.0141540807 2.861634e+01  
## 3 3.789116e+01 0.0076494661 2.962868e+01  
## 4 3.930606e+01 0.0065656458 3.066259e+01  
## 5 4.142046e+01 0.0028587406 3.234501e+01  
## 6 4.326723e+01 0.0024752947 3.385570e+01  
## 7 4.509748e+01 0.0013633354 3.541446e+01  
## 8 4.674500e+01 0.0014246065 3.678726e+01  
## 9 4.856265e+01 0.0014258901 3.818240e+01  
## 10 4.974819e+01 0.0016474825 3.904299e+01  
## 11 5.048636e+01 0.0018405363 3.955767e+01  
## 12 5.091217e+01 0.0018258953 3.986031e+01  
## 13 5.127850e+01 0.0017590056 4.009306e+01  
## 14 5.162712e+01 0.0016162040 4.029139e+01  
## 15 5.196320e+01 0.0013723077 4.049353e+01  
## 16 5.225482e+01 0.0012547820 4.067150e+01  
## 17 5.256880e+01 0.0011885845 4.084985e+01  
## 18 5.300055e+01 0.0010501900 4.114137e+01  
## 19 5.380775e+01 0.0009692891 4.158623e+01  
## 20 5.483917e+01 0.0009223097 4.209385e+01  
## 21 5.522431e+01 0.0009068171 4.225622e+01  
## 22 5.558012e+01 0.0009193425 4.240227e+01  
## 23 5.591282e+01 0.0009267561 4.253250e+01  
## 24 5.608452e+01 0.0009462771 4.258795e+01  
## 25 5.620958e+01 0.0009714210 4.264772e+01  
## 26 2.156421e+15 0.0012692598 1.732930e+15  
## 27 7.511462e+15 0.0010289876 6.034184e+15  
## 28 2.831204e+16 0.0025298567 2.263006e+16  
## 29 4.282583e+16 0.0043797169 3.386933e+16  
## 30 5.653343e+16 0.0041461779 4.482876e+16  
## 31 6.816377e+16 0.0036365074 5.386664e+16  
## 32 7.930837e+16 0.0040278438 6.245948e+16  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final value used for the model was ncomp = 1.

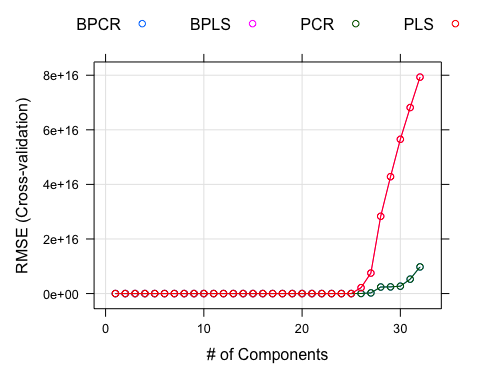
set.seed(123)  
Chol\_pls\_num <- train(x = Chol\_num\_tr\_X, y = Chol\_train\_y, method = "pls", tuneGrid = expand.grid(ncomp = 1:8), trControl = Chol\_control\_num)  
Chol\_pls\_num

## Partial Least Squares   
##   
## 3696 samples  
## 12 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 11, 9, 11, 10, 10, 11, ...   
## Resampling results across tuning parameters:  
##   
## ncomp RMSE Rsquared MAE   
## 1 40.88057 0.010748084 32.26603  
## 2 40.43823 0.018846454 31.51245  
## 3 44.01601 0.009087219 34.61430  
## 4 49.17924 0.006309826 38.63731  
## 5 55.84753 0.003800729 43.81716  
## 6 59.73218 0.001891028 46.82793  
## 7 68.64725 0.002096613 54.11983  
## 8 72.44313 0.002107833 57.07935  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final value used for the model was ncomp = 2.

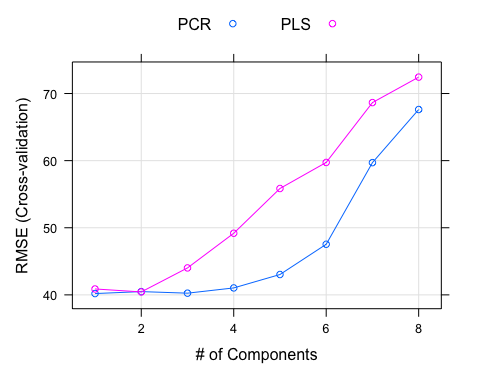
pls\_resamp <- Chol\_pls$results  
pls\_resamp$Model <- "PLS"  
  
pls\_box\_resamp <- Chol\_pls\_box$results  
pls\_box\_resamp$Model <- "BPLS"  
  
pls\_num\_resamp <- Chol\_pls\_num$results  
pls\_num\_resamp$Model <- "PLS"

### Compare

plot\_data <- rbind(pcr\_resamp, box\_pcr\_resamp, pls\_resamp, pls\_box\_resamp)  
xyplot(RMSE ~ ncomp, data = plot\_data, xlab = "# of Components", ylab = "RMSE (Cross-validation)", auto.key = list(columns = 4), groups = Model, type = c("o", "g"))



plot2\_data <- rbind(num\_pcr\_resamp, pls\_num\_resamp)  
xyplot(RMSE ~ ncomp, data = plot2\_data, xlab = "# of Components", ylab = "RMSE (Cross-validation)", auto.key = list(columns = 2), groups = Model, type = c("o", "g"))



## Penalized Models

### Ridge

set.seed(123)  
Chol\_ridge <- train(x = Chol\_tr\_X\_imp\_fin, y= Chol\_train\_y, method = "ridge", tuneGrid = expand.grid(lambda = seq(0, .1, length = 15)), trControl = Chol\_control)

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

Chol\_ridge

## Ridge Regression   
##   
## 3696 samples  
## 32 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 51, 50, 48, 50, 46, 46, ...   
## Resampling results across tuning parameters:  
##   
## lambda RMSE Rsquared MAE   
## 0.000000000 57.30783 0.001130497 43.28896  
## 0.007142857 51.91774 0.001676059 40.13528  
## 0.014285714 49.04024 0.001920656 38.16223  
## 0.021428571 47.56154 0.002198836 37.01972  
## 0.028571429 46.20969 0.002433663 36.03384  
## 0.035714286 45.58111 0.002704163 35.47685  
## 0.042857143 44.71095 0.002924084 34.84203  
## 0.050000000 44.11965 0.003150297 34.38012  
## 0.057142857 43.81931 0.003385763 34.08973  
## 0.064285714 43.87584 0.003621658 33.97768  
## 0.071428571 43.68077 0.003830483 33.76416  
## 0.078571429 42.69008 0.004004932 33.20837  
## 0.085714286 41.88986 0.004155063 32.71198  
## 0.092857143 41.82447 0.004371812 32.61636  
## 0.100000000 41.76289 0.004573558 32.52432  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final value used for the model was lambda = 0.1.

set.seed(123)  
Chol\_ridge\_box <- train(x = Chol\_tr\_boxcox, y= Chol\_train\_y, method = "ridge", tuneGrid = expand.grid(lambda = seq(0, .5, length = 15)), trControl = Chol\_control)

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

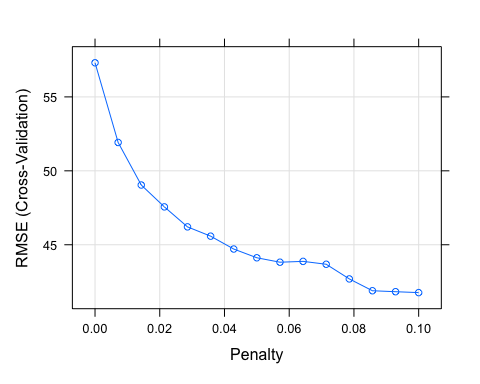
Chol\_ridge\_box

## Ridge Regression   
##   
## 3696 samples  
## 32 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 51, 50, 48, 50, 46, 46, ...   
## Resampling results across tuning parameters:  
##   
## lambda RMSE Rsquared MAE   
## 0.00000000 57.30783 0.001130497 43.28896  
## 0.03571429 45.58111 0.002704163 35.47685  
## 0.07142857 43.68077 0.003830483 33.76416  
## 0.10714286 42.33145 0.004775675 32.72236  
## 0.14285714 41.12908 0.005627556 31.92746  
## 0.17857143 40.92063 0.006387930 31.66375  
## 0.21428571 40.29552 0.007085350 31.26359  
## 0.25000000 39.78024 0.007719634 30.93971  
## 0.28571429 40.43944 0.008207511 31.17267  
## 0.32142857 39.89776 0.008727185 30.88678  
## 0.35714286 38.98136 0.009263185 30.40510  
## 0.39285714 39.12456 0.009632846 30.45972  
## 0.42857143 38.69120 0.010053743 30.21205  
## 0.46428571 38.60837 0.010387616 30.15638  
## 0.50000000 38.81339 0.010630063 30.26456  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final value used for the model was lambda = 0.4642857.

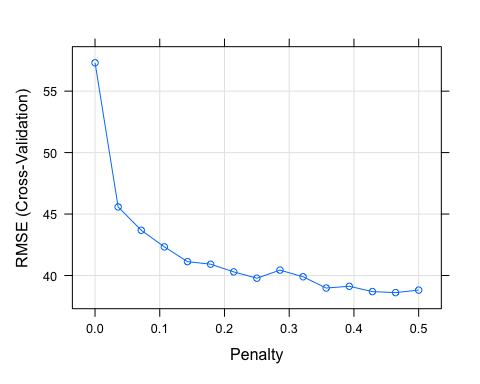
set.seed(123)  
Chol\_ridge\_num <- train(x = Chol\_num\_tr\_X, y= Chol\_train\_y, method = "ridge", tuneGrid = expand.grid(lambda = seq(0, .5, length = 15)), trControl = Chol\_control\_num)  
Chol\_ridge\_num

## Ridge Regression   
##   
## 3696 samples  
## 12 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 11, 9, 11, 10, 10, 11, ...   
## Resampling results across tuning parameters:  
##   
## lambda RMSE Rsquared MAE   
## 0.00000000 126.78810 0.001902577 98.22802  
## 0.03571429 49.92423 0.004292662 39.13308  
## 0.07142857 46.46868 0.006130216 36.39265  
## 0.10714286 45.10956 0.007630889 35.31193  
## 0.14285714 44.46566 0.008831610 34.78599  
## 0.17857143 44.14662 0.009791100 34.51837  
## 0.21428571 44.00180 0.010565551 34.39059  
## 0.25000000 43.96164 0.011199350 34.34531  
## 0.28571429 43.98981 0.011725150 34.35543  
## 0.32142857 44.06549 0.012166498 34.40250  
## 0.35714286 44.17588 0.012540441 34.47808  
## 0.39285714 44.31261 0.012859519 34.57403  
## 0.42857143 44.46999 0.013133145 34.68783  
## 0.46428571 44.64393 0.013368558 34.81479  
## 0.50000000 44.83143 0.013571444 34.95258  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final value used for the model was lambda = 0.25.

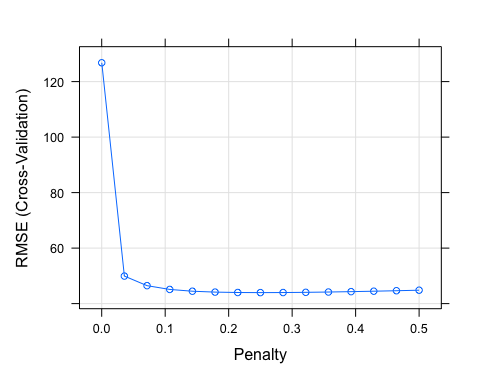
print(update(plot(Chol\_ridge), xlab = "Penalty"))



print(update(plot(Chol\_ridge\_box), xlab = "Penalty"))



print(update(plot(Chol\_ridge\_num), xlab = "Penalty"))



### Elastic Net

enet\_grid <- expand.grid(lambda = c(0, 0.01, 0.1), fraction = seq(0.05, 1, length = 20))  
  
Chol\_enet <- train(x = Chol\_tr\_X\_imp\_fin, y = Chol\_train\_y, method = "enet", tuneGrid = enet\_grid, trControl = Chol\_control)

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

Chol\_enet

## Elasticnet   
##   
## 3696 samples  
## 32 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 51, 50, 48, 50, 46, 46, ...   
## Resampling results across tuning parameters:  
##   
## lambda fraction RMSE Rsquared MAE   
## 0.00 0.05 35.83367 0.006222217 28.02459  
## 0.00 0.10 36.53421 0.005604617 28.56434  
## 0.00 0.15 37.13853 0.006832524 29.01152  
## 0.00 0.20 38.02442 0.007149682 29.68817  
## 0.00 0.25 39.23723 0.006728914 30.62965  
## 0.00 0.30 40.47220 0.005851752 31.59385  
## 0.00 0.35 41.91140 0.005101129 32.65975  
## 0.00 0.40 43.61118 0.004321812 33.85396  
## 0.00 0.45 45.54812 0.003643485 35.15439  
## 0.00 0.50 47.57574 0.003081583 36.44342  
## 0.00 0.55 49.67813 0.002663956 37.72115  
## 0.00 0.60 51.06576 0.002354932 38.67930  
## 0.00 0.65 52.16667 0.002088114 39.51067  
## 0.00 0.70 53.29875 0.001865415 40.35446  
## 0.00 0.75 54.49066 0.001671436 41.23217  
## 0.00 0.80 55.58236 0.001519706 42.03210  
## 0.00 0.85 56.23003 0.001380116 42.54420  
## 0.00 0.90 56.70633 0.001269371 42.87913  
## 0.00 0.95 57.13975 0.001189456 43.17937  
## 0.00 1.00 57.30783 0.001130497 43.28896  
## 0.01 0.05 35.62209 0.007230848 27.85396  
## 0.01 0.10 36.02062 0.006344525 28.16772  
## 0.01 0.15 36.43337 0.006044262 28.48772  
## 0.01 0.20 36.75450 0.006965389 28.72552  
## 0.01 0.25 37.15078 0.007315211 29.01283  
## 0.01 0.30 37.69742 0.007156375 29.41924  
## 0.01 0.35 38.37672 0.006667894 29.94267  
## 0.01 0.40 39.12008 0.005985806 30.52327  
## 0.01 0.45 39.82890 0.005237381 31.08088  
## 0.01 0.50 40.60420 0.004514830 31.70312  
## 0.01 0.55 41.43253 0.003883700 32.37354  
## 0.01 0.60 42.27977 0.003359763 33.05295  
## 0.01 0.65 43.13533 0.002941737 33.73224  
## 0.01 0.70 43.95068 0.002624328 34.36824  
## 0.01 0.75 44.80315 0.002385515 35.02851  
## 0.01 0.80 45.68679 0.002182495 35.71201  
## 0.01 0.85 46.61269 0.001976713 36.42605  
## 0.01 0.90 47.57257 0.001794043 37.15991  
## 0.01 0.95 48.50767 0.001644697 37.86719  
## 0.01 1.00 50.58689 0.001775240 39.23991  
## 0.10 0.05 35.54968 0.006041850 27.81362  
## 0.10 0.10 35.64349 0.007724957 27.86784  
## 0.10 0.15 35.84535 0.007276311 28.02691  
## 0.10 0.20 36.11859 0.006539789 28.23908  
## 0.10 0.25 36.34142 0.006494037 28.41280  
## 0.10 0.30 36.52007 0.006981548 28.55061  
## 0.10 0.35 36.68910 0.007573407 28.67644  
## 0.10 0.40 36.87370 0.008115886 28.81323  
## 0.10 0.45 37.09333 0.008165403 28.97116  
## 0.10 0.50 37.35938 0.008060185 29.16835  
## 0.10 0.55 37.66045 0.007910137 29.39602  
## 0.10 0.60 37.99432 0.007503438 29.65587  
## 0.10 0.65 38.35281 0.007026992 29.93892  
## 0.10 0.70 38.72448 0.006490591 30.23742  
## 0.10 0.75 39.07358 0.005980990 30.51533  
## 0.10 0.80 39.43525 0.005502920 30.80324  
## 0.10 0.85 39.81441 0.005044182 31.10668  
## 0.10 0.90 40.19457 0.004648780 31.40969  
## 0.10 0.95 40.56177 0.004310663 31.70159  
## 0.10 1.00 41.76289 0.004573558 32.52432  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final values used for the model were fraction = 0.05 and lambda = 0.1.

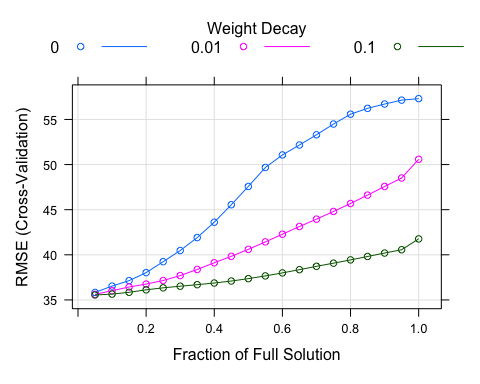
Chol\_enet\_box <- train(x = Chol\_tr\_X\_imp\_fin, y = Chol\_train\_y, method = "enet", tuneGrid = enet\_grid, trControl = Chol\_control)

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

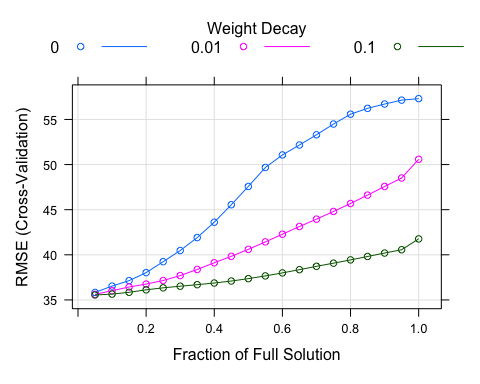
Chol\_enet\_box

## Elasticnet   
##   
## 3696 samples  
## 32 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 51, 50, 48, 50, 46, 46, ...   
## Resampling results across tuning parameters:  
##   
## lambda fraction RMSE Rsquared MAE   
## 0.00 0.05 35.83367 0.006222217 28.02459  
## 0.00 0.10 36.53421 0.005604617 28.56434  
## 0.00 0.15 37.13853 0.006832524 29.01152  
## 0.00 0.20 38.02442 0.007149682 29.68817  
## 0.00 0.25 39.23723 0.006728914 30.62965  
## 0.00 0.30 40.47220 0.005851752 31.59385  
## 0.00 0.35 41.91140 0.005101129 32.65975  
## 0.00 0.40 43.61118 0.004321812 33.85396  
## 0.00 0.45 45.54812 0.003643485 35.15439  
## 0.00 0.50 47.57574 0.003081583 36.44342  
## 0.00 0.55 49.67813 0.002663956 37.72115  
## 0.00 0.60 51.06576 0.002354932 38.67930  
## 0.00 0.65 52.16667 0.002088114 39.51067  
## 0.00 0.70 53.29875 0.001865415 40.35446  
## 0.00 0.75 54.49066 0.001671436 41.23217  
## 0.00 0.80 55.58236 0.001519706 42.03210  
## 0.00 0.85 56.23003 0.001380116 42.54420  
## 0.00 0.90 56.70633 0.001269371 42.87913  
## 0.00 0.95 57.13975 0.001189456 43.17937  
## 0.00 1.00 57.30783 0.001130497 43.28896  
## 0.01 0.05 35.62209 0.007230848 27.85396  
## 0.01 0.10 36.02062 0.006344525 28.16772  
## 0.01 0.15 36.43337 0.006044262 28.48772  
## 0.01 0.20 36.75450 0.006965389 28.72552  
## 0.01 0.25 37.15078 0.007315211 29.01283  
## 0.01 0.30 37.69742 0.007156375 29.41924  
## 0.01 0.35 38.37672 0.006667894 29.94267  
## 0.01 0.40 39.12008 0.005985806 30.52327  
## 0.01 0.45 39.82890 0.005237381 31.08088  
## 0.01 0.50 40.60420 0.004514830 31.70312  
## 0.01 0.55 41.43253 0.003883700 32.37354  
## 0.01 0.60 42.27977 0.003359763 33.05295  
## 0.01 0.65 43.13533 0.002941737 33.73224  
## 0.01 0.70 43.95068 0.002624328 34.36824  
## 0.01 0.75 44.80315 0.002385515 35.02851  
## 0.01 0.80 45.68679 0.002182495 35.71201  
## 0.01 0.85 46.61269 0.001976713 36.42605  
## 0.01 0.90 47.57257 0.001794043 37.15991  
## 0.01 0.95 48.50767 0.001644697 37.86719  
## 0.01 1.00 50.58689 0.001775240 39.23991  
## 0.10 0.05 35.54968 0.006041850 27.81362  
## 0.10 0.10 35.64349 0.007724957 27.86784  
## 0.10 0.15 35.84535 0.007276311 28.02691  
## 0.10 0.20 36.11859 0.006539789 28.23908  
## 0.10 0.25 36.34142 0.006494037 28.41280  
## 0.10 0.30 36.52007 0.006981548 28.55061  
## 0.10 0.35 36.68910 0.007573407 28.67644  
## 0.10 0.40 36.87370 0.008115886 28.81323  
## 0.10 0.45 37.09333 0.008165403 28.97116  
## 0.10 0.50 37.35938 0.008060185 29.16835  
## 0.10 0.55 37.66045 0.007910137 29.39602  
## 0.10 0.60 37.99432 0.007503438 29.65587  
## 0.10 0.65 38.35281 0.007026992 29.93892  
## 0.10 0.70 38.72448 0.006490591 30.23742  
## 0.10 0.75 39.07358 0.005980990 30.51533  
## 0.10 0.80 39.43525 0.005502920 30.80324  
## 0.10 0.85 39.81441 0.005044182 31.10668  
## 0.10 0.90 40.19457 0.004648780 31.40969  
## 0.10 0.95 40.56177 0.004310663 31.70159  
## 0.10 1.00 41.76289 0.004573558 32.52432  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final values used for the model were fraction = 0.05 and lambda = 0.1.

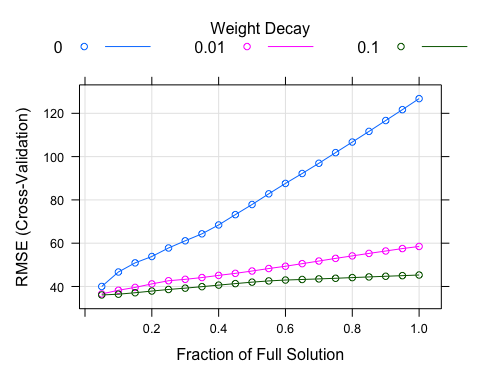
Chol\_enet\_num <- train(x = Chol\_num\_tr\_X, y = Chol\_train\_y, method = "enet", tuneGrid = enet\_grid, trControl = Chol\_control\_num)  
  
plot(Chol\_enet)



plot(Chol\_enet\_box)



plot(Chol\_enet\_num)



## Gather Results from Linear Models

Note: No diagnostic plot showed much of a difference between boxcox vs. regular (not suprising given normality in histograms), so we’ll just use the regular constructed models

### Create Data set of predictions and observed

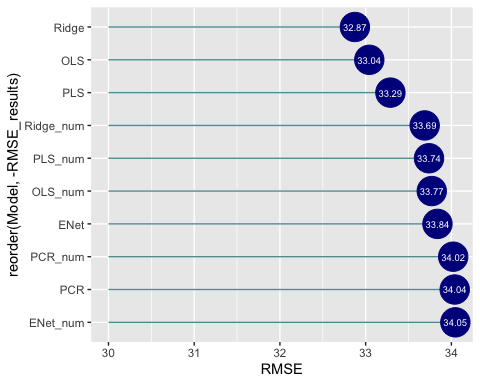
Res\_OLS <- predict(Chol\_ols, Chol\_test\_X\_imp)  
Res\_OLS\_num <- predict(Chol\_ols\_tune\_num, Chol\_num\_test\_X)

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit  
## may be misleading

Res\_PLS <- predict(Chol\_pls, Chol\_test\_X\_imp)  
Res\_PLS\_num <- predict(Chol\_pls\_num, Chol\_num\_test\_X)  
Res\_PCR <- predict(Chol\_pcr, Chol\_test\_X\_imp)  
Res\_PCR\_num <- predict(Chol\_pcr\_num, Chol\_num\_test\_X)  
Res\_Ridge <- predict(Chol\_ridge, Chol\_test\_X\_imp)  
Res\_Ridge\_num <- predict(Chol\_ridge\_num, Chol\_num\_test\_X)  
Res\_Enet <- predict(Chol\_enet, Chol\_test\_X\_imp)  
Res\_Enet\_num <- predict(Chol\_enet\_num, Chol\_num\_test\_X)  
  
Linear\_res <- cbind.data.frame(Observed = Chol\_test\_y, OLS = Res\_OLS, OLS\_num = Res\_OLS\_num, PLS = Res\_PLS, PLS\_num = Res\_PLS\_num, PCR = Res\_PCR, PCR\_num = Res\_PCR\_num ,Ridge = Res\_Ridge, Ridge\_num = Res\_Ridge\_num, ENet = Res\_Enet, ENet\_num = Res\_Enet\_num)

### Get RMSE and Plot

find\_rmse <- function(x){  
 caret::RMSE(x, Linear\_res[,"Observed"])  
}  
  
RMSE\_results <- apply(X = Linear\_res[,2:11], FUN = find\_rmse, MARGIN = 2)  
RMSE\_results <- data.frame(RMSE\_results)  
RMSE\_results$Model <- rownames(RMSE\_results)  
  
  
ggplot(RMSE\_results, aes(x=reorder(Model, -RMSE\_results), y=RMSE\_results)) + geom\_segment(aes(x=reorder(Model, -RMSE\_results), xend = reorder(Model, -RMSE\_results), y=30, yend=RMSE\_results), color = "cadetblue") + geom\_point(color = "darkblue", size = 10) + coord\_flip() + ylab("RMSE") + geom\_text(aes(label = round(RMSE\_results, 2)), color = "white", size = 2.5)



# Non-Linear Models - Brianne

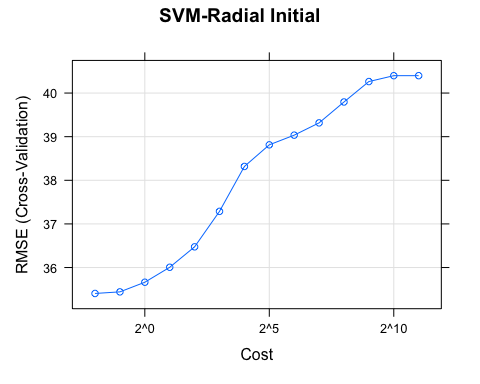
## Support Vector Machine (SVM)

### Create Initial Radial Model

# initial SVM model with radial basis and processed Chol\_tr\_X\_imp\_fin and Chol\_train\_y  
set.seed(123)  
svmR0 <- train(x=Chol\_tr\_X\_imp\_fin, y = Chol\_train\_y,  
 method = "svmRadial",  
 preProcess = c("center", "scale"),  
 tuneLength = 14,  
 trControl = Chol\_control)  
svmR0

## Support Vector Machines with Radial Basis Function Kernel   
##   
## 3696 samples  
## 32 predictor  
##   
## Pre-processing: centered (32), scaled (32)   
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 51, 50, 48, 50, 46, 46, ...   
## Resampling results across tuning parameters:  
##   
## C RMSE Rsquared MAE   
## 0.25 35.40559 0.019455264 27.76763  
## 0.50 35.44220 0.017157068 27.77777  
## 1.00 35.66280 0.014213297 27.95112  
## 2.00 36.00581 0.011582915 28.21196  
## 4.00 36.47452 0.010209766 28.54587  
## 8.00 37.28794 0.010549655 29.15644  
## 16.00 38.31727 0.009015845 29.92143  
## 32.00 38.81306 0.006477031 30.30348  
## 64.00 39.03490 0.004746663 30.45701  
## 128.00 39.31636 0.004238773 30.67175  
## 256.00 39.79527 0.004154623 31.05699  
## 512.00 40.26356 0.003937049 31.42013  
## 1024.00 40.39860 0.003866976 31.52589  
## 2048.00 40.39860 0.003866976 31.52589  
##   
## Tuning parameter 'sigma' was held constant at a value of 0.0207376  
## RMSE was used to select the optimal model using the smallest value.  
## The final values used for the model were sigma = 0.0207376 and C = 0.25.

# final model uses: sigma = 0.0207376 and C = 0.25  
 # RMSE: 35.40559, Rsquared: 0.019455264  
plot(svmR0, scales = list(x = list(log = 2)), main="SVM-Radial Initial")



### final radial model

# svm radial model v2   
 # issue causing variables in X: MIALANG\_Spanish, DMDEDUC2\_<9, MIALANG\_NA (zero var)  
Chol\_tr\_X\_impfin\_drop <- c("MIALANG\_Spanish", "DMDEDUC2\_<9", "MIALANG\_NA")  
Chol\_tr\_X\_impfin\_sv <- subset(Chol\_tr\_X\_imp\_fin,   
 select = !(names(Chol\_tr\_X\_imp\_fin) %in% Chol\_tr\_X\_impfin\_drop))  
  
#making test X have same columns available  
Chol\_te\_X\_sv <- subset(Chol\_test\_X\_imp, select = c(names(Chol\_tr\_X\_impfin\_sv)))  
dim(Chol\_tr\_X\_impfin\_sv)

## [1] 3696 29

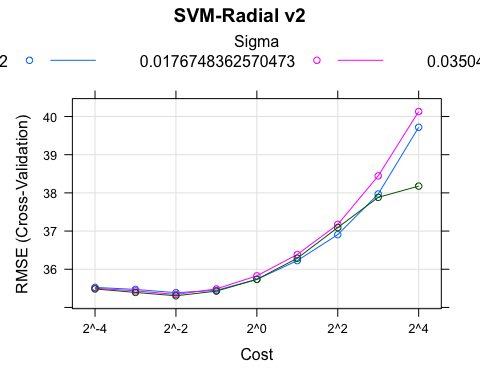
dim(Chol\_te\_X\_sv)

## [1] 921 29

# sigma grid instead of using tuneLength = 14  
sigmaEst <- kernlab::sigest(as.matrix(Chol\_tr\_X\_impfin\_sv[,1:29]))  
Csearch <- 2^seq(-4,+4)  
# sigma estimates using kernlab's sigest function  
svmgrid <- expand.grid(sigma = sigmaEst, C = Csearch)  
  
#model  
set.seed(123)  
svmR1 <- train(x=Chol\_tr\_X\_impfin\_sv, y = Chol\_train\_y,  
 method = "svmRadial",  
 preProcess = c("center", "scale"),  
 tuneGrid = svmgrid,  
 trControl = Chol\_control)  
svmR1

## Support Vector Machines with Radial Basis Function Kernel   
##   
## 3696 samples  
## 29 predictor  
##   
## Pre-processing: centered (29), scaled (29)   
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 51, 50, 48, 50, 46, 46, ...   
## Resampling results across tuning parameters:  
##   
## sigma C RMSE Rsquared MAE   
## 0.01122667 0.0625 35.52469 0.009696336 27.85830  
## 0.01122667 0.1250 35.47202 0.012142507 27.84388  
## 0.01122667 0.2500 35.38618 0.015125963 27.75710  
## 0.01122667 0.5000 35.44746 0.013676872 27.77376  
## 0.01122667 1.0000 35.73438 0.012288360 27.98960  
## 0.01122667 2.0000 36.22595 0.009125541 28.39371  
## 0.01122667 4.0000 36.90618 0.005909491 28.89325  
## 0.01122667 8.0000 37.96964 0.003556339 29.70350  
## 0.01122667 16.0000 39.71811 0.003114791 31.08638  
## 0.01767484 0.0625 35.50276 0.012385051 27.83878  
## 0.01767484 0.1250 35.43389 0.014960721 27.80456  
## 0.01767484 0.2500 35.33999 0.017190482 27.70085  
## 0.01767484 0.5000 35.48527 0.013325765 27.80045  
## 0.01767484 1.0000 35.82666 0.011217476 28.06930  
## 0.01767484 2.0000 36.38687 0.007423194 28.50946  
## 0.01767484 4.0000 37.17468 0.004931916 29.09620  
## 0.01767484 8.0000 38.44510 0.004210192 30.11222  
## 0.01767484 16.0000 40.13064 0.004339419 31.39401  
## 0.03504524 0.0625 35.48387 0.017296723 27.81143  
## 0.03504524 0.1250 35.39320 0.019603877 27.75034  
## 0.03504524 0.2500 35.30443 0.020245225 27.66402  
## 0.03504524 0.5000 35.42703 0.014725418 27.75156  
## 0.03504524 1.0000 35.73449 0.010524930 27.97035  
## 0.03504524 2.0000 36.28845 0.007206842 28.37461  
## 0.03504524 4.0000 37.09202 0.006669812 29.01808  
## 0.03504524 8.0000 37.88267 0.006635482 29.62419  
## 0.03504524 16.0000 38.17701 0.006262306 29.84817  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final values used for the model were sigma = 0.03504524 and C = 0.25.

# final model uses: sigma = 0.03504524 and C = 0.25.  
 # RMSE: 35.30443, Rsquared: 0.020245225   
plot(svmR1, scales = list(x = list(log = 2)), main="SVM-Radial v2")



svmR1$finalModel

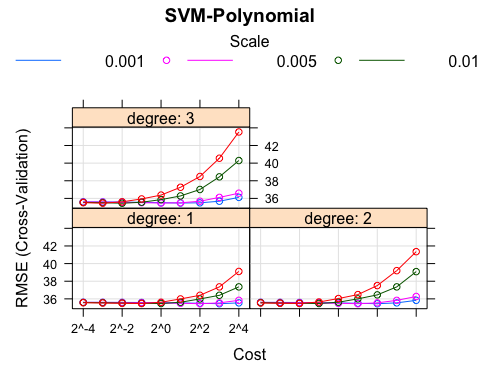
## Support Vector Machine object of class "ksvm"   
##   
## SV type: eps-svr (regression)   
## parameter : epsilon = 0.1 cost C = 0.25   
##   
## Gaussian Radial Basis kernel function.   
## Hyperparameter : sigma = 0.035045237945048   
##   
## Number of Support Vectors : 3377   
##   
## Objective Function Value : -557.6121   
## Training error : 0.82428

### Create Polynomial Model

# going to use the x training set from final svm-radial due to assuming there will be the same problem causing factors of nearZeroVar.  
set.seed(123)  
svmP <- train(x=Chol\_tr\_X\_impfin\_sv, y = Chol\_train\_y,  
 method = "svmPoly",  
 preProcess = c("center", "scale"),  
 tuneGrid = expand.grid(degree = 1:3,   
 scale = c(0.01, 0.005, 0.001, 0.0005),   
 C = Csearch),  
 trControl = Chol\_control)  
svmP

## Support Vector Machines with Polynomial Kernel   
##   
## 3696 samples  
## 29 predictor  
##   
## Pre-processing: centered (29), scaled (29)   
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 51, 50, 48, 50, 46, 46, ...   
## Resampling results across tuning parameters:  
##   
## degree scale C RMSE Rsquared MAE   
## 1 5e-04 0.0625 35.60550 0.003307868 27.88099  
## 1 5e-04 0.1250 35.60250 0.003307868 27.88108  
## 1 5e-04 0.2500 35.59695 0.003307868 27.88148  
## 1 5e-04 0.5000 35.58947 0.003467793 27.88969  
## 1 5e-04 1.0000 35.57179 0.004671825 27.89896  
## 1 5e-04 2.0000 35.53627 0.006769999 27.89302  
## 1 5e-04 4.0000 35.50309 0.008878237 27.88751  
## 1 5e-04 8.0000 35.44968 0.011756893 27.80268  
## 1 5e-04 16.0000 35.54443 0.014251476 27.87152  
## 1 1e-03 0.0625 35.60250 0.003307868 27.88108  
## 1 1e-03 0.1250 35.59695 0.003307868 27.88148  
## 1 1e-03 0.2500 35.58947 0.003467796 27.88969  
## 1 1e-03 0.5000 35.57179 0.004671830 27.89896  
## 1 1e-03 1.0000 35.53626 0.006770677 27.89301  
## 1 1e-03 2.0000 35.50304 0.008881237 27.88747  
## 1 1e-03 4.0000 35.44957 0.011758089 27.80254  
## 1 1e-03 8.0000 35.54429 0.014254316 27.87135  
## 1 1e-03 16.0000 35.83380 0.014070320 28.07687  
## 1 5e-03 0.0625 35.58555 0.003698012 27.89314  
## 1 5e-03 0.1250 35.56303 0.004991785 27.89895  
## 1 5e-03 0.2500 35.52389 0.007512235 27.89403  
## 1 5e-03 0.5000 35.49418 0.009250879 27.87423  
## 1 5e-03 1.0000 35.46922 0.012743776 27.82567  
## 1 5e-03 2.0000 35.61461 0.014743888 27.91266  
## 1 5e-03 4.0000 35.99156 0.012761259 28.18698  
## 1 5e-03 8.0000 36.40428 0.010880745 28.50431  
## 1 5e-03 16.0000 37.35864 0.007425673 29.23444  
## 1 1e-02 0.0625 35.56303 0.004991785 27.89895  
## 1 1e-02 0.1250 35.52389 0.007509264 27.89406  
## 1 1e-02 0.2500 35.49419 0.009252110 27.87432  
## 1 1e-02 0.5000 35.46929 0.012742860 27.82573  
## 1 1e-02 1.0000 35.61459 0.014742977 27.91265  
## 1 1e-02 2.0000 35.99140 0.012762456 28.18686  
## 1 1e-02 4.0000 36.40428 0.010881061 28.50433  
## 1 1e-02 8.0000 37.35862 0.007424646 29.23444  
## 1 1e-02 16.0000 39.10902 0.003741650 30.57409  
## 2 5e-04 0.0625 35.60254 0.003245942 27.88115  
## 2 5e-04 0.1250 35.59704 0.003245942 27.88162  
## 2 5e-04 0.2500 35.58973 0.003415482 27.89020  
## 2 5e-04 0.5000 35.57213 0.004636091 27.89975  
## 2 5e-04 1.0000 35.53660 0.006759380 27.89384  
## 2 5e-04 2.0000 35.50346 0.008837641 27.88771  
## 2 5e-04 4.0000 35.44947 0.011748428 27.80205  
## 2 5e-04 8.0000 35.54528 0.014231517 27.87110  
## 2 5e-04 16.0000 35.83080 0.014074038 28.07184  
## 2 1e-03 0.0625 35.59713 0.003185589 27.88177  
## 2 1e-03 0.1250 35.58999 0.003364486 27.89071  
## 2 1e-03 0.2500 35.57248 0.004595661 27.90050  
## 2 1e-03 0.5000 35.53690 0.006745486 27.89463  
## 2 1e-03 1.0000 35.50377 0.008798054 27.88784  
## 2 1e-03 2.0000 35.44929 0.011746076 27.80151  
## 2 1e-03 4.0000 35.54667 0.014194715 27.87117  
## 2 1e-03 8.0000 35.82844 0.014072990 28.06726  
## 2 1e-03 16.0000 36.27133 0.011281994 28.40449  
## 2 5e-03 0.0625 35.56898 0.004675354 27.91039  
## 2 5e-03 0.1250 35.52724 0.007353543 27.90071  
## 2 5e-03 0.2500 35.49628 0.008843266 27.86915  
## 2 5e-03 0.5000 35.47375 0.012598208 27.82442  
## 2 5e-03 1.0000 35.62169 0.014496701 27.91070  
## 2 5e-03 2.0000 36.00041 0.012688034 28.17946  
## 2 5e-03 4.0000 36.45283 0.010013853 28.52389  
## 2 5e-03 8.0000 37.35891 0.006296414 29.18117  
## 2 5e-03 16.0000 39.08906 0.002981805 30.51211  
## 2 1e-02 0.0625 35.52886 0.007195341 27.90346  
## 2 1e-02 0.1250 35.49604 0.008629937 27.86368  
## 2 1e-02 0.2500 35.47985 0.012410247 27.82312  
## 2 1e-02 0.5000 35.64375 0.013745143 27.91973  
## 2 1e-02 1.0000 36.03523 0.012112360 28.19583  
## 2 1e-02 2.0000 36.48285 0.009102593 28.52459  
## 2 1e-02 4.0000 37.50797 0.005221784 29.27932  
## 2 1e-02 8.0000 39.20045 0.002436746 30.56414  
## 2 1e-02 16.0000 41.35322 0.001526747 32.31228  
## 3 5e-04 0.0625 35.59978 0.003184050 27.88144  
## 3 5e-04 0.1250 35.59307 0.003200857 27.88509  
## 3 5e-04 0.2500 35.58112 0.004045184 27.89614  
## 3 5e-04 0.5000 35.55677 0.005468165 27.90211  
## 3 5e-04 1.0000 35.51454 0.008065033 27.89271  
## 3 5e-04 2.0000 35.47619 0.009817538 27.84036  
## 3 5e-04 4.0000 35.49937 0.013156932 27.84709  
## 3 5e-04 8.0000 35.67856 0.014705663 27.96125  
## 3 5e-04 16.0000 36.11653 0.012065024 28.28594  
## 3 1e-03 0.0625 35.59346 0.003085484 27.88584  
## 3 1e-03 0.1250 35.58167 0.003977699 27.89746  
## 3 1e-03 0.2500 35.55753 0.005441261 27.90395  
## 3 1e-03 0.5000 35.51493 0.008018994 27.89327  
## 3 1e-03 1.0000 35.47560 0.009805631 27.83870  
## 3 1e-03 2.0000 35.50051 0.013103169 27.84620  
## 3 1e-03 4.0000 35.67891 0.014672152 27.95868  
## 3 1e-03 8.0000 36.11278 0.012065777 28.27977  
## 3 1e-03 16.0000 36.58833 0.009826878 28.63440  
## 3 5e-03 0.0625 35.54673 0.006234945 27.90907  
## 3 5e-03 0.1250 35.51257 0.007931940 27.89299  
## 3 5e-03 0.2500 35.44947 0.011299979 27.79547  
## 3 5e-03 0.5000 35.56464 0.013292681 27.87132  
## 3 5e-03 1.0000 35.86151 0.012861573 28.06298  
## 3 5e-03 2.0000 36.27928 0.010185822 28.37759  
## 3 5e-03 4.0000 37.01044 0.006477946 28.91038  
## 3 5e-03 8.0000 38.44411 0.003204919 29.98999  
## 3 5e-03 16.0000 40.29246 0.001719569 31.44658  
## 3 1e-02 0.0625 35.51793 0.007230139 27.89140  
## 3 1e-02 0.1250 35.45366 0.011119576 27.78774  
## 3 1e-02 0.2500 35.61148 0.012236872 27.89471  
## 3 1e-02 0.5000 35.93955 0.011281108 28.10440  
## 3 1e-02 1.0000 36.37681 0.008474946 28.43641  
## 3 1e-02 2.0000 37.26057 0.004695041 29.08827  
## 3 1e-02 4.0000 38.48464 0.002886420 30.04071  
## 3 1e-02 8.0000 40.55448 0.002274583 31.69958  
## 3 1e-02 16.0000 43.53313 0.002657192 33.99113  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final values used for the model were degree = 2, scale = 0.001 and C = 2.

# final model uses: degree = 2, scale = 0.001, offset = 1   
 # sigma = 0.02231109 and C = 2.  
 # RMSE: 35.44929, Rsquared: 0.011746076  
plot(svmP, scales = list(x = list(log = 2),  
 between=list(x=.5, y=1)), main="SVM-Polynomial")



svmP$finalModel

## Support Vector Machine object of class "ksvm"   
##   
## SV type: eps-svr (regression)   
## parameter : epsilon = 0.1 cost C = 2   
##   
## Polynomial kernel function.   
## Hyperparameters : degree = 2 scale = 0.001 offset = 1   
##   
## Number of Support Vectors : 3366   
##   
## Objective Function Value : -4741.157   
## Training error : 0.932111

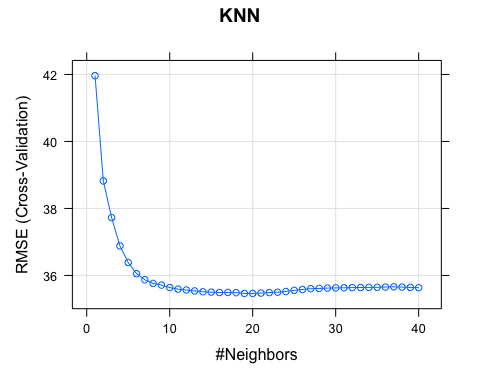
## K Nearest Neighbors (KNN)

### Create Initial KNN Model

# KNN Model needs to have NZV removed so again we are using the x=Chol\_tr\_X\_impfin\_sv to train and Chol\_te\_X\_sv to test  
set.seed(123)  
knnTune <- train(x=Chol\_tr\_X\_impfin\_sv, y = Chol\_train\_y,  
 method = "knn",  
 preProcess = c("center", "scale"),  
 tuneGrid = data.frame(k=1:40),  
 trControl = Chol\_control)  
knnTune

## k-Nearest Neighbors   
##   
## 3696 samples  
## 29 predictor  
##   
## Pre-processing: centered (29), scaled (29)   
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 51, 50, 48, 50, 46, 46, ...   
## Resampling results across tuning parameters:  
##   
## k RMSE Rsquared MAE   
## 1 41.96292 0.0030270508 32.73341  
## 2 38.81981 0.0049086814 30.44225  
## 3 37.72731 0.0041360203 29.62242  
## 4 36.88208 0.0063779767 28.90377  
## 5 36.38651 0.0092146818 28.47635  
## 6 36.05320 0.0123437417 28.22668  
## 7 35.87239 0.0141203786 28.12946  
## 8 35.76213 0.0147297257 28.04586  
## 9 35.71201 0.0144672222 28.01596  
## 10 35.63941 0.0149570471 27.94913  
## 11 35.59267 0.0148245075 27.90115  
## 12 35.56590 0.0144195646 27.87503  
## 13 35.53830 0.0142288217 27.87037  
## 14 35.51263 0.0141684880 27.86313  
## 15 35.50258 0.0137494913 27.87540  
## 16 35.49106 0.0134916898 27.87377  
## 17 35.49338 0.0128433667 27.89116  
## 18 35.48547 0.0127536372 27.89219  
## 19 35.46435 0.0133963137 27.87610  
## 20 35.46258 0.0132317143 27.87753  
## 21 35.47397 0.0123928657 27.89122  
## 22 35.48890 0.0113586027 27.90008  
## 23 35.50080 0.0104341679 27.91220  
## 24 35.52090 0.0091278754 27.92292  
## 25 35.55519 0.0070742295 27.94176  
## 26 35.58109 0.0055734543 27.95844  
## 27 35.60477 0.0041524173 27.97520  
## 28 35.61258 0.0036244328 27.98172  
## 29 35.62353 0.0028171451 27.98479  
## 30 35.62839 0.0023164659 27.98704  
## 31 35.63114 0.0019792885 27.98673  
## 32 35.63792 0.0015551359 27.98812  
## 33 35.64019 0.0013457651 27.98466  
## 34 35.64424 0.0010453753 27.98622  
## 35 35.64671 0.0009183578 27.98157  
## 36 35.65396 0.0004922536 27.98663  
## 37 35.66156 0.0001961450 27.98852  
## 38 35.65257 0.0001741643 27.97793  
## 39 35.64370 0.0003003220 27.96893  
## 40 35.63394 0.0006650456 27.95697  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final value used for the model was k = 20.

# final model uses: k=40  
 # RMSE: 35.63394, Rsquared: 0.0006650456  
plot(knnTune, main="KNN")



knnTune$finalModel

## 20-nearest neighbor regression model

## Multivariate Adaptive Regression Splines (MARS)

### Create Initial MARS Model

# MARS model doesn't need preprocessing, so first rendition will be with Chol\_tr\_X\_imp\_fin   
set.seed(123)  
mars1 <- train(x = Chol\_tr\_X\_imp\_fin, y = Chol\_train\_y,  
 method = "earth",  
 tuneGrid = expand.grid(degree = 1:3, nprune = 2:38),  
 trControl = Chol\_control)

## Loading required package: earth

## Loading required package: Formula

## Loading required package: plotmo

## Loading required package: plotrix

## Loading required package: TeachingDemos

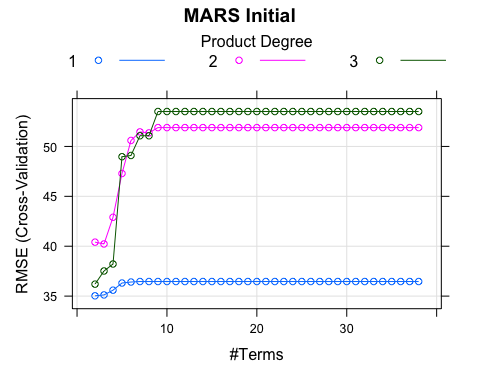
mars1$finalModel

## Selected 2 of 18 terms, and 1 of 32 predictors (nprune=2)  
## Termination condition: RSq changed by less than 0.001 at 18 terms  
## Importance: RIDAGEYR, INDFMPIR-unused, BMXHT-unused, BMXBMI-unused, ...  
## Number of terms at each degree of interaction: 1 1 (additive model)  
## GCV 1192.459 RSS 4400176 GRSq 0.05246562 RSq 0.05349109

mars1$results

## degree nprune RMSE Rsquared MAE RMSESD RsquaredSD MAESD  
## 1 1 2 35.02993 0.05354368 27.23479 0.8776969 0.02164487 0.8388177  
## 38 2 2 40.40751 0.02722680 28.84092 8.0742255 0.03196192 2.2457613  
## 75 3 2 36.19850 0.01851951 28.10321 1.1359076 0.02897909 0.9241746  
## 2 1 3 35.11904 0.04859923 27.25761 0.7739781 0.02098486 0.7361259  
## 39 2 3 40.22536 0.03489353 28.61387 8.1668305 0.02990257 2.2954634  
## 76 3 3 37.51833 0.03203798 27.98807 5.5496554 0.01988232 1.6758328  
## 3 1 4 35.59312 0.03971912 27.69936 0.8737576 0.01716185 0.8183737  
## 40 2 4 42.89990 0.02407639 29.68482 11.6118373 0.02138903 3.0606821  
## 77 3 4 38.21781 0.02811604 28.48588 5.4394166 0.01939410 1.6187438  
## 4 1 5 36.31180 0.03050825 28.37861 1.0518493 0.01216991 0.9764692  
## 41 2 5 47.28733 0.02008981 31.50484 17.6678866 0.01875604 5.1150224  
## 78 3 5 48.96919 0.02177852 30.88872 19.2957817 0.01597779 3.9485600  
## 5 1 6 36.41079 0.03003684 28.45269 1.3275591 0.01312629 1.1882532  
## 42 2 6 50.59945 0.01356533 32.88542 18.2410611 0.01316059 5.2931544  
## 79 3 6 49.09454 0.01938229 31.26730 18.9198893 0.01432148 4.1544405  
## 6 1 7 36.46136 0.03003537 28.49695 1.4741756 0.01312943 1.3184185  
## 43 2 7 51.45988 0.01307332 33.46189 17.7833728 0.01328287 5.2389106  
## 80 3 7 51.06975 0.01883827 32.14162 20.7509319 0.01469495 4.7520450  
## 7 1 8 36.46136 0.03003537 28.49695 1.4741756 0.01312943 1.3184185  
## 44 2 8 51.35546 0.01302901 33.50902 17.6130277 0.01332896 5.3091567  
## 81 3 8 51.06975 0.01883827 32.14162 20.7509319 0.01469495 4.7520450  
## 8 1 9 36.46136 0.03003537 28.49695 1.4741756 0.01312943 1.3184185  
## 45 2 9 51.88763 0.01303077 33.72722 18.5263044 0.01332710 5.6742169  
## 82 3 9 53.49901 0.01879599 32.85927 25.2811024 0.01472897 6.1394170  
## 9 1 10 36.46136 0.03003537 28.49695 1.4741756 0.01312943 1.3184185  
## 46 2 10 51.88763 0.01303077 33.72722 18.5263044 0.01332710 5.6742169  
## 83 3 10 53.49901 0.01879599 32.85927 25.2811024 0.01472897 6.1394170  
## 10 1 11 36.46136 0.03003537 28.49695 1.4741756 0.01312943 1.3184185  
## 47 2 11 51.88763 0.01303077 33.72722 18.5263044 0.01332710 5.6742169  
## 84 3 11 53.49901 0.01879599 32.85927 25.2811024 0.01472897 6.1394170  
## 11 1 12 36.46136 0.03003537 28.49695 1.4741756 0.01312943 1.3184185  
## 48 2 12 51.88763 0.01303077 33.72722 18.5263044 0.01332710 5.6742169  
## 85 3 12 53.49901 0.01879599 32.85927 25.2811024 0.01472897 6.1394170  
## 12 1 13 36.46136 0.03003537 28.49695 1.4741756 0.01312943 1.3184185  
## 49 2 13 51.88763 0.01303077 33.72722 18.5263044 0.01332710 5.6742169  
## 86 3 13 53.49901 0.01879599 32.85927 25.2811024 0.01472897 6.1394170  
## 13 1 14 36.46136 0.03003537 28.49695 1.4741756 0.01312943 1.3184185  
## 50 2 14 51.88763 0.01303077 33.72722 18.5263044 0.01332710 5.6742169  
## 87 3 14 53.49901 0.01879599 32.85927 25.2811024 0.01472897 6.1394170  
## 14 1 15 36.46136 0.03003537 28.49695 1.4741756 0.01312943 1.3184185  
## 51 2 15 51.88763 0.01303077 33.72722 18.5263044 0.01332710 5.6742169  
## 88 3 15 53.49901 0.01879599 32.85927 25.2811024 0.01472897 6.1394170  
## 15 1 16 36.46136 0.03003537 28.49695 1.4741756 0.01312943 1.3184185  
## 52 2 16 51.88763 0.01303077 33.72722 18.5263044 0.01332710 5.6742169  
## 89 3 16 53.49901 0.01879599 32.85927 25.2811024 0.01472897 6.1394170  
## 16 1 17 36.46136 0.03003537 28.49695 1.4741756 0.01312943 1.3184185  
## 53 2 17 51.88763 0.01303077 33.72722 18.5263044 0.01332710 5.6742169  
## 90 3 17 53.49901 0.01879599 32.85927 25.2811024 0.01472897 6.1394170  
## 17 1 18 36.46136 0.03003537 28.49695 1.4741756 0.01312943 1.3184185  
## 54 2 18 51.88763 0.01303077 33.72722 18.5263044 0.01332710 5.6742169  
## 91 3 18 53.49901 0.01879599 32.85927 25.2811024 0.01472897 6.1394170  
## 18 1 19 36.46136 0.03003537 28.49695 1.4741756 0.01312943 1.3184185  
## 55 2 19 51.88763 0.01303077 33.72722 18.5263044 0.01332710 5.6742169  
## 92 3 19 53.49901 0.01879599 32.85927 25.2811024 0.01472897 6.1394170  
## 19 1 20 36.46136 0.03003537 28.49695 1.4741756 0.01312943 1.3184185  
## 56 2 20 51.88763 0.01303077 33.72722 18.5263044 0.01332710 5.6742169  
## 93 3 20 53.49901 0.01879599 32.85927 25.2811024 0.01472897 6.1394170  
## 20 1 21 36.46136 0.03003537 28.49695 1.4741756 0.01312943 1.3184185  
## 57 2 21 51.88763 0.01303077 33.72722 18.5263044 0.01332710 5.6742169  
## 94 3 21 53.49901 0.01879599 32.85927 25.2811024 0.01472897 6.1394170  
## 21 1 22 36.46136 0.03003537 28.49695 1.4741756 0.01312943 1.3184185  
## 58 2 22 51.88763 0.01303077 33.72722 18.5263044 0.01332710 5.6742169  
## 95 3 22 53.49901 0.01879599 32.85927 25.2811024 0.01472897 6.1394170  
## 22 1 23 36.46136 0.03003537 28.49695 1.4741756 0.01312943 1.3184185  
## 59 2 23 51.88763 0.01303077 33.72722 18.5263044 0.01332710 5.6742169  
## 96 3 23 53.49901 0.01879599 32.85927 25.2811024 0.01472897 6.1394170  
## 23 1 24 36.46136 0.03003537 28.49695 1.4741756 0.01312943 1.3184185  
## 60 2 24 51.88763 0.01303077 33.72722 18.5263044 0.01332710 5.6742169  
## 97 3 24 53.49901 0.01879599 32.85927 25.2811024 0.01472897 6.1394170  
## 24 1 25 36.46136 0.03003537 28.49695 1.4741756 0.01312943 1.3184185  
## 61 2 25 51.88763 0.01303077 33.72722 18.5263044 0.01332710 5.6742169  
## 98 3 25 53.49901 0.01879599 32.85927 25.2811024 0.01472897 6.1394170  
## 25 1 26 36.46136 0.03003537 28.49695 1.4741756 0.01312943 1.3184185  
## 62 2 26 51.88763 0.01303077 33.72722 18.5263044 0.01332710 5.6742169  
## 99 3 26 53.49901 0.01879599 32.85927 25.2811024 0.01472897 6.1394170  
## 26 1 27 36.46136 0.03003537 28.49695 1.4741756 0.01312943 1.3184185  
## 63 2 27 51.88763 0.01303077 33.72722 18.5263044 0.01332710 5.6742169  
## 100 3 27 53.49901 0.01879599 32.85927 25.2811024 0.01472897 6.1394170  
## 27 1 28 36.46136 0.03003537 28.49695 1.4741756 0.01312943 1.3184185  
## 64 2 28 51.88763 0.01303077 33.72722 18.5263044 0.01332710 5.6742169  
## 101 3 28 53.49901 0.01879599 32.85927 25.2811024 0.01472897 6.1394170  
## 28 1 29 36.46136 0.03003537 28.49695 1.4741756 0.01312943 1.3184185  
## 65 2 29 51.88763 0.01303077 33.72722 18.5263044 0.01332710 5.6742169  
## 102 3 29 53.49901 0.01879599 32.85927 25.2811024 0.01472897 6.1394170  
## 29 1 30 36.46136 0.03003537 28.49695 1.4741756 0.01312943 1.3184185  
## 66 2 30 51.88763 0.01303077 33.72722 18.5263044 0.01332710 5.6742169  
## 103 3 30 53.49901 0.01879599 32.85927 25.2811024 0.01472897 6.1394170  
## 30 1 31 36.46136 0.03003537 28.49695 1.4741756 0.01312943 1.3184185  
## 67 2 31 51.88763 0.01303077 33.72722 18.5263044 0.01332710 5.6742169  
## 104 3 31 53.49901 0.01879599 32.85927 25.2811024 0.01472897 6.1394170  
## 31 1 32 36.46136 0.03003537 28.49695 1.4741756 0.01312943 1.3184185  
## 68 2 32 51.88763 0.01303077 33.72722 18.5263044 0.01332710 5.6742169  
## 105 3 32 53.49901 0.01879599 32.85927 25.2811024 0.01472897 6.1394170  
## 32 1 33 36.46136 0.03003537 28.49695 1.4741756 0.01312943 1.3184185  
## 69 2 33 51.88763 0.01303077 33.72722 18.5263044 0.01332710 5.6742169  
## 106 3 33 53.49901 0.01879599 32.85927 25.2811024 0.01472897 6.1394170  
## 33 1 34 36.46136 0.03003537 28.49695 1.4741756 0.01312943 1.3184185  
## 70 2 34 51.88763 0.01303077 33.72722 18.5263044 0.01332710 5.6742169  
## 107 3 34 53.49901 0.01879599 32.85927 25.2811024 0.01472897 6.1394170  
## 34 1 35 36.46136 0.03003537 28.49695 1.4741756 0.01312943 1.3184185  
## 71 2 35 51.88763 0.01303077 33.72722 18.5263044 0.01332710 5.6742169  
## 108 3 35 53.49901 0.01879599 32.85927 25.2811024 0.01472897 6.1394170  
## 35 1 36 36.46136 0.03003537 28.49695 1.4741756 0.01312943 1.3184185  
## 72 2 36 51.88763 0.01303077 33.72722 18.5263044 0.01332710 5.6742169  
## 109 3 36 53.49901 0.01879599 32.85927 25.2811024 0.01472897 6.1394170  
## 36 1 37 36.46136 0.03003537 28.49695 1.4741756 0.01312943 1.3184185  
## 73 2 37 51.88763 0.01303077 33.72722 18.5263044 0.01332710 5.6742169  
## 110 3 37 53.49901 0.01879599 32.85927 25.2811024 0.01472897 6.1394170  
## 37 1 38 36.46136 0.03003537 28.49695 1.4741756 0.01312943 1.3184185  
## 74 2 38 51.88763 0.01303077 33.72722 18.5263044 0.01332710 5.6742169  
## 111 3 38 53.49901 0.01879599 32.85927 25.2811024 0.01472897 6.1394170

#used 1 of 32 predictors, 2 of 18 terms (nprune =2) degree=1  
 # RMSE: 35.02993, Rsquared: 0.05354368   
plot(mars1, main="MARS Initial")



## Multivariate Adaptive Regression Splines (MARS)

### Create Secondary MARS Model

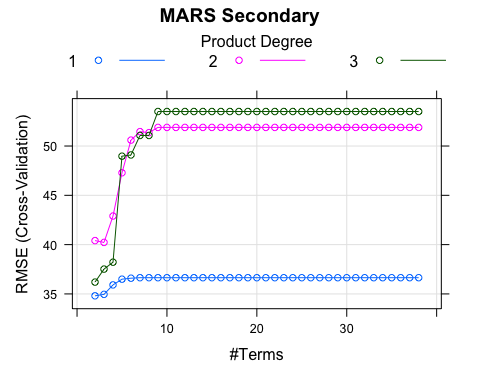
# MARS model using same X sets as SVM models:   
set.seed(123)  
mars2 <- train(x = Chol\_tr\_X\_impfin\_sv, y = Chol\_train\_y,  
 method = "earth",  
 tuneGrid = expand.grid(degree = 1:3, nprune = 2:38),  
 trControl = Chol\_control)

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

mars2$finalModel

## Selected 2 of 18 terms, and 1 of 29 predictors (nprune=2)  
## Termination condition: RSq changed by less than 0.001 at 18 terms  
## Importance: RIDAGEYR, INDFMPIR-unused, BMXHT-unused, BMXBMI-unused, ...  
## Number of terms at each degree of interaction: 1 1 (additive model)  
## GCV 1192.459 RSS 4400176 GRSq 0.05246562 RSq 0.05349109

plot(mars2, main="MARS Secondary")



# No change between the two MARS models. Drops all but two factors for both.  
 #nprune=2, degree=1, RMSE: 34.80857, Rsquared: 0.05457991

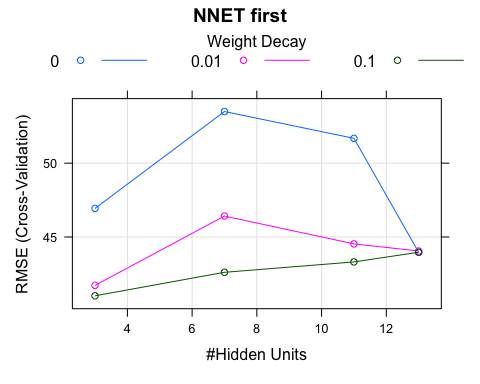
## Neural Network Model (nnet)

### Create Initial NNET Model

set.seed(123)  
nnetGrid <- expand.grid(decay = c(0, 0.01, .1), size = c(3, 7, 11, 13))  
# NNET first rendition will be with Chol\_tr\_X\_imp\_fin   
set.seed(100)  
nnet1 <- train(x = Chol\_tr\_X\_imp\_fin, y = Chol\_train\_y,  
 method = "nnet",  
 tuneGrid = nnetGrid,  
 trControl = Chol\_control,  
 preProc = c("center", "scale"),  
 linout = TRUE,  
 trace = FALSE,  
 MaxNWts = 13 \* (ncol(Chol\_tr\_X\_imp\_fin) + 1) + 13 + 1,  
 maxit = 100)  
nnet1

## Neural Network   
##   
## 3696 samples  
## 32 predictor  
##   
## Pre-processing: centered (32), scaled (32)   
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 51, 50, 48, 50, 46, 46, ...   
## Resampling results across tuning parameters:  
##   
## decay size RMSE Rsquared MAE   
## 0.00 3 46.93574 0.0030605097 35.85699  
## 0.00 7 53.49676 0.0019465703 40.63233  
## 0.00 11 51.67898 0.0026194114 40.58513  
## 0.00 13 43.96067 0.0006635379 34.48169  
## 0.01 3 41.72586 0.0021627217 32.84491  
## 0.01 7 46.41820 0.0031789071 36.05993  
## 0.01 11 44.53251 0.0031900494 34.98660  
## 0.01 13 44.05154 0.0014003744 34.48739  
## 0.10 3 41.01832 0.0050800112 32.18821  
## 0.10 7 42.60989 0.0034464589 33.52421  
## 0.10 11 43.31281 0.0024589450 33.88722  
## 0.10 13 43.96715 0.0028314273 34.44477  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final values used for the model were size = 3 and decay = 0.1.

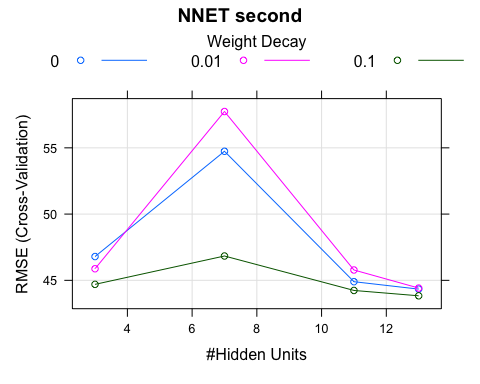
# size=3, decay=0.1  
 # RMSE: 40.97813, Rsquared: 0.0051328366  
plot(nnet1, main="NNET first")



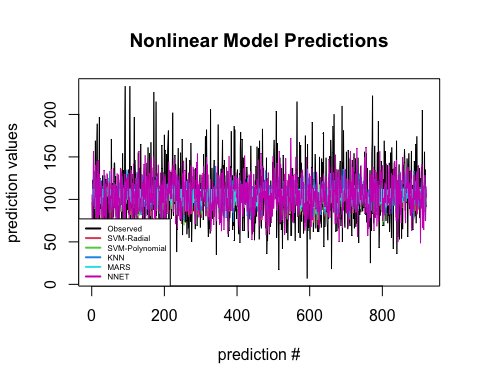
set.seed(123)  
# NNET second rendition will be with Chol\_tr\_X\_impfin\_sv   
set.seed(100)  
nnet2 <- train(x = Chol\_tr\_X\_impfin\_sv, y = Chol\_train\_y,  
 method = "nnet",  
 tuneGrid = nnetGrid,  
 trControl = Chol\_control,  
 preProc = c("center", "scale"),  
 linout = TRUE,  
 trace = FALSE,  
 MaxNWts = 13 \* (ncol(Chol\_tr\_X\_impfin\_sv) + 1) + 13 + 1,  
 maxit = 100)  
nnet2

## Neural Network   
##   
## 3696 samples  
## 29 predictor  
##   
## Pre-processing: centered (29), scaled (29)   
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 51, 50, 48, 50, 46, 46, ...   
## Resampling results across tuning parameters:  
##   
## decay size RMSE Rsquared MAE   
## 0.00 3 46.79483 0.002847267 35.64307  
## 0.00 7 54.74047 0.002393215 42.78625  
## 0.00 11 44.89697 0.001922672 35.31086  
## 0.00 13 44.33676 0.002345836 34.77825  
## 0.01 3 45.87404 0.003545346 35.17972  
## 0.01 7 57.74093 0.003905650 42.06573  
## 0.01 11 45.78004 0.002195915 35.93089  
## 0.01 13 44.41660 0.001767701 34.89326  
## 0.10 3 44.69636 0.002842232 34.90825  
## 0.10 7 46.83613 0.002421236 36.27177  
## 0.10 11 44.23575 0.002229325 34.68948  
## 0.10 13 43.83204 0.001984790 34.40128  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final values used for the model were size = 13 and decay = 0.1.

# size=13, decay=0.1  
 # RMSE: 44.38061, Rsquared: 0.001743654  
plot(nnet2, main="NNET second")

 ## Comparing Nonlinear Models: ### Saving Results

NonLpred <- data.frame(obs=Chol\_test\_y)  
NonLpred$svmR <- predict(svmR1, Chol\_te\_X\_sv)  
NonLpred$svmP <- predict(svmP, Chol\_te\_X\_sv)  
NonLpred$KNN <- predict(knnTune, Chol\_te\_X\_sv)  
NonLpred$MARS2 <- predict(mars2, Chol\_te\_X\_sv)  
NonLpred$NNET2 <- predict(nnet2, Chol\_te\_X\_sv)  
plotpred <- data.frame(x=1:921, y1=NonLpred$obs, y2=NonLpred$svmR,   
 y3=NonLpred$svmP, y4=NonLpred$KNN,   
 y5=NonLpred$MARS2[,"y"], y6=NonLpred$NNET2)  
plot(plotpred$x, plotpred$y1, type = "l", col = 1,   
 xlab = "prediction #", ylab = "prediction values",   
 main = "Nonlinear Model Predictions")  
lines(plotpred$x, plotpred$y2, col = 2)  
lines(plotpred$x, plotpred$y3, col = 3)  
lines(plotpred$x, plotpred$y4, col = 4)  
lines(plotpred$x, plotpred$y5, col = 5)  
lines(plotpred$x, plotpred$y6, col = 6)  
legend("bottomleft", cex=0.5, legend = c("Observed", "SVM-Radial",  
 "SVM-Polynomial", "KNN", "MARS", "NNET"),  
 col = 1:6, lwd = 2)

 ### Getting RMSE and Plotting

# RMSE = sqrt(sum((obs-pred)^2)/n), n=921  
getRMSE <- function(x,y) {  
 sqrt(sum((x-y)^2)/length(x))  
}  
nonlin\_rmse <- data.frame(c("svmRad", "svmPoly", "KNN",  
 "MARS", "NNet"))  
nonlin\_rmse$RMSE <- c(getRMSE(NonLpred$obs, NonLpred$svmR),  
 getRMSE(NonLpred$obs, NonLpred$svmP),  
 getRMSE(NonLpred$obs, NonLpred$KNN),  
 getRMSE(NonLpred$obs, NonLpred$MARS2),  
 getRMSE(NonLpred$obs, NonLpred$NNET2))  
colnames(nonlin\_rmse)[1]<-"Model Type"  
nonlin\_rmse[order(nonlin\_rmse$RMSE),]

## Model Type RMSE  
## 1 svmRad 32.94845  
## 2 svmPoly 33.03741  
## 4 MARS 33.52618  
## 3 KNN 33.85140  
## 5 NNet 36.94634

# best non linear model is svm radial with   
 # sigma = sigma = 0.03504524 and C = 0.25. (RMSE is 32.95)

### Messing around with predictor set and log transforming the response

expDropC <- c("RIDRETH1\_Black", "RIDRETH1\_Mex", "RIDRETH1\_OHis",   
 "RIDRETH1\_Oth", "RIDRETH1\_White", "BMXBMI")  
Xtrial\_tr <- subset(Chol\_train\_X\_imp, select = !(names(Chol\_train\_X\_imp) %in% expDropC))  
# looking for near zero var:  
Xtri\_nzv <- nearZeroVar(Xtrial\_tr)  
 # "DMDBORN4\_Ref", "DMDBORN4\_DK", "DMDEDUC2\_Ref", "DMDEDUC2\_DK", "DMDMARTZ\_Ref",  
 # "DMDMARTZ\_DK", "AIALANGA\_Asian"  
Xtri\_tr\_nz <- subset(Xtrial\_tr, select = -c(Xtri\_nzv))  
print(paste("Xtrial\_tr ncol: ", ncol(Xtrial\_tr), " NZV removed, new ncol: ", ncol(Xtri\_tr\_nz)))

## [1] "Xtrial\_tr ncol: 47 NZV removed, new ncol: 40"

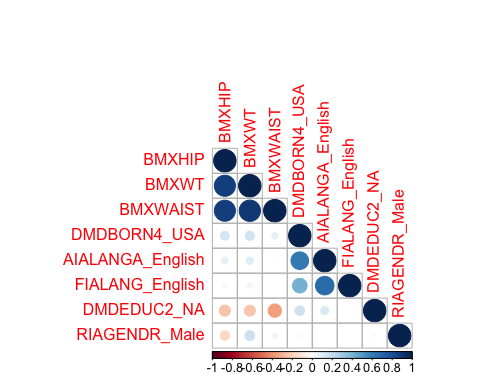
# dropping SIALANG groups (sample person interview instrument lang)  
expDropC <- c("SIALANG\_English", "SIALANG\_Spanish",   
 "MIALANG\_English", "MIALANG\_Spanish", "MIALANG\_NA")  
Xtri\_tr\_nz <- subset(Xtri\_tr\_nz, select = !(names(Xtri\_tr\_nz) %in% expDropC))  
print(paste("Xtrial\_tr ncol: ", ncol(Xtrial\_tr), " NZV removed, new ncol: ", ncol(Xtri\_tr\_nz)))

## [1] "Xtrial\_tr ncol: 47 NZV removed, new ncol: 35"

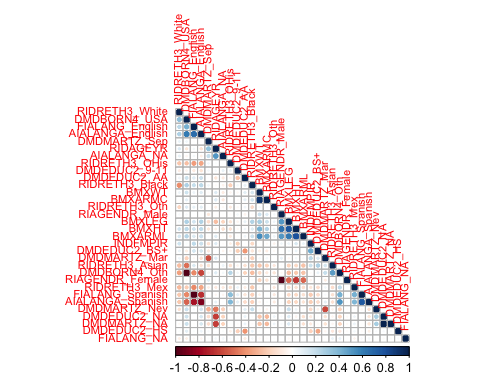
#looking for high corr:  
Xtritr\_hiC <- findCorrelation(cor(Xtri\_tr\_nz), cutoff = 0.8)  
Xtritr\_hiC

## [1] 33 18 3 30 8 9 25 10

# "AIALANGA\_English", "DMDBORN4\_USA", "BMXWT", "FIALANG\_English",   
 # "BMXWAIST", "BMXHIP", "DMDEDUC2\_NA", "RIAGENDR\_Male"  
  
Xtritr\_hiC <- subset(Xtri\_tr\_nz, select = c("AIALANGA\_English", "DMDBORN4\_USA",   
 "BMXWT", "FIALANG\_English", "BMXWAIST",  
 "BMXHIP", "DMDEDUC2\_NA", "RIAGENDR\_Male"))  
invisible(cor(Xtritr\_hiC)) # invisible used to reduce extensive output  
corrplot(cor(Xtritr\_hiC), order = "hclust", type="lower")



#dropping "BMXHIP", "BMXWAIST", DMDEDUC2\_<9 (recurring issues in model attempts)  
drophiC <- c("BMXHIP", "BMXWAIST", "DMDEDUC2\_<9")  
X\_trial\_train <- subset(Xtri\_tr\_nz, select = !(names(Xtri\_tr\_nz) %in% drophiC))  
corrplot(cor(X\_trial\_train), order = "hclust", type="lower", tl.cex = 0.7)

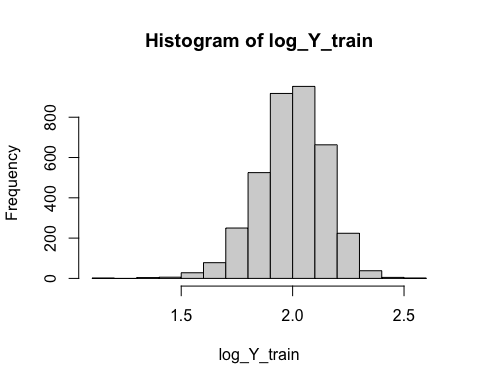


### Making test and train columns match

keepsies <- colnames(X\_trial\_train)  
X\_trial\_test <- subset(Chol\_test\_X\_imp, select = c(keepsies))

### Radial SVM with the matched data and log adjusted y

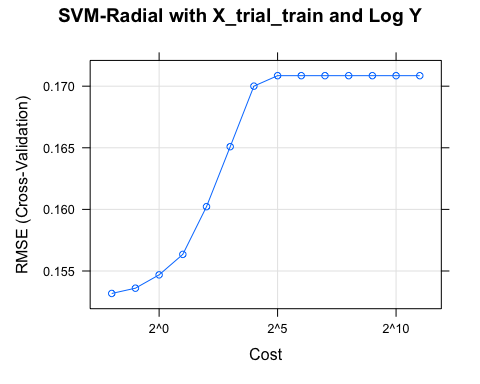
#svm radial with X\_trial\_train and log adjusted y  
log\_Y\_train <- log10(Chol\_train\_y)  
hist(log\_Y\_train)



log\_Y\_test <- log10(Chol\_test\_y)  
  
#model  
set.seed(123)  
svmR\_trial <- train(x=X\_trial\_train, y = log\_Y\_train,  
 method = "svmRadial",  
 preProcess = c("center", "scale"),  
 tuneLength = 14,  
 trControl = Chol\_control)  
svmR\_trial

## Support Vector Machines with Radial Basis Function Kernel   
##   
## 3696 samples  
## 32 predictor  
##   
## Pre-processing: centered (32), scaled (32)   
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 51, 50, 48, 50, 46, 46, ...   
## Resampling results across tuning parameters:  
##   
## C RMSE Rsquared MAE   
## 0.25 0.1531765 0.021637157 0.1185587  
## 0.50 0.1536049 0.016237646 0.1189650  
## 1.00 0.1546869 0.012994159 0.1199085  
## 2.00 0.1563482 0.008328590 0.1217328  
## 4.00 0.1602217 0.005763568 0.1253818  
## 8.00 0.1650901 0.005535310 0.1294877  
## 16.00 0.1700027 0.005258265 0.1334787  
## 32.00 0.1708529 0.005056529 0.1341204  
## 64.00 0.1708529 0.005056529 0.1341204  
## 128.00 0.1708529 0.005056529 0.1341204  
## 256.00 0.1708529 0.005056529 0.1341204  
## 512.00 0.1708529 0.005056529 0.1341204  
## 1024.00 0.1708529 0.005056529 0.1341204  
## 2048.00 0.1708529 0.005056529 0.1341204  
##   
## Tuning parameter 'sigma' was held constant at a value of 0.02019005  
## RMSE was used to select the optimal model using the smallest value.  
## The final values used for the model were sigma = 0.02019005 and C = 0.25.

# issues in: DMDEDUC2\_<9,   
 # final model uses: sigma = 0.02019005 and C = 0.25.  
 # RMSE: 0.1531765, Rsquared: 0.021637157  
 # while these are the lowest values, the graph is identical to the non-log adjusted SVM radial model with just different RMSE values.   
plot(svmR\_trial, scales = list(x = list(log = 2)), main="SVM-Radial with X\_trial\_train and Log Y")



svmR\_trial$finalModel

## Support Vector Machine object of class "ksvm"   
##   
## SV type: eps-svr (regression)   
## parameter : epsilon = 0.1 cost C = 0.25   
##   
## Gaussian Radial Basis kernel function.   
## Hyperparameter : sigma = 0.0201900463201106   
##   
## Number of Support Vectors : 3367   
##   
## Objective Function Value : -574.0137   
## Training error : 0.850126

trialPred <- data.frame(obs=log\_Y\_test)  
trialPred$svmRad <- predict(svmR\_trial, X\_trial\_test)

# Regression Trees - Eva

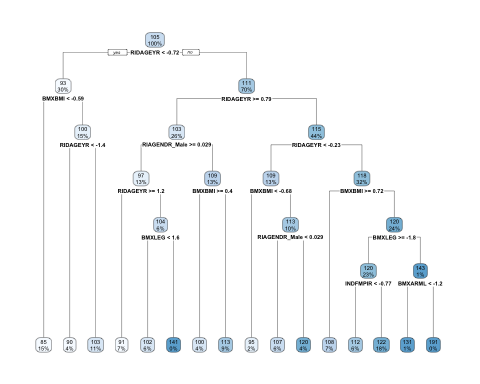
## Single Regression Trees

### Tree with CART based splits (rpart) and optimization with default parameters

set.seed(123)  
# let's see how it splits the training data  
Chol\_rpart <- rpart(Chol\_train\_y ~., data = Chol\_tr\_X\_imp\_fin, cp = 0.003)  
  
summary(Chol\_rpart)

## Call:  
## rpart(formula = Chol\_train\_y ~ ., data = Chol\_tr\_X\_imp\_fin, cp = 0.003)  
## n= 3696   
##   
## CP nsplit rel error xerror xstd  
## 1 0.054164414 0 1.0000000 1.0004973 0.03300310  
## 2 0.020563310 1 0.9458356 0.9554923 0.03188785  
## 3 0.013308632 2 0.9252723 0.9464258 0.03207678  
## 4 0.007112960 3 0.9119636 0.9311717 0.03205759  
## 5 0.005855210 4 0.9048507 0.9260955 0.03176146  
## 6 0.004972343 6 0.8931403 0.9256549 0.03150486  
## 7 0.004824224 7 0.8881679 0.9238887 0.03142018  
## 8 0.004142728 8 0.8833437 0.9264225 0.03141473  
## 9 0.003828087 10 0.8750582 0.9339156 0.03236363  
## 10 0.003774630 11 0.8712302 0.9329465 0.03238441  
## 11 0.003599490 12 0.8674555 0.9316994 0.03236427  
## 12 0.003384505 13 0.8638560 0.9351374 0.03241022  
## 13 0.003021559 14 0.8604715 0.9307203 0.03231526  
## 14 0.003000000 15 0.8574500 0.9348469 0.03254034  
##   
## Variable importance  
## RIDAGEYR BMXBMI DMDEDUC2\_NA DMDMARTZ\_Nev BMXARMC   
## 32 13 12 9 8   
## BMXARML AIALANGA\_NA BMXLEG BMXHT RIAGENDR\_Male   
## 5 5 4 4 4   
## INDFMPIR DMDMARTZ\_Sep DMDEDUC2\_BS+ DMDMARTZ\_Mar   
## 1 1 1 1   
##   
## Node number 1: 3696 observations, complexity param=0.05416441  
## mean=105.2411, MSE=1257.805   
## left son=2 (1098 obs) right son=3 (2598 obs)  
## Primary splits:  
## RIDAGEYR < -0.7177214 to the left, improve=0.05416441, (0 missing)  
## DMDEDUC2\_NA < 0.9305478 to the right, improve=0.04807959, (0 missing)  
## DMDMARTZ\_Nev < -0.2411516 to the right, improve=0.02845850, (0 missing)  
## BMXBMI < -0.7742706 to the left, improve=0.02378829, (0 missing)  
## BMXARMC < -0.8110234 to the left, improve=0.01974683, (0 missing)  
## Surrogate splits:  
## DMDEDUC2\_NA < 0.9305478 to the right, agree=0.862, adj=0.536, (0 split)  
## DMDMARTZ\_Nev < -0.2411516 to the right, agree=0.846, adj=0.482, (0 split)  
## BMXBMI < -0.9678924 to the left, agree=0.735, adj=0.108, (0 split)  
## BMXARMC < -1.151012 to the left, agree=0.730, adj=0.091, (0 split)  
##   
## Node number 2: 1098 observations, complexity param=0.01330863  
## mean=92.54463, MSE=805.3737   
## left son=4 (536 obs) right son=5 (562 obs)  
## Primary splits:  
## BMXBMI < -0.5935568 to the left, improve=0.06996470, (0 missing)  
## BMXARMC < -0.113152 to the left, improve=0.06346379, (0 missing)  
## RIDAGEYR < -1.398075 to the left, improve=0.04298442, (0 missing)  
## DMDEDUC2\_NA < 0.9305478 to the right, improve=0.03832009, (0 missing)  
## DMDEDUC2\_AA < -0.4815614 to the right, improve=0.02453409, (0 missing)  
## Surrogate splits:  
## BMXARMC < -0.4352465 to the left, agree=0.884, adj=0.763, (0 split)  
## BMXARML < -0.3506071 to the left, agree=0.622, adj=0.226, (0 split)  
## RIDAGEYR < -1.203688 to the left, agree=0.605, adj=0.190, (0 split)  
## DMDEDUC2\_NA < 0.9305478 to the right, agree=0.604, adj=0.188, (0 split)  
## DMDEDUC2\_BS+ < -0.3327601 to the right, agree=0.587, adj=0.153, (0 split)  
##   
## Node number 3: 2598 observations, complexity param=0.02056331  
## mean=110.607, MSE=1352.096   
## left son=6 (968 obs) right son=7 (1630 obs)  
## Primary splits:  
## RIDAGEYR < 0.7887756 to the right, improve=0.027213960, (0 missing)  
## AIALANGA\_NA < 0.7694033 to the right, improve=0.019766670, (0 missing)  
## BMXBMI < 0.6714394 to the right, improve=0.007005182, (0 missing)  
## DMDBORN4\_Oth < 0.5241397 to the left, improve=0.006384595, (0 missing)  
## BMXARML < 0.8611078 to the right, improve=0.005282776, (0 missing)  
## Surrogate splits:  
## AIALANGA\_NA < 0.7694033 to the right, agree=0.787, adj=0.428, (0 split)  
## DMDMARTZ\_Sep < 0.6618522 to the right, agree=0.653, adj=0.068, (0 split)  
## BMXLEG < -2.129812 to the left, agree=0.630, adj=0.007, (0 split)  
## BMXHT < -2.253535 to the left, agree=0.629, adj=0.005, (0 split)  
## BMXARML < -2.618245 to the left, agree=0.628, adj=0.001, (0 split)  
##   
## Node number 4: 536 observations  
## mean=84.85821, MSE=627.8344   
##   
## Node number 5: 562 observations, complexity param=0.003828087  
## mean=99.87544, MSE=864.6108   
## left son=10 (146 obs) right son=11 (416 obs)  
## Primary splits:  
## RIDAGEYR < -1.398075 to the left, improve=0.03662437, (0 missing)  
## BMXBMI < 0.32292 to the left, improve=0.02895393, (0 missing)  
## BMXARMC < 0.3163073 to the left, improve=0.02481897, (0 missing)  
## DMDEDUC2\_NA < 0.9305478 to the right, improve=0.02327611, (0 missing)  
## DMDEDUC2\_AA < -0.4815614 to the right, improve=0.01764382, (0 missing)  
## Surrogate splits:  
## DMDEDUC2\_NA < 0.9305478 to the right, agree=0.826, adj=0.329, (0 split)  
## INDFMPIR < -1.53041 to the left, agree=0.746, adj=0.021, (0 split)  
## BMXBMI < -0.5548324 to the left, agree=0.746, adj=0.021, (0 split)  
## BMXARMC < -1.043647 to the left, agree=0.746, adj=0.021, (0 split)  
## BMXHT < -2.298376 to the left, agree=0.744, adj=0.014, (0 split)  
##   
## Node number 6: 968 observations, complexity param=0.00711296  
## mean=102.7355, MSE=1416.277   
## left son=12 (495 obs) right son=13 (473 obs)  
## Primary splits:  
## RIAGENDR\_Male < 0.02868757 to the right, improve=0.02411971, (0 missing)  
## BMXHT < -0.4649097 to the right, improve=0.02055340, (0 missing)  
## BMXARML < 0.8611078 to the right, improve=0.01581797, (0 missing)  
## BMXBMI < 0.3706801 to the right, improve=0.01427778, (0 missing)  
## BMXARMC < -0.1310462 to the right, improve=0.01285898, (0 missing)  
## Surrogate splits:  
## BMXHT < -0.06633024 to the right, agree=0.853, adj=0.700, (0 split)  
## BMXLEG < -0.2429857 to the right, agree=0.773, adj=0.535, (0 split)  
## BMXARML < 0.2864087 to the right, agree=0.770, adj=0.529, (0 split)  
## DMDMARTZ\_Mar < -0.1600155 to the right, agree=0.616, adj=0.214, (0 split)  
## DMDMARTZ\_Sep < 0.6618522 to the left, agree=0.606, adj=0.195, (0 split)  
##   
## Node number 7: 1630 observations, complexity param=0.00585521  
## mean=115.2816, MSE=1255.334   
## left son=14 (464 obs) right son=15 (1166 obs)  
## Primary splits:  
## RIDAGEYR < -0.2317546 to the left, improve=0.011590290, (0 missing)  
## BMXBMI < 0.7617963 to the right, improve=0.011509240, (0 missing)  
## BMXLEG < -1.843139 to the right, improve=0.007419219, (0 missing)  
## DMDBORN4\_Oth < 0.5241397 to the left, improve=0.005611986, (0 missing)  
## BMXARMC < 1.085755 to the right, improve=0.004611308, (0 missing)  
## Surrogate splits:  
## BMXBMI < -1.613299 to the left, agree=0.718, adj=0.009, (0 split)  
## BMXARMC < -1.741519 to the left, agree=0.718, adj=0.009, (0 split)  
## BMXLEG < 2.27452 to the right, agree=0.717, adj=0.006, (0 split)  
## BMXARML < 2.817162 to the right, agree=0.717, adj=0.006, (0 split)  
##   
## Node number 10: 146 observations  
## mean=90.37671, MSE=687.0841   
##   
## Node number 11: 416 observations  
## mean=103.2091, MSE=884.1366   
##   
## Node number 12: 495 observations, complexity param=0.004824224  
## mean=97.02222, MSE=1260.83   
## left son=24 (257 obs) right son=25 (238 obs)  
## Primary splits:  
## RIDAGEYR < 1.177549 to the right, improve=0.03593447, (0 missing)  
## AIALANGA\_NA < 0.7694033 to the right, improve=0.03260307, (0 missing)  
## BMXLEG < 1.59693 to the left, improve=0.02390759, (0 missing)  
## BMXBMI < -0.2837618 to the right, improve=0.02065733, (0 missing)  
## BMXARMC < 1.318379 to the right, improve=0.01047926, (0 missing)  
## Surrogate splits:  
## AIALANGA\_NA < 0.7694033 to the right, agree=0.970, adj=0.937, (0 split)  
## RIDRETH3\_White < 0.3645385 to the right, agree=0.586, adj=0.139, (0 split)  
## BMXARMC < 0.2268366 to the left, agree=0.578, adj=0.122, (0 split)  
## RIDRETH3\_Black < 0.5542921 to the left, agree=0.570, adj=0.105, (0 split)  
## AIALANGA\_English < -2.469158 to the right, agree=0.570, adj=0.105, (0 split)  
##   
## Node number 13: 473 observations, complexity param=0.00377463  
## mean=108.7146, MSE=1509.045   
## left son=26 (156 obs) right son=27 (317 obs)  
## Primary splits:  
## BMXBMI < 0.3964963 to the right, improve=0.02458421, (0 missing)  
## INDFMPIR < -0.05039062 to the left, improve=0.02189095, (0 missing)  
## BMXARMC < 1.19312 to the right, improve=0.01510029, (0 missing)  
## BMXLEG < -2.520729 to the right, improve=0.01028420, (0 missing)  
## BMXARML < 0.6187648 to the right, improve=0.01009494, (0 missing)  
## Surrogate splits:  
## BMXARMC < 0.2304154 to the right, agree=0.854, adj=0.558, (0 split)  
## BMXARML < 0.7226261 to the right, agree=0.706, adj=0.109, (0 split)  
## INDFMPIR < -1.308407 to the left, agree=0.672, adj=0.006, (0 split)  
## BMXHT < 0.6610774 to the right, agree=0.672, adj=0.006, (0 split)  
##   
## Node number 14: 464 observations, complexity param=0.004972343  
## mean=109.2349, MSE=1026.456   
## left son=28 (91 obs) right son=29 (373 obs)  
## Primary splits:  
## BMXBMI < -0.6839137 to the left, improve=0.04853424, (0 missing)  
## RIAGENDR\_Male < 0.02868757 to the left, improve=0.03776510, (0 missing)  
## BMXARMC < -0.6499762 to the left, improve=0.03739263, (0 missing)  
## BMXLEG < 0.03326233 to the left, improve=0.01273680, (0 missing)  
## DMDBORN4\_Oth < 0.5241397 to the left, improve=0.01123883, (0 missing)  
## Surrogate splits:  
## BMXARMC < -0.8110234 to the left, agree=0.894, adj=0.462, (0 split)  
##   
## Node number 15: 1166 observations, complexity param=0.00585521  
## mean=117.6878, MSE=1326.074   
## left son=30 (265 obs) right son=31 (901 obs)  
## Primary splits:  
## BMXBMI < 0.7230719 to the right, improve=0.019870600, (0 missing)  
## RIDAGEYR < 0.6915823 to the right, improve=0.007605915, (0 missing)  
## BMXARMC < 1.085755 to the right, improve=0.006445164, (0 missing)  
## BMXLEG < -1.843139 to the right, improve=0.005717149, (0 missing)  
## BMXARML < 0.5841444 to the right, improve=0.004316487, (0 missing)  
## Surrogate splits:  
## BMXARMC < 1.085755 to the right, agree=0.886, adj=0.498, (0 split)  
## BMXLEG < -2.416484 to the left, agree=0.776, adj=0.015, (0 split)  
## BMXARML < 2.869092 to the right, agree=0.774, adj=0.004, (0 split)  
##   
## Node number 24: 257 observations  
## mean=90.54475, MSE=899.7889   
##   
## Node number 25: 238 observations, complexity param=0.003021559  
## mean=104.0168, MSE=1556.462   
## left son=50 (228 obs) right son=51 (10 obs)  
## Primary splits:  
## BMXLEG < 1.570869 to the left, improve=0.03791935, (0 missing)  
## DMDEDUC2\_9-11 < 1.224965 to the left, improve=0.03211565, (0 missing)  
## BMXBMI < -0.3483024 to the right, improve=0.02225825, (0 missing)  
## BMXHT < 1.478165 to the left, improve=0.02098695, (0 missing)  
## DMDMARTZ\_Nev < 0.7669317 to the right, improve=0.01875647, (0 missing)  
## Surrogate splits:  
## BMXHT < 2.235466 to the left, agree=0.966, adj=0.2, (0 split)  
##   
## Node number 26: 156 observations  
## mean=100.0321, MSE=1199.569   
##   
## Node number 27: 317 observations  
## mean=112.9874, MSE=1605.987   
##   
## Node number 28: 91 observations  
## mean=94.94505, MSE=776.6014   
##   
## Node number 29: 373 observations, complexity param=0.003384505  
## mean=112.7212, MSE=1025.44   
## left son=58 (215 obs) right son=59 (158 obs)  
## Primary splits:  
## RIAGENDR\_Male < 0.02868757 to the left, improve=0.04113595, (0 missing)  
## BMXLEG < 0.03326233 to the left, improve=0.01961291, (0 missing)  
## BMXBMI < 2.071971 to the right, improve=0.01668855, (0 missing)  
## DMDBORN4\_Oth < 0.5241397 to the left, improve=0.01517982, (0 missing)  
## BMXHT < -0.0165078 to the left, improve=0.01417552, (0 missing)  
## Surrogate splits:  
## BMXHT < 0.5016456 to the left, agree=0.831, adj=0.601, (0 split)  
## BMXARML < 0.2033197 to the left, agree=0.769, adj=0.456, (0 split)  
## BMXLEG < 0.2938737 to the left, agree=0.753, adj=0.418, (0 split)  
## INDFMPIR < -0.1527586 to the left, agree=0.619, adj=0.101, (0 split)  
## BMXARMC < -0.0952579 to the left, agree=0.590, adj=0.032, (0 split)  
##   
## Node number 30: 265 observations  
## mean=108.2226, MSE=1089.109   
##   
## Node number 31: 901 observations, complexity param=0.004142728  
## mean=120.4717, MSE=1361.67   
## left son=62 (867 obs) right son=63 (34 obs)  
## Primary splits:  
## BMXLEG < -1.843139 to the right, improve=0.014884930, (0 missing)  
## BMXARMC < 1.211014 to the left, improve=0.009461288, (0 missing)  
## RIDRETH3\_Black < 0.5542921 to the right, improve=0.008043446, (0 missing)  
## INDFMPIR < -0.7231829 to the left, improve=0.007061246, (0 missing)  
## BMXBMI < -0.2708537 to the left, improve=0.006776726, (0 missing)  
## Surrogate splits:  
## BMXHT < -1.954601 to the right, agree=0.971, adj=0.235, (0 split)  
## BMXARML < -2.323971 to the right, agree=0.966, adj=0.088, (0 split)  
##   
## Node number 50: 228 observations  
## mean=102.4079, MSE=1440.917   
##   
## Node number 51: 10 observations  
## mean=140.7, MSE=2786.21   
##   
## Node number 58: 215 observations  
## mean=107.1535, MSE=854.0927   
##   
## Node number 59: 158 observations  
## mean=120.2975, MSE=1159.019   
##   
## Node number 62: 867 observations, complexity param=0.00359949  
## mean=119.5802, MSE=1272.986   
## left son=124 (210 obs) right son=125 (657 obs)  
## Primary splits:  
## INDFMPIR < -0.7743669 to the left, improve=0.015161550, (0 missing)  
## BMXARMC < 1.211014 to the left, improve=0.011285510, (0 missing)  
## BMXBMI < -0.2708537 to the left, improve=0.008664953, (0 missing)  
## RIDAGEYR < 0.6915823 to the right, improve=0.006674608, (0 missing)  
## RIDRETH3\_Black < 0.5542921 to the right, improve=0.006344337, (0 missing)  
## Surrogate splits:  
## BMXARMC < -2.036772 to the left, agree=0.760, adj=0.010, (0 split)  
## BMXBMI < -1.548758 to the left, agree=0.759, adj=0.005, (0 split)  
##   
## Node number 63: 34 observations, complexity param=0.004142728  
## mean=143.2059, MSE=3085.987   
## left son=126 (27 obs) right son=127 (7 obs)  
## Primary splits:  
## BMXARML < -1.164187 to the left, improve=0.19305520, (0 missing)  
## BMXHT < -1.112601 to the left, improve=0.16323750, (0 missing)  
## INDFMPIR < -0.8033506 to the right, improve=0.11564990, (0 missing)  
## RIDAGEYR < 0.1084221 to the left, improve=0.07737871, (0 missing)  
## DMDEDUC2\_AA < 0.3718316 to the right, improve=0.07447832, (0 missing)  
## Surrogate splits:  
## BMXHT < -1.022921 to the left, agree=0.912, adj=0.571, (0 split)  
## BMXBMI < 0.4971797 to the left, agree=0.824, adj=0.143, (0 split)  
##   
## Node number 124: 210 observations  
## mean=111.8095, MSE=1186.621   
##   
## Node number 125: 657 observations  
## mean=122.0639, MSE=1275.122   
##   
## Node number 126: 27 observations  
## mean=130.7778, MSE=1924.099   
##   
## Node number 127: 7 observations  
## mean=191.1429, MSE=4673.837

rpart.plot(Chol\_rpart)



# tune and predict  
Chol\_rpart\_tune <- train(x = Chol\_tr\_X\_imp\_fin, y = Chol\_train\_y, method = "rpart", cp = 0.003)  
Chol\_rpart\_pred <- predict(Chol\_rpart\_tune, Chol\_test\_X\_imp)  
postResample(pred = Chol\_rpart\_pred, obs = Chol\_test\_y)

## RMSE Rsquared MAE   
## 33.0113573 0.0659519 25.8550917

# Rsquared value of 0.066 isn't too great. RMSE of 33.0. Let's compare to other trees

### Tree with CART based splits (rpart2 to tune over max depth)

set.seed(123)  
Chol\_rpart2\_tune <- train(x = Chol\_tr\_X\_imp\_fin, y = Chol\_train\_y, method = "rpart2", cp = 0.003)  
Chol\_rpart2\_pred <- predict(Chol\_rpart2\_tune, Chol\_test\_X\_imp)  
postResample(pred = Chol\_rpart2\_pred, obs = Chol\_test\_y)

## RMSE Rsquared MAE   
## 32.93397878 0.07117066 25.65328561

# minor improvement? Rsquared value of 0.071 and RMSE of 32.9

## Bagged Trees

set.seed(123)  
Chol\_bagtree <- train(x=Chol\_tr\_X\_imp\_fin, y = Chol\_train\_y, method = "treebag", nbagg = 70, trControl = Chol\_control)  
Chol\_bagtree

## Bagged CART   
##   
## 3696 samples  
## 32 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 51, 50, 48, 50, 46, 46, ...   
## Resampling results:  
##   
## RMSE Rsquared MAE   
## 35.31841 0.03281359 27.35183

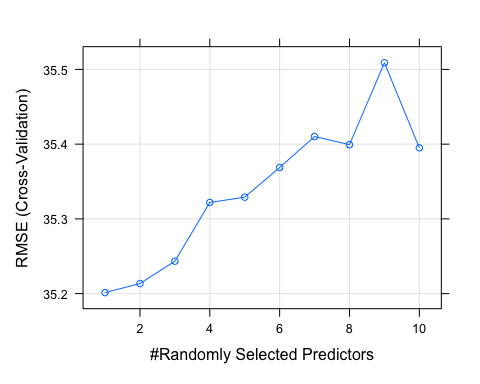
# Rsquared value of 0.033. RMSE is 35.3. Still not fantastic

## Random Forest

set.seed(123)  
  
rfmtryValues <- seq(1,10,1)  
  
Chol\_rf <- train(x = Chol\_tr\_X\_imp\_fin, y = Chol\_train\_y, method = "rf", ntree = 300, tuneGrid = data.frame(mtry=rfmtryValues), trControl=Chol\_control) # 300 trees provides the highest Rsquared value when checking with test data  
Chol\_rf

## Random Forest   
##   
## 3696 samples  
## 32 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 51, 50, 48, 50, 46, 46, ...   
## Resampling results across tuning parameters:  
##   
## mtry RMSE Rsquared MAE   
## 1 35.20133 0.02518717 27.50798  
## 2 35.21350 0.02576961 27.47209  
## 3 35.24339 0.02768960 27.48311  
## 4 35.32197 0.02699749 27.51524  
## 5 35.32907 0.02816671 27.51454  
## 6 35.36871 0.02846881 27.52456  
## 7 35.41023 0.02775607 27.55243  
## 8 35.39932 0.02857164 27.53620  
## 9 35.50890 0.02651411 27.61687  
## 10 35.39513 0.03059209 27.53761  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final value used for the model was mtry = 1.

plot(Chol\_rf)



Chol\_rf\_pred <- predict(Chol\_rf, Chol\_test\_X\_imp)  
postResample(pred = Chol\_rf\_pred, obs = Chol\_test\_y)

## RMSE Rsquared MAE   
## 33.06977523 0.07717907 26.17119444

# Rsquared value of 0.077. RMSE of 33.1

## Boosted Trees

# some control parameters  
gbmGrid <- expand.grid(interaction.depth = c(1,3,5,7,9), n.trees=300, shrinkage = c(0.01, 0.1), n.minobsinnode=5)  
  
set.seed(123)  
Chol\_gbm <- train(x = Chol\_tr\_X\_imp\_fin, y = Chol\_train\_y, method = "gbm", tuneGrid = gbmGrid, verbose = FALSE, trControl = Chol\_control)  
Chol\_gbm

## Stochastic Gradient Boosting   
##   
## 3696 samples  
## 32 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 51, 50, 48, 50, 46, 46, ...   
## Resampling results across tuning parameters:  
##   
## shrinkage interaction.depth RMSE Rsquared MAE   
## 0.01 1 35.11055 0.036008596 27.22850  
## 0.01 3 35.48463 0.027497218 27.51374  
## 0.01 5 35.49972 0.027023116 27.51478  
## 0.01 7 35.52798 0.026970125 27.54038  
## 0.01 9 35.49190 0.027126226 27.52603  
## 0.10 1 37.54685 0.018575218 29.19602  
## 0.10 3 38.45922 0.011161826 29.96295  
## 0.10 5 38.40583 0.012043967 29.92585  
## 0.10 7 38.65801 0.009083163 30.15651  
## 0.10 9 38.57613 0.009973257 30.08218  
##   
## Tuning parameter 'n.trees' was held constant at a value of 300  
## Tuning  
## parameter 'n.minobsinnode' was held constant at a value of 5  
## RMSE was used to select the optimal model using the smallest value.  
## The final values used for the model were n.trees = 300, interaction.depth =  
## 1, shrinkage = 0.01 and n.minobsinnode = 5.

Chol\_gbm\_pred <- predict(Chol\_gbm, Chol\_test\_X\_imp)  
postResample(pred = Chol\_gbm\_pred, obs = Chol\_test\_y)

## RMSE Rsquared MAE   
## 32.86446826 0.08137564 25.82653202

# RMSE 32.9 and Rsquared 0.081

## Model Trees

### Model Trees (M5)

# decision tree with linear regression at terminal nodes to predict continuous variables  
set.seed(123)  
Chol\_M5 <- train(x = Chol\_tr\_X\_imp\_fin, y = Chol\_train\_y, method = "M5", trControl = Chol\_control, control = Weka\_control(M=10))  
  
plot(Chol\_M5)



Chol\_M5\_pred <- predict(Chol\_M5, Chol\_test\_X\_imp)  
postResample(pred = Chol\_M5\_pred, obs = Chol\_test\_y)

## RMSE Rsquared MAE   
## 33.61012224 0.07641278 26.16774336

# Rsquared value of 0.076 and RMSE of 33.6

### Model Tree (Rule Based M5)

# decision tree with linear regression at terminal nodes to predict continuous variables  
set.seed(123)  
Chol\_M5rules <- train(x = Chol\_tr\_X\_imp\_fin, y = Chol\_train\_y, method = "M5Rules", trControl = Chol\_control, control = Weka\_control(M=10))  
  
plot(Chol\_M5rules)



Chol\_M5rules\_pred <- predict(Chol\_M5rules, Chol\_test\_X\_imp)  
postResample(pred = Chol\_M5rules\_pred, obs = Chol\_test\_y)

## RMSE Rsquared MAE   
## 32.75281564 0.08668215 25.26610200

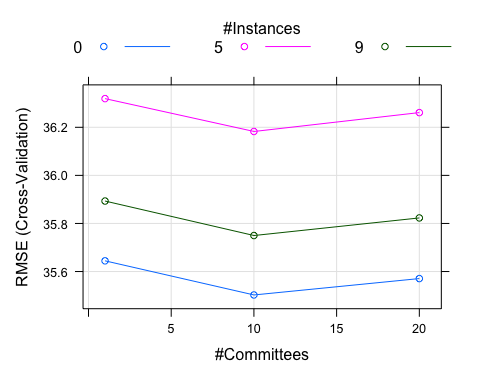
# Slight improvement, Rsquared value of 0.087 and RMSE of 32.8

## Cubist

set.seed(123)  
Chol\_cube <- train(x = Chol\_tr\_X\_imp\_fin, y = Chol\_train\_y, method = "cubist", trControl = Chol\_control)

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

plot(Chol\_cube)



Chol\_cube\_pred <- predict(Chol\_cube, Chol\_test\_X\_imp)  
postResample(pred = Chol\_cube\_pred, obs = Chol\_test\_y)

## RMSE Rsquared MAE   
## 32.63815970 0.09218122 25.11596950

# "Best performing" so far, but ever so slightly over the other models. Rsquared of 0.092 and RMSE of 32.6

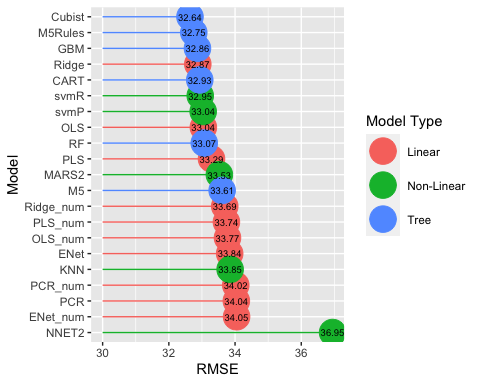
# Results

### Create Data set with all of the models

NonLpred <- NonLpred %>% select(!obs)  
Trees <- cbind.data.frame(Cubist = Chol\_cube\_pred, GBM = Chol\_gbm\_pred, M5 = Chol\_M5\_pred, M5Rules = Chol\_M5rules\_pred, RF = Chol\_rf\_pred, CART = Chol\_rpart2\_pred)  
  
Results <- cbind(Linear\_res, NonLpred, Trees)

### Get RMSE and Plot

find\_rmse <- function(x){  
 caret::RMSE(x, Results[,"Observed"])  
}  
  
RMSE\_results <- apply(X = Results[,2:22], FUN = find\_rmse, MARGIN = 2)  
RMSE\_results <- data.frame(RMSE\_results)  
RMSE\_results$Model <- rownames(RMSE\_results)  
RMSE\_results$Model\_Type <- "Linear"  
RMSE\_results$Model\_Type[11:15] <- "Non-Linear"  
RMSE\_results$Model\_Type[16:21] <- "Tree"  
  
ggplot(RMSE\_results, aes(x=reorder(Model, -RMSE\_results), y=RMSE\_results)) + geom\_segment(aes(x=reorder(Model, -RMSE\_results), xend = reorder(Model, -RMSE\_results), y=30, yend=RMSE\_results, color = Model\_Type)) + geom\_point(aes(color=Model\_Type), size = 9) + coord\_flip() + ylab("RMSE") + geom\_text(aes(label = round(RMSE\_results, 2)), color = "black", size = 2.5) + labs(color = "Model Type") + xlab("Model")



### Plot best model (cubist) against the original data

cubist\_plot <- Results %>% select(Observed, Cubist)  
cubist\_plot$x <- c(1:921)  
cubist\_plot <- cubist\_plot %>% gather(key = "Data Source", value = "LDL", -x)  
# cubist\_plot$alpha <- ifelse(cubist\_plot$Observed == "Observed", 0.8, 1)  
ggplot(cubist\_plot, aes(x = x, y = LDL)) + geom\_line(aes(color = `Data Source`, alpha = `Data Source`)) + scale\_color\_manual(values = c("royalblue1", "royalblue4")) + scale\_alpha\_manual(values = c(1,.3)) + theme\_classic()

