NHANES Prediction

Machine Learning Project

Sarah Salter

3/20/2018

I. Introduction

The National Health and Nutrition Examination Survey (NHANES) is a cross-sectional, nationally representative survey that assesses demographic, dietary and health-related questions and can be used to better understand differences in health and nutrition across the life-span. The goal of this analysis is to predict 9-year survival status (for those of age 50 years and older) and the age for all NHANES participants of the 2003-2004 survey. This dataset consists of 10122 participants and 813 variables related to patient demographics, dietary characteristics, body measurements, health status, etc. There are a total of 5 survey sections (Demographics, Dietary, Laboratory, Examination, Questionnaire), each containing at least one dataset [1]. A majority of the variables from these individual datasets are used within this data; however, not all are used.

It should be noted that each collected variable within the data had a corresponding 'target population' in regards to age. Whether a person was in that prespecified age range determined if they had their data collected corresponding to that variable. For instance, 'BMXHEAD' represents head circumference (cm) measurements for females and males ages 0-6 months only. All people that did not met the 0-6 month target age criteria were given a missing 'BMXHEAD' value.

One of the problems with this dataset is that there is a lot of informative missingness based on the variable specific target population. For some variables both informative missingness, due to a patient not being in the variable's target age range, and missingness cauesd by other reasons not related to the target age range (i.e- a question/measurement not being completed by mistake or refusal) were combined. Due to the fact that there is a lot of informative missingness, we must acknowledge that the data is not missing at random. Since dealing with informative missingness is beyond the scope of this course, I choose to proceed under the assumption that the data was missing at random; thus, the results will be biased and this should be taken into consideration when assessing each prediction performance. In an attempt to reduce the amount of bias, I chose to work mostly with variables that had the least amount of missing data and the widest target population age ranges. However, for certain categorical variables that I thought were exceptionally important for prediction, I recoded the missingness to its own level. For all other variables, I worked with complete data only and evaluated several combinations of variables to ensure the biggest patient population possible.

II. Predicting Age

A. Exploratory Analysis

The exam ages within the NHANES dataset had a range from about 1 month - 84.8 years, with a median age at approximately 19 years old. A total of 692 participants had missing exam age, whom were not included in this analysis. A histogram and summary of age can be found in the appendix [2]. Since the goal is to predict exam age, I removed all patients that had this value missing. As a result, the new dataset contained 9430 patients and 813 predictors. Next, I did a brief evaluation of the data to determine if there were any other age variables in the dataset. I identified three predictors: 'RIDAGEMN', 'RIDAGEYR', and 'DMDHRAGE'. The variables 'RIDAGEMN' and 'RIDAGEYR' (which was coded in years) represent the age the individual was screened for participation in the survey. Most people were likely screened and examined within a similar timeframe, thus their ages would most likely be very similar if not identical. For this reason, I removed both of these age variables from the data set to in order to not have an unscientific prediction advantage. Next, I removed 'DMDHRAGE' variable which was defined as the age in years of the household reference person at the time of HH screening. Although not all people in the survey were reference persons (only one for each houseld), if the reference person participated in the survey than this age would be very similar if not identical to their exam age, depending on how duration of time between their screening and exam. After removing patients who had missing outcome and these three age predictors, the refined dataset contained contained 9430 patients and 810 predictors. This data was used as the starting point (throughout the prediction analysis for exam age) before performing variable and patient selection based on incomplete data.

B. Data

My initial step in the exploratory data analysis was to define some of the variables within the dataset. First, I looked through the individual datasets to get a sense of the predictors that were available. Then using apriori knowledge, I made a list of variables I thought would be good predictors of age, such as body measurements, medical conditions, dietary profiles, etc. Afterwards, I determined the names and characteristics of these variables within the dataset.

My second step was to evaluate all of the predictors with less than 25% missing data. For these predictors I calculated the correlation associated with each indivdual predictor and exam age (using only the combined complete data from exam age and the predictor of interest). For variables with the highest absolute correlation, I assessed the variable definition and the target population associated with this variable. In this process a few variables were identified that were essentially meaningless predictors, such as patient id 'SEQN' and if body measurement exam was completed 'BMDSTATS'. These variables were removed, the absolute value correlations were calculated again, and the top variables were definied that were not defined in the previous correlation assessment.

Next, using the complete data associated with the predictors that had less than 25% missingness, I used a lasso method and assessed the non-zero coefficient variables. I also conducted random forest/bagging and assessed the variable importance plot. For both of these techniques I assessed the variable definitions and the target population associated with the top variables (non-zero coefficient variables in the lasso technique and the top 20% of variables listed for the boosting technique).

Using these variable definitions, the correlation calculations, and the lasso and random forest/bagging results, I created several datasets with different variables. Since the NHANES dataset contained a large amount of informative missingness based on patients not meeting the age criteria for certain variables, I tried to hand select variables that targeted all patients 0-150 years of age and other variables with 'large' age ranges to reduce potential bias from missing not at random data. For variables that had more restricted age ranges, I tried techniques which seperately (1) left the missing as is (2) redefined the missingness as its own level for categorical variables that I identified as being potential good predictors. Afterwards, I analyzed the missingness patterns and characteristics related to each patient and various chosen variables.

Furthermore, I identified some categorical variables that did not have good distribution amongst the levels. As a result, I combined some levels when necessary. For each dataset, I factored variables that were categorical and left all other variables defined as numeric. For some variables that I thought could have temporal issues (e.g- 9-year morality since this occurs after age was assessed) or time issues (education for people <19, pregnancy, time living in the US), I evaluated a dataset without these variables. In each dataset, I evaluated only patients with complete data across all of the variables, where missingness was assessed in both ways as mentioned above (i.e- (1) missing was left as is or (2) recoded as its own level within the factored variable).

C. Finalized Patient Populations

In total I created 10 datasets, specifically 5 datasets with unique variable selection, each individually evaluated with both missingness approaches (and therefore different patient popualations). I performed several prediction techniques for each dataset. Using a validation approach, I trained each model using a random selection of 70% of the patients. Then I tested the model using the other 30% of the patients. I calculated a test error rate using my test data and assessed my prediction ability for each. Due to the fact I was only using a validation approach in the initial stages to assess dataset performances, I took into account the amount of predictors (relative the number of patients) to consider whether I was at great risk of potentially overfitting the models to the training data. I selected the datasets that had good (if not the best) prediction ability and a reasonable amount of predictors based on the dataset size. The datasets consisted of the following dimensions: (1) consisted of 7193 patients & 72 predictors (2) consisted of 7698 patients & 69 predictors (3) consisted of 7139 patients & 55 predictors. Further to the appendix to see the variables that were included in each dataset.

D. Methods

I performed ridge regression, the lasso, boosting, bagging, and random forest methods for all three data sets. Across all datasets the tree based methods outperformed the shrinkage methods, as would be expected. Specifically, the boosting and regreesion tree method yield the smallest mean square errors.

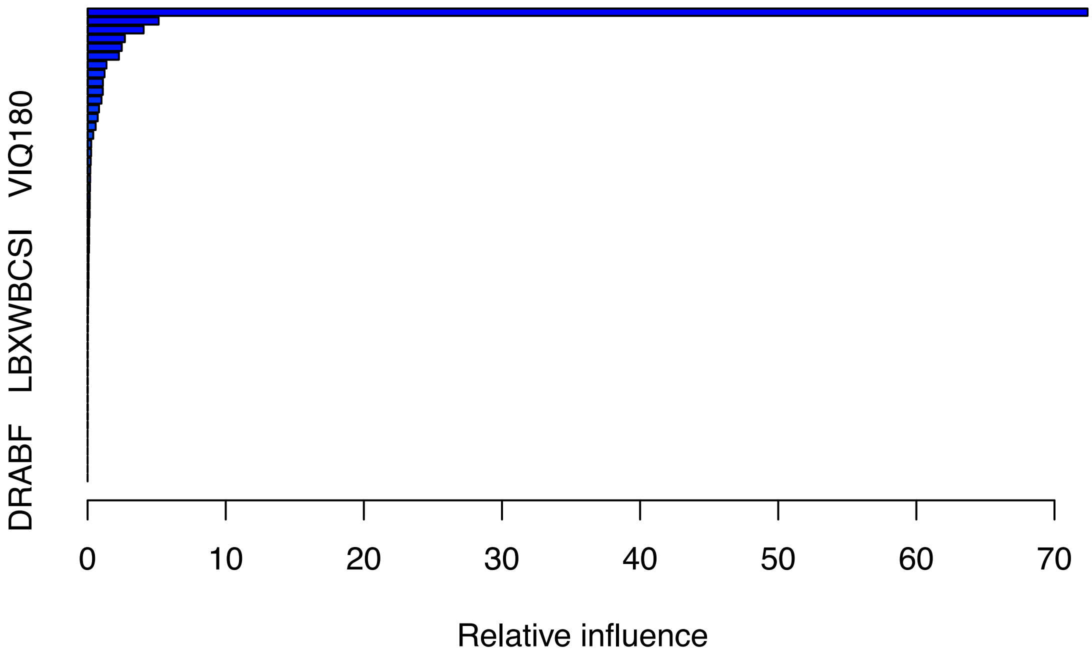
In addition to evaluating several different methods, I also evaluated different parameters within each method. For the shrinkage techniques, I let the result determine the appropriate lamba and gamma values to be used in the model. For the tree based methods, I evaluated different parameter value for each method. For boosting, a shrinkage parameter (lamba) of 0.2 had smaller MSEs than the default lambda of 0.01. Bagging performed better when 'mtry', the number of variables randomly sampled as candidates at each split, was increased from 15 to 18. All details regarding the methods can be found in the appendix.

E. Results

```
####################################
#Boosting
####################################
library(gbm)
```

```
## Loading required package: survival

## Loading required package: lattice

## Loading required package: splines

## Loading required package: parallel

## Loaded gbm 2.1.3
```

```
set.seed(1)
train = sample(1:nrow(complete_data7), round(nrow(complete_data7)*.70,0))
complete.test=complete_data7[-train ,"RIDAGEEX"]
boost.complete=gbm(RIDAGEEX~., data=complete_data7[train,],
                   distribution="gaussian", n.trees=5000, interaction.depth=4)
summary(boost.complete)
```




```
##                var      rel.inf
## BAQ110      BAQ110 7.239430e+01
## DMDMARTL DMDMARTL 5.141586e+00
## DMDEDUC2 DMDEDUC2 4.055232e+00
## MCQ160A    MCQ160A 2.696638e+00
## BPQ080      BPQ080 2.472152e+00
## WTMEC2YR WTMEC2YR 2.264375e+00
## VIQ200      VIQ200 1.363166e+00
## DMDHHSIZ DMDHHSIZ 1.224446e+00
## MCQ160B    MCQ160B 1.105935e+00
## BPQ040A    BPQ040A 1.101414e+00
## BMXHT        BMXHT 1.002820e+00
## MCQ160E    MCQ160E 8.239928e-01
## BPACSZ      BPACSZ 7.308963e-01
## WHQ030      WHQ030 5.803421e-01
## MCQ220      MCQ220 4.065777e-01
## VIQ180      VIQ180 2.614757e-01
## BPXPULS    BPXPULS 2.543025e-01
## PEASCTM1 PEASCTM1 2.249607e-01
## LBXMCVSI LBXMCVSI 1.927162e-01
## MCQ160G    MCQ160G 1.854764e-01
## BPQ020      BPQ020 1.709519e-01
## BMXWAIST BMXWAIST 1.563349e-01
## MCQ092      MCQ092 1.521335e-01
## BPXML1      BPXML1 1.492247e-01
## DR1TKCAL DR1TKCAL 1.166898e-01
## BMXARML    BMXARML 1.096073e-01
## MCQ160D    MCQ160D 1.081964e-01
## VIQ220      VIQ220 9.814768e-02
## DR1TSUGR DR1TSUGR 7.004912e-02
## BMXARMC    BMXARMC 6.934214e-02
## MCQ160C    MCQ160C 6.623720e-02
## DR1TCARB DR1TCARB 6.400086e-02
```

```
## DIQ010     DIQ010 4.248198e-02
## DR1TFIBE DR1TFIBE 3.595401e-02
## LBXWBCSI LBXWBCSI 2.082596e-02
## DR1TCAFF DR1TCAFF 1.643595e-02
## DR1TSFAT DR1TSFAT 1.339797e-02
## HSD010     HSD010 8.755418e-03
## DMDBORN   DMDBORN 8.429233e-03
## DR1TCHOL DR1TCHOL 6.203059e-03
## DR1TSODI DR1TSODI 5.117287e-03
## DR1TIRON DR1TIRON 4.807717e-03
## BMXBMI     BMXBMI 4.755276e-03
## DR1TPROT DR1TPROT 4.613053e-03
## DR1TTFAT DR1TTFAT 4.478910e-03
## RIAGENDR RIAGENDR 4.112659e-03
## DR1TALCO DR1TALCO 3.591463e-03
## MCQ160F   MCQ160F 1.721571e-03
## INDHHINC INDHHINC 5.943381e-04
## DMDCITZN DMDCITZN 0.000000e+00
## SIAPROXY SIAPROXY 0.000000e+00
## SIAINTRP SIAINTRP 0.000000e+00
## DIQ050     DIQ050 0.000000e+00
## DRABF       DRABF 0.000000e+00
```

```r
yhat.boost=predict(boost.complete, newdata= complete_data7[-train,], n.trees=5000)
mean((yhat.boost - complete.test)^2)
```

```
## [1] 4656.878
```

```r
set.seed(1)
boost.complete=gbm(RIDAGEEX~.,data=complete_data7[train ,], distribution= "gaussian", n.trees=5000,
                   interaction.depth=4, shrinkage =0.2, verbose=F)
yhat.boost=predict(boost.complete, newdata= complete_data7[-train,], n.trees=5000)
mean((yhat.boost - complete.test)^2)
```

```
## [1] 4604.231
```

```r
###################################
#Bagging/ Regression Tree
###################################
library(tree)
```

```
## Warning: package 'tree' was built under R version 3.4.4
```

```r
set.seed(1)
train = sample(1:nrow(complete_data7), nrow(complete_data7)/2)
library(randomForest)
```

```
## randomForest 4.6-12

## Type rfNews() to see new features/changes/bug fixes.
```

```r
set.seed(1)
bag.data = randomForest(RIDAGEEX~., complete_data7, subset=train, mtry=15, importance =TRUE)
yhat.bag = predict(bag.data , newdata=complete_data7[-train ,])
complete.test=complete_data7[-train ,"RIDAGEEX"]

set.seed(1)
rf.complete = randomForest(RIDAGEEX ~., data=complete_data7, subset=train, mtry=18)
```

```r
yhat.rf = predict(rf.complete , newdata = complete_data7[-train ,])
mean((yhat.rf - complete.test)^2)
```

```
## [1] 4070.722
```

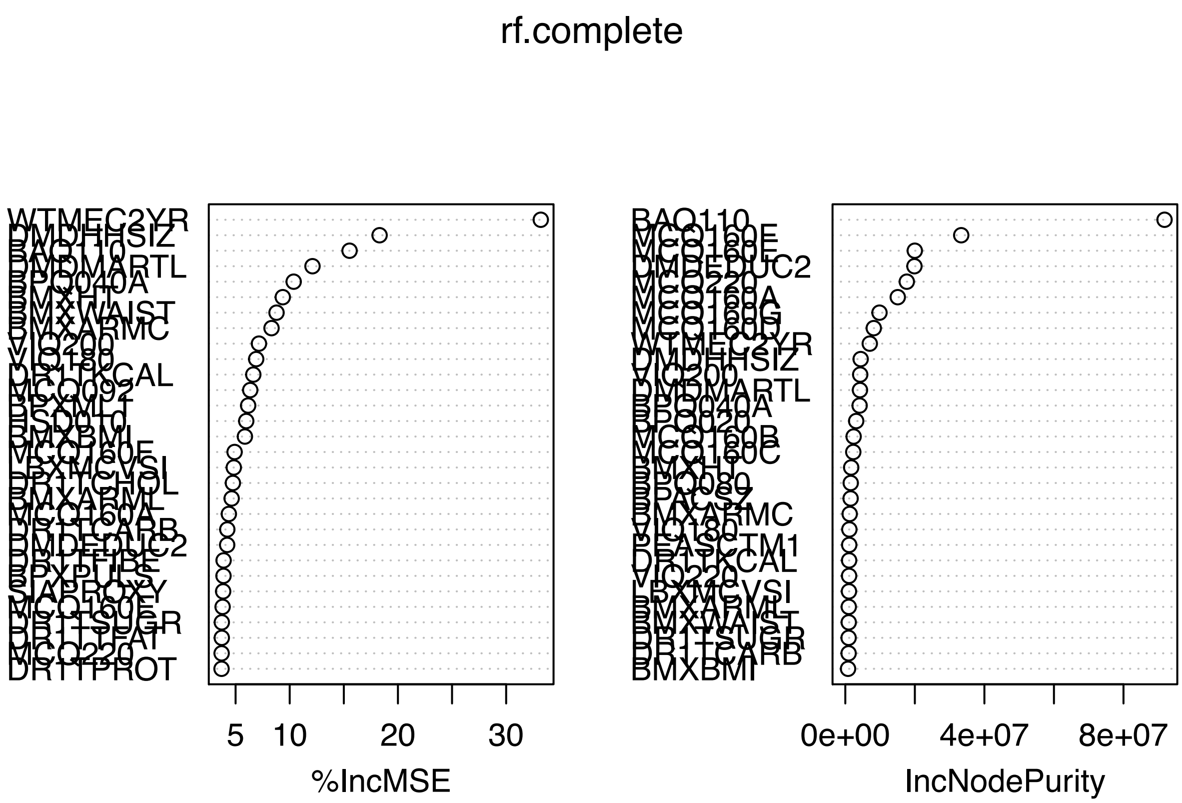
```r
set.seed(1)
rf.complete = randomForest(RIDAGEEX ~., data=complete_data7, subset=train, importance =TRUE, ntree=100)
yhat.rf = predict(rf.complete , newdata = complete_data7[-train ,])
mean((yhat.rf - complete.test)^2)
```

```
## [1] 4139.895
```

```r
mse = mean((yhat.rf - complete.test)^2)
head(importance(rf.complete))
```

```
##             %IncMSE IncNodePurity
## BAQ110    15.531965    91809992.2
## BMXHT      9.359169     1689301.6
## BMXBMI     5.858725      785703.5
## BMXARML    4.636248      990935.3
## BMXARMC    8.319988     1260717.2
## BMXWAIST   8.782981      989342.4
```

```r
head(varImpPlot(rf.complete))
```




```
##             %IncMSE IncNodePurity
## BAQ110    15.531965    91809992.2
## BMXHT      9.359169     1689301.6
## BMXBMI     5.858725      785703.5
## BMXARML    4.636248      990935.3
## BMXARMC    8.319988     1260717.2
```

```
## BMXWAIST  8.782981      989342.4
```

Although dataset 2 yield the smallest test error (MSE = 3178 from a bagging/random forest technique), I think that there could be some potential underestimation of this test error due to the fact there are a large amount of predictors in this dataset and a total of 133 levels (between the numeric and factored variables). For this reason, I think that results from dataset 3 are more trust worthy. In dataset 3 the smallest test MSE was calculated to be 4139.8951601 from a bagging/random forest technique.

Unfortunately, due to time constraints I only had time to evaluate my test error for each methods using a validation approach. A 10-fold cross validation should be performed to ensure consistency of the results that were found.

III. Predicting Mortality

A. Data

The second prediction we are interested is predicting patient mortality (using 9-year follow-up data), specifically within patients that are 50 years and older. As a result, first I removed all patients from the dataset that had missing values for mortality, leaving a total of 5610 patients. Next, I removed all patients that were younger than 50 years old, based on patient exam age, which resulted in final 'base' level dataset of 2132 patients and 813 predictors. It should be noted that exam age was used to determine if a patient was 50 years or old, as opposed to survey age, for consistency since exam age was used in the previous prediction.

Due to the fact that we are in a more restrictive age range, and the lower bound of target ages is between 20-50, there is not as much of a concern with informative missingness here. Although, it is still likely the data is not missing at random, I assumed that the was for reasons mentioned in the previous section.

Since mortality is a categorical variable I re-factored some variables that had many levels with a few number of data points and/or a small amount of cases. I also factored variables that were categorical predictors and made all other predictors numeric. Next, I conducted similar explortory techniques for mortality as were used for age, as well as similar variable selection methods in the creation of the 5 datasets used to evaluate mortality prediction. After creating my datasets, I created duplicate versions of each dataset and recoded the missing data as it's own level for each cateogorial variable, as done in the previous section.

B. Methods

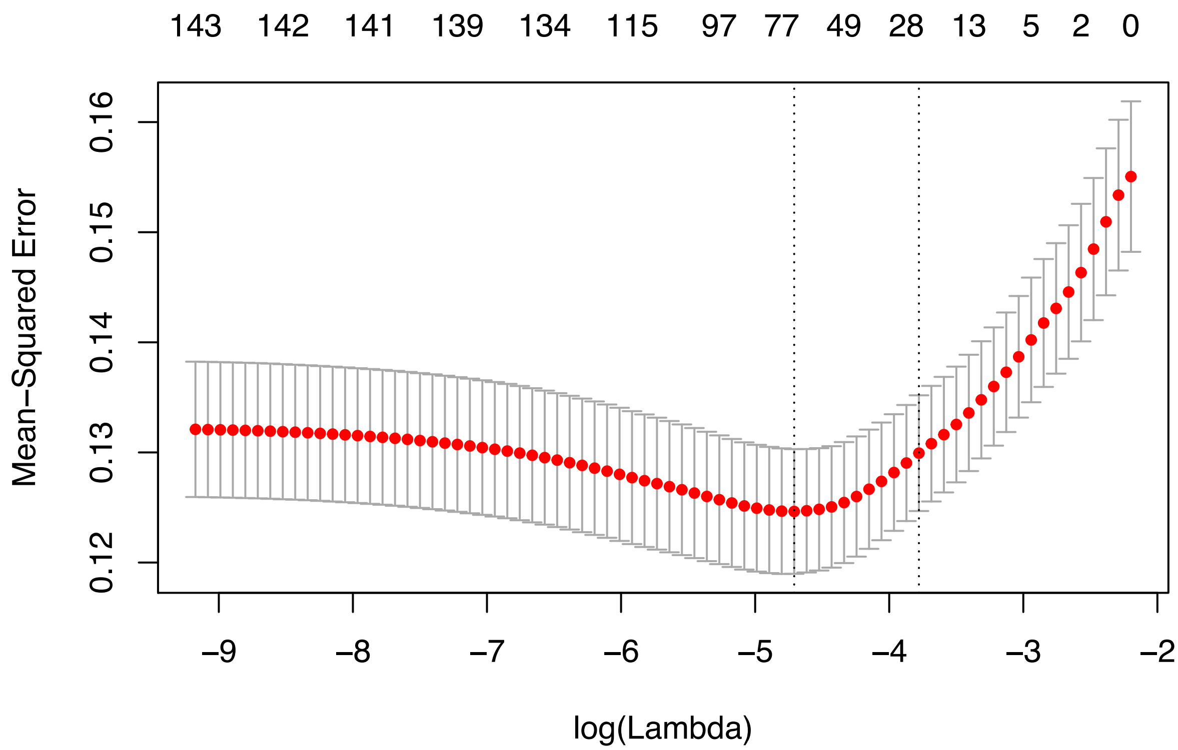
Two final datasets were chosen to assess patient mortality prediction. The dimensions of these datasets are: (1) 1523 patients & 64 predictors (2) 1781 patients & 25. Detailed information about these datasets can be found in the appendix. Lasso, Boosting, and a SMV were performed on all datasets, with different tuning parameters and kernels evaluated for the SVM.

E. Results

```r
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 3.4.2

## Loading required package: Matrix

## Loading required package: foreach

## Loaded glmnet 2.0-13
```

```r
x = model.matrix(mortstat ~ ., family = binomial(), data = new_data4)
y = new_data4$mortstat
y <- as.numeric(y)
grid = 10^seq(10, -2, length =100)
set.seed(1)
train = sample(1781, 1246)
test = (-train)
y.test = y[test]
lasso.mod = glmnet(x[train,], y[train], alpha = 1, lambda = grid)

cv.out = cv.glmnet(x[train,], y[train], alpha = 1)
plot(cv.out)
```




```r
bestlam = cv.out$lambda.min
lasso.pred = predict(lasso.mod, s = bestlam, newx = x[test,])
mean((lasso.pred - y.test)^2)
```

```
## [1] 0.138928
```

```r
out = glmnet(x, y, alpha = 1, lambda = grid)
lasso.coef = predict(out, type = "coefficients", s = bestlam)[1:43,]
lasso.coef[lasso.coef!=0]
```

```
##   (Intercept)       BAQ1102       BAQ1103     RIAGENDR2      RIDAGEEX
##  9.002580e-01  8.915141e-02  8.946703e-02 -6.649303e-02  6.662743e-04
##     DMQMILIT2     DMDMARTL2     DMDMARTL3     DMDMARTL5      DMDHHSIZ
## -2.512140e-02  4.224902e-02  4.480473e-02  5.564528e-02 -4.070466e-03
##      WTMEC2YR       DIQ0102       DIQ0502      DR1TSODI       HSD0102
## -8.524615e-06 -3.645735e-02 -4.070202e-02 -2.133170e-06  2.002525e-02
##       HSD0103       HSD0104  LBXWBCSI10.1  LBXWBCSI10.2  LBXWBCSI10.3
##  1.414235e-01  3.931380e-02 -1.172064e-01  3.250851e-01  9.790779e-02
```

```
## LBXWBCSI10.4  LBXWBCSI10.5  LBXWBCSI10.6  LBXWBCSI10.7    LBXWBCSI11
## 2.647731e-01  9.513987e-02  7.207326e-02 -1.147795e-01  3.146294e-02
## LBXWBCSI11.5
## 1.232138e-01
```

The second finalized dataset with the smallest about of predictors performed best. While radial kernal SVM slightly out outperformed linear kernal SVMs (and significantly out performed polynomial kernels), Lasso appeared to have the smallest misclassification test error of 0.1389, suggesting a 86.1% accurary rate. I was surprised by this result, as I would have anticipated SVMs having done a better job at prediction. Cross validation resuts should be further assessed to ensure that appropriate misclassification error rates are being compared acrossed models, so the best technique can be chosen. All details from analysis can be found in the appendix.

IV. Conclusions

As mentioned previously, these results must be preceived with some skepticism due to the data being missing at random, and since not all of methods had test errors created with a k-fold validation approach. Cross validation should be performed on all of the techniques evaluated once more, to ensure appropriate test errors are being compared across models, and also to ensure that the test errors are not being under-estimated.

In both predictions, the datasets with the smallest amount of predictors were chosen to make the final assessments. There is still a chance, that considerably over-fitting to the training data occurred, and more parismonious models should be looked into. Other prediction techniques, specifically more supervised learning techniques should be considered. due to the fact that some techniques like discriminany analysis or the KNN method may have been more appropriate for the data at hand. Another thing to consider is that no interaction terms, transformations of predictors, non-lienar predictors, were evaluated in any of these models. Evaluating the relationship of the outcome to each predictor, as well as assessing some intereaction plots, may have provided additional insight that could have help our prediction. Lastly, handling of missing data could have been most likely handled in a better way. Unforunately, all of these things were not possible based on time constraints; however, I look forward on exploring them in the future.

V. Appendix Predicting Age

[2]. Histogram/ Summary of Exam Age

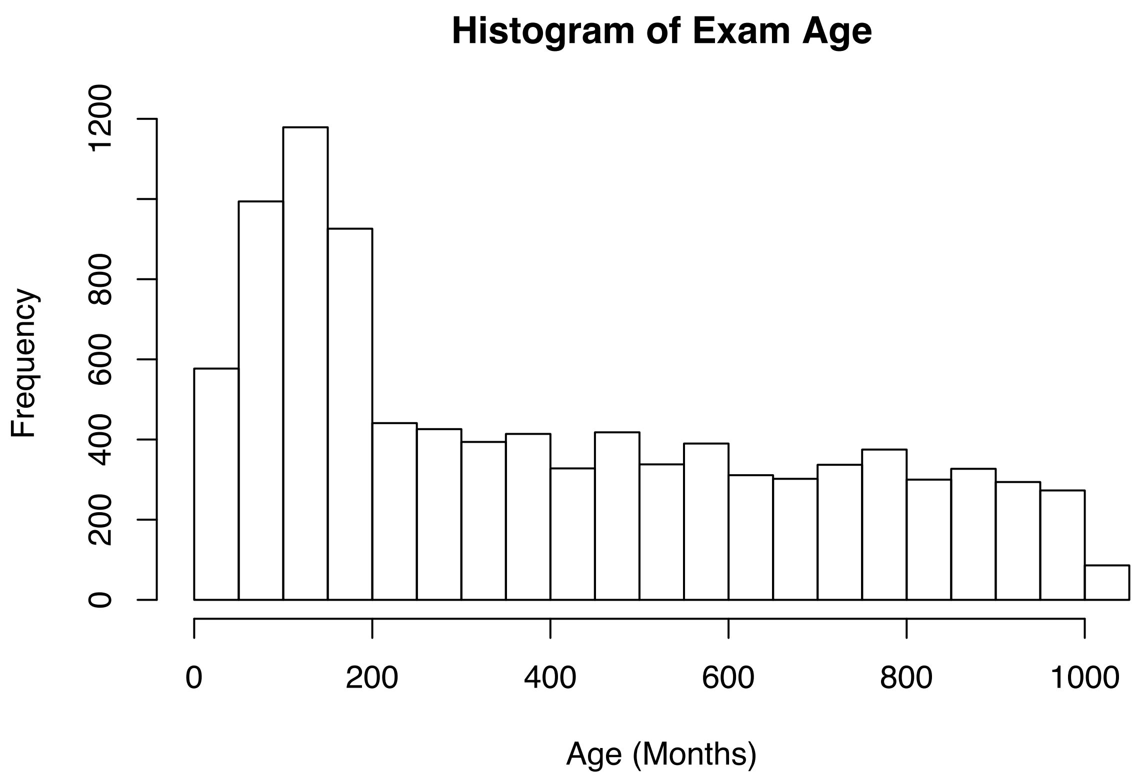
```r
load("/Users/sarahsalter/Downloads/nhanes2003-2004.Rda")
nhanes_data <- nhanes2003_2004
nhanes_data$RIDAGEEX <- as.numeric(nhanes_data$RIDAGEEX)
summary(nhanes_data$RIDAGEEX)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##     1.0   134.0   324.0   396.3   640.8  1018.0     692
```

```r
summary(nhanes_data$RIDAGEEX/12)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##  0.0833 11.1667 27.0000 33.0260 53.3958 84.8333     692
```

```r
hist(nhanes_data$RIDAGEEX, xlab="Age (Months)", main = "Histogram of Exam Age")
```




```
hist(nhanes_data$RIDAGEEX/12, xlab="Age (Yrs)", main = "Histogram of Exam Age")
```

[3a]. Variables Contained in Age Dataset 1

```
colnames(complete_data5)
```

```
##  [1] "BAQ110"   "BMXWT"    "BMXHT"    "BMXBMI"   "BMXARML"  "BMXARMC"
##  [7] "BMXWAIST" "BPQ020"   "BPQ040A"  "BPQ080"   "PEASCTM1" "BPACSZ"
## [13] "BPXPULS"  "BPXML1"   "RIAGENDR" "RIDAGEEX" "RIDRETH1" "DMDBORN"
## [19] "DMDCITZN" "DMDYRSUS" "DMDEDUC3" "DMDEDUC2" "DMDSCHOL" "DMDMARTL"
## [25] "DMDHHSIZ" "INDHHINC" "RIDEXPRG" "SIAPROXY" "SIAINTRP" "WTMEC2YR"
## [31] "DIQ010"   "DIQ050"   "DRABF"    "DR1TKCAL" "DR1TPROT" "DR1TCARB"
## [37] "DR1TSUGR" "DR1TFIBE" "DR1TTFAT" "DR1TSFAT" "DR1TCHOL" "DR1TATOC"
## [43] "DR1TVARA" "DR1TBCAR" "DR1TFA"   "DR1TVC"   "DR1TVK"   "DR1TCALC"
## [49] "DR1TMAGN" "DR1TIRON" "DR1TSODI" "DR1TPOTA" "DR1TCAFF" "DR1TALCO"
## [55] "HSD010"   "HSQ520"   "LBXWBCSI" "LBXMCVSI" "MCQ092"   "MCQ160A"
## [61] "MCQ160B"  "MCQ160C"  "MCQ160D"  "MCQ160E"  "MCQ160F"  "MCQ160G"
## [67] "MCQ220"   "VIQ180"   "VIQ200"   "VIQ220"   "WHQ030"   "WHQ070"
## [73] "WHQ090"
```

1. DIQ010: Doctor told you have diabetes (1-150 Years)
2. DIQ050: Taking insulin now (1-150 Years)
3. HSQ520: Flu, pneumonia, ear infection in past 30 days? (1-150 Years)
4. HSD010: General Health Condition (12-150 Years)
5. VIQ180: Eye surgery for near sightedness (12-150 Years)
6. VIQ200: Eye surgery for cataracts (12-150 Years)
7. VIQ220: Glasses/ contacts worn for distance; (12-150 Years)
8. DR1TKCAL: Energy (Calories) (0-150 Years)
9. DR1TPROT: Protein (0-150 Years)
10. DR1TCARB: Carbohydrate (0-150 Years)
11. DR1TSUGR: Total sugars (0-150 Years)
12. DR1TFIBE: Dietary Fiber (0-150 Years)
13. DR1TTFAT: Total Fat (0-150 Years)
14. DR1TCHOL: Cholesterol (0-150 Years)
15. DR1TATOC: Vitamin E (0-150 Years)
16. DR1TVARA: Vitamin A (0-150 Years)
17. DR1TBCAR: Beta-carotene (0-150 Years)
18. DR1TFA: Folic Acid (0-150 Years)
19. DR1TVC: Vitamin C (0-150 Years)
20. DR1TVK: Vitamin K (0-150 Years)
21. DR1TCALC: Calcium (0-150 Years)
22. DR1TMAGN: Magnesium (0-150 Years)
23. DR1TIRON: Iron (0-150 Years)
24. DR1TSODI: Sodium (0-150 Years)
25. DR1TPOTA: Potassium (0-150 Years)
26. DR1TCAFF: Caffeine (0-150 Years)
27. DR1TALCO: Alcohol (0-150 Years)
28. WHQ070: Tried to lose weight in past year (16-150 Years)
29. WHQ090: Tried not to gain weight in past year (16-150 Years)
30. BMXWT: Weight (0-150 Years)
31. BMXARML: Upper Arm Length (0-150 Years)
32. BMXARMC: Arm Circumference (0-150 Years)
33. BMXHT: Standing Height (2-150 Years)
34. BMXBMI: Body Mass Index (2-150 Years)
35. BMXWAIST: Waist Circumference (2-150 Years)
36. PEASCTM1: Blood Pressure Time in Seconds (0-150 Years)

```
37. BPACSZ: Coded cuff size; Recode missing (8-150 Years)
38. BPQ020: Ever told you had high blood pressure (16-150 Years)
39. DMDMARTL: Martial Status
40. DMDEDUC3: Education (6-19 Years)
41. DMDEDUC2: Eudcation (20-150 Years)
42. RIAGENDR: Gender (0-150 Years)
43. RIDRETH1: Race/Ethnicity (0-150 Years)
44. DMDBORN: Country of Birth (0-150 Years)
45. DMDCITZN: Citizenship Status (0-150 Years)
46. DMDYRSUS: Length of time in US (0-150 Years)
47. DMDSCHOL: Now attending school (6-19 Years)
48. DMDHHSIZ: Total number of people in the household (0-150 Years)
49. INDHHIC: Annual Household Income (0-150 Years)
50. RIDEXPRG: Pregnancy Status at Exam (8-59 Years)
51. SIAPROXY: Was a proxy used in SP interview
52. SIAINTRP: Was an interpreter used in SP interview
53. WTMEC2YR: Full Sample 2 Year MEC Exam Weight (0-150 Years)
54. DRABF: Breast-fed infant (either day) (0-150 Years)
55. BAQ110: Can you stand on your own? (40-150 Years)
56. BPXML1: Pulse Maximum Inflation Levels
57. LBXWBCSI: White blood cell count (1-150 Years)
58. LBXMCVSI: Mean cell volume (1-150 Years)
59. WHQ030: How do you consider your weight (16-150 Years)
60. MCQ160G: Ever told you had emphysema (20-150 Years)
61. BPQ040A: Taking prescription for hypertension
62. BPQ080: Doctor told you had high cholesterol (20-150 Years)
63. MCQ160A: Ever told you had arthritis (20-150 Years)
64. MCQ160B: Ever told you had congestive heart failure (20-150 Years)
65. MCQ160C: Ever told you had coronary heart disease (20-150 Years)
66. MCQ160D: Ever told you had angina (20-150 Years)
67. MCQ160E: Ever told you had heart attack; (20-150 Years)
68. MCQ160F: Ever told you had stroke; (20-150 Years)
69. MCQ220: Ever told you have cancer; (20-150 Years)
70. MCQ092: Ever receive blood transfusion; (6-150 Years)
71. DR1TSFAT: Total Saturated Fat (0-150 Years)
72. BPXPULS: Is pulse irregular?
73. RIDAGEEX: Patient age when exam was given
```

[3b]. Rational for Variables Contained in Age Dataset 1

[3c]. Refactoring/Coding Missingness

```
#1/3 Yes/Borderline-1; 2-No; Missing/Unkown-3
data5$DIQ010 <- ifelse(data5$DIQ010==1 | data5$DIQ010==3, 1, data5$DIQ010)
data5$DIQ010  <- replace(data5$DIQ010 , is.na(data5$DIQ010 ), 3)
data5$DIQ010 <- ifelse(data5$DIQ010==9, 3, data5$DIQ010)
#3-Missing
data5$DIQ050 <- replace(data5$DIQ050 , is.na(data5$DIQ050 ), 3)
#3-Refused/Don't Know/Missing
data5$HSQ520 <- ifelse(data5$HSQ520==7 | data5$HSQ520==9, 3, data5$HSQ520)
data5$HSQ520  <- replace(data5$HSQ520 , is.na(data5$HSQ520 ), 3)
#1-Excellent/Good; 2-Good/Fair; 3-Poor; 4-Refused/Don'tKnow/Missing
data5$HSD010 <- ifelse(data5$HSD010==1 | data5$HSD010==2, 1, data5$HSD010)
```

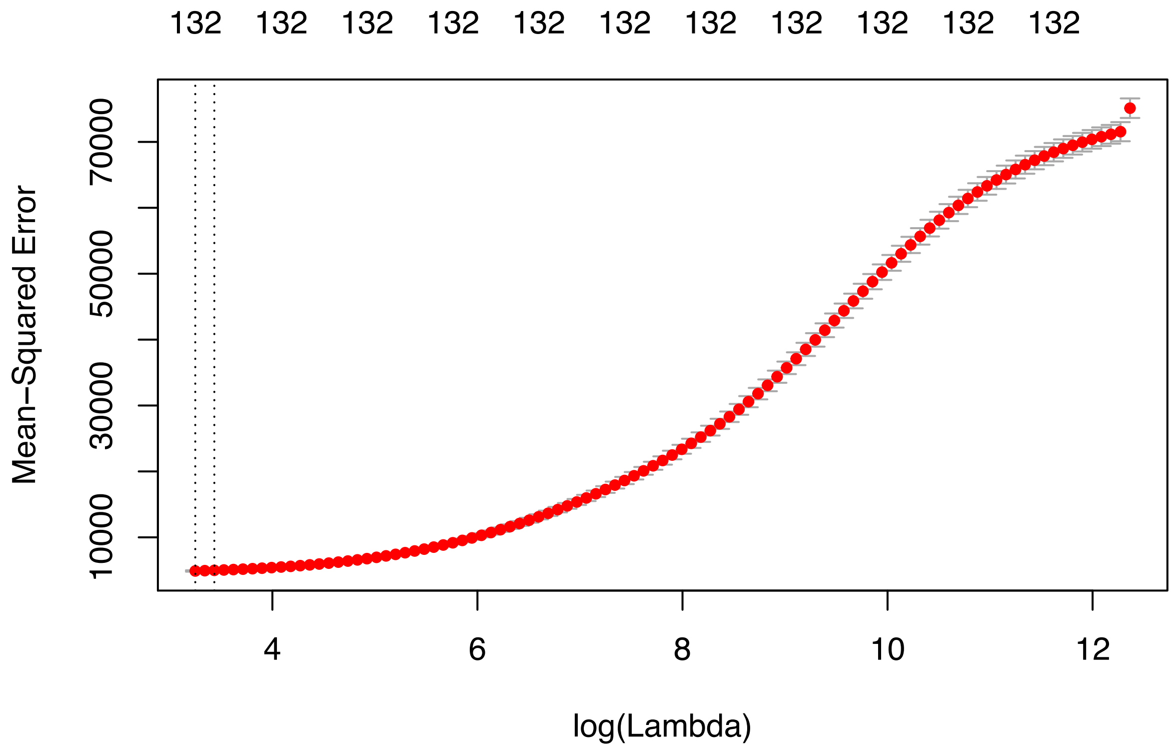
```
data5$HSD010 <- ifelse(data5$HSD010==3 | data5$HSD010==4, 2, data5$HSD010)
data5$HSD010 <- ifelse(data5$HSD010==5, 3, data5$HSD010)
data5$HSD010 <- ifelse(data5$HSD010==7 | data5$HSD010==9, 4, data5$HSD010)
data5$HSD010 <- replace(data5$HSD010, is.na(data5$HSD010), 4)
#3-Don'tKnow/Missing
data5$VIQ180 <- ifelse(data5$VIQ180==9, 3, data5$VIQ180)
data5$VIQ180  <- replace(data5$VIQ180 , is.na(data5$VIQ180 ), 3)
#3-Don'tKnow/Missing
data5$VIQ200 <- ifelse(data5$VIQ200==9, 3, data5$VIQ200)
data5$VIQ200  <- replace(data5$VIQ200 , is.na(data5$VIQ200 ), 3)
#3-Don'tKnow/Missing
data5$VIQ220 <- ifelse(data5$VIQ220==9, 3, data5$VIQ220)
data5$VIQ220  <- replace(data5$VIQ220 , is.na(data5$VIQ220 ), 3)
#3-Don'tKnow/Missing
data5$WHQ070 <- ifelse(data5$WHQ070==9, 3, data5$WHQ070)
data5$WHQ070  <- replace(data5$WHQ070 , is.na(data5$WHQ070 ), 3)
#3-Don'tKnow/Missing
data5$WHQ090 <- ifelse(data5$WHQ090==9, 3, data5$WHQ090)
data5$WHQ090  <- replace(data5$WHQ090 , is.na(data5$WHQ090 ), 3)
#6-Missing
data5$BPACSZ  <- replace(data5$WHQ090 , is.na(data5$WHQ090 ), 6)
#3-Don'tKnow/Missing
data5$BPQ020 <- ifelse(data5$BPQ020==9, 3, data5$BPQ020)
data5$BPQ020  <- replace(data5$BPQ020 , is.na(data5$BPQ020 ), 3)
#1-Married/Living with Partner; 2-Widowed; 3-Divorced/Separated; 4-Never Married; 5-Refused/Missing
data5$DMDMARTL <- ifelse(data5$DMDMARTL==1 | data5$DMDMARTL==6, 1, data5$DMDMARTL)
data5$DMDMARTL <- ifelse(data5$DMDMARTL==3 | data5$DMDMARTL==4, 3, data5$DMDMARTL)
data5$DMDMARTL <- ifelse(data5$DMDMARTL==5, 4, data5$DMDMARTL)
data5$DMDMARTL[ is.na(data5$DMDMARTL) ] <- 5
#1-Less than HS D; 2-HS D/AA D; 3- College Grad; 4-Missing/Unknown
data5$DMDEDUC2 <- ifelse(data5$DMDEDUC2==1 | data5$DMDEDUC2==2, 1, data5$DMDEDUC2)
data5$DMDEDUC2 <- ifelse(data5$DMDEDUC2==3 | data5$DMDEDUC2==4, 2, data5$DMDEDUC2)
data5$DMDEDUC2 <- ifelse(data5$DMDEDUC2==5, 3, data5$DMDEDUC2)
data5$DMDEDUC2 <- ifelse(data5$DMDEDUC2==7 | data5$DMDEDUC2==9, 4, data5$DMDEDUC2)
data5$DMDEDUC2[ is.na(data5$DMDEDUC2) ] <- 4
#16-Less than HS/ Less 9th/ Less 5th/Unknown/Missing
data5$DMDEDUC3 <- ifelse(data5$DMDEDUC3==55 | data5$DMDEDUC3==66 | data5$DMDEDUC3==77 | data5$DMDEDUC3=
data5$DMDEDUC3[ is.na(data5$DMDEDUC3) ] <- 16
#10-Refused/Missing
data5$DMDYRSUS <- ifelse(data5$DMDYRSUS==77 | data5$DMDYRSUS==88 | data5$DMDYRSUS==99, 10, data5$DMDYRS
data5$DMDYRSUS[ is.na(data5$DMDYRSUS) ] <- 10
#1-In School/Vacation; 2-No; 3-Unknown/Missing
data5$DMDSCHOL <- ifelse(data5$DMDSCHOL==1 | data5$DMDSCHOL==2, 1, data5$DMDSCHOL)
data5$DMDSCHOL <- ifelse(data5$DMDSCHOL==3, 2, data5$DMDSCHOL)
data5$DMDSCHOL <- ifelse(data5$DMDSCHOL==7 | data5$DMDSCHOL==9, 3, data5$DMDSCHOL)
data5$DMDSCHOL[ is.na(data5$DMDSCHOL) ] <- 3
#3-Don'tKnow/Missing
data5$BAQ110  <- replace(data5$BAQ110 , is.na(data5$BAQ110 ), 3)
#0-Below Median; 1-Above Median; 2-Missing
data5$BPXML1  <- replace(data5$BPXML1, is.na(data5$BPXML1 ), 2)
data5$BPXML1 <- ifelse(data5$BPXML1>=140 & data5$BPXML1!=2, 1 ,data5$BPXML1)
data5$BPXML1 <- ifelse(data5$BPXML1<140 & data5$BPXML1!=2 & data5$BPXML1!=1, 0 ,data5$BPXML1)
#4-Refused/Unkown/Missing
```

```
data5$WHQ030 <- ifelse(data5$WHQ030==7 | data5$WHQ030==9, 4, data5$WHQ030)
data5$WHQ030[ is.na(data5$WHQ030) ] <- 4
#3-Don'tKnown/Missing
data5$MCQ160G <- ifelse(data5$MCQ160G==9, 3, data5$MCQ160G)
data5$MCQ160G[ is.na(data5$MCQ160G) ] <- 3
#3-Don'tKnown/Missing
data5$BPQ040A <- ifelse(data5$BPQ040A==9, 3, data5$BPQ040A)
data5$BPQ040A[ is.na(data5$BPQ040A) ] <- 3
#3-Don'tKnown/Missing
data5$BPQ080 <- ifelse(data5$BPQ080==9, 3, data5$BPQ080)
data5$BPQ080[ is.na(data5$BPQ080) ] <- 3
#3-Don'tKnown/Missing
data5$MCQ160A <- ifelse(data5$MCQ160A==9, 3, data5$MCQ160A)
data5$MCQ160A[ is.na(data5$MCQ160A) ] <- 3
#3-Don'tKnown/Missing
data5$MCQ160B <- ifelse(data5$MCQ160B==9, 3, data5$MCQ160B)
data5$MCQ160B[ is.na(data5$MCQ160B) ] <- 3
#3-Don'tKnown/Missing
data5$MCQ160C <- ifelse(data5$MCQ160C==9, 3, data5$MCQ160C)
data5$MCQ160C[ is.na(data5$MCQ160C) ] <- 3
#3-Don'tKnown/Missing
data5$MCQ160D <- ifelse(data5$MCQ160D==9, 3, data5$MCQ160D)
data5$MCQ160D[ is.na(data5$MCQ160D) ] <- 3
#3-Don'tKnown/Missing/Refuse
data5$MCQ160E <- ifelse(data5$MCQ160E==7 | data5$MCQ160E==9, 3, data5$MCQ160E)
data5$MCQ160E[ is.na(data5$MCQ160E) ] <- 3
#3-Don'tKnown/Missing
data5$MCQ160F <- ifelse(data5$MCQ160F==9, 3, data5$MCQ160F)
data5$MCQ160F[ is.na(data5$MCQ160F) ] <- 3
#3-Don'tKnown/Missing
data5$MCQ220 <- ifelse(data5$MCQ220==9, 3, data5$MCQ220)
data5$MCQ220[ is.na(data5$MCQ220) ] <- 3
#3-Don'tKnown/Missing
data5$MCQ092 <- ifelse(data5$MCQ092==9, 3, data5$MCQ092)
data5$MCQ092[ is.na(data5$MCQ092) ] <- 3
#3-Don'tKnown/Missing
data5$DRABF[ is.na(data5$DRABF) ] <- 3
#3-Don'tKnown/Missing
data5$BPXPULS[ is.na(data5$BPXPULS) ] <- 3
#3-Don'tKnown/Missing
data5$DMDBORN <- ifelse(data5$DMDBORN==7,3,data5$DMDBORN)
#3-Don'tKnown/Missing
data5$DMDCITZN <- ifelse(data5$DMDCITZN==7,3,data5$DMDCITZN)
#10-Don'tKnown/Missing/Refused
data5$DMDYRSUS <- ifelse(data5$DMDYRSUS==77 | data5$DMDYRSUS==99 | data5$DMDYRSUS==88, 10, data5$DMDYRSU
data5$DMDYRSUS[ is.na(data5$DMDYRSUS) ] <- 10
#4-Missing/Unknown
data5$INDHHINC <- ifelse(data5$INDHHINC==2, 1, data5$INDHHINC)
data5$INDHHINC <- ifelse(data5$INDHHINC==3, 1, data5$INDHHINC)
data5$INDHHINC <- ifelse(data5$INDHHINC==4, 1, data5$INDHHINC)
data5$INDHHINC <- ifelse(data5$INDHHINC==13, 1, data5$INDHHINC)
data5$INDHHINC <- ifelse(data5$INDHHINC==5, 2, data5$INDHHINC)
data5$INDHHINC <- ifelse(data5$INDHHINC==6, 2, data5$INDHHINC)
```

```r
data5$INDHHINC <- ifelse(data5$INDHHINC==7, 2, data5$INDHHINC)
data5$INDHHINC <- ifelse(data5$INDHHINC==8, 2, data5$INDHHINC)
data5$INDHHINC <- ifelse(data5$INDHHINC==9, 2, data5$INDHHINC)
data5$INDHHINC <- ifelse(data5$INDHHINC==10, 2, data5$INDHHINC)
data5$INDHHINC <- ifelse(data5$INDHHINC==12, 2, data5$INDHHINC)
data5$INDHHINC <- ifelse(data5$INDHHINC==11, 3, data5$INDHHINC)
data5$INDHHINC <- ifelse(data5$INDHHINC==14, 4, data5$INDHHINC)
data5$INDHHINC <- ifelse(data5$INDHHINC==15, 4, data5$INDHHINC)
data5$INDHHINC <- ifelse(data5$INDHHINC==77 | data5$INDHHINC==99,4,data5$INDHHINC)
data5$INDHHINC[ is.na(data5$INDHHINC) ] <- 4
#3-Missing/Unkown
data5$RIDEXPRG[ is.na(data5$RIDEXPRG) ] <- 3
#Factor variables
data5$DIQ010 <-as.factor(data5$DIQ010); data5$DIQ050 <-as.factor(data5$DIQ050)
data5$HSQ520 <-as.factor(data5$HSQ520); data5$HSD010 <-as.factor(data5$HSD010)
data5$VIQ180 <-as.factor(data5$VIQ180); data5$VIQ200 <-as.factor(data5$VIQ200)
data5$VIQ220 <-as.factor(data5$VIQ220); data5$WHQ070 <-as.factor(data5$WHQ070)
data5$WHQ090 <-as.factor(data5$WHQ090); data5$BPACSZ <-as.factor(data5$BPACSZ)
data5$BPQ020 <-as.factor(data5$BPQ020); data5$DMDMARTL <-as.factor(data5$DMDMARTL)
data5$DMDEDUC2 <-as.factor(data5$DMDEDUC2); data5$DMDEDUC3 <-as.factor(data5$DMDEDUC3)
data5$RIDRETH1 <-as.factor(data5$RIDRETH1); data5$DMDBORN <-as.factor(data5$DMDBORN)
data5$DMDCITZN <-as.factor(data5$DMDCITZN); data5$DMDYRSUS <-as.factor(data5$DMDYRSUS)
data5$DMDSCHOL <-as.factor(data5$DMDSCHOL); data5$INDHHINC <-as.factor(data5$INDHHINC)
data5$DRABF <-as.factor(data5$DRABF); data5$BAQ110 <-as.factor(data5$BAQ110)
data5$MCQ160G <-as.factor(data5$MCQ160G); data5$BPQ040A <-as.factor(data5$BPQ040A)
data5$BPQ080 <-as.factor(data5$BPQ080 ); data5$MCQ160A <-as.factor(data5$MCQ160A)
data5$MCQ160B <-as.factor(data5$MCQ160B); data5$MCQ160C <-as.factor(data5$MCQ160C)
data5$MCQ160D <-as.factor(data5$MCQ160D); data5$MCQ160E <-as.factor(data5$MCQ160E)
data5$MCQ160F <-as.factor(data5$MCQ160F); data5$MCQ220 <-as.factor(data5$MCQ220)
data5$MCQ092 <-as.factor(data5$MCQ092); data5$BPXPULS <-as.factor(data5$BPXPULS)
data5$RIDEXPRG <- as.factor(data5$RIDEXPRG)
complete_data5 <- data5[complete.cases(data5),]
```

[3c]. Methods/Reults for Predicting Age in Age Dataset 1

```r
#complete_data5
###################################
#(1) Ridge Regression
###################################
library(glmnet)
x = model.matrix(RIDAGEEX~., data = complete_data5)[,-1]
y = complete_data5$RIDAGEEX
grid = 10^seq(10, -2, length =100)
ridge.mod = glmnet(x, y, alpha = 0, lambda = grid)

set.seed(1)
train = sample(1:nrow(x), round(nrow(x)*.70,0))
test = (-train)
y.test = y[test]
cv.out = cv.glmnet(x[train,], y[train], alpha = 0)
plot(cv.out)
```




```
bestlam = cv.out$lambda.min
bestlam
```

```
## [1] 25.7323
```

```
set.seed(1)
ridge.pred = predict(ridge.mod, s = bestlam, newx = x[test,])
mean((ridge.pred - y.test)^2)
```

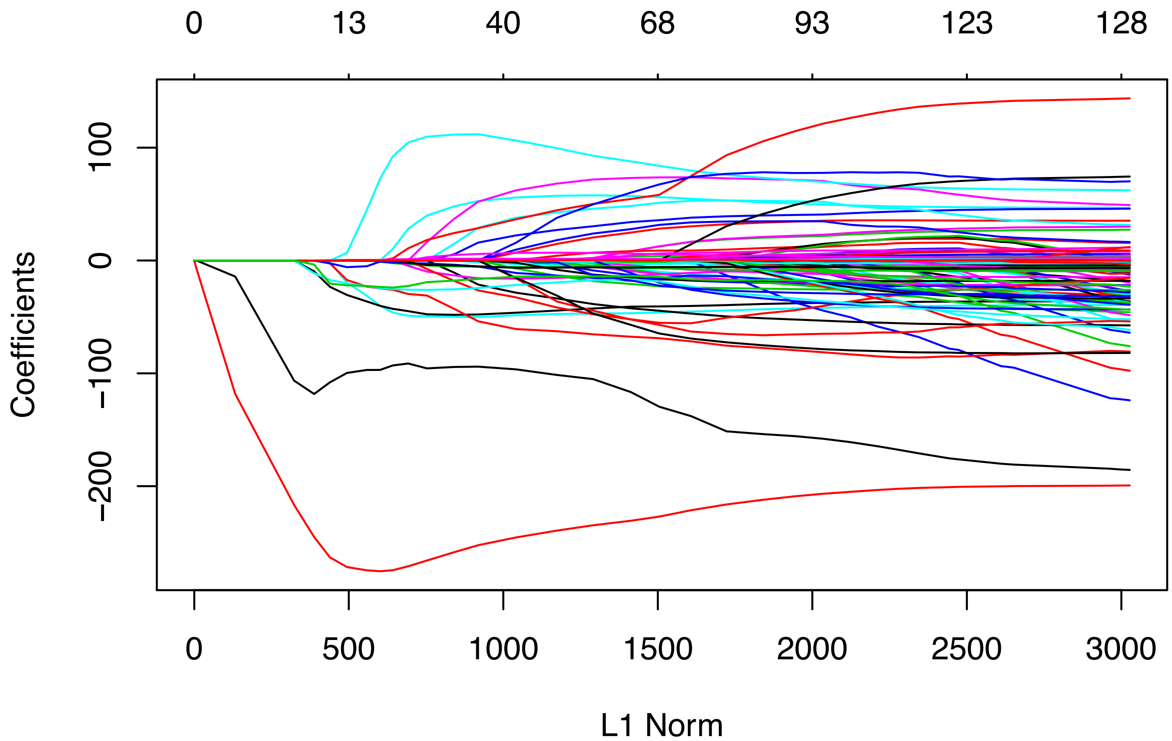
```
## [1] 4808.889
```

```
out = glmnet(x, y, alpha = 0)
predict(out, type = "coefficients", s = bestlam)[1:136,]
```

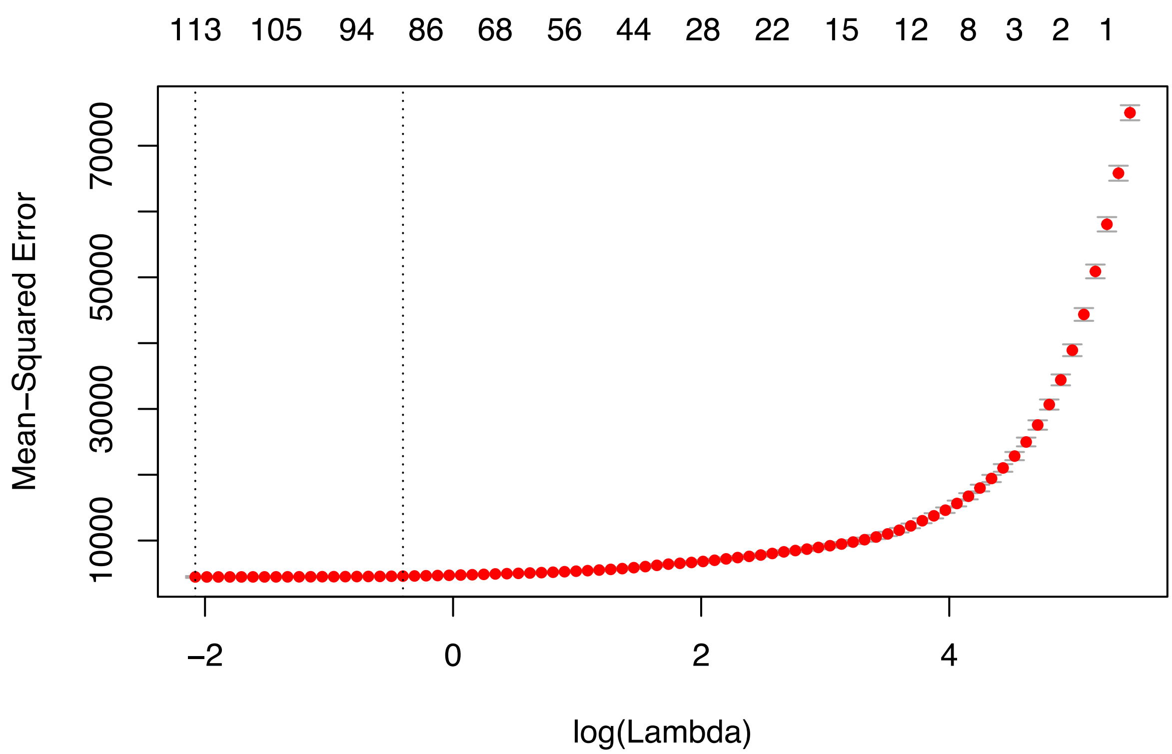
```
##   (Intercept)       BAQ1102       BAQ1103         BMXWT         BMXHT
##  5.570367e+02  2.850542e+01 -1.841816e+02 -3.498076e-01 -2.609477e-02
##        BMXBMI       BMXARML       BMXARMC      BMXWAIST       BPQ0202
##  1.181103e-01  2.131814e+00 -1.307745e+00  1.010007e+00 -1.109361e+01
##       BPQ0203      BPQ040A2      BPQ040A3       BPQ0802       BPQ0803
## -1.383252e+01 -3.003095e+01 -3.764640e+01 -6.644655e+00 -4.654154e+01
##      PEASCTM1       BPACSZ2       BPACSZ3      BPXPULS2      BPXPULS3
##  5.942971e-02  3.408638e+00 -1.423029e+01  5.847038e+01  8.747344e+00
##        BPXML1      RIAGENDR     RIDRETH12     RIDRETH13     RIDRETH14
##  5.067322e+00  2.501616e+01  1.178084e+01  3.116790e+01 -5.631529e+00
##     RIDRETH15      DMDBORN2      DMDBORN3     DMDCITZN2     DMDCITZN3
##  1.093679e+01 -4.453170e+00  1.218743e+01  1.480891e+00 -2.740290e+01
##     DMDYRSUS2     DMDYRSUS3     DMDYRSUS4     DMDYRSUS5     DMDYRSUS6
## -2.597072e+01 -1.639381e+01 -1.145791e+01 -7.595874e+00 -1.294521e+01
##     DMDYRSUS7     DMDYRSUS8     DMDYRSUS9    DMDYRSUS10     DMDEDUC31
##  2.966064e+01  5.605201e+01  7.527748e+01 -2.927997e+00 -8.957305e+00
##     DMDEDUC32     DMDEDUC33     DMDEDUC34     DMDEDUC35     DMDEDUC36
##  1.068387e+00  8.745300e+00  1.104507e+01  7.556231e+00  1.098067e+01
```

```
##    DMDEDUC37     DMDEDUC38     DMDEDUC39    DMDEDUC310    DMDEDUC311
##  1.547819e+01  1.976209e+01  1.019761e+01 -1.791517e+00  6.504388e+00
##    DMDEDUC312    DMDEDUC313    DMDEDUC314    DMDEDUC315    DMDEDUC316
##  1.694311e+01  1.593591e+01 -9.225666e-01  1.274429e+01 -4.570725e+00
##    DMDEDUC22     DMDEDUC23     DMDEDUC24     DMDSCHOL2     DMDSCHOL3
## -1.290531e+01 -3.165563e+00 -2.399408e+01  3.542845e-01 -1.610409e+01
##    DMDMARTL2     DMDMARTL3     DMDMARTL4     DMDMARTL5      DMDHHSIZ
##  8.407031e+01 -9.730553e+00 -4.882861e+01 -5.299534e+01 -6.347440e+00
##    INDHHINC2     INDHHINC3     INDHHINC4     RIDEXPRG2     RIDEXPRG3
##  3.585231e+00 -3.397866e+00  6.220202e+00  1.022512e+01  6.682234e+01
##     SIAPROXY      SIAINTRP      WTMEC2YR       DIQ0102       DIQ0103
##  2.874656e+00 -3.516389e+00 -9.560312e-04 -1.748108e+01 -1.899569e+01
##      DIQ0502       DIQ0503        DRABF2        DRABF3      DR1TKCAL
##  2.336630e+00  0.000000e+00  0.000000e+00  0.000000e+00 -5.498292e-03
##     DR1TPROT      DR1TCARB      DR1TSUGR      DR1TFIBE      DR1TTFAT
## -1.363398e-01 -5.430200e-02 -4.470231e-02  6.702387e-01  3.302805e-02
##     DR1TSFAT      DR1TCHOL      DR1TATOC      DR1TVARA      DR1TBCAR
## -1.255570e-01  2.223282e-03 -2.324112e-02  4.698083e-03  4.205638e-05
##       DR1TFA        DR1TVC        DR1TVK      DR1TCALC      DR1TMAGN
##  1.288748e-03  9.400320e-03  9.518232e-03 -4.277335e-03  2.036022e-02
##     DR1TIRON      DR1TSODI      DR1TPOTA      DR1TCAFF      DR1TALCO
##  1.843328e-01 -1.155016e-03  2.575688e-03  1.431566e-02 -1.641410e-01
##      HSD0102       HSD0103       HSD0104       HSQ5202       HSQ5203
##  2.947246e+00 -1.230967e+00 -4.060307e+01  4.100426e+00  5.750965e+01
##     LBXWBCSI      LBXMCVSI       MCQ0922       MCQ0923      MCQ160A2
## -1.272568e+00  1.526115e+00 -2.288296e+01 -4.820278e+01 -2.705154e+01
##     MCQ160A3      MCQ160B2      MCQ160B3      MCQ160C2      MCQ160C3
## -2.591037e+01  4.351724e+00 -1.986906e+01 -1.360057e+01 -1.864023e+01
##     MCQ160D2      MCQ160D3      MCQ160E2      MCQ160E3      MCQ160F2
##  3.719983e+00 -2.222569e+01 -3.081156e+00 -2.611255e+01  4.581966e+00
##     MCQ160F3      MCQ160G2      MCQ160G3       MCQ2202       MCQ2203
## -2.622077e+01  9.287069e+00 -2.400881e+01 -2.650435e+01 -2.583275e+01
##      VIQ1802       VIQ1803       VIQ2002       VIQ2003       VIQ2202
##  8.329540e+00 -1.046355e+01 -6.292086e+01 -1.054929e+01 -1.097969e+01
##      VIQ2203        WHQ030       WHQ0702       WHQ0703       WHQ0902
## -1.555892e+01  1.124902e+00  4.346569e+00 -8.930275e+00  3.600486e+00
##      WHQ0903
## -1.460283e+01
```

```
###################################
#(2) Lasso
###################################
library(glmnet)
set.seed(1)
x = model.matrix(RIDAGEEX~., data = complete_data5)[,-1]
y = complete_data5$RIDAGEEX
train = sample(1:nrow(x), round(nrow(x)*.70,0))
test = (-train)
y.test = y[test]

grid = 10^seq(10, -2, length =100)
lasso.mod = glmnet(x[train,], y[train], alpha = 1, lambda = grid)
plot(lasso.mod)
```




```r
set.seed(1)
cv.out = cv.glmnet(x[train,], y[train], alpha = 1)
plot(cv.out)
```




```r
bestlam = cv.out$lambda.min
lasso.pred = predict(lasso.mod, s = bestlam, newx = x[test,])
mean((lasso.pred - y.test)^2)
```

```
## [1] 4467.087
```

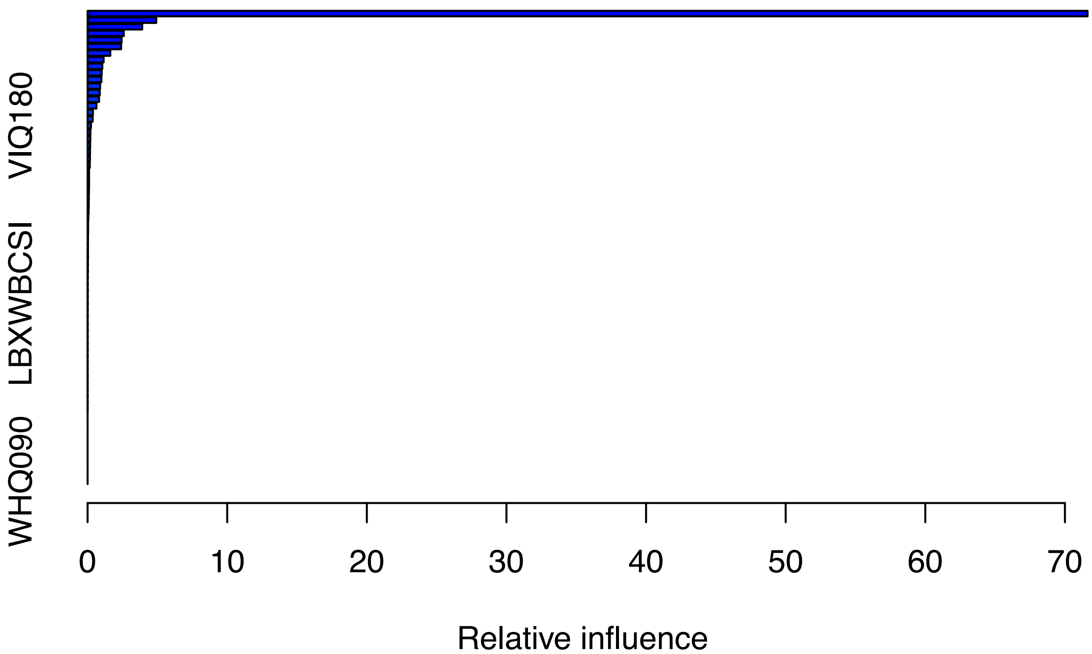
```r
out = glmnet(x, y, alpha = 1, lambda = grid)
lasso.coef = predict(out, type = "coefficients", s = bestlam)[1:136,]
lasso.coef[lasso.coef!=0]
```

```
##   (Intercept)       BAQ1102       BAQ1103         BMXWT         BMXHT 
##  6.244372e+02  8.959362e+00 -2.037327e+02 -1.248220e+00 -3.231828e-01 
##        BMXBMI       BMXARML       BMXARMC      BMXWAIST       BPQ0203 
##  1.100542e+00  3.179239e+00 -3.357654e+00  2.334980e+00 -5.324563e+00 
##       BPQ040A2      BPQ040A3       BPQ0802       BPQ0803      PEASCTM1 
## -2.323632e+01 -3.664620e+01 -4.078247e+00 -3.800880e+01  3.720727e-02 
##       BPACSZ2       BPACSZ3      BPXPULS2      BPXPULS3        BPXML1 
##  5.229704e+00 -1.501297e+00  4.585260e+01  9.220250e+00  6.347421e+00 
##      RIAGENDR     RIDRETH12     RIDRETH13     RIDRETH14     RIDRETH15 
##  3.518349e+01  2.645849e+01  4.846966e+01  4.591487e+00  2.708770e+01 
##      DMDBORN3     DMDCITZN2     DMDCITZN3     DMDYRSUS2     DMDYRSUS3 
##  1.546988e+01  1.527148e+00 -1.842683e+01 -2.441936e+01 -1.707183e+01 
##     DMDYRSUS4     DMDYRSUS5     DMDYRSUS6     DMDYRSUS7     DMDYRSUS8 
## -1.185410e+01 -9.212293e+00 -2.022606e+01  1.753140e+01  4.238587e+01 
##     DMDYRSUS9     DMDEDUC31     DMDEDUC32     DMDEDUC33     DMDEDUC34 
##  5.687572e+01 -3.817318e+01 -1.734561e+01 -5.586549e+00 -8.276461e-01 
##     DMDEDUC35     DMDEDUC36     DMDEDUC37     DMDEDUC38     DMDEDUC39 
## -2.857259e+00  5.184362e-02  6.837029e-01 -2.593090e+00 -9.713251e-01 
##    DMDEDUC310    DMDEDUC311    DMDEDUC312    DMDEDUC313    DMDEDUC315 
## -8.220180e+00  2.702100e+00  1.709912e+01  1.247525e+01  1.134238e+01 
##    DMDEDUC316     DMDEDUC22     DMDEDUC24     DMDSCHOL2     DMDSCHOL3 
## -2.003657e+01 -1.300772e+01 -1.898618e+01  3.453803e+00 -6.720314e+01 
##     DMDMARTL2     DMDMARTL3     DMDMARTL4     DMDMARTL5      DMDHHSIZ 
##  6.267284e+01 -1.985672e+01 -5.853075e+01 -8.770588e+01 -5.338859e+00 
##     INDHHINC2     INDHHINC3     INDHHINC4     RIDEXPRG2     RIDEXPRG3 
##  6.486972e+00  1.662761e+00  5.591227e+00  7.906471e+01  1.480233e+02 
##      SIAPROXY      WTMEC2YR       DIQ0102       DIQ0502      DR1TPROT 
## -7.053270e+00 -1.276839e-03 -9.664335e+00  4.814126e+00 -1.578236e-01 
##      DR1TCARB      DR1TFIBE      DR1TTFAT      DR1TSFAT      DR1TCHOL 
## -1.027415e-01  6.281911e-01  2.914993e-02 -1.521375e-01  1.387638e-03 
##      DR1TATOC      DR1TVARA      DR1TBCAR        DR1TFA        DR1TVC 
## -4.774386e-02  4.169306e-03 -4.289441e-05  6.527101e-03  9.671391e-03 
##        DR1TVK      DR1TCALC      DR1TMAGN      DR1TIRON      DR1TSODI 
##  2.476981e-03 -3.269831e-03  2.672010e-02  1.593549e-01 -1.108354e-04 
##      DR1TPOTA      DR1TCAFF      DR1TALCO       HSD0102       HSD0103 
##  2.878104e-03  6.460464e-03 -1.750060e-01 -2.423210e-01 -1.160915e+01 
##       HSD0104       HSQ5202       HSQ5203      LBXWBCSI      LBXMCVSI 
## -7.253519e+01  5.316305e+00  8.741958e+01 -9.231134e-01  1.202785e+00 
##       MCQ0922       MCQ0923      MCQ160A2      MCQ160A3      MCQ160B2 
## -1.651835e+01 -2.567198e+01 -2.310777e+01 -3.301512e+01 -6.134050e-01 
##      MCQ160C2      MCQ160D3      MCQ160E2      MCQ160E3      MCQ160F2 
## -3.161341e+01 -1.520478e+01 -1.550695e+00 -6.040924e+01 -4.610524e+00 
##      MCQ160F3      MCQ160G3       MCQ2202       MCQ2203       VIQ1802 
## -1.356765e+02 -9.781572e+00 -4.154019e+01 -5.306326e+01  2.062979e+00 
##       VIQ2002       VIQ2202       VIQ2203        WHQ030       WHQ0702 
## -7.589469e+01 -8.293226e+00 -2.659541e+01  3.301751e+00  5.121810e-01 
##       WHQ0703       WHQ0903 
## -9.028559e+00 -2.174886e+01
```

```r
length(lasso.coef[lasso.coef!=0])
```

```
## [1] 117
```

```r
####################################
#(3) Boosting
####################################
library(gbm)
set.seed(1)
train = sample(1:nrow(complete_data5), round(nrow(complete_data5)*.70,0))
complete.test=complete_data5[-train ,"RIDAGEEX"]
boost.complete=gbm(RIDAGEEX~., data=complete_data5[train,], distribution="gaussian", n.trees=5000,
                   interaction.depth=4)
summary(boost.complete)
```




```
##               var      rel.inf
## BAQ110     BAQ110 7.162229e+01
## DMDMARTL DMDMARTL 4.916016e+00
## DMDEDUC2 DMDEDUC2 3.909600e+00
## MCQ160A   MCQ160A 2.592194e+00
## BPQ080     BPQ080 2.435003e+00
## RIDEXPRG RIDEXPRG 2.401218e+00
## WTMEC2YR WTMEC2YR 1.625185e+00
## BMXHT       BMXHT 1.149476e+00
## VIQ200     VIQ200 1.052755e+00
## MCQ160B   MCQ160B 1.007877e+00
## BPQ040A   BPQ040A 9.877196e-01
## DMDHHSIZ DMDHHSIZ 8.968240e-01
## BPACSZ     BPACSZ 8.700893e-01
## MCQ160E   MCQ160E 8.198413e-01
## WHQ030     WHQ030 6.332557e-01
## MCQ220     MCQ220 3.799118e-01
## DR1TKCAL DR1TKCAL 3.628622e-01
```

```
## VIQ180     VIQ180 2.413110e-01
## RIDRETH1 RIDRETH1 2.007928e-01
## BPQ020     BPQ020 1.946399e-01
## MCQ160G   MCQ160G 1.825662e-01
## DMDYRSUS DMDYRSUS 1.746104e-01
## BPXPULS   BPXPULS 1.678314e-01
## PEASCTM1 PEASCTM1 1.576443e-01
## MCQ160D   MCQ160D 1.057256e-01
## BMXARML   BMXARML 1.054911e-01
## MCQ092     MCQ092 1.043512e-01
## LBXMCVSI LBXMCVSI 9.667658e-02
## VIQ220     VIQ220 8.705143e-02
## BMXWAIST BMXWAIST 8.122694e-02
## BMXWT       BMXWT 7.914398e-02
## BPXML1     BPXML1 5.116085e-02
## DR1TCARB DR1TCARB 3.777800e-02
## DIQ010     DIQ010 3.770745e-02
## BMXARMC   BMXARMC 3.039087e-02
## DR1TSUGR DR1TSUGR 2.674755e-02
## RIAGENDR RIAGENDR 2.490007e-02
## MCQ160C   MCQ160C 2.262349e-02
## DR1TPROT DR1TPROT 1.718969e-02
## MCQ160F   MCQ160F 1.638452e-02
## WHQ070     WHQ070 9.531169e-03
## DR1TBCAR DR1TBCAR 9.168245e-03
## DR1TALCO DR1TALCO 9.014415e-03
## HSD010     HSD010 6.955252e-03
## LBXWBCSI LBXWBCSI 6.678747e-03
## DR1TCAFF DR1TCAFF 6.260457e-03
## DR1TFIBE DR1TFIBE 5.665035e-03
## DR1TSFAT DR1TSFAT 5.233758e-03
## DR1TVC     DR1TVC 5.168597e-03
## DMDEDUC3 DMDEDUC3 4.548604e-03
## DR1TVARA DR1TVARA 4.420993e-03
## DR1TTFAT DR1TTFAT 3.309405e-03
## DR1TCHOL DR1TCHOL 2.766564e-03
## DR1TVK     DR1TVK 2.244570e-03
## DR1TSODI DR1TSODI 1.948579e-03
## DR1TPOTA DR1TPOTA 1.771555e-03
## BMXBMI     BMXBMI 1.770580e-03
## DR1TMAGN DR1TMAGN 1.550574e-03
## DR1TFA     DR1TFA 1.450068e-03
## HSQ520     HSQ520 1.438487e-03
## DMDSCHOL DMDSCHOL 1.374330e-03
## INDHHINC INDHHINC 5.680237e-04
## DR1TIRON DR1TIRON 4.890355e-04
## DR1TATOC DR1TATOC 2.874125e-04
## SIAPROXY SIAPROXY 1.890862e-04
## DMDBORN   DMDBORN 1.358891e-04
## DMDCITZN DMDCITZN 0.000000e+00
## SIAINTRP SIAINTRP 0.000000e+00
## DIQ050     DIQ050 0.000000e+00
## DRABF       DRABF 0.000000e+00
## DR1TCALC DR1TCALC 0.000000e+00
```

```
## WHQ090     WHQ090 0.000000e+00
```

```r
yhat.boost=predict(boost.complete, newdata= complete_data5[-train,], n.trees=5000)
mean((yhat.boost - complete.test)^2)
```

```
## [1] 4030.353
```

```r
set.seed(1)
boost.complete=gbm(RIDAGEEX~.,data=complete_data5[train ,], distribution= "gaussian", n.trees=5000,
                   interaction.depth=4, shrinkage =0.2, verbose=F)
yhat.boost=predict(boost.complete, newdata= complete_data5[-train,], n.trees=5000)
mean((yhat.boost - complete.test)^2)
```

```
## [1] 3790.108
```

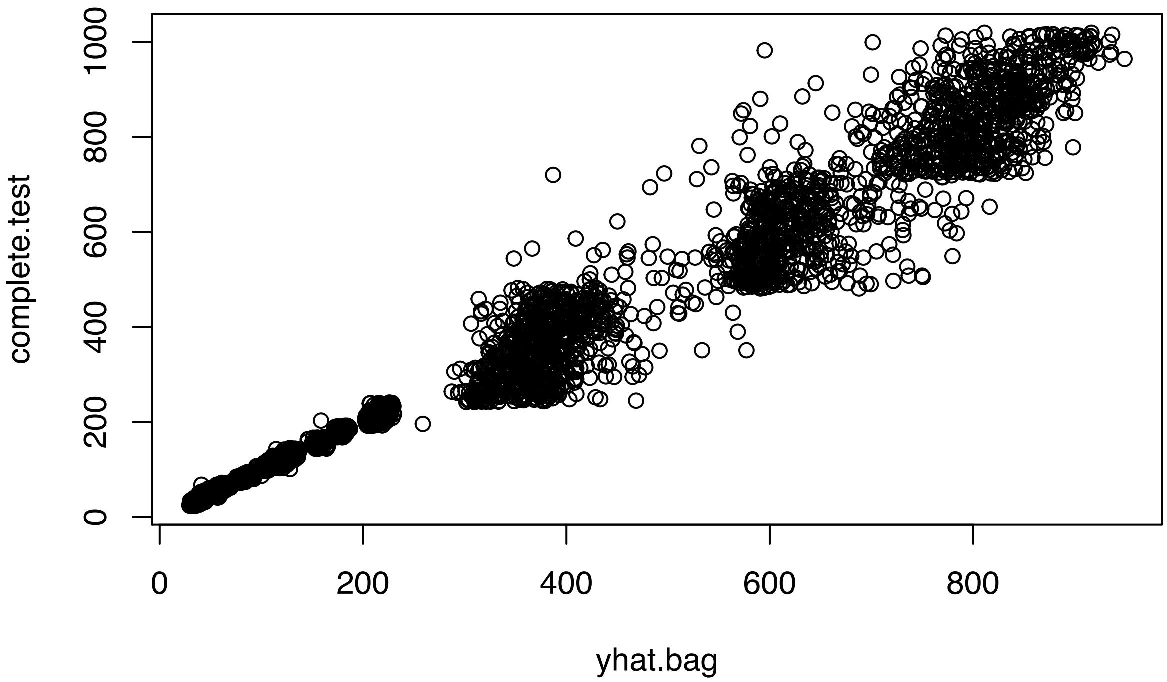
```r
###################################
#(4) Bagging/ Regression Tree
###################################
library(tree)
set.seed(1)
train = sample(1:nrow(complete_data5), nrow(complete_data5)/2)
tree.data = tree(RIDAGEEX~., complete_data5, subset=train)
summary(tree.data)
```

```
##
## Regression tree:
## tree(formula = RIDAGEEX ~ ., data = complete_data5, subset = train)
## Variables actually used in tree construction:
## [1] "BAQ110"   "MCQ160A"  "VIQ180"   "RIDEXPRG" "WTMEC2YR"
## Number of terminal nodes:  6
## Residual mean deviance:  7854 = 28200000 / 3590
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -317.500  -40.450    0.447    0.000   37.450  630.600
```

```r
library(randomForest)
set.seed(1)
bag.data = randomForest(RIDAGEEX~., complete_data5, subset=train, mtry=15, importance =TRUE)
bag.data
```

```
##
## Call:
##  randomForest(formula = RIDAGEEX ~ ., data = complete_data5, mtry = 15,      importance = TRUE, subs
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 15
##
##           Mean of squared residuals: 3482.866
##                     % Var explained: 95.43
```

```r
yhat.bag = predict(bag.data , newdata=complete_data5[-train ,])
complete.test=complete_data5[-train ,"RIDAGEEX"]
plot(yhat.bag , complete.test)
```




```
mean((yhat.bag - complete.test)^2)
```

```
## [1] 3397.338
```

```
bag.complete = randomForest(RIDAGEEX~., data=complete_data5 , subset=train, mtry=15, ntree=25)
yhat.bag = predict(bag.complete , newdata=complete_data5[-train ,])
mean((yhat.bag - complete.test)^2)
```

```
## [1] 3656.222
```

```
set.seed(1)
rf.complete = randomForest(RIDAGEEX ~., data=complete_data5, subset=train, mtry=18, ntree=30)
yhat.rf = predict(rf.complete , newdata = complete_data5[-train ,])
mean((yhat.rf - complete.test)^2)
```

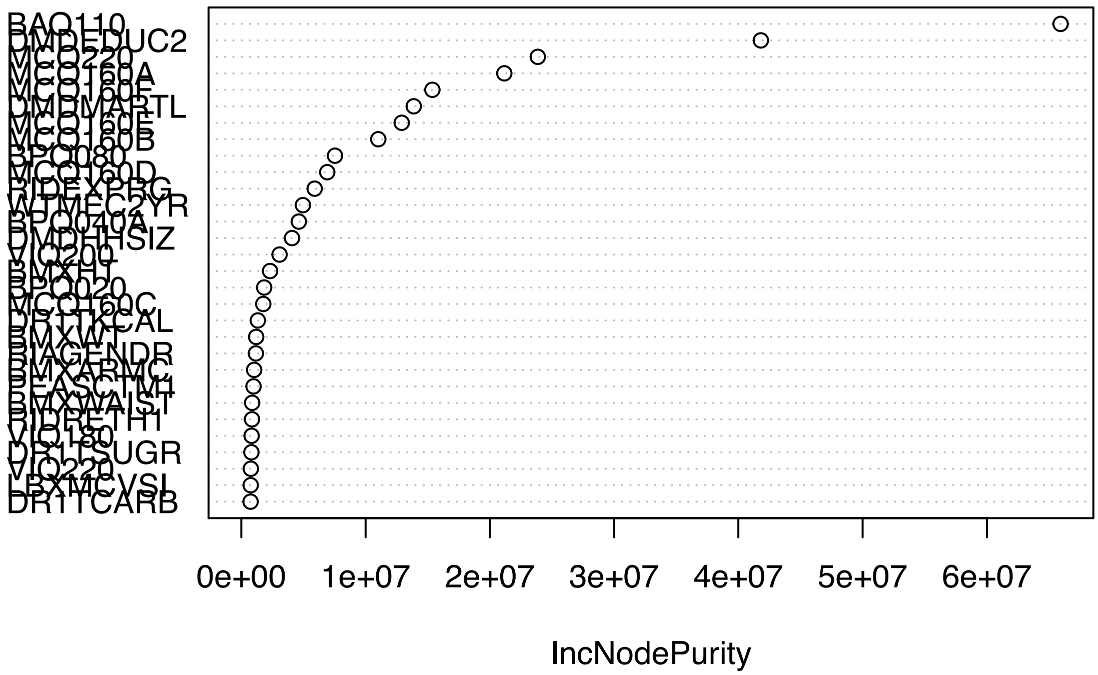
```
## [1] 3612.438
```

```
head(importance(rf.complete))
```

```
##          IncNodePurity
## BAQ110      65930320.1
## BMXWT        1197240.5
## BMXHT        2311932.2
## BMXBMI        472927.3
## BMXARML       599465.1
## BMXARMC      1033756.4
```

```
head(varImpPlot(rf.complete))
```

rf.complete




```
##          IncNodePurity
## BAQ110     65930320.1
## BMXWT       1197240.5
## BMXHT       2311932.2
## BMXBMI       472927.3
## BMXARML      599465.1
## BMXARMC     1033756.4
```

```r
set.seed(1)
rf.complete = randomForest(RIDAGEEX ~., data=complete_data5, subset=train, importance =TRUE, ntree=100)
yhat.rf = predict(rf.complete , newdata = complete_data5[-train ,])
mean((yhat.rf - complete.test)^2)
```

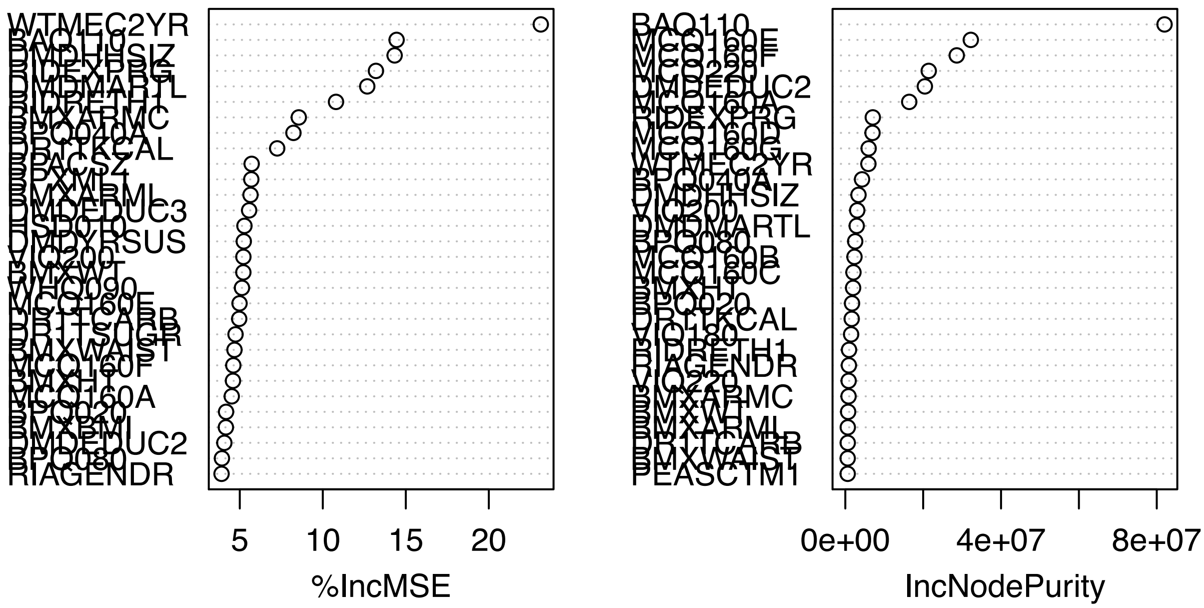
```
## [1] 3271.013
```

```r
head(importance(rf.complete))
```

```
##            %IncMSE IncNodePurity
## BAQ110   14.446409    81999873.2
## BMXWT     5.214026      725073.7
## BMXHT     4.589086     1994497.2
## BMXBMI    4.162237      418995.5
## BMXARML   5.645939      634393.7
## BMXARMC   8.552824      827100.7
```

```r
head(varImpPlot(rf.complete))
```

rf.complete




```
##           %IncMSE IncNodePurity
## BAQ110  14.446409    81999873.2
## BMXWT    5.214026      725073.7
## BMXHT    4.589086     1994497.2
## BMXBMI   4.162237      418995.5
## BMXARML  5.645939      634393.7
## BMXARMC  8.552824      827100.7
```

[4a]. Variables Contained in Age Dataset 2

```r
colnames(complete_data6)
```

```
##  [1] "BAQ110"   "BMXWT"    "BMXHT"    "BMXBMI"   "BMXARML"  "BMXARMC"
##  [7] "BPQ020"   "BPQ040A"  "BPQ080"   "PEASCTM1" "BPACSZ"   "BPXPULS"
## [13] "BPXML1"   "RIAGENDR" "RIDAGEEX" "RIDRETH1" "DMDBORN"  "DMDCITZN"
## [19] "DMDYRSUS" "DMDEDUC3" "DMDEDUC2" "DMDSCHOL" "DMDMARTL" "DMDHHSIZ"
## [25] "INDHHINC" "RIDEXPRG" "SIAPROXY" "SIAINTRP" "WTMEC2YR" "DIQ010"
## [31] "DIQ050"   "DRABF"    "DR1TKCAL" "DR1TPROT" "DR1TCARB" "DR1TSUGR"
## [37] "DR1TFIBE" "DR1TTFAT" "DR1TSFAT" "DR1TCHOL" "DR1TATOC" "DR1TVARA"
## [43] "DR1TBCAR" "DR1TFA"   "DR1TVC"   "DR1TVK"   "DR1TCALC" "DR1TMAGN"
## [49] "DR1TIRON" "DR1TSODI" "DR1TPOTA" "DR1TCAFF" "DR1TALCO" "HSD010"
## [55] "HSQ520"   "MCQ092"   "MCQ160A"  "MCQ160B"  "MCQ160C"  "MCQ160D"
## [61] "MCQ160E"  "MCQ160F"  "MCQ160G"  "MCQ220"   "VIQ180"   "VIQ200"
## [67] "VIQ220"   "WHQ030"   "WHQ070"   "WHQ090"
```

1. DIQ010: Doctor told you have diabetes (1-150 Years)
2. DIQ050: Taking insulin now (1-150 Years)
3. HSQ520: Flu, pneumonia, ear infection in past 30 days? (1-150 Years)

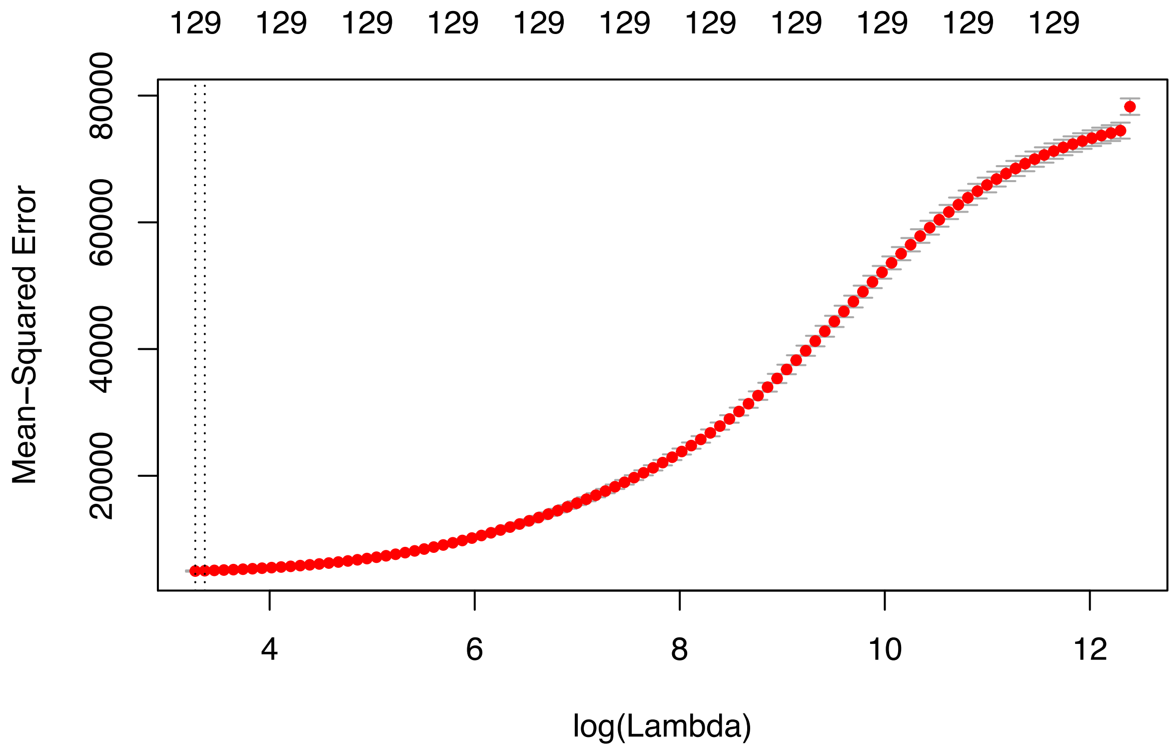
4. HSD010: General Health Condition (12-150 Years)
5. VIQ180: Eye surgery for near sightedness (12-150 Years)
6. VIQ200: Eye surgery for cataracts (12-150 Years)
7. VIQ220: Glasses/ contacts worn for distance; (12-150 Years)
8. DR1TKCAL: Energy (Calories) (0-150 Years)
9. DR1TPROT: Protein (0-150 Years)
10. DR1TCARB: Carbohydrate (0-150 Years)
11. DR1TSUGR: Total sugars (0-150 Years)
12. DR1TFIBE: Dietary Fiber (0-150 Years)
13. DR1TTFAT: Total Fat (0-150 Years)
14. DR1TCHOL: Cholesterol (0-150 Years)
15. DR1TATOC: Vitamin E (0-150 Years)
16. DR1TVARA: Vitamin A (0-150 Years)
17. DR1TBCAR: Beta-carotene (0-150 Years)
18. DR1TFA: Folic Acid (0-150 Years)
19. DR1TVC: Vitamin C (0-150 Years)
20. DR1TVK: Vitamin K (0-150 Years)
21. DR1TCALC: Calcium (0-150 Years)
22. DR1TMAGN: Magnesium (0-150 Years)
23. DR1TIRON: Iron (0-150 Years)
24. DR1TSODI: Sodium (0-150 Years)
25. DR1TPOTA: Potassium (0-150 Years)
26. DR1TCAFF: Caffeine (0-150 Years)
27. DR1TALCO: Alcohol (0-150 Years)
28. WHQ070: Tried to lose weight in past year (16-150 Years)
29. WHQ090: Tried not to gain weight in past year (16-150 Years)
30. BMXWT: Weight (0-150 Years)
31. BMXARML: Upper Arm Length (0-150 Years)
32. BMXARMC: Arm Circumference (0-150 Years)
33. BMXHT: Standing Height (2-150 Years)
34. BMXBMI: Body Mass Index (2-150 Years)
35. PEASCTM1: Blood Pressure Time in Seconds (0-150 Years)
36. BPACSZ: Coded cuff size; Recode missing (8-150 Years)
37. BPQ020: Ever told you had high blood pressure (16-150 Years)
38. DMDMARTL: Martial Status
39. DMDEDUC3: Education (6-19 Years)
40. DMDEDUC2: Eudcation (20-150 Years)
41. RIAGENDR: Gender (0-150 Years)
42. RIDRETH1: Race/Ethnicity (0-150 Years)
43. DMDBORN: Country of Birth (0-150 Years)
44. DMDCITZN: Citizenship Status (0-150 Years)
45. DMDYRSUS: Length of time in US (0-150 Years)
46. DMDSCHOL: Now attending school (6-19 Years)
47. DMDHHSIZ: Total number of people in the household (0-150 Years)
48. INDHHIC: Annual Household Income (0-150 Years)
49. RIDEXPRG: Pregnancy Status at Exam (8-59 Years)
50. SIAPROXY: Was a proxy used in SP interview
51. SIAINTRP: Was an interpreter used in SP interview
52. WTMEC2YR: Full Sample 2 Year MEC Exam Weight (0-150 Years)
53. DRABF: Breast-fed infant (either day) (0-150 Years)
54. BAQ110: Can you stand on your own? (40-150 Years)
55. BPXML1: Pulse Maximum Inflation Levels
56. WHQ030: How do you consider your weight (16-150 Years)
57. MCQ160G: Ever told you had emphysema (20-150 Years)

58. BPQ040A: Taking prescription for hypertension
59. BPQ080: Doctor told you had high cholesterol (20-150 Years)
60. MCQ160A: Ever told you had arthritis (20-150 Years)
61. MCQ160B: Ever told you had congestive heart failure (20-150 Years)
62. MCQ160C: Ever told you had coronary heart disease (20-150 Years)
63. MCQ160D: Ever told you had angina (20-150 Years)
64. MCQ160E: Ever told you had heart attack; (20-150 Years)
65. MCQ160F: Ever told you had stroke; (20-150 Years)
66. MCQ220: Ever told you have cancer; (20-150 Years)
67. MCQ092: Ever receive blood transfusion; (6-150 Years)
68. DR1TSFAT: Total Saturated Fat (0-150 Years)
69. BPXPULS: Is pulse irregular?
70. RIDAGEEX: Patient age when exam was given

[4b]. Rational Variables Contained in Age Dataset 2

[4d]. Methods/Reults for Predicting Age in Age Dataset 2

```
#complete_data6
####################################
#(1) Ridge Regression
####################################
x = model.matrix(RIDAGEEX~., data = complete_data6)[,-1]
y = complete_data6$RIDAGEEX
grid = 10^seq(10, -2, length =100)
ridge.mod = glmnet(x, y, alpha = 0, lambda = grid)

set.seed(1)
train = sample(1:nrow(x), round(nrow(x)*.70,0))
test = (-train)
y.test = y[test]
cv.out = cv.glmnet(x[train,], y[train], alpha = 0)
plot(cv.out)
```




```r
bestlam = cv.out$lambda.min
bestlam
```

```
## [1] 26.42882
```

```r
set.seed(1)
ridge.pred = predict(ridge.mod, s = bestlam, newx = x[test,])
mean((ridge.pred - y.test)^2)
```

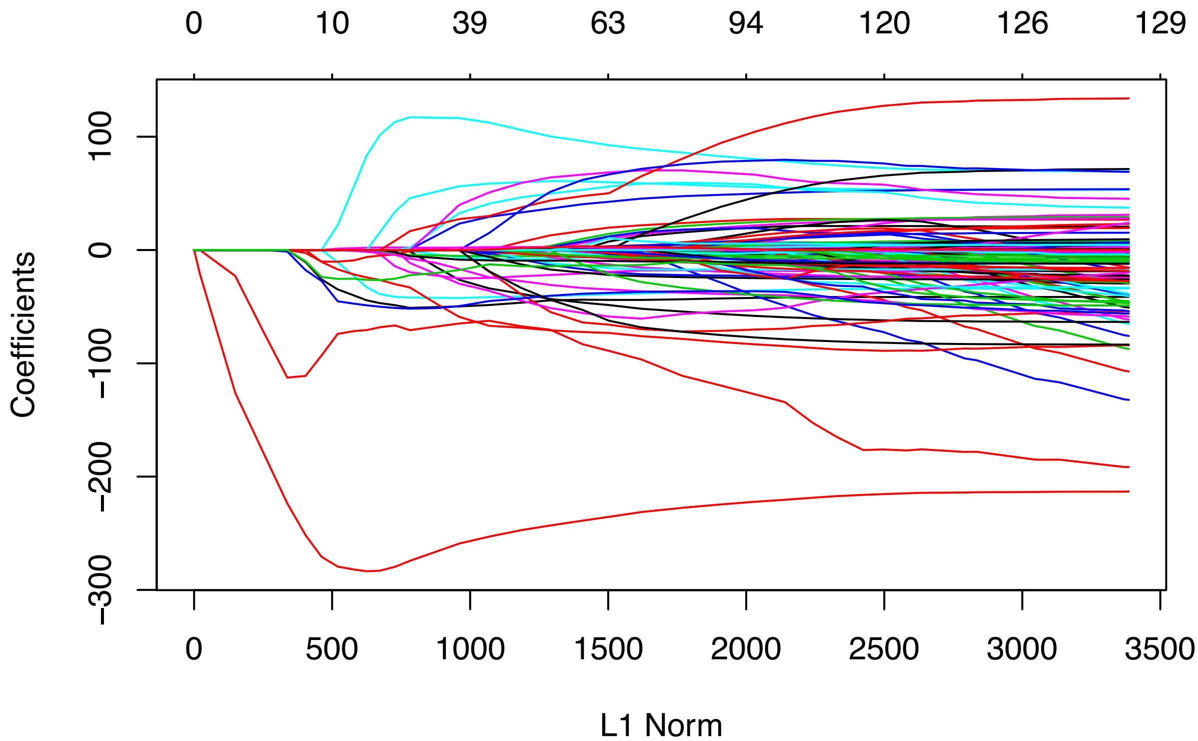
```
## [1] 4940.001
```

```r
out = glmnet(x, y, alpha = 0)
predict(out, type = "coefficients", s = bestlam)[1:133,]
```

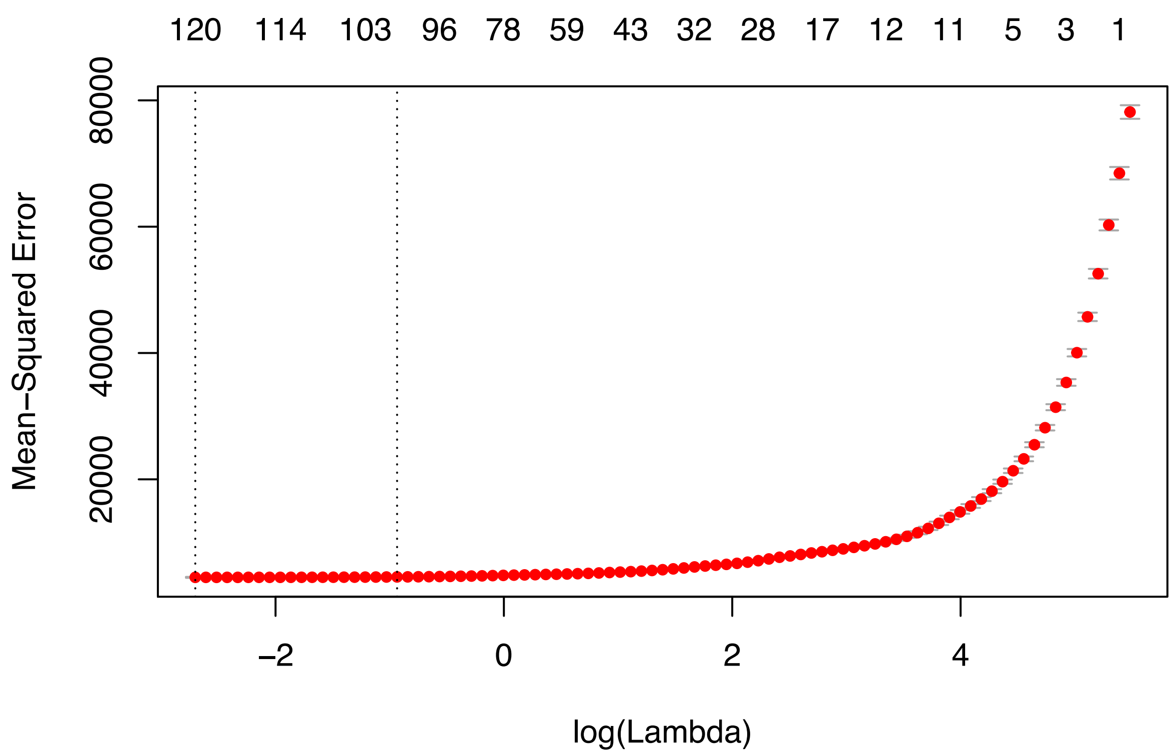
```
##   (Intercept)       BAQ1102       BAQ1103         BMXWT         BMXHT 
##  7.147868e+02  3.387531e+01 -1.859072e+02 -1.219960e-01  1.202634e-01 
##        BMXBMI       BMXARML       BMXARMC       BPQ0202       BPQ0203 
##  9.641353e-01  2.446244e+00 -1.021507e+00 -1.134476e+01 -1.387782e+01 
##      BPQ040A2      BPQ040A3       BPQ0802       BPQ0803      PEASCTM1 
## -3.146859e+01 -3.949393e+01 -7.498627e+00 -4.931558e+01  5.799037e-02 
##       BPACSZ2       BPACSZ3      BPXPULS2      BPXPULS3        BPXML1 
##  4.051828e+00 -1.533815e+01  6.042689e+01  1.219001e+01  4.238189e+00 
##      RIAGENDR     RIDRETH12     RIDRETH13     RIDRETH14     RIDRETH15 
##  2.251772e+01  1.184098e+01  3.235366e+01 -9.604635e+00  9.796766e+00 
##      DMDBORN2      DMDBORN3     DMDCITZN2     DMDCITZN3     DMDYRSUS2 
## -3.203872e+00  1.142053e+01  2.216237e-01 -3.603007e+01 -2.576380e+01 
##     DMDYRSUS3     DMDYRSUS4     DMDYRSUS5     DMDYRSUS6     DMDYRSUS7 
## -1.463680e+01 -1.100242e+01 -7.267108e+00 -1.327972e+01  3.097851e+01 
##     DMDYRSUS8     DMDYRSUS9    DMDYRSUS10     DMDEDUC31     DMDEDUC32 
##  5.919930e+01  7.912438e+01 -2.326467e+00 -1.010929e+01  1.540998e+00 
##     DMDEDUC33     DMDEDUC34     DMDEDUC35     DMDEDUC36     DMDEDUC37 
##  9.119945e+00  1.276348e+01  8.836354e+00  1.077415e+01  1.620441e+01
```

```
##     DMDEDUC38     DMDEDUC39    DMDEDUC310    DMDEDUC311    DMDEDUC312
##  2.114376e+01  1.089655e+01 -9.452507e-01  6.231686e+00  1.578960e+01
##    DMDEDUC313    DMDEDUC314    DMDEDUC315    DMDEDUC316     DMDEDUC22
##  1.687552e+01 -3.955813e+00  1.315025e+01 -6.316078e+00 -1.344225e+01
##     DMDEDUC23     DMDEDUC24     DMDSCHOL2     DMDSCHOL3     DMDMARTL2
## -5.681238e+00 -2.352141e+01  7.815126e-01 -1.655293e+01  9.162208e+01
##     DMDMARTL3     DMDMARTL4     DMDMARTL5      DMDHHSIZ     INDHHINC2
## -9.428372e+00 -4.949334e+01 -5.382095e+01 -6.482398e+00  3.136157e+00
##     INDHHINC3     INDHHINC4     RIDEXPRG2     RIDEXPRG3      SIAPROXY
## -3.241738e+00  5.300199e+00  7.941986e+00  6.235171e+01 -5.743622e-03
##      SIAINTRP      WTMEC2YR       DIQ0102       DIQ0103       DIQ0502
## -9.027704e-01 -9.852790e-04 -1.903000e+01 -2.675219e+01  3.168463e+00
##       DIQ0503        DRABF2        DRABF3       DR1TKCAL       DR1TPROT
##  0.000000e+00  0.000000e+00  0.000000e+00 -5.613357e-03 -1.454188e-01
##      DR1TCARB      DR1TSUGR      DR1TFIBE      DR1TTFAT      DR1TSFAT
## -5.406081e-02 -4.928944e-02  6.973026e-01  3.017503e-02 -1.356390e-01
##      DR1TCHOL      DR1TATOC      DR1TVARA      DR1TBCAR        DR1TFA
##  4.174828e-03 -8.202491e-02  4.403282e-03 -1.085485e-05  4.642239e-04
##        DR1TVC        DR1TVK      DR1TCALC      DR1TMAGN      DR1TIRON
##  1.013092e-02  1.213891e-02 -4.889524e-03  2.347896e-02  2.367992e-01
##      DR1TSODI      DR1TPOTA      DR1TCAFF      DR1TALCO       HSD0102
## -1.037541e-03  2.626022e-03  1.730222e-02 -1.537689e-01  4.141622e+00
##       HSD0103       HSD0104       HSQ5202       HSQ5203       MCQ0922
##  2.881540e+00 -4.007265e+01  3.886018e+00  5.733985e+01 -2.110925e+01
##       MCQ0923      MCQ160A2      MCQ160A3      MCQ160B2      MCQ160B3
## -4.897246e+01 -2.861064e+01 -2.780040e+01  3.778498e+00 -2.198587e+01
##      MCQ160C2      MCQ160C3      MCQ160D2      MCQ160D3      MCQ160E2
## -1.297169e+01 -2.062659e+01  3.608714e+00 -2.368246e+01 -2.751691e+00
##      MCQ160E3      MCQ160F2      MCQ160F3      MCQ160G2      MCQ160G3
## -2.755719e+01  4.102849e+00 -2.758036e+01  9.658863e+00 -2.538330e+01
##       MCQ2202       MCQ2203       VIQ1802       VIQ1803       VIQ2002
## -2.759151e+01 -2.594768e+01  8.994288e+00 -1.018928e+01 -6.348045e+01
##       VIQ2003       VIQ2202       VIQ2203        WHQ030       WHQ0702
## -1.062560e+01 -1.189286e+01 -1.488822e+01  2.760960e-01  4.468285e+00
##       WHQ0703       WHQ0902       WHQ0903
## -9.506782e+00  4.153724e+00 -1.565731e+01
```

```
###################################
#(2) Lasso
###################################
set.seed(1)
x = model.matrix(RIDAGEEX~., data = complete_data6)[,-1]
y = complete_data6$RIDAGEEX
train = sample(1:nrow(x), round(nrow(x)*.70,0))
test = (-train)
y.test = y[test]

grid = 10^seq(10, -2, length =100)
lasso.mod = glmnet(x[train,], y[train], alpha = 1, lambda = grid)
plot(lasso.mod)
```




```
set.seed(1)
cv.out = cv.glmnet(x[train,], y[train], alpha = 1)
plot(cv.out)
```




```
bestlam = cv.out$lambda.min
lasso.pred = predict(lasso.mod, s = bestlam, newx = x[test,])
mean((lasso.pred - y.test)^2)
```

```
## [1] 4780.98
```

```
out = glmnet(x, y, alpha = 1, lambda = grid)
lasso.coef = predict(out, type = "coefficients", s = bestlam)[1:133,]
lasso.coef[lasso.coef!=0]
```

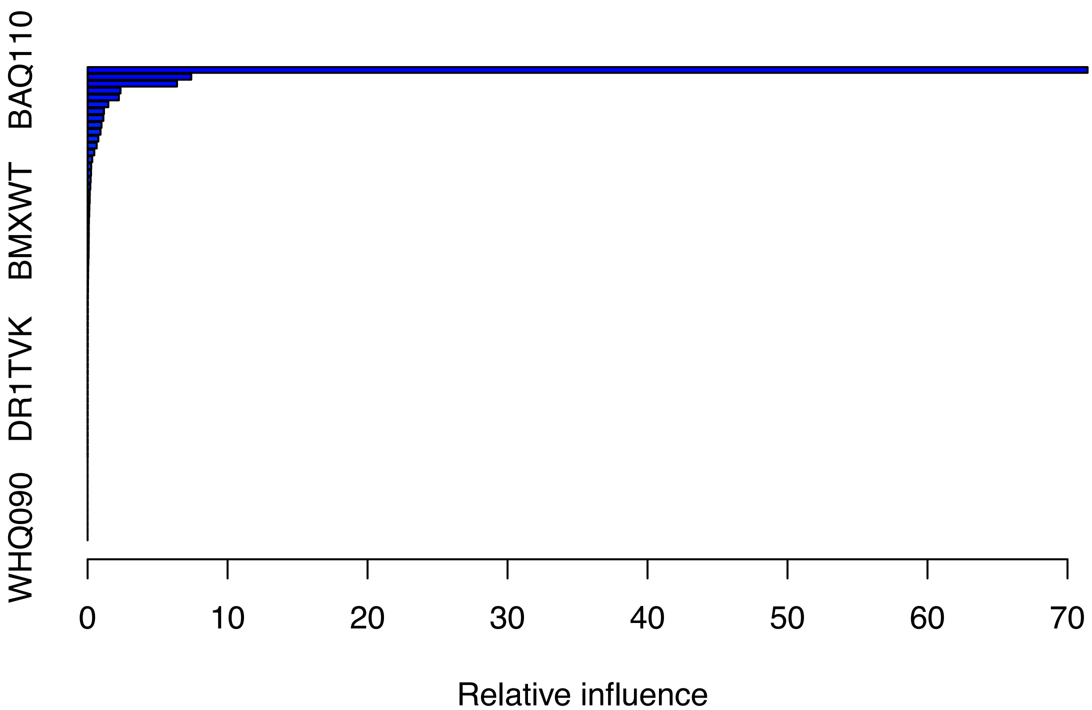
```
##   (Intercept)       BAQ1102       BAQ1103         BMXWT        BMXBMI
##  8.517192e+02  1.697197e+01 -2.087118e+02 -7.374139e-01  5.235773e+00
##       BMXARML       BMXARMC       BPQ0203      BPQ040A2      BPQ040A3
##  3.964687e+00 -4.376676e+00 -2.351978e+00 -2.542539e+01 -3.876846e+01
##       BPQ0802       BPQ0803      PEASCTM1       BPACSZ2       BPACSZ3
## -7.190623e+00 -4.173013e+01  3.551868e-02  6.807603e+00 -6.058680e+00
##      BPXPULS2      BPXPULS3        BPXML1      RIAGENDR     RIDRETH12
##  4.853400e+01  6.823117e+00  6.189174e+00  3.028525e+01  2.659507e+01
##     RIDRETH13     RIDRETH14     RIDRETH15      DMDBORN3     DMDCITZN2
##  5.051872e+01 -3.168107e+00  2.666211e+01  1.396548e+01  1.783092e+00
##     DMDCITZN3     DMDYRSUS2     DMDYRSUS3     DMDYRSUS4     DMDYRSUS5
## -2.740417e+01 -2.637374e+01 -1.893651e+01 -1.354515e+01 -1.198199e+01
##     DMDYRSUS6     DMDYRSUS7     DMDYRSUS8     DMDYRSUS9    DMDYRSUS10
## -2.197260e+01  2.035056e+01  4.351011e+01  6.005356e+01 -2.044114e+00
##     DMDEDUC31     DMDEDUC32     DMDEDUC33     DMDEDUC34     DMDEDUC35
## -4.909807e+01 -2.573701e+01 -1.329162e+01 -6.723886e+00 -7.480140e+00
##     DMDEDUC36     DMDEDUC37     DMDEDUC38     DMDEDUC39    DMDEDUC310
## -3.587800e+00 -2.076633e+00 -6.753074e+00 -5.045505e+00 -9.466870e+00
##    DMDEDUC311    DMDEDUC312    DMDEDUC313    DMDEDUC314    DMDEDUC315
## -2.399951e-02  1.280037e+01  1.108555e+01 -5.133416e+00  9.751599e+00
##    DMDEDUC316     DMDEDUC22     DMDEDUC23     DMDSCHOL2     DMDSCHOL3
## -2.986981e+01 -1.441914e+01 -3.914121e+00  5.850451e+00 -7.583290e+01
##     DMDMARTL2     DMDMARTL3     DMDMARTL4     DMDMARTL5      DMDHHSIZ
##  6.889937e+01 -2.103140e+01 -6.196685e+01 -9.044537e+01 -5.414385e+00
##     INDHHINC2     INDHHINC3     INDHHINC4     RIDEXPRG2     RIDEXPRG3
##  5.670383e+00  9.174085e-01  4.379960e+00  6.838274e+01  1.339556e+02
##      SIAPROXY      SIAINTRP      WTMEC2YR       DIQ0102       DIQ0103
## -2.810893e+01  3.194619e+00 -1.325421e-03 -1.218869e+01 -1.205650e+01
##       DIQ0502      DR1TKCAL      DR1TPROT      DR1TCARB      DR1TSUGR
##  6.939434e+00 -1.823330e-05 -1.893298e-01 -1.066784e-01 -2.099857e-03
##      DR1TFIBE      DR1TTFAT      DR1TSFAT      DR1TCHOL      DR1TATOC
##  6.729968e-01  7.183498e-02 -2.556159e-01  4.617368e-03 -2.345810e-01
##      DR1TVARA      DR1TBCAR        DR1TFA        DR1TVC        DR1TVK
##  3.811149e-03 -1.509218e-04  5.420686e-03  1.035097e-02  7.368983e-03
##      DR1TCALC      DR1TMAGN      DR1TIRON      DR1TSODI      DR1TPOTA
## -3.711308e-03  3.200761e-02  2.596337e-01 -1.059508e-04  2.851840e-03
##      DR1TCAFF      DR1TALCO       HSD0102       HSD0103       HSD0104
##  9.567941e-03 -1.613738e-01  4.076851e-01 -1.069744e+01 -6.429210e+01
##       HSQ5202       HSQ5203       MCQ0922       MCQ0923      MCQ160A2
##  5.564912e+00  8.021817e+01 -1.560212e+01 -2.215062e+01 -2.588887e+01
##      MCQ160A3      MCQ160B2      MCQ160B3      MCQ160C2      MCQ160D2
## -7.662913e+01 -2.886050e+00 -1.792985e+00 -3.237648e+01  5.018999e-02
##      MCQ160D3      MCQ160E2      MCQ160E3      MCQ160F2      MCQ160F3
## -2.481979e+00 -3.653836e+00 -5.782970e+01 -1.012243e+01 -1.601611e+02
##      MCQ160G2      MCQ160G3       MCQ2202       MCQ2203       VIQ1802
## -1.460260e+00 -3.961164e+01 -4.490419e+01 -3.111716e+01  7.421659e+00
##       VIQ2002       VIQ2202       VIQ2203        WHQ030       WHQ0703
## -8.237741e+01 -9.598683e+00 -2.752404e+01  2.981867e+00 -1.057471e+01
##       WHQ0902       WHQ0903
```

```
##  3.708354e-01 -3.636853e+01
```

```r
length(lasso.coef[lasso.coef!=0])
```

```
## [1] 122
```

```r
###################################
#(3) Boosting
###################################
library(gbm)
set.seed(1)
train = sample(1:nrow(complete_data6), round(nrow(complete_data6)*.70,0))
complete.test=complete_data6[-train ,"RIDAGEEX"]
boost.complete=gbm(RIDAGEEX~., data=complete_data6[train,], distribution="gaussian", n.trees=5000,
                   interaction.depth=4)
summary(boost.complete)
```




```
##                var      rel.inf
## BAQ110      BAQ110 7.143535e+01
## DMDEDUC2 DMDEDUC2 7.404063e+00
## DMDMARTL DMDMARTL 6.388204e+00
## RIDEXPRG RIDEXPRG 2.351285e+00
## BPQ080      BPQ080 2.232385e+00
## WTMEC2YR WTMEC2YR 1.479533e+00
## WHQ030      WHQ030 1.169594e+00
## BMXHT        BMXHT 1.125285e+00
## VIQ200      VIQ200 9.949143e-01
## BPQ040A    BPQ040A 9.242118e-01
## DMDHHSIZ DMDHHSIZ 7.682504e-01
## MCQ160A    MCQ160A 6.504267e-01
## DR1TKCAL DR1TKCAL 4.768298e-01
## BPACSZ      BPACSZ 3.323495e-01
## MCQ220      MCQ220 2.558173e-01
```

```
## RIDRETH1 RIDRETH1 2.449365e-01
## VIQ180      VIQ180 2.182878e-01
## BPXPULS    BPXPULS 1.921902e-01
## PEASCTM1 PEASCTM1 1.570518e-01
## MCQ092      MCQ092 1.507354e-01
## MCQ160G    MCQ160G 1.261704e-01
## BMXARML    BMXARML 1.148099e-01
## BMXWT        BMXWT 9.330742e-02
## BPXML1      BPXML1 8.528355e-02
## DR1TCARB DR1TCARB 8.510056e-02
## DMDYRSUS DMDYRSUS 7.965813e-02
## VIQ220      VIQ220 7.676979e-02
## BPQ020      BPQ020 7.136279e-02
## MCQ160E    MCQ160E 5.344210e-02
## DIQ010      DIQ010 4.037293e-02
## MCQ160F    MCQ160F 3.665424e-02
## BMXARMC    BMXARMC 2.560991e-02
## MCQ160C    MCQ160C 2.395323e-02
## MCQ160B    MCQ160B 1.371290e-02
## RIAGENDR RIAGENDR 1.131580e-02
## DR1TSUGR DR1TSUGR 1.104332e-02
## WHQ070      WHQ070 1.057529e-02
## DR1TPROT DR1TPROT 1.042185e-02
## BMXBMI      BMXBMI 9.148658e-03
## SIAPROXY SIAPROXY 7.019139e-03
## DMDSCHOL DMDSCHOL 6.730304e-03
## DMDEDUC3 DMDEDUC3 5.302524e-03
## HSD010      HSD010 5.299303e-03
## MCQ160D    MCQ160D 5.265898e-03
## DR1TCAFF DR1TCAFF 5.014389e-03
## DR1TVK      DR1TVK 4.672647e-03
## DR1TFIBE DR1TFIBE 4.647239e-03
## DR1TALCO DR1TALCO 3.488876e-03
## DR1TSODI DR1TSODI 2.975580e-03
## DR1TBCAR DR1TBCAR 2.914565e-03
## DR1TSFAT DR1TSFAT 2.580943e-03
## DR1TVARA DR1TVARA 2.207115e-03
## DR1TVC      DR1TVC 1.966783e-03
## DR1TMAGN DR1TMAGN 1.616674e-03
## DR1TFA      DR1TFA 1.511914e-03
## DR1TCHOL DR1TCHOL 1.400078e-03
## DR1TPOTA DR1TPOTA 9.813242e-04
## HSQ520      HSQ520 9.482384e-04
## DR1TTFAT DR1TTFAT 9.141468e-04
## INDHHINC INDHHINC 7.068246e-04
## DR1TCALC DR1TCALC 6.697678e-04
## DR1TIRON DR1TIRON 5.158035e-04
## DR1TATOC DR1TATOC 2.374112e-04
## DMDBORN    DMDBORN 0.000000e+00
## DMDCITZN DMDCITZN 0.000000e+00
## SIAINTRP SIAINTRP 0.000000e+00
## DIQ050      DIQ050 0.000000e+00
## DRABF        DRABF 0.000000e+00
## WHQ090      WHQ090 0.000000e+00
```

```r
yhat.boost=predict(boost.complete, newdata= complete_data6[-train,], n.trees=5000)
mean((yhat.boost - complete.test)^2)
```

```
## [1] 4275.505
```

```r
boost.complete=gbm(RIDAGEEX~.,data=complete_data6[train,], distribution= "gaussian", n.trees=5000,
                   interaction.depth=4, shrinkage =0.2, verbose=F)
yhat.boost=predict(boost.complete, newdata= complete_data6[-train,], n.trees=5000)
mean((yhat.boost - complete.test)^2)
```

```
## [1] 4037.711
```

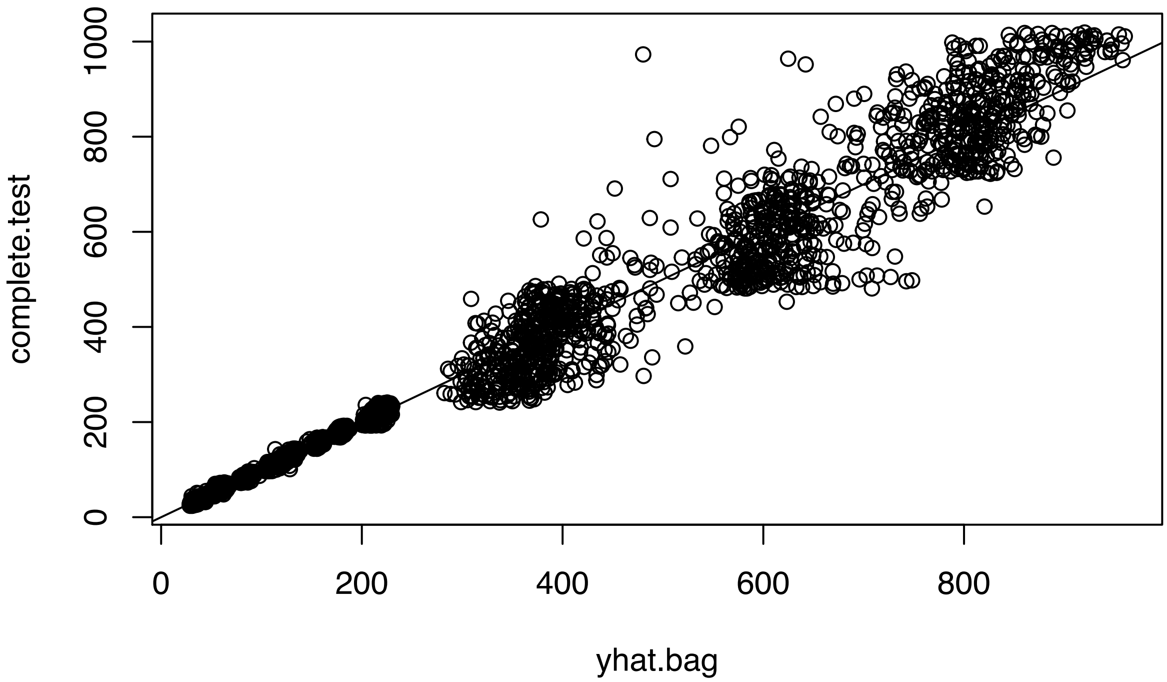
```r
###################################
#(4) Bagging/ Regression Tree
###################################
library(tree)
set.seed(1)
train = sample(1:nrow(complete_data6), round(nrow(complete_data6)*.70,0))
tree.data = tree(RIDAGEEX~., complete_data6, subset=train)
summary(tree.data)
```

```
##
## Regression tree:
## tree(formula = RIDAGEEX ~ ., data = complete_data6, subset = train)
## Variables actually used in tree construction:
## [1] "BAQ110"   "DMDEDUC2" "BMXHT"    "RIDEXPRG" "WTMEC2YR"
## Number of terminal nodes:  6
## Residual mean deviance:  7849 = 42250000 / 5383
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -320.400  -40.200   -1.395    0.000   37.800  632.600
```

```r
library(randomForest)
set.seed(1)
bag.data = randomForest(RIDAGEEX~., complete_data6, subset=train, mtry=15, importance =TRUE)
bag.data
```

```
##
## Call:
##  randomForest(formula = RIDAGEEX ~ ., data = complete_data6, mtry = 15,      importance = TRUE, subs
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 15
##
##           Mean of squared residuals: 3198.12
##                     % Var explained: 95.91
```

```r
yhat.bag = predict(bag.data , newdata=complete_data6[-train,])
complete.test=complete_data6[-train ,"RIDAGEEX"]
plot(yhat.bag , complete.test); abline (0,1)
```




```r
mean((yhat.bag - complete.test)^2)
```

```
## [1] 3239.908
```

```r
bag.complete = randomForest(RIDAGEEX~., data=complete_data6 , subset=train, mtry=15, ntree=25)
yhat.bag = predict(bag.complete , newdata=complete_data6[-train ,])
mean((yhat.bag - complete.test)^2)
```

```
## [1] 3447.584
```

```r
set.seed(1)
rf.complete = randomForest(RIDAGEEX ~., data=complete_data6, subset=train, mtry=18, ntree=30)
yhat.rf = predict(rf.complete , newdata = complete_data6[-train ,])
mean((yhat.rf - complete.test)^2)
```

```
## [1] 3332.395
```

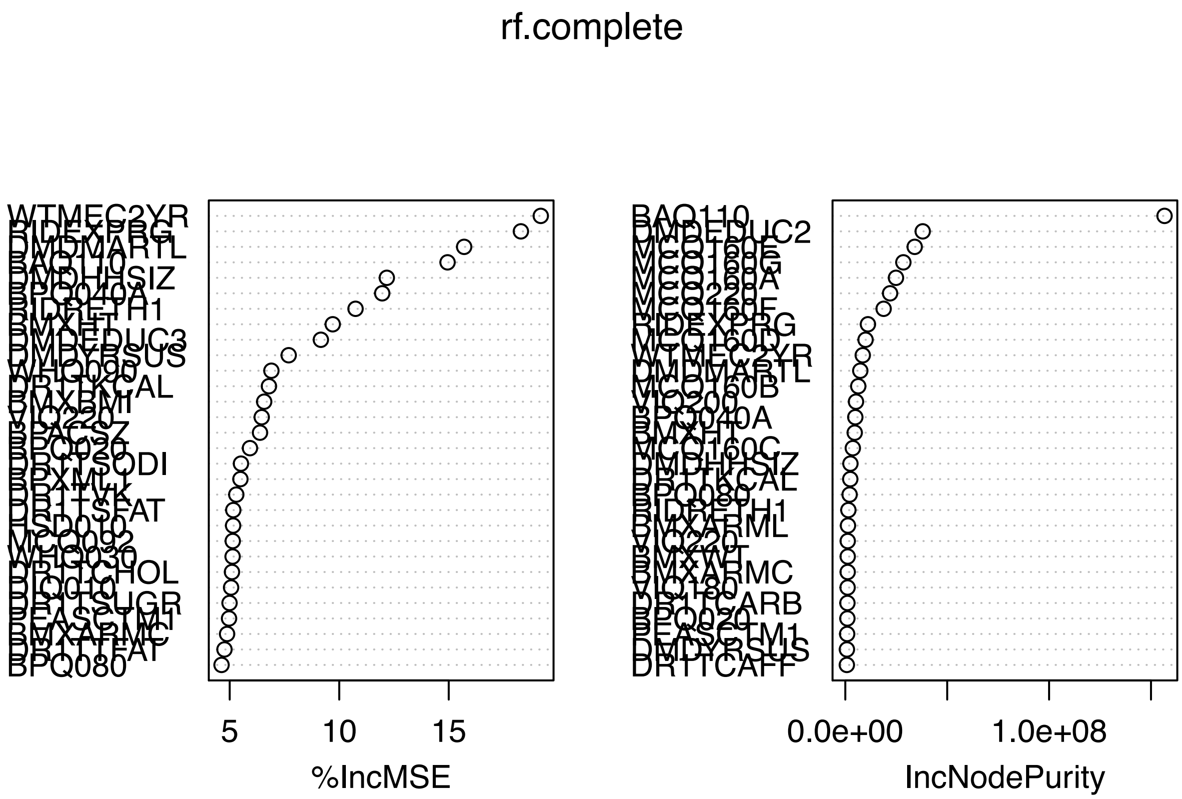
```r
set.seed(1)
rf.complete = randomForest(RIDAGEEX ~., data=complete_data6, subset=train, importance =TRUE, ntree=100)
yhat.rf = predict(rf.complete , newdata = complete_data6[-train ,])
mean((yhat.rf - complete.test)^2)
```

```
## [1] 3125.45
```

```r
head(importance(rf.complete))
```

```
##           %IncMSE IncNodePurity
## BAQ110  14.946437   156649195.2
## BMXWT    4.421037     1178238.7
## BMXHT    9.703259     4617524.0
## BMXBMI   6.573052      745396.1
## BMXARML  4.367520     1299176.8
## BMXARMC  4.880180     1155448.9
```

```r
head(varImpPlot(rf.complete))
```




```
##            %IncMSE IncNodePurity
## BAQ110  14.946437   156649195.2
## BMXWT    4.421037     1178238.7
## BMXHT    9.703259     4617524.0
## BMXBMI   6.573052      745396.1
## BMXARML  4.367520     1299176.8
## BMXARMC  4.880180     1155448.9
```

[5a]. Variables Contained in Age Dataset 3

```r
colnames(complete_data7)
```

```
##  [1] "BAQ110"   "BMXHT"    "BMXBMI"   "BMXARML"  "BMXARMC"  "BMXWAIST"
##  [7] "BPQ020"   "BPQ040A"  "BPQ080"   "PEASCTM1" "BPACSZ"   "BPXPULS"
## [13] "BPXML1"   "RIAGENDR" "RIDAGEEX" "DMDBORN"  "DMDCITZN" "DMDEDUC2"
## [19] "DMDMARTL" "DMDHHSIZ" "INDHHINC" "SIAPROXY" "SIAINTRP" "WTMEC2YR"
## [25] "DIQ010"   "DIQ050"   "DRABF"    "DR1TKCAL" "DR1TPROT" "DR1TCARB"
## [31] "DR1TSUGR" "DR1TFIBE" "DR1TTFAT" "DR1TSFAT" "DR1TCHOL" "DR1TIRON"
## [37] "DR1TSODI" "DR1TCAFF" "DR1TALCO" "HSD010"   "LBXWBCSI" "LBXMCVSI"
## [43] "MCQ092"   "MCQ160A"  "MCQ160B"  "MCQ160C"  "MCQ160D"  "MCQ160E"
## [49] "MCQ160F"  "MCQ160G"  "MCQ220"   "VIQ180"   "VIQ200"   "VIQ220"
## [55] "WHQ030"
```

[1] "BAQ110" "BMXHT" "BMXBMI" "BMXARML" "BMXARMC" "BMXWAIST" "BPQ020" "BPQ040A" "BPQ080" "PEASCTM1" "BPACSZ" "BPXPULS" "BPXML1"
[14] "RIAGENDR" "RIDAGEEX" "DMDBORN" "DMDCITZN" "DMDEDUC2" "DMDMARTL" "DMD-HHSIZ" "INDHHINC" "SIAPROXY" "SIAINTRP" "WTMEC2YR" "DIQ010" "DIQ050"
[27] "DRABF" "DR1TKCAL" "DR1TPROT" "DR1TCARB" "DR1TSUGR" "DR1TFIBE" "DR1TTFAT"

"DR1TSFAT" "DR1TCHOL" "DR1TIRON" "DR1TSODI" "DR1TCAFF" "DR1TALCO" [40] "HSD010" "LBXWBCSI" "LBXMCVSI" "MCQ092" "MCQ160A" "MCQ160B" "MCQ160C" "MCQ160D" "MCQ160E" "MCQ160F" "MCQ160G" "MCQ220" "VIQ180"
[53] "VIQ200" "VIQ220" "WHQ030"

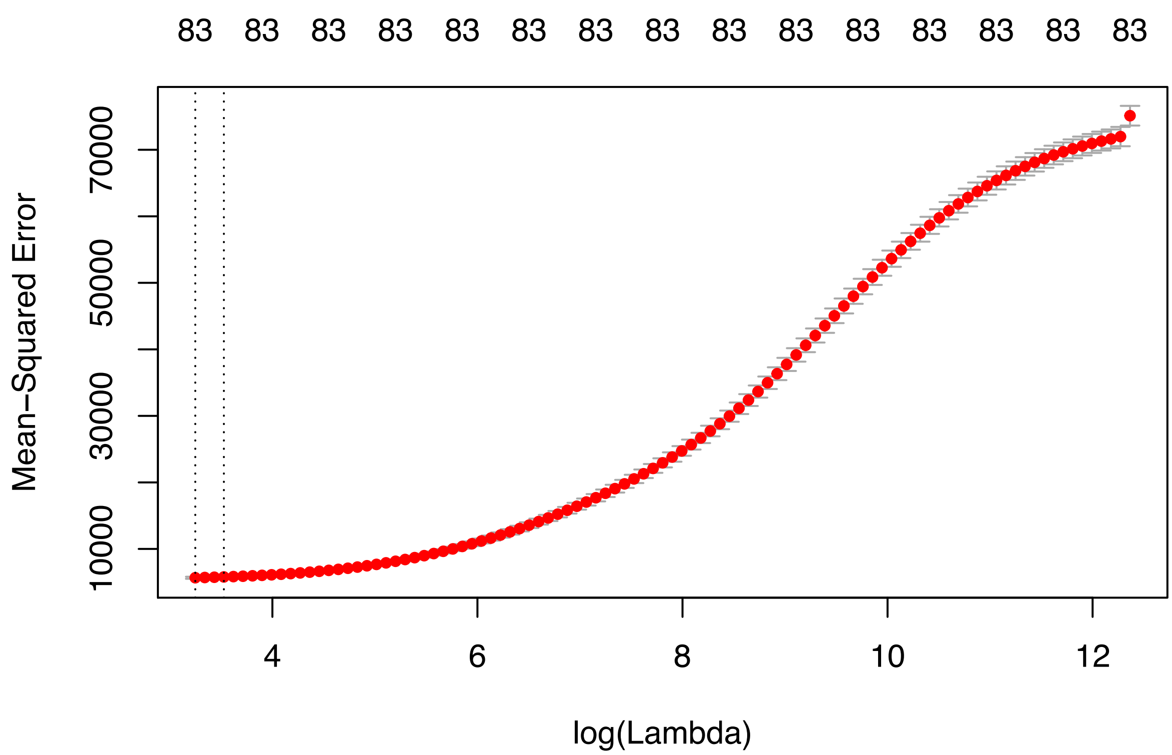
1. DIQ010: Doctor told you have diabetes (1-150 Years)
2. DIQ050: Taking insulin now (1-150 Years)
3. HSD010: General Health Condition (12-150 Years)
4. VIQ180: Eye surgery for near sightedness (12-150 Years)
5. VIQ200: Eye surgery for cataracts (12-150 Years)
6. VIQ220: Glasses/ contacts worn for distance; (12-150 Years)
7. DR1TKCAL: Energy (Calories) (0-150 Years)
8. DR1TPROT: Protein (0-150 Years)
9. DR1TCARB: Carbohydrate (0-150 Years)
10. DR1TSUGR: Total sugars (0-150 Years)
11. DR1TFIBE: Dietary Fiber (0-150 Years)
12. DR1TTFAT: Total Fat (0-150 Years)
13. DR1TSFAT: Total Saturated Fat (0-150 Years)
14. DR1TCHOL: Cholesterol (0-150 Years)
15. DR1TIRON: Iron (0-150 Years)
16. DR1TSODI: Sodium (0-150 Years)
17. DR1TCAFF: Caffeine (0-150 Years)
18. DR1TALCO: Alcohol (0-150 Years)
19. BMXARML: Upper Arm Length (0-150 Years)
20. BMXARMC: Arm Circumference (0-150 Years)
21. BMXHT: Standing Height (2-150 Years)
22. BMXBMI: Body Mass Index (2-150 Years)
23. BMXWAIST: Waist Circumference (2-150 Years)
24. PEASCTM1: Blood Pressure Time in Seconds (0-150 Years)
25. BPACSZ: Coded cuff size; Recode missing (8-150 Years)
26. BPQ020: Ever told you had high blood pressure (16-150 Years)
27. DMDMARTL: Martial Status
28. DMDEDUC2: Eudcation (20-150 Years)
29. RIAGENDR: Gender (0-150 Years)
30. DMDBORN: Country of Birth (0-150 Years)
31. DMDCITZN: Citizenship Status (0-150 Years)
32. DMDHHSIZ: Total number of people in the household (0-150 Years)
33. INDHHINC: Annual Household Income (0-150 Years)
34. SIAPROXY: Was a proxy used in SP interview
35. SIAINTRP: Was an interpreter used in SP interview
36. WTMEC2YR: Full Sample 2 Year MEC Exam Weight (0-150 Years)
37. DRABF: Breast-fed infant (either day) (0-150 Years)
38. BAQ110: Can you stand on your own? (40-150 Years)
39. BPXML1: Pulse Maximum Inflation Levels
40. LBXWBCSI: White blood cell count (1-150 Years)
41. LBXMCVSI: Mean cell volume (1-150 Years)
42. WHQ030: How do you consider your weight (16-150 Years)
43. MCQ160G: Ever told you had emphysema (20-150 Years)
44. BPQ040A: Taking prescription for hypertension
45. BPQ080: Doctor told you had high cholesterol (20-150 Years)
46. MCQ160A: Ever told you had arthritis (20-150 Years)
47. MCQ160B: Ever told you had congestive heart failure (20-150 Years)
48. MCQ160C: Ever told you had coronary heart disease (20-150 Years)
49. MCQ160D: Ever told you had angina (20-150 Years)

```
50. MCQ160E: Ever told you had heart attack; (20-150 Years)
51. MCQ160F: Ever told you had stroke; (20-150 Years)
52. MCQ220: Ever told you have cancer; (20-150 Years)
53. MCQ092: Ever receive blood transfusion; (6-150 Years)
54. BPXPULS: Is pulse irregular?
55. RIDAGEEX: Patient age when exam was given
```

[5b]. Rational Variables Contained in Age Dataset 3

[5d]. Methods/Reults for Predicting Age in Age Dataset 3

```
#complete_data7
####################################
#(1) Ridge Regression
####################################
x = model.matrix(RIDAGEEX~., data = complete_data7)[,-1]
y = complete_data7$RIDAGEEX
grid = 10^seq(10, -2, length =100)
ridge.mod = glmnet(x, y, alpha = 0, lambda = grid)

set.seed(1)
train = sample(1:nrow(x), round(nrow(x)*.70,0))
test = (-train)
y.test = y[test]
cv.out = cv.glmnet(x[train,], y[train], alpha = 0)
plot(cv.out)
```




```
bestlam = cv.out$lambda.min
bestlam
```

```
## [1] 25.7323
```

```
set.seed(1)
ridge.pred = predict(ridge.mod, s = bestlam, newx = x[test,])
mean((ridge.pred - y.test)^2)
```

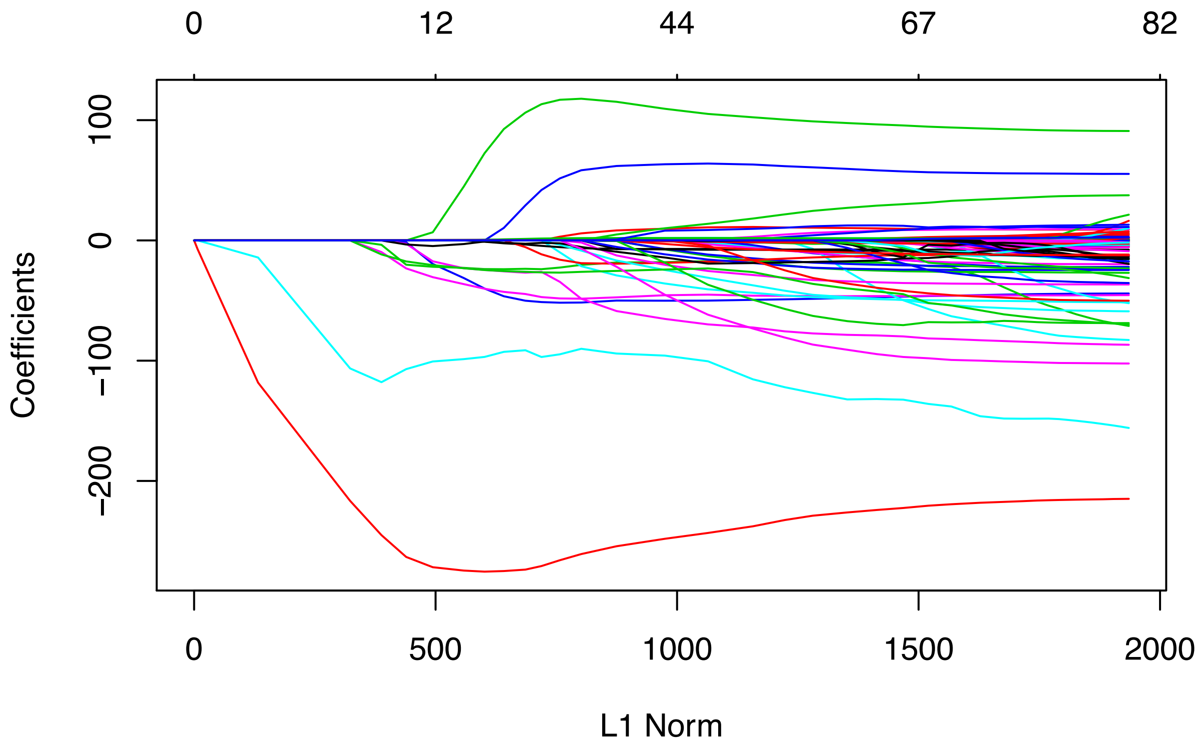
```
## [1] 5625.824
```

```
out = glmnet(x, y, alpha = 0)
predict(out, type = "coefficients", s = bestlam)[1:86,]
```

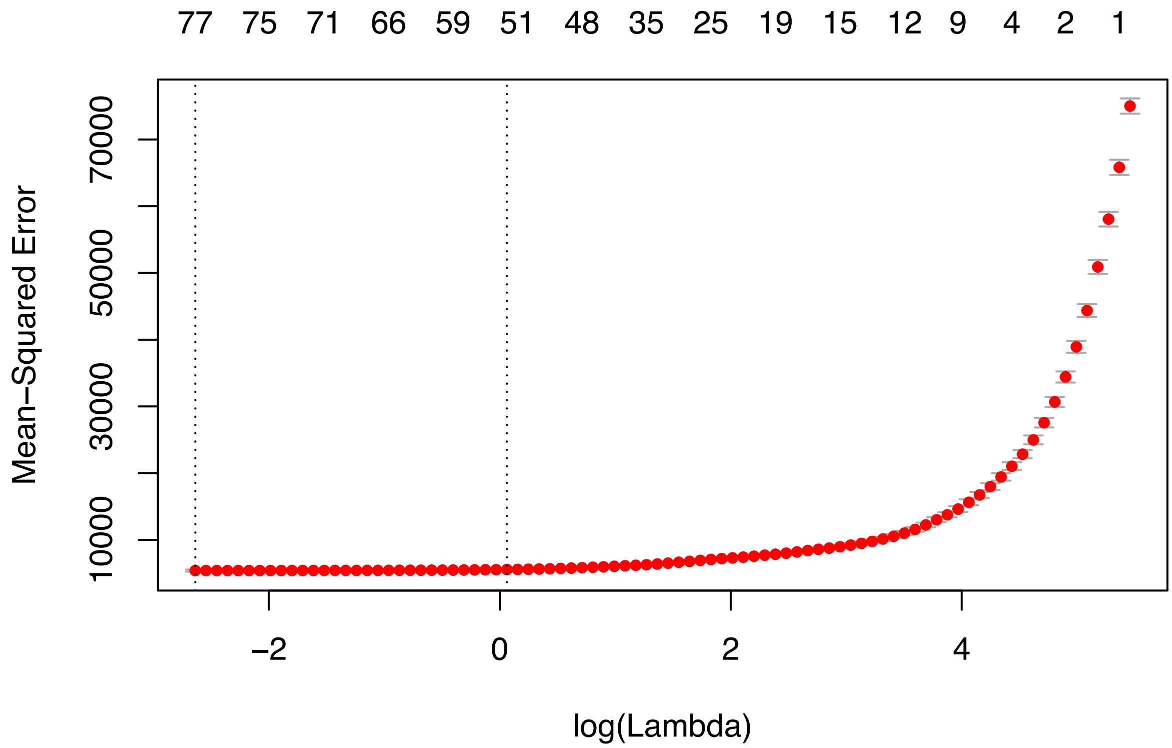
```
##   (Intercept)       BAQ1102       BAQ1103         BMXHT        BMXBMI 
##  6.123824e+02  3.338510e+01 -1.931061e+02  1.967967e-02 -8.771797e-01 
##       BMXARML       BMXARMC      BMXWAIST       BPQ0202       BPQ0203 
##  2.354587e+00 -1.343511e+00  1.011061e+00 -9.852070e+00 -1.805479e+01 
##      BPQ040A2      BPQ040A3       BPQ0802       BPQ0803      PEASCTM1 
## -3.381653e+01 -4.214942e+01 -1.174804e+01 -5.172194e+01  6.775465e-02 
##       BPACSZ2       BPACSZ3      BPXPULS2      BPXPULS3        BPXML1 
##  1.221838e+01 -1.979377e+01  6.476203e+01 -7.655404e+00  9.235582e+00 
##      RIAGENDR      DMDBORN2      DMDBORN3     DMDCITZN2     DMDCITZN3 
## -6.343498e+00  3.472176e+00  2.701942e+01 -1.738995e+01 -5.104767e+01 
##     DMDEDUC22     DMDEDUC23     DMDEDUC24     DMDMARTL2     DMDMARTL3 
## -1.695746e+01 -8.148760e+00 -2.100033e+01  1.015104e+02 -1.318431e+01 
##     DMDMARTL4     DMDMARTL5      DMDHHSIZ     INDHHINC2     INDHHINC3 
## -4.866297e+01 -5.776690e+01 -8.068927e+00  3.772617e+00 -2.590254e+00 
##     INDHHINC4      SIAPROXY      SIAINTRP      WTMEC2YR       DIQ0102 
##  8.602917e+00  4.339202e+00 -7.087747e+00 -7.070220e-04 -1.861583e+01 
##       DIQ0103       DIQ0502       DIQ0503        DRABF2        DRABF3 
## -2.934046e+01  9.274214e-01  0.000000e+00  0.000000e+00  0.000000e+00 
##      DR1TKCAL      DR1TPROT      DR1TCARB      DR1TSUGR      DR1TFIBE 
## -5.764236e-03 -7.351280e-02 -5.556045e-02 -3.222578e-02  1.074360e+00 
##      DR1TTFAT      DR1TSFAT      DR1TCHOL      DR1TIRON      DR1TSODI 
##  1.548272e-02 -1.364691e-01  4.332729e-04  3.546102e-01 -1.228206e-03 
##      DR1TCAFF      DR1TALCO       HSD0102       HSD0103       HSD0104 
##  2.308481e-02 -1.530535e-01 -8.293172e-01 -6.589120e+00 -1.866798e+01 
##      LBXWBCSI      LBXMCVSI       MCQ0922       MCQ0923      MCQ160A2 
## -1.103603e+00  2.078643e+00 -2.753796e+01 -5.016759e+01 -3.467796e+01 
##      MCQ160A3      MCQ160B2      MCQ160B3      MCQ160C2      MCQ160C3 
## -2.402963e+01  3.205490e+00 -1.876897e+01 -1.525800e+01 -1.745458e+01 
##      MCQ160D2      MCQ160D3      MCQ160E2      MCQ160E3      MCQ160F2 
##  1.166616e+00 -2.220802e+01 -3.202785e+00 -2.459353e+01  3.836596e+00 
##      MCQ160F3      MCQ160G2      MCQ160G3       MCQ2202       MCQ2203 
## -2.574890e+01  8.343701e+00 -2.348117e+01 -3.113632e+01 -2.671038e+01 
##       VIQ1802       VIQ1803       VIQ2002       VIQ2003       VIQ2202 
##  1.321332e+01 -1.574565e+01 -7.296598e+01 -1.674912e+01 -1.280387e+01 
##       VIQ2203 
## -2.112555e+01
```

```
####################################
#(2) Lasso
####################################
set.seed(1)
x = model.matrix(RIDAGEEX~., data = complete_data7)[,-1]
y = complete_data7$RIDAGEEX
train = sample(1:nrow(x), round(nrow(x)*.70,0))
test = (-train)
```

```r
y.test = y[test]

grid = 10^seq(10, -2, length =100)
lasso.mod = glmnet(x[train,], y[train], alpha = 1, lambda = grid)
plot(lasso.mod)
```




```r
set.seed(1)
cv.out = cv.glmnet(x[train,], y[train], alpha = 1)
plot(cv.out)
```




```
bestlam = cv.out$lambda.min
lasso.pred = predict(lasso.mod, s = bestlam, newx = x[test,])
mean((lasso.pred - y.test)^2)
```

```
## [1] 5499.755
```

```
out = glmnet(x, y, alpha = 1, lambda = grid)
lasso.coef = predict(out, type = "coefficients", s = bestlam)[1:86,]
lasso.coef[lasso.coef!=0]
```

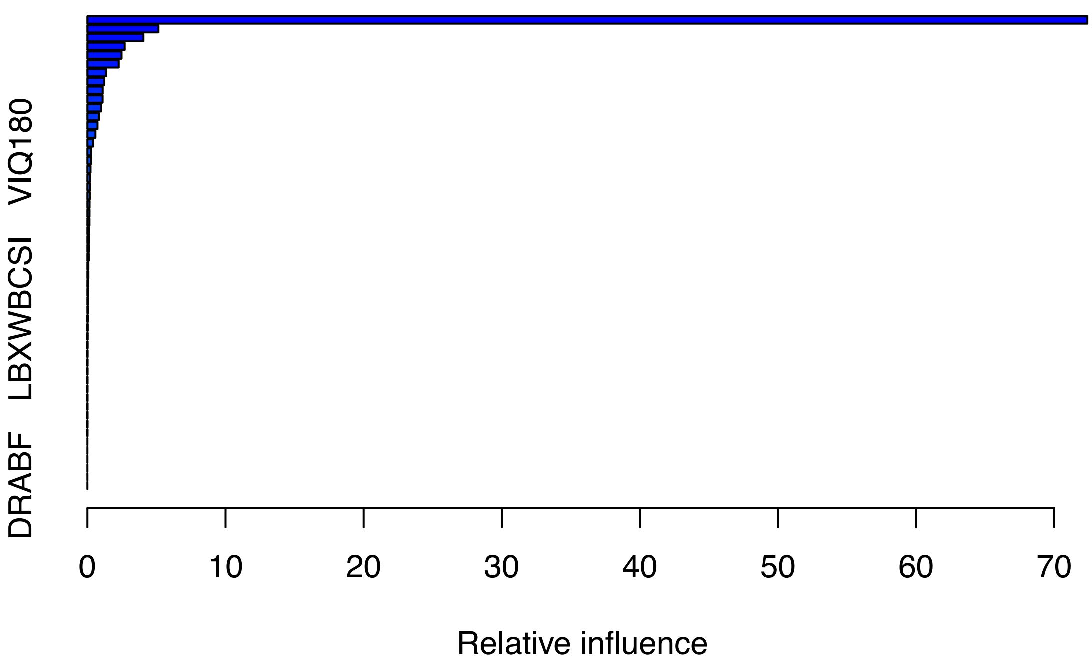
```
##   (Intercept)       BAQ1102       BAQ1103         BMXHT        BMXBMI
##  8.386766e+02  1.983021e+01 -2.195543e+02 -8.252432e-01 -2.864310e+00
##       BMXARML       BMXARMC      BMXWAIST       BPQ0203      BPQ040A2
##  4.361098e+00 -2.043337e+00  2.041245e+00 -3.775629e+00 -2.972615e+01
##      BPQ040A3       BPQ0802       BPQ0803      PEASCTM1       BPACSZ2
## -4.245286e+01 -1.362233e+01 -4.699984e+01  6.641887e-02  9.616008e+00
##       BPACSZ3      BPXPULS2        BPXML1      RIAGENDR      DMDBORN3
## -5.413929e+01  5.424982e+01  7.877762e+00 -5.285981e+00  2.919034e+01
##     DMDCITZN2     DMDCITZN3     DMDEDUC22     DMDEDUC23     DMDMARTL2
## -1.649461e+01 -5.993455e+01 -1.905507e+01 -1.017729e+01  8.908249e+01
##     DMDMARTL3     DMDMARTL4     DMDMARTL5      DMDHHSIZ     INDHHINC2
## -2.066332e+01 -5.471888e+01 -8.888980e+01 -7.395627e+00  5.568880e+00
##     INDHHINC3     INDHHINC4      SIAPROXY      SIAINTRP      WTMEC2YR
##  4.380906e-01  9.950436e+00 -3.176655e+01 -4.507603e+00 -7.530993e-04
##       DIQ0102       DIQ0103       DIQ0502       DR1TKCAL      DR1TPROT
## -9.166444e+00 -1.937249e+01  1.226031e+00 -4.156100e-03 -5.879091e-02
##      DR1TCARB      DR1TSUGR      DR1TFIBE      DR1TTFAT      DR1TSFAT
## -9.096111e-02  1.135281e-02  1.060412e+00  3.626073e-02 -1.642804e-01
##      DR1TCHOL      DR1TIRON      DR1TSODI      DR1TCAFF      DR1TALCO
## -3.023833e-03  4.561507e-01 -3.712504e-04  1.754065e-02 -1.343254e-01
##       HSD0102       HSD0103       HSD0104      LBXWBCSI      LBXMCVSI
```

```
## -4.317601e+00 -1.918333e+01 -1.799366e+01 -1.204375e+00  1.767397e+00
##       MCQ0922       MCQ0923      MCQ160A2      MCQ160A3      MCQ160B2
## -2.393161e+01 -5.634195e+01 -3.475030e+01 -2.602023e+01 -3.429380e+00
##      MCQ160C2      MCQ160C3      MCQ160D2      MCQ160D3      MCQ160E2
## -3.460778e+01  5.886460e-01 -3.160353e+00 -7.570516e-01 -2.422198e-01
##      MCQ160E3      MCQ160F2      MCQ160F3      MCQ160G3       MCQ2202
## -1.116600e+01 -1.039991e+01 -1.494125e+02 -8.769852e+00 -5.019763e+01
##       MCQ2203       VIQ1802       VIQ2002       VIQ2003       VIQ2202
## -6.536366e+01  8.953294e+00 -9.886716e+01 -1.608516e+01 -1.167050e+01
##       VIQ2203
## -6.298471e+01
```

```r
length(lasso.coef[lasso.coef!=0])
```

```
## [1] 76
```

```r
###################################
#(3) Boosting
###################################
library(gbm)
set.seed(1)
train = sample(1:nrow(complete_data7), round(nrow(complete_data7)*.70,0))
complete.test=complete_data7[-train ,"RIDAGEEX"]
boost.complete=gbm(RIDAGEEX~., data=complete_data7[train,],
                   distribution="gaussian", n.trees=5000, interaction.depth=4)
summary(boost.complete)
```




```
##               var      rel.inf
## BAQ110     BAQ110 7.239430e+01
## DMDMARTL DMDMARTL 5.141586e+00
## DMDEDUC2 DMDEDUC2 4.055232e+00
## MCQ160A   MCQ160A 2.696638e+00
## BPQ080     BPQ080 2.472152e+00
## WTMEC2YR WTMEC2YR 2.264375e+00
```

```
## VIQ200      VIQ200 1.363166e+00
## DMDHHSIZ DMDHHSIZ 1.224446e+00
## MCQ160B   MCQ160B 1.105935e+00
## BPQ040A   BPQ040A 1.101414e+00
## BMXHT       BMXHT 1.002820e+00
## MCQ160E   MCQ160E 8.239928e-01
## BPACSZ     BPACSZ 7.308963e-01
## WHQ030     WHQ030 5.803421e-01
## MCQ220     MCQ220 4.065777e-01
## VIQ180     VIQ180 2.614757e-01
## BPXPULS   BPXPULS 2.543025e-01
## PEASCTM1 PEASCTM1 2.249607e-01
## LBXMCVSI LBXMCVSI 1.927162e-01
## MCQ160G   MCQ160G 1.854764e-01
## BPQ020     BPQ020 1.709519e-01
## BMXWAIST BMXWAIST 1.563349e-01
## MCQ092     MCQ092 1.521335e-01
## BPXML1     BPXML1 1.492247e-01
## DR1TKCAL DR1TKCAL 1.166898e-01
## BMXARML   BMXARML 1.096073e-01
## MCQ160D   MCQ160D 1.081964e-01
## VIQ220     VIQ220 9.814768e-02
## DR1TSUGR DR1TSUGR 7.004912e-02
## BMXARMC   BMXARMC 6.934214e-02
## MCQ160C   MCQ160C 6.623720e-02
## DR1TCARB DR1TCARB 6.400086e-02
## DIQ010     DIQ010 4.248198e-02
## DR1TFIBE DR1TFIBE 3.595401e-02
## LBXWBCSI LBXWBCSI 2.082596e-02
## DR1TCAFF DR1TCAFF 1.643595e-02
## DR1TSFAT DR1TSFAT 1.339797e-02
## HSD010     HSD010 8.755418e-03
## DMDBORN   DMDBORN 8.429233e-03
## DR1TCHOL DR1TCHOL 6.203059e-03
## DR1TSODI DR1TSODI 5.117287e-03
## DR1TIRON DR1TIRON 4.807717e-03
## BMXBMI     BMXBMI 4.755276e-03
## DR1TPROT DR1TPROT 4.613053e-03
## DR1TTFAT DR1TTFAT 4.478910e-03
## RIAGENDR RIAGENDR 4.112659e-03
## DR1TALCO DR1TALCO 3.591463e-03
## MCQ160F   MCQ160F 1.721571e-03
## INDHHINC INDHHINC 5.943381e-04
## DMDCITZN DMDCITZN 0.000000e+00
## SIAPROXY SIAPROXY 0.000000e+00
## SIAINTRP SIAINTRP 0.000000e+00
## DIQ050     DIQ050 0.000000e+00
## DRABF       DRABF 0.000000e+00
```

```r
yhat.boost=predict(boost.complete, newdata= complete_data7[-train,], n.trees=5000)
mean((yhat.boost - complete.test)^2)
```

```
## [1] 4656.878
```

```r
set.seed(1)
boost.complete=gbm(RIDAGEEX~.,data=complete_data7[train ,], distribution= "gaussian", n.trees=5000,
                   interaction.depth=4, shrinkage =0.2, verbose=F)
yhat.boost=predict(boost.complete, newdata= complete_data7[-train,], n.trees=5000)
mean((yhat.boost - complete.test)^2)
```

```
## [1] 4604.231
```

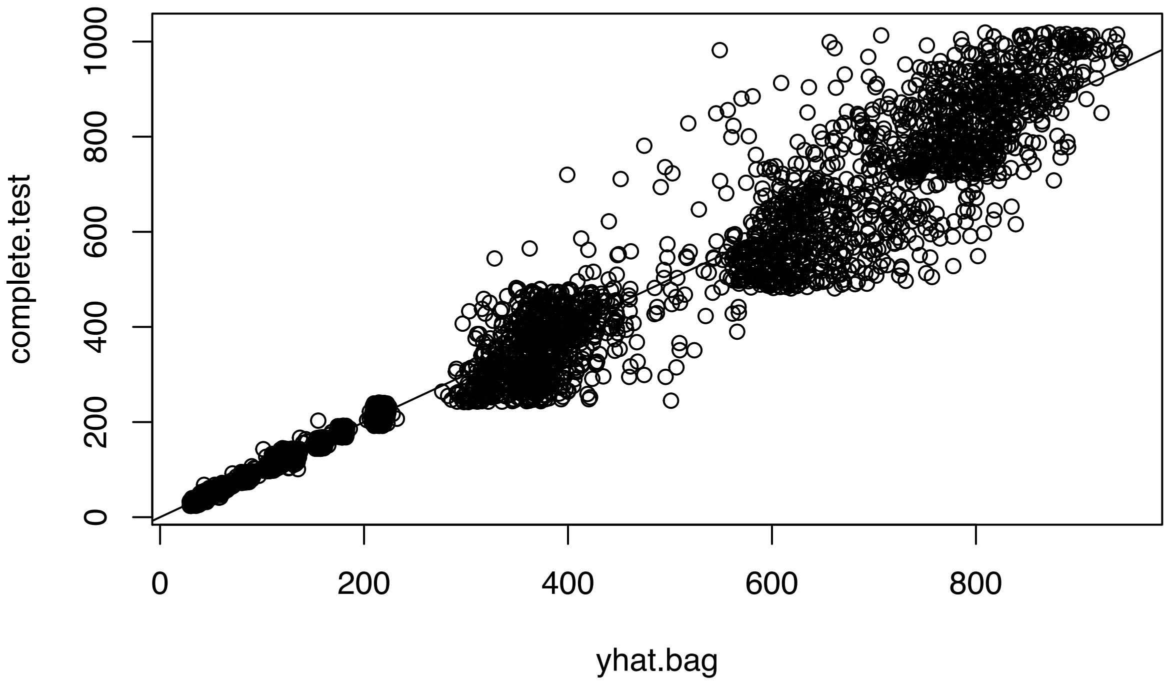
```r
###################################
#(4) Bagging/ Regression Tree
###################################
library(tree)
set.seed(1)
train = sample(1:nrow(complete_data7), nrow(complete_data7)/2)
tree.data = tree(RIDAGEEX~., complete_data7, subset=train)
summary(tree.data)
```

```
##
## Regression tree:
## tree(formula = RIDAGEEX ~ ., data = complete_data7, subset = train)
## Variables actually used in tree construction:
## [1] "BAQ110"   "MCQ160A"  "VIQ180"   "WTMEC2YR" "DMDHHSIZ"
## Number of terminal nodes:  6
## Residual mean deviance:  8553 = 30700000 / 3590
## Distribution of residuals:
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -323.2000  -43.0300   -0.1778    0.0000   37.8200  630.6000
```

```r
library(randomForest)
set.seed(1)
bag.data = randomForest(RIDAGEEX~., complete_data7, subset=train, mtry=15, importance =TRUE)
bag.data
```

```
##
## Call:
##  randomForest(formula = RIDAGEEX ~ ., data = complete_data7, mtry = 15,      importance = TRUE, subs
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 15
##
##           Mean of squared residuals: 4208.967
##                     % Var explained: 94.47
```

```r
yhat.bag = predict(bag.data , newdata=complete_data7[-train ,])
complete.test=complete_data7[-train ,"RIDAGEEX"]
plot(yhat.bag , complete.test); abline (0,1)
```




```
mean((yhat.bag - complete.test)^2)
```

```
## [1] 4134.786
```

```
bag.complete = randomForest(RIDAGEEX~., data=complete_data7 , subset=train, mtry=15, ntree=25)
yhat.bag = predict(bag.complete , newdata=complete_data7[-train ,])
mean((yhat.bag - complete.test)^2)
```

```
## [1] 4348.715
```

```
set.seed(1)
rf.complete = randomForest(RIDAGEEX ~., data=complete_data7, subset=train, mtry=18, ntree=30)
yhat.rf = predict(rf.complete , newdata = complete_data7[-train ,])
mean((yhat.rf - complete.test)^2)
```

```
## [1] 4200.583
```

```
set.seed(1)
rf.complete = randomForest(RIDAGEEX ~., data=complete_data7, subset=train, importance =TRUE, ntree=100)
yhat.rf = predict(rf.complete , newdata = complete_data7[-train ,])
mean((yhat.rf - complete.test)^2)
```

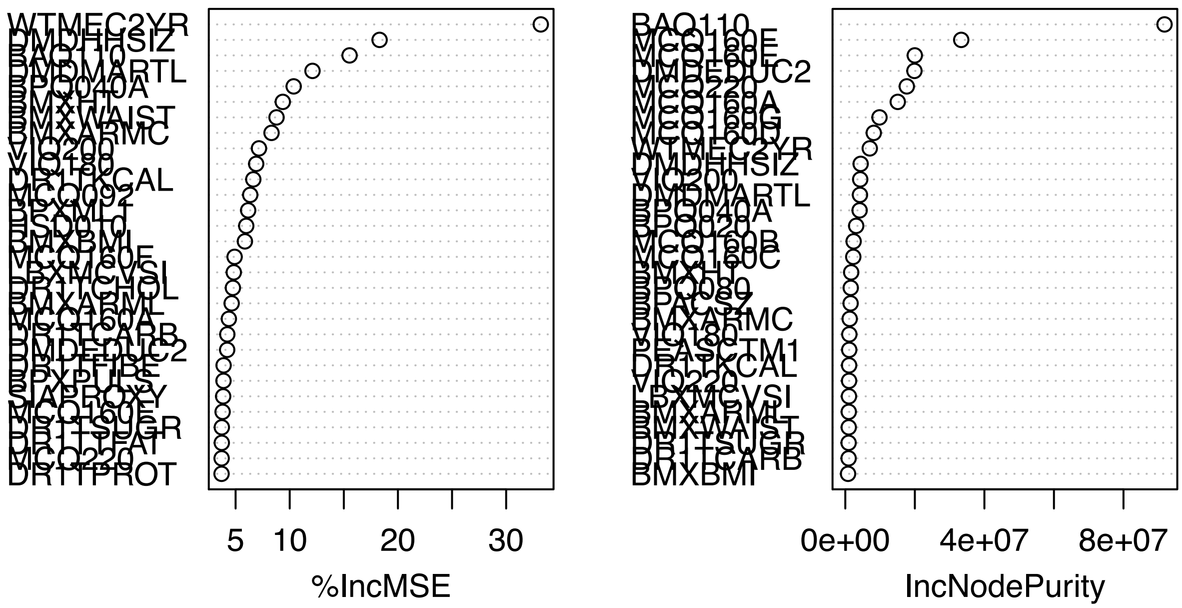
```
## [1] 4139.895
```

```
head(importance(rf.complete))
```

```
##             %IncMSE IncNodePurity
## BAQ110    15.531965    91809992.2
## BMXHT      9.359169     1689301.6
## BMXBMI     5.858725      785703.5
## BMXARML    4.636248      990935.3
## BMXARMC    8.319988     1260717.2
## BMXWAIST   8.782981      989342.4
```

```
head(varImpPlot(rf.complete))
```

rf.complete

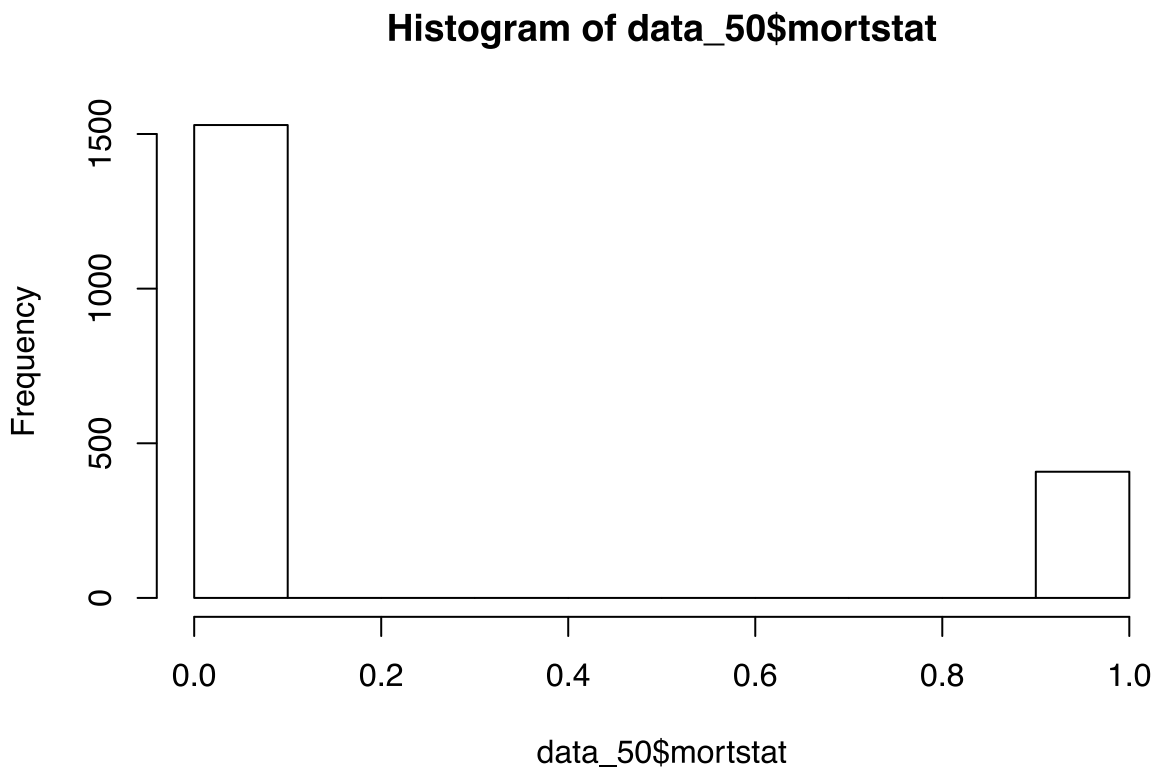



```
##              %IncMSE IncNodePurity
## BAQ110    15.531965    91809992.2
## BMXHT      9.359169     1689301.6
## BMXBMI     5.858725      785703.5
## BMXARML    4.636248      990935.3
## BMXARMC    8.319988     1260717.2
## BMXWAIST   8.782981      989342.4
```

VI. Appendix Predicting Mortaltiy

[6]. Histogram/ Summary of Exam Mortality

```
data_50 <- nhanes_data[which(as.numeric(nhanes_data$RIDAGEEX)>=50*12),]
hist(data_50$mortstat)
```




```
data_50$mortstat <- as.factor(data_50$mortstat)
summary(data_50$mortstat)
```

```
##    0    1 NA's 
## 1529  408  683
```

[7a]. Variables Contained in Mortality Dataset 1

```
colnames(new_data3)
```

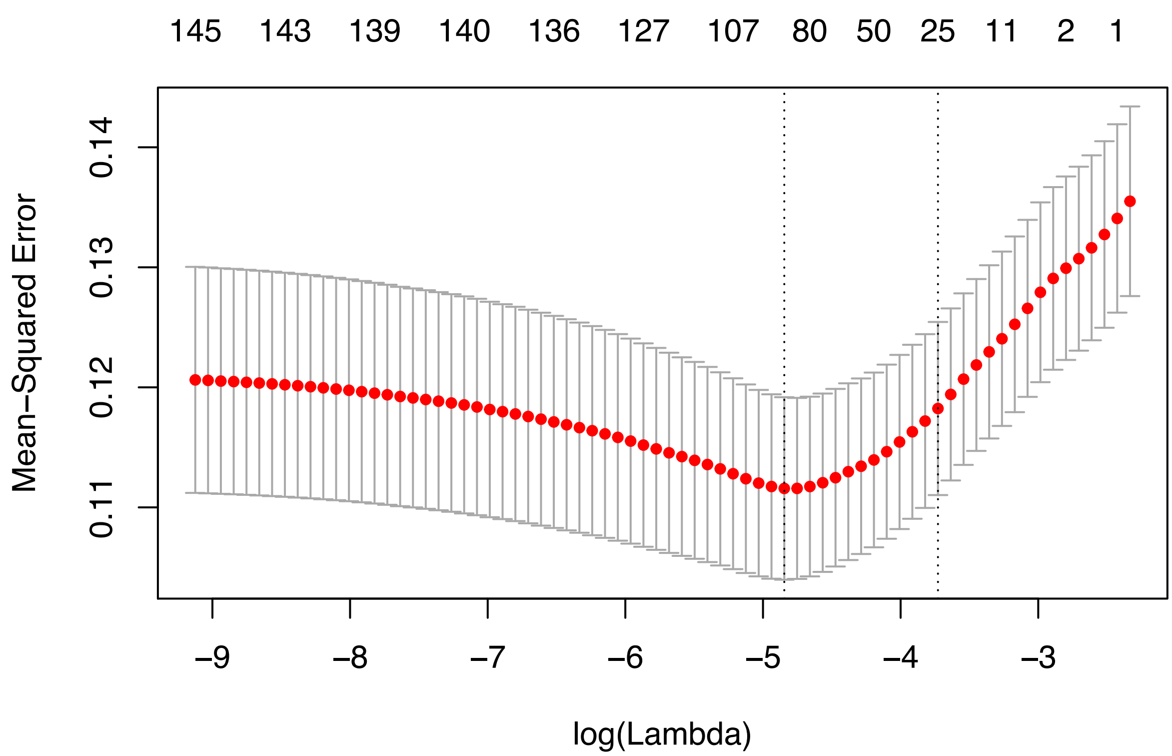
```
##  [1] "BAQ110"   "BMXWT"    "BMXBMI"   "BMXARMC"  "BMXWAIST" "BPQ020"  
##  [7] "BPQ040A"  "BPQ080"   "BPQ150D"  "BPXPULS"  "BPXML1"   "RIAGENDR"
## [13] "RIDAGEEX" "RIDRETH1" "DMQMILIT" "DMDCITZN" "DMDEDUC2" "DMDMARTL"
## [19] "DMDHHSIZ" "INDHHINC" "INDFMPIR" "SIAPROXY" "WTINT2YR" "WTMEC2YR"
## [25] "DIQ010"   "DIQ050"   "DIQ070"   "DRQSDT1"  "DRQSDT7"  "DR1TKCAL"
## [31] "DR1TCARB" "DR1TSUGR" "DR1TTFAT" "DR1TSFAT" "DR1TCHOL" "DR1TSODI"
## [37] "DR1TCAFF" "DR1_320"  "HSD010"   "HSQ480"   "HSQ490"   "LBXWBCSI"
## [43] "LBXLYPCT" "LBXNEPCT" "LBDNENO"  "LBXRBCSI" "LBXHGB"   "LBXHCT"  
## [49] "LBXRDW"   "MCQ010"   "MCQ092"   "MCQ160A"  "MCQ160B"  "MCQ160C" 
## [55] "MCQ160D"  "MCQ160E"  "MCQ160F"  "MCQ160G"  "MCQ220"   "SSQ011"  
## [61] "VIQ180"   "VIQ200"   "WHQ030"   "mortstat"
```

[7b]. Rational Variables Contained in Mortality Dataset 1

[7c]. Methods/Result in Mortality Dataset 1

```
####################################
#(1) LASSO
```

```
####################################
new_data3$mortstat <- as.factor(new_data3$mortstat)
set.seed(1)
library(glmnet)
x = model.matrix(mortstat ~ ., family = binomial(), data = new_data3)[,-1]
y = new_data3$mortstat
y <- as.numeric(y)
grid = 10^seq(10, -2, length =100)
set.seed(1)
train = sample(nrow(new_data3), round(nrow(new_data3)*.70),1)
test = (-train)
y.test = y[test]
lasso.mod = glmnet(x[train,], y[train], alpha = 1, lambda = grid)

cv.out = cv.glmnet(x[train,], y[train], alpha = 1)
plot(cv.out)
```




```
bestlam = cv.out$lambda.min
lasso.pred = predict(lasso.mod, s = bestlam, newx = x[test,])
mean((lasso.pred - y.test)^2)
```

```
## [1] 0.1400119
```

```
out = glmnet(x, y, alpha = 1, lambda = grid)
lasso.coef = predict(out, type = "coefficients", s = bestlam)[1:197,]
lasso.coef[lasso.coef!=0]
```

```
##   (Intercept)       BAQ1102         BMXWT      BPQ150D2     RIAGENDR2
##  1.249243e+00  1.239113e-01 -1.040379e-05 -5.001568e-02 -5.650472e-02
##      RIDAGEEX     DMQMILIT2     DMDMARTL2     DMDMARTL3      DMDHHSIZ
##  6.075250e-04 -2.608340e-02  4.440393e-02  5.160392e-02 -5.944580e-03
```

```
##      INDFMPIR      WTMEC2YR       DIQ0102       DIQ0502       DIQ0702
## -1.292140e-05 -1.138559e-05 -4.022663e-02 -6.085373e-02  8.080110e-04
##       DR1TTFAT      DR1TSFAT       DR1TSODI       HSD0102       HSD0103
##  6.972747e-07 -8.130286e-07 -7.059259e-06  1.302925e-02  1.545123e-01
##        HSQ490      LBXWBCSI      LBXLYPCT      LBXNEPCT       LBDNENO
##  7.336629e-04 -4.560889e-04 -8.059031e-05  5.938068e-05  3.718482e-04
##      LBXRBCSI        LBXHGB    LBXRDW11.2    LBXRDW11.7    LBXRDW11.8
## -3.598318e-04 -1.277783e-04 -2.374182e-02 -1.572168e-04 -2.174533e-02
##      LBXRDW12    LBXRDW12.1    LBXRDW12.2    LBXRDW12.4    LBXRDW12.5
## -1.054971e-02 -5.412828e-02 -3.547138e-03 -3.097887e-02 -3.968242e-02
##    LBXRDW12.8    LBXRDW13.3    LBXRDW13.4    LBXRDW13.8      LBXRDW14
##  5.936288e-02  2.354970e-02  1.738092e-02 -2.724373e-02  5.543967e-02
##    LBXRDW14.3    LBXRDW14.4    LBXRDW14.5    LBXRDW14.6    LBXRDW15.1
##  3.541417e-02  9.898820e-02  9.055567e-02  1.175875e-01  5.953711e-02
##    LBXRDW15.4    LBXRDW15.9      LBXRDW16    LBXRDW16.3      LBXRDW17
## -1.043760e-01  1.515357e-01  3.699877e-01  1.309459e-01  2.567196e-01
##    LBXRDW17.3    LBXRDW17.9    LBXRDW18.1    LBXRDW18.7    LBXRDW18.9
##  3.682311e-01  2.490907e-01  2.006548e-01  4.015453e-01 -2.786714e-03
##    LBXRDW21.2    LBXRDW22.4       MCQ0922      MCQ160B2      MCQ160B3
##  7.890160e-02  3.134633e-01 -5.699282e-02 -7.293410e-02 -4.821758e-02
##      MCQ160E2      MCQ160G2      MCQ160G3       MCQ2203       SSQ0112
## -3.088246e-04 -1.567894e-01 -1.657830e-01  3.898853e-03 -5.041225e-02
##       WHQ0302       WHQ0303
##  9.900324e-02  1.347301e-02
```

```r
length(lasso.coef[lasso.coef!=0])
```

```
## [1] 67
```

```r
###################################
#(2) Boosting
###################################
library(gbm)
set.seed(1)
new_data3$mortstat <- as.numeric(new_data3$mortstat)
new_data3$mortstat <- new_data3$mortstat - 1
new_data3$mortstat <- as.numeric(new_data3$mortstat)
train = sample(1:nrow(new_data3), round(nrow(new_data3)*.70,0))
complete.test=new_data3[-train ,"mortstat"]
boost.complete=gbm(mortstat~., data=new_data3[train,], distribution="bernoulli", n.trees=5000,
                   interaction.depth=4)
yhat.boost=predict(boost.complete, newdata= new_data3[-train,], n.trees=5000)
mean((yhat.boost - complete.test)^2)
```

```
## [1] 6.201917
```

```r
set.seed(1)
boost.complete=gbm(mortstat~.,data=new_data3[train,], distribution= "bernoulli", n.trees=5000,
                   interaction.depth=4, shrinkage =0.2, verbose=F)
yhat.boost=predict(boost.complete, newdata= new_data3[-train,], n.trees=5000)
mean((yhat.boost - complete.test)^2)
```

```
## [1] 2489.238
```

```r
set.seed(1)
boost.complete=gbm(mortstat~.,data=new_data3[train,], distribution= "bernoulli", n.trees=5000,
                   interaction.depth=4, shrinkage =0.1, verbose=F)
```

```r
yhat.boost=predict(boost.complete, newdata= new_data3[-train,], n.trees=5000)
mean((yhat.boost - complete.test)^2)
```

```
## [1] 701.4015
```

```r
####################################
#(3) SVM Classifier
####################################
library(e1071)
set.seed(2)
new_data3$mortstat <- as.factor(new_data3$mortstat)
train = sample(nrow(new_data3), round(nrow(new_data3)*.70,0))
tune.out = tune(svm, mortstat~., data=new_data3[train,], kernel ="linear",
                ranges=list(cost=c(0.00001, 0.0001, 0.001, 0.01, 0.1, 1)  ))
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##     1
##
## - best performance: 0.1791747
##
## - Detailed performance results:
##    cost     error dispersion
## 1 1e-05 0.1819961 0.02838594
## 2 1e-04 0.1819961 0.02838594
## 3 1e-03 0.1819961 0.02838594
## 4 1e-02 0.1819961 0.02838594
## 5 1e-01 0.1819697 0.02282519
## 6 1e+00 0.1791747 0.01788158
```

```r
bestmod = tune.out$best.model
table(true=new_data3[-train,"mortstat"], pred=predict(tune.out$best.model, newdata=new_data3[-train,]))
```

```
##     pred
## true   0   1
##    0 344  27
##    1  68  18
```

```r
svmfit=svm(mortstat ~., data=new_data3 , kernel="linear", cost=bestmod$cost, gamma=bestmod$gamma, scale=
summary(svmfit)
```

```
##
## Call:
## svm(formula = mortstat ~ ., data = new_data3, kernel = "linear",
##     cost = bestmod$cost, gamma = bestmod$gamma, scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
```

```
##        cost:  1
##       gamma:  0.005076142
##
## Number of Support Vectors:  589
##
##  ( 321 268 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

```
(table(true=new_data3[-train,"mortstat"], pred=predict(tune.out$best.model, newdata=new_data3[-train,]))
 table(true=new_data3[-train,"mortstat"], pred=predict(tune.out$best.model, newdata=new_data3[-train,]))
(table(true=new_data3[-train,"mortstat"], pred=predict(tune.out$best.model, newdata=new_data3[-train,]))
 table(true=new_data3[-train,"mortstat"], pred=predict(tune.out$best.model, newdata=new_data3[-train,]))
 table(true=new_data3[-train,"mortstat"], pred=predict(tune.out$best.model, newdata=new_data3[-train,]))
 table(true=new_data3[-train,"mortstat"], pred=predict(tune.out$best.model, newdata=new_data3[-train,]))
```

```
## [1] 0.2078775
```

```
set.seed(2)
tune.out = tune(svm, mortstat~., data=new_data3[train,], kernel ="radial",
                ranges=list(cost=c(0.001, 0.01, 0.1, 1, 5, 10)))
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##     5
##
## - best performance: 0.1735585
##
## - Detailed performance results:
##    cost     error dispersion
## 1 1e-03 0.1819961 0.02666999
## 2 1e-02 0.1819961 0.02666999
## 3 1e-01 0.1819961 0.02666999
## 4 1e+00 0.1819961 0.02666999
## 5 5e+00 0.1735585 0.03329578
## 6 1e+01 0.1810880 0.02679073
```

```
bestmod = tune.out$best.model
table(true=new_data3[-train,"mortstat"], pred=predict(tune.out$best.model, newdata=new_data3[-train,]))
```

```
##     pred
## true   0   1
##    0 365   6
##    1  77   9
```

```
svmfit=svm(mortstat ~., data=new_data3 , kernel="radial", cost=bestmod$cost, gamma=bestmod$gamma, scale=
summary(svmfit)
```

```
##
## Call:
## svm(formula = mortstat ~ ., data = new_data3, kernel = "radial",
##     cost = bestmod$cost, gamma = bestmod$gamma, scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  5
##       gamma:  0.005076142
##
## Number of Support Vectors:  673
##
##  ( 394 279 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

```
(table(true=new_data3[-train,"mortstat"], pred=predict(tune.out$best.model, newdata=new_data3[-train,])
 table(true=new_data3[-train,"mortstat"], pred=predict(tune.out$best.model, newdata=new_data3[-train,])
(table(true=new_data3[-train,"mortstat"], pred=predict(tune.out$best.model, newdata=new_data3[-train,])
 table(true=new_data3[-train,"mortstat"], pred=predict(tune.out$best.model, newdata=new_data3[-train,])
 table(true=new_data3[-train,"mortstat"], pred=predict(tune.out$best.model, newdata=new_data3[-train,])
 table(true=new_data3[-train,"mortstat"], pred=predict(tune.out$best.model, newdata=new_data3[-train,])
```

```
## [1] 0.1816193
```

[8a]. Variables Contained in Mortality Dataset 2

```
colnames(new_data4)
```

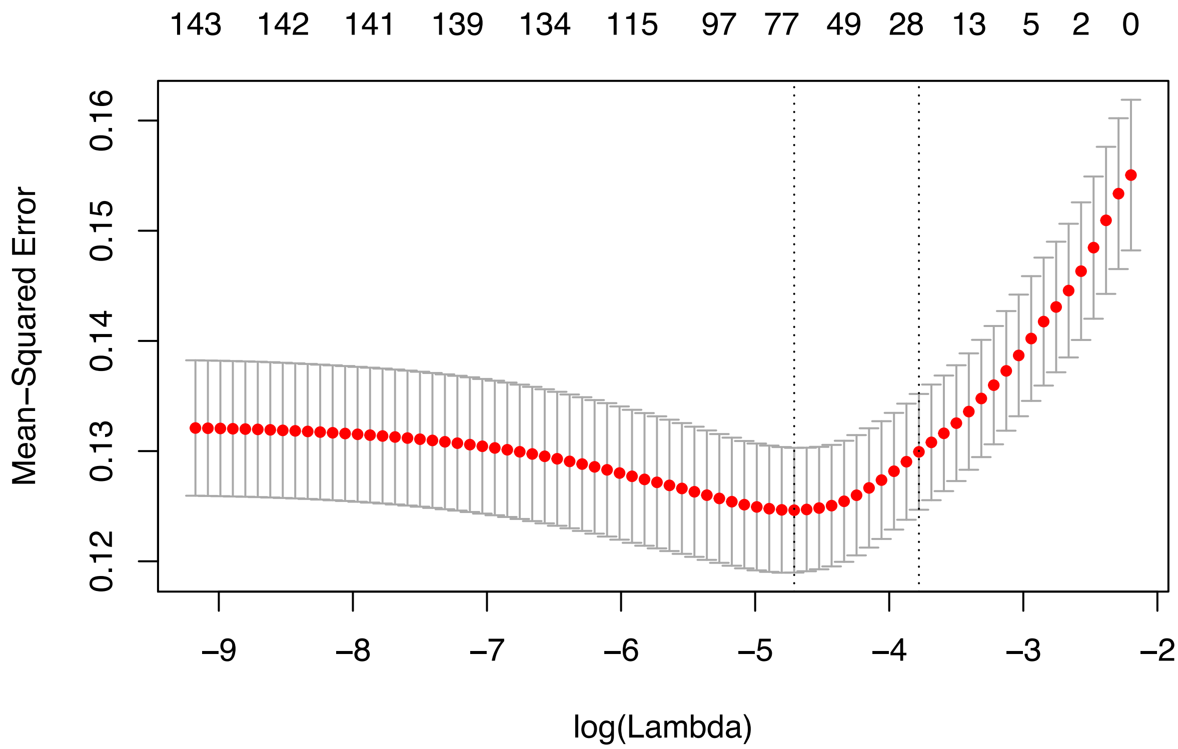
```
##  [1] "BAQ110"   "RIAGENDR" "RIDAGEEX" "DMQMILIT" "DMDMARTL" "DMDHHSIZ"
##  [7] "INDHHINC" "WTMEC2YR" "DIQ010"   "DIQ050"   "DR1TSFAT" "DR1TSODI"
## [13] "HSD010"   "LBXWBCSI" "LBXLYPCT" "LBXNEPCT" "LBDNENO"  "LBXRBCSI"
## [19] "LBXRDW"   "MCQ092"   "MCQ160B"  "MCQ160G"  "SSQ011"   "WHQ030"
## [25] "mortstat"
```

[8b]. Rational for Variables Contained in Mortality Dataset 2

[8c]. Methods/Results for Mortality Dataset 2

```
###################################
#(1) LASSO
###################################
new_data4$mortstat <- as.factor(new_data4$mortstat)
```

```r
library(glmnet)
x = model.matrix(mortstat ~ ., family = binomial(), data = new_data4)
y = new_data4$mortstat
y <- as.numeric(y)
grid = 10^seq(10, -2, length =100)
set.seed(1)
train = sample(1781, 1246)
test = (-train)
y.test = y[test]
lasso.mod = glmnet(x[train,], y[train], alpha = 1, lambda = grid)

cv.out = cv.glmnet(x[train,], y[train], alpha = 1)
plot(cv.out)
```




```r
bestlam = cv.out$lambda.min
lasso.pred = predict(lasso.mod, s = bestlam, newx = x[test,])
mean((lasso.pred - y.test)^2)
```

```
## [1] 0.138928
```

```r
out = glmnet(x, y, alpha = 1, lambda = grid)
lasso.coef = predict(out, type = "coefficients", s = bestlam)[1:43,]
lasso.coef[lasso.coef!=0]
```

```
##   (Intercept)       BAQ1102       BAQ1103     RIAGENDR2      RIDAGEEX
##  9.002580e-01  8.915141e-02  8.946703e-02 -6.649303e-02  6.662743e-04
##     DMQMILIT2     DMDMARTL2     DMDMARTL3     DMDMARTL5      DMDHHSIZ
## -2.512140e-02  4.224902e-02  4.480473e-02  5.564528e-02 -4.070466e-03
##      WTMEC2YR       DIQ0102       DIQ0502      DR1TSODI       HSD0102
## -8.524615e-06 -3.645735e-02 -4.070202e-02 -2.133170e-06  2.002525e-02
##       HSD0103       HSD0104  LBXWBCSI10.1  LBXWBCSI10.2  LBXWBCSI10.3
```

```
##  1.414235e-01  3.931380e-02 -1.172064e-01  3.250851e-01  9.790779e-02
##  LBXWBCSI10.4  LBXWBCSI10.5  LBXWBCSI10.6  LBXWBCSI10.7    LBXWBCSI11
##  2.647731e-01  9.513987e-02  7.207326e-02 -1.147795e-01  3.146294e-02
##  LBXWBCSI11.5
##  1.232138e-01
```

```
###################################
#(2) Boosting
###################################
library(gbm)
set.seed(1)
new_data3$mortstat <- as.numeric(new_data3$mortstat)
train = sample(1:nrow(new_data4), round(nrow(new_data4)*.70,0))
complete.test=new_data4[-train ,"mortstat"]
boost.complete=gbm(mortstat~., data=new_data4[train,],
                   distribution="bernoulli", n.trees=5000, interaction.depth=4)
yhat.boost=predict(boost.complete, newdata= new_data4[-train,], n.trees=5000)
mean((yhat.boost - complete.test)^2)
```

```
## Warning in Ops.factor(yhat.boost, complete.test): '-' not meaningful for
## factors
```

```
## [1] NA
```

```
boost.complete=gbm(mortstat~.,data=new_data4[train ,], distribution= "bernoulli", n.trees=5000,
                   interaction.depth=4, shrinkage =0.2, verbose=F)
yhat.boost=predict(boost.complete, newdata= new_data4[-train,], n.trees=5000)
mean((yhat.boost - complete.test)^2)
```

```
## Warning in Ops.factor(yhat.boost, complete.test): '-' not meaningful for
## factors
```

```
## [1] NA
```

```
###################################
#(5) SVM Classifier
###################################
library(e1071)
set.seed(2)
train = sample(nrow(new_data3), round(nrow(new_data3)*.70,0))
tune.out = tune(svm, mortstat~., data=new_data4[train,], kernel ="linear",
                ranges=list(cost=c(0.00001, 0.0001,0.001, 0.01, 0.1, 1) ))
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##   0.1
##
## - best performance: 0.1734967
##
## - Detailed performance results:
##    cost     error dispersion
```

```
## 1 1e-05 0.2129254 0.02514597
## 2 1e-04 0.2129254 0.02514597
## 3 1e-03 0.2129254 0.02514597
## 4 1e-02 0.2119820 0.02592822
## 5 1e-01 0.1734967 0.02968943
## 6 1e+00 0.1913243 0.03062784
```

```
bestmod = tune.out$best.model
summary(bestmod)
```

```
##
## Call:
## best.tune(method = svm, train.x = mortstat ~ ., data = new_data4[train,
##     ], ranges = list(cost = c(1e-05, 1e-04, 0.001, 0.01, 0.1,
##     1)), kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.1
##       gamma:  0.004854369
##
## Number of Support Vectors:  452
##
##  ( 238 214 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

```
table(true=new_data4[-train,"mortstat"], pred=predict(tune.out$best.model, newdata=new_data4[-train,]))
```

```
##     pred
## true   0   1
##    0 556  24
##    1 105  30
```

```
svmfit=svm(mortstat ~., data=new_data4 , kernel="linear", cost=bestmod$cost, gamma=bestmod$gamma, scale=
summary(svmfit)
```

```
##
## Call:
## svm(formula = mortstat ~ ., data = new_data4, kernel = "linear",
##     cost = bestmod$cost, gamma = bestmod$gamma, scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.1
##       gamma:  0.004854369
##
## Number of Support Vectors:  733
```

```
##
##  ( 386 347 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

```
(table(true=new_data4[-train,"mortstat"], pred=predict(tune.out$best.model, newdata=new_data4[-train,])
 table(true=new_data4[-train,"mortstat"], pred=predict(tune.out$best.model, newdata=new_data4[-train,])
(table(true=new_data4[-train,"mortstat"], pred=predict(tune.out$best.model, newdata=new_data4[-train,])
 table(true=new_data4[-train,"mortstat"], pred=predict(tune.out$best.model, newdata=new_data4[-train,])
 table(true=new_data4[-train,"mortstat"], pred=predict(tune.out$best.model, newdata=new_data4[-train,])
 table(true=new_data4[-train,"mortstat"], pred=predict(tune.out$best.model, newdata=new_data4[-train,])
```

```
## [1] 0.1804196
```

```
set.seed(2)
tune.out = tune(svm, mortstat~., data=new_data4[train,], kernel ="radial",
                ranges=list(cost=c(0.001, 0.01, 0.1, 1, 5, 10)))
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##    10
##
## - best performance: 0.1763446
##
## - Detailed performance results:
##    cost     error dispersion
## 1 1e-03 0.2129078 0.03718399
## 2 1e-02 0.2129078 0.03718399
## 3 1e-01 0.2129078 0.03718399
## 4 1e+00 0.2119644 0.03950960
## 5 5e+00 0.1782137 0.03323982
## 6 1e+01 0.1763446 0.03011651
```

```
bestmod = tune.out$best.model
summary(bestmod)
```

```
##
## Call:
## best.tune(method = svm, train.x = mortstat ~ ., data = new_data4[train,
##     ], ranges = list(cost = c(0.001, 0.01, 0.1, 1, 5, 10)), kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  10
```

```
##       gamma:  0.004854369
##
## Number of Support Vectors:  458
##
##  ( 245 213 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

```r
table(true=new_data4[-train,"mortstat"], pred=predict(tune.out$best.model, newdata=new_data4[-train,]))
```

```
##     pred
## true   0   1
##    0 558  22
##    1 103  32
```

```r
svmfit=svm(mortstat ~., data=new_data4 , kernel="radial", cost=bestmod$cost, gamma=bestmod$gamma, scale=
summary(svmfit)
```

```
##
## Call:
## svm(formula = mortstat ~ ., data = new_data4, kernel = "radial",
##     cost = bestmod$cost, gamma = bestmod$gamma, scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  10
##       gamma:  0.004854369
##
## Number of Support Vectors:  743
##
##  ( 398 345 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

```r
(table(true=new_data4[-train,"mortstat"], pred=predict(tune.out$best.model, newdata=new_data4[-train,])
 table(true=new_data4[-train,"mortstat"], pred=predict(tune.out$best.model, newdata=new_data4[-train,])
(table(true=new_data4[-train,"mortstat"], pred=predict(tune.out$best.model, newdata=new_data4[-train,])
 table(true=new_data4[-train,"mortstat"], pred=predict(tune.out$best.model, newdata=new_data4[-train,])
 table(true=new_data4[-train,"mortstat"], pred=predict(tune.out$best.model, newdata=new_data4[-train,])
 table(true=new_data4[-train,"mortstat"], pred=predict(tune.out$best.model, newdata=new_data4[-train,])
```

```
## [1] 0.1748252
```

VII. Reference

[1] https://wwwn.cdc.gov/nchs/nhanes/ContinuousNhanes/Default.aspx?BeginYear=2003