title: "Blood-Pressure-Analysis"
output: pdf_document

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.0.5

## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.1.1     v dplyr   1.0.6
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1

## Warning: package 'ggplot2' was built under R version 4.0.5

## Warning: package 'tibble' was built under R version 4.0.5

## Warning: package 'tidyr' was built under R version 4.0.5

## Warning: package 'readr' was built under R version 4.0.5

## Warning: package 'purrr' was built under R version 4.0.5

## Warning: package 'dplyr' was built under R version 4.0.5

## Warning: package 'stringr' was built under R version 4.0.4

## Warning: package 'forcats' was built under R version 4.0.5

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(NHANES)
```

```
## Warning: package 'NHANES' was built under R version 4.0.5
```

```
library(car)
```

```
## Warning: package 'car' was built under R version 4.0.5

## Loading required package: carData

## Warning: package 'carData' was built under R version 4.0.3
```

```
##
## Attaching package: 'car'

## The following object is masked from 'package:dplyr':
##
##     recode

## The following object is masked from 'package:purrr':
##
##     some
```

```r
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 4.0.5

## Loading required package: Matrix

##
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack

## Loaded glmnet 4.1-1
```

```r
library(rms)
```

```
## Warning: package 'rms' was built under R version 4.0.5

## Loading required package: Hmisc

## Warning: package 'Hmisc' was built under R version 4.0.5

## Loading required package: lattice

## Loading required package: survival

## Loading required package: Formula

## Warning: package 'Formula' was built under R version 4.0.3

##
## Attaching package: 'Hmisc'

## The following objects are masked from 'package:dplyr':
##
##     src, summarize
## The following objects are masked from 'package:base':
##
##     format.pval, units

## Loading required package: SparseM

## Warning: package 'SparseM' was built under R version 4.0.4

##
## Attaching package: 'SparseM'

## The following object is masked from 'package:base':
##
##     backsolve

##
## Attaching package: 'rms'

## The following objects are masked from 'package:car':
##
##     Predict, vif
```

```
library(MASS)
```

```
##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select
```

```
small.nhanes <- na.omit(NHANES[NHANES$SurveyYr=="2011_12"
& NHANES$Age > 17,c(1,3,4,8:11,13,17,20,21,25,46,50,51,52,61)])
small.nhanes <- as.data.frame(small.nhanes %>%
group_by(ID) %>% filter(row_number()==1) )
nrow(small.nhanes)
```

```
## [1] 743
```

```
## Checking whether there are any ID that was repeated. If not ##
## then length(unique(small.nhanes$ID)) and nrow(small.nhanes) are same ##
length((small.nhanes$ID))
```

```
## [1] 743
```

```
mean((small.nhanes$Age))
```

```
## [1] 50.67026
```

```
mean((small.nhanes$Gender) == 'male')
```

```
## [1] 0.5854643
```

```
set.seed(1006092577)
train <- small.nhanes[sample(seq_len(nrow(small.nhanes)), size = 500),]
nrow(train)
```

```
## [1] 500
```

```
length(which(small.nhanes$ID %in% train$ID))
```

```
## [1] 500
```

```
test <- small.nhanes[!small.nhanes$ID %in% train$ID,]
nrow(test)
```

```
## [1] 243
```

```
train <- train[-c(1)]
test <- test[-c(1)]
```

```
train <- train[-c(6)] # remove Income
model.full <- lm(BPSysAve ~ ., data = train)
vif(model.full)
```

```
##                Gendermale                       Age                Race3Black
##                  2.324307                  1.700961                  3.659003
##             Race3Hispanic              Race3Mexican                Race3White
##                  2.314496                  3.092994                  6.560808
##                Race3Other   Education9 - 11th Grade       EducationHigh School
##                  2.037498                  2.982943                  3.755887
##     EducationSome College     EducationCollege Grad  MaritalStatusLivePartner
##                  4.432732                  4.145793                  1.892999
##      MaritalStatusMarried MaritalStatusNeverMarried    MaritalStatusSeparated
##                  2.867711                  2.366799                  1.169672
##      MaritalStatusWidowed                   Poverty                    Weight
##                  1.666645                  1.442439                115.049094
##                    Height                       BMI          DepressedSeveral
##                 30.961186                 97.926426                  1.137605
##             DepressedMost             SleepHrsNight           SleepTroubleYes
##                  1.318728                  1.111570                  1.259948
##             PhysActiveYes               SmokeNowYes
##                  1.289039                  1.363561
```

```
train <- train[-c(8)] # remove poverty
model.full <- lm(BPSysAve ~ ., data = train)
vif(model.full)
```

```
##                 Gendermale                        Age                 Race3Black
##                   2.257597                   1.673187                   3.648253
##              Race3Hispanic               Race3Mexican                 Race3White
##                   2.312977                   3.092541                   6.544272
##                 Race3Other   Education9 - 11th Grade      EducationHigh School
##                   2.036117                   2.982873                   3.754540
##     EducationSome College     EducationCollege Grad MaritalStatusLivePartner
##                   4.431085                   4.145203                   1.887349
##      MaritalStatusMarried MaritalStatusNeverMarried    MaritalStatusSeparated
##                   2.832880                   2.365250                   1.165646
##      MaritalStatusWidowed                   Poverty                     Weight
##                   1.656863                   1.432865                   8.322221
##                        BMI          DepressedSeveral              DepressedMost
##                   7.930600                   1.137563                   1.302233
##             SleepHrsNight           SleepTroubleYes             PhysActiveYes
##                   1.111564                   1.231223                   1.287279
##               SmokeNowYes
##                   1.363481
```

First let's take a brief look at the 17 different predictors that we can use. ### Gender ### Categorical - This is an interesting one to look at, to see if there are any biological differences between male and female in terms of blood pressure.

Age

Continuous - Age definitely will play a role in determine how healthy one is.

Race

Categorical - Is there a clear distinction between races in terms of blood pressure? Perhaps biological? Or maybe it is a result of different cultures/lifestyles? It is unclear whether this predictor actually will play a role in prediction.

Education

Categorical - Level of education is an interesting predictor to check - maybe those more highly educated or healthier as a result?

MaritalStatus

Categorical - A predictor that doesn't seem to be able to help explain/predict one's blood pressure, but this analysis will still try to take it into account.

HHIncome

Categorical - A predictor that separates individuals into brackets of income. Perhaps higher income results in healthier lifestyles?

Poverty

Continuous - Similar to HHIncome.

Weight

Continuous - Weight is always a decent indicator for health. However it can be misleading - heavier individuals can either be unhealthy in terms of obesity or healthy due to larger muscle mass or height.

Height

Continuous - Height seems less likely to be an adequate predictor on blood pressure, however it may be indicator of other factors that affect one's blood pressure.

BMI

Continuous - Similar to weight.

Depressed

Categorical - This is a very interesting predictor to look out. One can say that mental health can play a role in physical health. Specifically those who are happier with themselves may be healthier vs. depressed individuals who neglect themselves.

SleepTrouble

Categorical - This is an interesting predictor. Trouble sleeping may be a symptom of unhealthy blood pressure instead of the other way around, as well as just deeply correlated with the Depressed predictor.

PhysActive

Categorical - This is self-explanatory. Being physically active surely must play an effect on blood pressure.

SmokeNow

Categorical - Perhaps the most important predictor in this dataset.

Now that we have looked at the predictors, we should clean up the dataset. We are primarily concerned with the predictor SmokeNow. There are a number of predictors here that are not going to be very useful for analysis. For instance Education, MaritalStatus, HHIncome, Poverty, Depressed are not very useful predictors (or if they are, then they are explained by other factors). We can remove BMI as well, since Weight and Height are both used to find BMI, and remove SleepHrsNight, since this is very correlated with SleepTrouble. Then,

For the heck of it, let's try to make a full model based on all 17 variables. Since we have a very large observation pool (500 in training dataset and 143 in the testing dataset), the risk of overfitting is not as high.

```r
train <- small.nhanes[sample(seq_len(nrow(small.nhanes)), size = 500),]
test <- small.nhanes[!small.nhanes$ID %in% train$ID,]
train <- train[-c(1)]
test <- test[-c(1)]
train <- train[-c(5, 6, 10, 12, 14)]
test <- test[-c(5, 6, 10, 12, 14)]
# Refit the full model and check for VIFs again
```

```r
model.full <- lm(BPSysAve ~ ., data = train)
summary(model.full)
```

```
##
## Call:
## lm(formula = BPSysAve ~ ., data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -36.729 -10.521  -1.083   9.359  79.942
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)             125.280352  19.543642   6.410 3.47e-10 ***
## Gendermale                5.847632   2.083821   2.806  0.00522 **
## Age                       0.412391   0.049427   8.343 7.66e-16 ***
## Race3Black                6.447169   4.548265   1.418  0.15698
## Race3Hispanic             4.453693   4.953415   0.899  0.36904
## Race3Mexican              3.796192   4.865722   0.780  0.43566
## Race3White                1.662817   4.051335   0.410  0.68167
## Race3Other               -2.714326   5.488148  -0.495  0.62112
## Education9 - 11th Grade   1.064642   3.456626   0.308  0.75822
## EducationHigh School      1.459522   3.250078   0.449  0.65358
## EducationSome College     1.988971   3.310862   0.601  0.54829
## EducationCollege Grad    -0.942636   3.587172  -0.263  0.79283
## Poverty                  -0.617969   0.513416  -1.204  0.22932
## Weight                    0.008115   0.041865   0.194  0.84639
## Height                   -0.158828   0.118581  -1.339  0.18107
## SleepHrsNight             0.102679   0.568329   0.181  0.85670
## PhysActiveYes            -1.356258   1.612418  -0.841  0.40069
## SmokeNowYes              -1.461263   1.726404  -0.846  0.39774
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 16.19 on 482 degrees of freedom
## Multiple R-squared:  0.2114, Adjusted R-squared:  0.1836
## F-statistic: 7.601 on 17 and 482 DF,  p-value: < 2.2e-16
```

```r
anova(model.full)
```

```
## Analysis of Variance Table
##
## Response: BPSysAve
##                Df Sum Sq Mean Sq  F value    Pr(>F)
## Gender          1   1937  1937.1   7.3866  0.006808 **
## Age             1  27030 27030.2 103.0721 < 2.2e-16 ***
## Race3           5   2472   494.4   1.8853  0.095381 .
## Education       4   1068   267.0   1.0180  0.397514
## Poverty         1    502   502.3   1.9152  0.167027
## Weight          1      2     2.2   0.0083  0.927623
## Height          1    543   543.0   2.0704  0.150827
## SleepHrsNight   1     21    21.0   0.0801  0.777337
## PhysActive      1    124   123.8   0.4721  0.492353
## SmokeNow         1    188   187.9   0.7164  0.397738
## Residuals      482 126402   262.2
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
p <- length(model.full$coefficients) - 1
vif(model.full)
```

```
##              Gendermale                     Age              Race3Black
##                2.031488                1.380063                4.044485
##           Race3Hispanic            Race3Mexican              Race3White
##                2.638458                3.016036                6.831929
##              Race3Other Education9 - 11th Grade    EducationHigh School
##                2.099287                2.840384                3.755939
##   EducationSome College   EducationCollege Grad                 Poverty
##                4.606668                4.070168                1.352454
##                  Weight                  Height            SleepHrsNight
##                1.334822                2.419930                1.124613
##           PhysActiveYes             SmokeNowYes
##                1.235357                1.374619
```

Overall the VIFs are less than 5. Lucky!

Okay this model is messy. Nearly every predictor has a high p-value, including SmokeNow. This obviously is a problem. (Ovefitting?) Check the relationship with SmokeNow.

```
# The important one!
model.15 <- lm(BPSysAve ~ SmokeNow, data = train)
summary(model.15)
```

```
##
## Call:
## lm(formula = BPSysAve ~ SmokeNow, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -41.278 -12.439  -1.439   9.722  90.722
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  126.278      1.035 121.975   <2e-16 ***
## SmokeNowYes   -4.839      1.617  -2.993   0.0029 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 17.78 on 498 degrees of freedom
## Multiple R-squared:  0.01767,    Adjusted R-squared:  0.0157
## F-statistic: 8.957 on 1 and 498 DF,  p-value: 0.002901
```

```
anova(model.15)
```

```
## Analysis of Variance Table
```

```
##
## Response: BPSysAve
##            Df Sum Sq Mean Sq F value   Pr(>F)
## SmokeNow    1   2832 2832.09  8.9572 0.002901 **
## Residuals 498 157458  316.18
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Clearly we see there is a linear relationship between SmokeNow and BPSysAve.

Diagnostic Checking for Full Model

Let's do some model diagnostics!!! Check fitted values vs standardized residuals. If we see any pattern then we know

```r
resid <- rstudent(model.full)
fitted <- predict(model.full)

pdf("model_full.pdf", height = 8, width = 16)
par(family = 'mono', mfrow = c(1,2))
plot(model.full)
#qqnorm(resid)
#qqline(resid)
#plot(resid ~ fitted, type = "p", xlab = "Fitted Values",
#     ylab = "Standardized Residual", cex.lab = 1.2,
#     col = "red")
#lines(lowess(fitted, resid), col = "blue")
#dev.off()
plot(train$BPSysAve ~ fitted, type = "p", xlab = "Fitted Values",
     ylab = "BPSysAve", cex.lab = 1.2,
     col = "red")
abline(lm(train$BPSysAve ~ fitted), lwd = 2, col = "blue")
lines(lowess(fitted, train$BPSysAve), col = "red")
dev.off()
```

```
## pdf
##   2
```

The Q-Q plot with the standardized residuals are are one-to-one except for the extreme quantiles. This likely means the standard residuals deviate from the normal. Our fitted values to Standardized Residual plot indicates a somewhat linear pattern by lowess, however clearly the errors have high variance, as the plot instead of looking linear, looks more uniformly "cloud-like" around a line. This indicates errors are linear but also have high variance. We can also take a look at the response vs. fitted plots.

```r
pdf("model_full2.pdf", height = 8, width = 16)
plot(model.full)
par(family = 'mono')
plot(train$BPSysAve ~ fitted, type = "p", xlab = "Fitted Values",
     ylab = "BPSysAve", cex.lab = 1.2,
     col = "red")
abline(lm(train$BPSysAve ~ fitted), lwd = 2, col = "blue")
lines(lowess(fitted, train$BPSysAve), col = "red")
dev.off()
```

```
## pdf
##   2
```

Once again we see that the lowess line is close to the actual line. That is, the relationship between fitted values and the actual values is linear in nature. Therefore once again we see that the linearity assumption is met, however the large variance around the line is concerning, especially since the main goal is prediction and not accuracy. Therefore no transformation is necessary at the moment. We can check for outliers to see if we can improve our model. (Remember that increasing the number of predictors usually results in higher variance, which is why shrinkage methods are needed.)

Leverage/Influential/Outlier Points for Full Model

```
h <- hatvalues(model.full)
thresh <- 2 * (dim(model.matrix(model.full))[2])/nrow(small.nhanes)
w <- which(h > thresh)
w
```

```
## 144 624 459 519 452 390 257 228  82 492 262  37 533 569 139 158 302 113  57 367
##   3   6  10  11  17  19  26  27  38  53  55  58  62  67  68  73  74  79 101 105
## 626 127  19 528  66 202 444 454 598 197 470 343 159 501 594 348 330 552 298 620
## 121 123 127 133 137 142 146 147 151 155 161 167 171 183 184 185 194 196 197 203
## 556  13 245 541   7 107 171 412 476 145 190  40 111  30 632 290 371 561 723 437
## 205 216 219 220 230 231 233 235 241 252 260 263 278 279 285 292 293 296 299 304
## 702 405  23 691 544 374 283  43 184 338 206 223 188 215 517  67 270 646 307 126
## 305 311 314 317 318 327 334 335 337 345 349 355 356 358 368 375 377 380 381 387
## 194  45 688 701 416  81 641 120 178 705 502 170 679 196 351 409 179 531 666  65
## 390 391 392 394 395 398 400 406 407 408 415 417 428 430 432 436 439 444 445 449
## 152 475 586 448  97 716 514 654 411 315
## 464 466 468 470 479 487 489 491 494 500
```

```
#small.nhanes[w,]
```

We clearly have a number of leverage points - specifically points that are "influential" on the full model. However this does not mean these points are "bad" or "good" for our model without further diagnostics.

```
# We have 15 predictors...
D <- cooks.distance(model.full)
which(D > qf(0.5, p+1, nrow(train)-p-1))
```

```
## named integer(0)
```

```
# Using a cutoff comparing to 50-th percentile of F dist with p+1 and n-p-1 df
# results in no points.
# Try plotting Cook's distance to find.

#Look at plots of Cook's
pdf("cook_full.pdf", height = 8, width = 12)
par(family = 'mono')
plot(train$BPSysAve, D, type = "p", xlab = "BpSysAve", ylab = "Cook's Distances",
     main = "Cook's Distance")
```

```
abline(h = 2, lty = 2)
abline(h = -2, lty = 2)
text(train$BPSysAve[D > 4/(nrow(train)-p-1)]+3, D[D > 4/(nrow(train)-p-1)],
     labels = which(D > 4/(nrow(train)-p-1)))
dev.off()
```

```
## pdf
##   2
```

Interestingly there are no values over the threshold of Cook's distance using the normal MLR cutoff.

We'll move onto DFFITS and DFBETAS.

```
D <- cooks.distance(model.full)
which(D > qf(0.5, p+1, nrow(train)-p-1))
```

```
## named integer(0)
```

```
## DFFITS ##
dfits <- dffits(model.full)
which(abs(dfits) > 2*sqrt((p+1)/nrow(train)))
```

```
## 336 610 122  38 471 209 444 598 470 343 159 594 348 522 107 171 161 723 437 108
##   1  15  40  70  72  91 146 151 161 167 171 184 185 206 231 233 283 299 304 316
##  43 231 126 688 448  97 662 315
## 335 363 387 392 470 479 485 500
```

```
## DFBETAS ##
dfb <- dfbetas(model.full)
which(abs(dfb[,1]) > 2/sqrt(nrow(train)))
```

```
## 592 214 471 376 396 303 655  47 594 225 597 161 632 461 437 108  43 424 215 213
##   8  31  72  96 117 130 153 164 184 215 234 283 285 301 304 316 335 342 358 366
## 512 126 669 571 104 448  97 551 315
## 385 387 399 409 426 470 479 499 500
```

DFFITS provides a larger amount of "influential points" than DFBETAS. Since we have 500 observations, it may be hard to determine which influential points should be removed, especially since Cook's Distance did not determine any points to be influential.

Now that model diagnostics are complete, let's try looking at different models.

DIFFERENT MODELS

For curiosity's sake, let me create my own model based on assumptions, presumptions, intuition and prior knowledge, i.e. create a model without any selection tools, based off my own bias and knowledge. We can compare this fun model with the calculated models later on to see if my pre-disposed knowledge works out.

(1) Typically we want smaller models (less predictors). This is to help with simplicity, as well as avoid multicollinearity.

(2) From the 15 possible predictors I can easily determine which predictors wouldn't help with determining blood pressure. The most obvious are - Education, MaritalStatus and Depressed. Poverty is by no means a determining factor of blood pressure as well.

(3) There are a number of predictors that easily are correllated. For example, weight, height and BMI are correlated. Since both weight and height are accounted for BMI, I will exclude weight and height in preference of BMI. Poverty and HHIncome are also similar. I will choose Poverty due to simplicity (not categorical) a nd being continuous means it will be more precise. SleepHrsNight and SleepTrouble are very similar, and so for similar reasons I will choose SleepHrsNight.

As a result I will make a model as follows: BPSysAve ~ Gender, Age, Poverty, SleepHrsNight, PhysActive, SmokeNow, for a total of 6 predictors.

```
model.fun <- lm(BPSysAve ~ Gender + Age + Height + Weight
                + PhysActive + SmokeNow, data = train)
summary(model.fun)
```

```
##
## Call:
## lm(formula = BPSysAve ~ Gender + Age + Height + Weight + PhysActive +
##     SmokeNow, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -36.974 -10.466  -0.911   9.270  81.388
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   138.97294   17.28626   8.040 6.74e-15 ***
## Gendermale      6.33824    1.89462   3.345 0.000884 ***
## Age             0.39313    0.04681   8.398 4.85e-16 ***
## Height         -0.23275    0.10776  -2.160 0.031258 *
## Weight          0.02897    0.04116   0.704 0.481830
## PhysActiveYes  -1.90483    1.54672  -1.232 0.218713
## SmokeNowYes    -0.40230    1.62729  -0.247 0.804841
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 16.21 on 493 degrees of freedom
## Multiple R-squared:  0.1919, Adjusted R-squared:  0.1821
## F-statistic: 19.51 on 6 and 493 DF,  p-value: < 2.2e-16
```

```
anova(model.fun)
```

```
## Analysis of Variance Table
##
## Response: BPSysAve
##             Df Sum Sq Mean Sq  F value    Pr(>F)
## Gender       1   1937  1937.1   7.3727  0.006855 **
## Age          1  27030 27030.2 102.8777 < 2.2e-16 ***
## Height       1   1163  1163.1   4.4267  0.035886 *
## Weight       1    228   227.7   0.8667  0.352326
## PhysActive   1    384   384.1   1.4617  0.227232
## SmokeNow     1     16    16.1   0.0611  0.804841
## Residuals  493 129532   262.7
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Notice once again we have relatively high p-values for peculiar predictors - specifically SmokeNow, despite being one of the most "obvious" cases of having an effect on blood pressure. Before we do any diagnostics/analysis, let use build a few other models to compare. This means model selection!

```r
criteria <- function(model) {
    n <- length(model$residuals)
    p <- length(model$coefficients) - 1
    RSS <- sum(model$residuals^2)
    R2 <- summary(model)$r.squared
    R2.adj <- summary(model)$adj.r.squared
    AIC <- n*log(RSS/n) + 2*p
    BIC <- n*log(RSS/n) + (p+2)*log(n)
    results <- c(R2, R2.adj, AIC, BIC)
    names(results) <- c("R Squared", "Adjusted R Squared", "AIC", "BIC")
    return(results)
}
```

Let us make two models from stepwise selection - one from AIC and one from BIC. We do not care about AIC Corrected, as the dataset is large enough. Let us use both direction stepwise selection. First is AIC...

```r
model.AIC <- step(model.full, trace = 0, k = 2, direction = "both")
summary(model.AIC)
```

```
##
## Call:
## lm(formula = BPSysAve ~ Gender + Age + Poverty + Height, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -35.594 -10.768  -0.842   8.866  80.964
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 131.53046   17.05507   7.712 6.85e-14 ***
## Gendermale    5.90613    1.89425   3.118  0.00193 **
## Age           0.42213    0.04371   9.658  < 2e-16 ***
## Poverty      -0.86293    0.45275  -1.906  0.05723 .
## Height       -0.17477    0.10134  -1.725  0.08524 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 16.16 on 495 degrees of freedom
## Multiple R-squared:  0.1939, Adjusted R-squared:  0.1874
## F-statistic: 29.77 on 4 and 495 DF,  p-value: < 2.2e-16
```

```r
anova(model.AIC)
```

```
## Analysis of Variance Table
##
## Response: BPSysAve
##            Df Sum Sq Mean Sq  F value    Pr(>F)
## Gender      1   1937  1937.1   7.4210  0.006675 **
## Age         1  27030 27030.2 103.5512 < 2.2e-16 ***
## Poverty     1   1335  1335.1   5.1146  0.024159 *
## Height      1    776   776.3   2.9740  0.085239 .
## Residuals 495 129211   261.0
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
empty_model <- lm(BPSysAve ~ 1, data = train)
model.AIC2 <-step(empty_model, trace = 0, k = 2, direction = "forward",
                  scope=list(upper = model.full, lower=empty_model))
```

Huh, interestingly SmokeNow is not present.

```
# Add this line to the beginning to make things easier.
n <- nrow(train)
model.BIC <- step(model.full, trace = 0, k = log(n), direction = "backward")
summary(model.BIC)
```

```
##
## Call:
## lm(formula = BPSysAve ~ Gender + Age, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -38.432 -10.196  -1.079   8.510  82.279
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 100.54341    2.39568  41.969   <2e-16 ***
## Gendermale    3.71041    1.46777   2.528   0.0118 *
## Age           0.42722    0.04224  10.114   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 16.26 on 497 degrees of freedom
## Multiple R-squared:  0.1807, Adjusted R-squared:  0.1774
## F-statistic: 54.81 on 2 and 497 DF,  p-value: < 2.2e-16
```

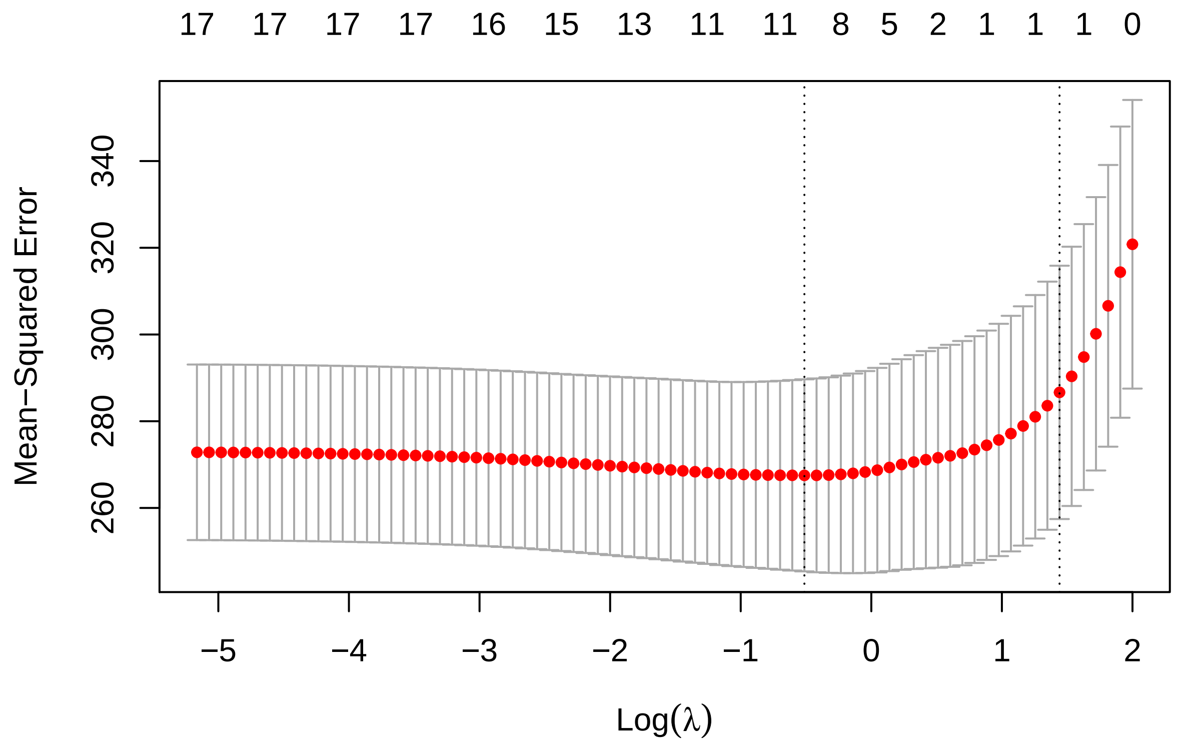
```
anova(model.BIC)
```

```
## Analysis of Variance Table
##
## Response: BPSysAve
##            Df Sum Sq Mean Sq  F value  Pr(>F)
## Gender      1   1937  1937.1   7.3311 0.00701 **
## Age         1  27030 27030.2 102.2980 < 2e-16 ***
## Residuals 497 131322   264.2
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Once again, we are missing SmokeNow. This doesn't seen to make the most sense in terms of my own intuition. One more model can be made using one more variation selection method - LASSO. Notice we don't use ridge regression since it is not very useful for variable selection. Then,

```
set.seed(1006092577)
x_train <- model.matrix(BPSysAve ~ .,data = train)
cv.lasso <- cv.glmnet(x = x_train, y = train$BPSysAve, standardize = T, alpha = 1)

plot(cv.lasso)
```



```
best.lambda <- cv.lasso$lambda.1se
best.lambda
```

```
## [1] 4.22672
```

```
co<-coef(cv.lasso, s = "lambda.1se")
co
```

```
## 19 x 1 sparse Matrix of class "dgCMatrix"
##                                 1
## (Intercept)           114.9948784
## (Intercept)             .
## Gendermale              .
## Age                     0.1835594
## Race3Black                .
## Race3Hispanic             .
## Race3Mexican              .
## Race3White                .
## Race3Other                .
## Education9 - 11th Grade   .
## EducationHigh School      .
## EducationSome College     .
## EducationCollege Grad     .
## Poverty                   .
## Weight                    .
## Height                    .
## SleepHrsNight             .
## PhysActiveYes             .
## SmokeNowYes               .
```

Wow! After performing LASSO penalty on the full model we result in only one predictor left - Age. Therefore let us create a model based off LASSO. This means fitting a single linear regression with merely age.

```
model.lasso <- lm(BPSysAve ~ Age, data = train)
summary(model.lasso)
```

```
##
## Call:
## lm(formula = BPSysAve ~ Age, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -36.884 -10.219  -1.449   9.161  80.116
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 102.55505    2.27185   45.14   <2e-16 ***
## Age           0.42911    0.04246   10.11   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 16.34 on 498 degrees of freedom
## Multiple R-squared:  0.1702, Adjusted R-squared:  0.1685
## F-statistic: 102.1 on 1 and 498 DF,  p-value: < 2.2e-16
```

```
anova(model.lasso)
```

```
## Analysis of Variance Table
##
## Response: BPSysAve
##            Df Sum Sq Mean Sq F value    Pr(>F)
## Age         1  27279 27278.8  102.13 < 2.2e-16 ***
## Residuals 498 133011   267.1
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Awesome, we have a small p-value for this SLR. This means that Age does have a linear relationship with BPSysAve. We can easily check this with a plot comparing fitted values with the actual values (due to it being Simple Linear Regression). Then,

```
pdf("LASSO_Plot_FittedVSBPSysAve.pdf", height = 8, width = 16)
plot(train$Age, train$BPSysAve, type = "p", xlab = "Age", ylab = "BPSysAve",
     main = "Age vs. BPSysave - LASSO")
abline(lm(BPSysAve ~ Age, data = train), lwd = 2, col = "blue")
dev.off()
```

```
## pdf
##   2
```

Let us check the variance inflation factor of each model.

```
vif(model.full)
```

```
##              Gendermale                    Age             Race3Black
##                2.031488               1.380063               4.044485
##           Race3Hispanic            Race3Mexican             Race3White
##                2.638458               3.016036               6.831929
##              Race3Other Education9 - 11th Grade    EducationHigh School
##                2.099287               2.840384               3.755939
##    EducationSome College    EducationCollege Grad                Poverty
##                4.606668               4.070168               1.352454
##                  Weight                 Height           SleepHrsNight
##                1.334822               2.419930               1.124613
##           PhysActiveYes            SmokeNowYes
##                1.235357               1.374619
```

We see that with the full model we have some high GVIF (general VIF) values. These occur with the expected predictors - between HHIncome and Poverty, and between Weight, Height and BMI. Let us not remove any predictors since this model's entire purpose is to represent the full model.

```
vif(model.fun)
```

```
##    Gendermale           Age        Height        Weight PhysActiveYes
##      1.676170      1.235474      1.994590      1.287979      1.134594
##   SmokeNowYes
##      1.219010
```

This is great, as there is low correlation between the predictors. Our assumptions resulted in a model that does not suffer from multicollinearity.

```
vif(model.AIC)
```

```
## Gendermale        Age    Poverty     Height
##   1.686479   1.084170   1.056590   1.775706
```

```r
vif(model.BIC)
```

```
## Gendermale         Age
##   1.000314    1.000314
```

```r
criteria(model.full)
```

```
##           R Squared Adjusted R Squared                AIC                BIC
##           0.2114128          0.1835996       2800.3090818       2884.3866357
```

```r
criteria(model.fun)
```

```
##           R Squared Adjusted R Squared                AIC                BIC
##           0.1918916          0.1820566       2790.5356967       2828.2525615
```

```r
criteria(model.AIC)
```

```
##           R Squared Adjusted R Squared                AIC                BIC
##           0.1938908          0.1873768       2785.2972371       2814.5848857
```

```r
criteria(model.BIC)
```

```
##           R Squared Adjusted R Squared                AIC                BIC
##           0.1807186          0.1774217       2789.4014303       2810.2598627
```

```r
criteria(model.lasso)
```

```
##           R Squared Adjusted R Squared                AIC                BIC
##           0.1701843          0.1685180       2793.7894153       2810.4332396
```

Model Diagnostics for all Models

We have finished with our models, now time to look at plots!

```r
resid <- rstudent(model.fun)
fitted <- predict(model.fun)

# The 4 plots -> residual vs fitted, Q-Q plots, std residual vs fitted, std residual
# vs leverage
pdf("model_fun.pdf", height = 8, width = 16)
par(family = 'mono', mfrow = c(1,2))
plot(model.fun)
# Plot the fitted vs response(BPSysAve) for normality assumption
plot(train$BPSysAve ~ fitted, type = "p", xlab = "Fitted Values",
     ylab = "BPSysAve", cex.lab = 1.2,
     col = "red")
abline(lm(train$BPSysAve ~ fitted), lwd = 2, col = "blue")
lines(lowess(fitted, train$BPSysAve), col = "red")
dev.off()
```

```
## pdf
##   2
```

We can do this for the other models as well.

```r
resid <- rstudent(model.AIC)
fitted <- predict(model.AIC)

pdf("model_AIC.pdf", height = 8, width = 16)
par(family = 'mono', mfrow = c(1,2))
plot(model.AIC)
# Plot the fitted vs response(BPSysAve) for normality assumption
plot(train$BPSysAve ~ fitted, type = "p", xlab = "Fitted Values",
     ylab = "BPSysAve", cex.lab = 1.2,
     col = "red")
abline(lm(train$BPSysAve ~ fitted), lwd = 2, col = "blue")
lines(lowess(fitted, train$BPSysAve), col = "red")
dev.off()
```

```
## pdf
##   2
```

```r
resid <- rstudent(model.BIC)
fitted <- predict(model.BIC)

pdf("model_BIC.pdf", height = 8, width = 16)
par(family = 'mono', mfrow = c(1,2))
plot(model.BIC)
# Plot the fitted vs response(BPSysAve) for normality assumption
plot(train$BPSysAve ~ fitted, type = "p", xlab = "Fitted Values",
     ylab = "BPSysAve", cex.lab = 1.2,
     col = "red")
abline(lm(train$BPSysAve ~ fitted), lwd = 2, col = "blue")
lines(lowess(fitted, train$BPSysAve), col = "red")
dev.off()
```

```
## pdf
##   2
```

```r
resid <- rstudent(model.lasso)
fitted <- predict(model.lasso)

pdf("model_LASSO.pdf", height = 8, width = 16)
par(family = 'mono', mfrow = c(1,2))
plot(model.lasso)
# Plot the fitted vs response(BPSysAve) for normality assumption
plot(train$BPSysAve ~ fitted, type = "p", xlab = "Fitted Values",
     ylab = "BPSysAve", cex.lab = 1.2,
     col = "red")
abline(lm(train$BPSysAve ~ fitted), lwd = 2, col = "blue")
lines(lowess(fitted, train$BPSysAve), col = "red")
dev.off()
```

```
## pdf
##   2
```

Plot 1/3: This is a plot of fitted values vs. residuals. This is a very similar plot to plot 3, which is instead a plot of square roots of standardized residuals instead. Both are used in the same interpretation. These plots are used to test assumptions related to the errors such as homoscedasticity. If the plots result in a somewhat flat line with points randomly (uniformly) scattered around the line, then we say that the assumption of constant variance is met, and possibly that the errors are independent. All 5 plots from our models do not display no **evident pattern**. This means all 5 of our models satisfy the independence of errors and homoscedasticity assumptions. This means we may not need to use transformations on our models.

Plot 2: This is a Normal Q-Q plot, which seeks to challenge the assumption of normality of errors. Notice that this assumption and the constant variance assumption are the most important assumptions to have to be able to apply inferential tools. We know if the Normality assumption is met if we see a one-to-one relationship between the quantiles, as this proves normality. Notice all 5 models provide a one-to-one relationship except on the extreme values. However this is not very prominent, since this only occurs on the extreme quantiles (and thus is rare). We say that the normality assumption is met on every model.

Plot 4: This plot is a Cook's distance plot. This is used to help find influential points. Alongside DFFITS and DFBETAs, we can determine if there are any bad leverage points that we may wish to remove from the data. We see from every model that the Cook's distance values are very low - with the highest being from the full model. There are two things to consider with Cook's distance values - the cutoff, and the gaps in the plot. Every plot has the points densely packed closed, with a few outliers outside of the main group, however these outliers do not deviate very much. The cutoff of Cook's distance is based off the 50th percentile of the F distribution with degrees of freedom $p+1$ and $n-p-1$. Since we have such a large n value, being 500 in the train data, then our cutoff value will be very large - close to 1. It is safe to say that none of the points in any plot are anywhere near to 1. This means that we do not have any influential points to consider removing from our data. We can further look at DFBETAs and DFFITS for more clarification.

```r
## Cook's Distance ##
D <- cooks.distance(model.full)
which(D > qf(0.5, p+1, nrow(train)-p-1))
```

```
## named integer(0)
```

```r
## DFFITS ##
dfits <- dffits(model.full)
which(abs(dfits) > 2*sqrt((p+1)/nrow(train)))
```

```
## 336 610 122  38 471 209 444 598 470 343 159 594 348 522 107 171 161 723 437 108
##   1  15  40  70  72  91 146 151 161 167 171 184 185 206 231 233 283 299 304 316
##  43 231 126 688 448  97 662 315
## 335 363 387 392 470 479 485 500
```

```r
## DFBETAS ##
dfb <- dfbetas(model.full)
which(abs(dfb[,1]) > 2/sqrt(nrow(train)))
```

```
## 592 214 471 376 396 303 655  47 594 225 597 161 632 461 437 108  43 424 215 213
##   8  31  72  96 117 130 153 164 184 215 234 283 285 301 304 316 335 342 358 366
## 512 126 669 571 104 448  97 551 315
## 385 387 399 409 426 470 479 499 500
```

```r
D <- cooks.distance(model.fun)
which(D > qf(0.5, p+1, nrow(train)-p-1))
```

```
## named integer(0)
```

```r
## DFFITS ##
dfits <- dffits(model.fun)
which(abs(dfits) > 2*sqrt((p+1)/nrow(train)))
```

```
##  38 471 126 688
##  70  72 387 392
```

```r
## DFBETAS ##
dfb <- dfbetas(model.fun)
which(abs(dfb[,1]) > 2/sqrt(nrow(train)))
```

```
## 592 471 376 655 470 594 324 225 597 161 632 108  43 424 694 215 512 126 669 571
##   8  72  96 153 161 184 212 215 234 283 285 316 335 342 344 358 385 387 399 409
##  97 315
## 479 500
```

```r
D <- cooks.distance(model.AIC)
which(D > qf(0.5, p+1, nrow(train)-p-1))
```

```
## named integer(0)
```

```r
## DFFITS ##
dfits <- dffits(model.AIC)
which(abs(dfits) > 2*sqrt((p+1)/nrow(train)))
```

```
##  38 471 522 688
##  70  72 206 392
```

```r
## DFBETAS ##
dfb <- dfbetas(model.AIC)
which(abs(dfb[,1]) > 2/sqrt(nrow(train)))
```

```
## 592 471 376 655 470 594 522 597 233 161 632 461 108  43 424 694 215 213 512 126
##   8  72  96 153 161 184 206 234 262 283 285 301 316 335 342 344 358 366 385 387
## 701 669 571  97 315
## 394 399 409 479 500
```

```r
D <- cooks.distance(model.BIC)
which(D > qf(0.5, p+1, nrow(train)-p-1))
```

```
## named integer(0)
## DFFITS ##
dfits <- dffits(model.BIC)
which(abs(dfits) > 2*sqrt((p+1)/nrow(train)))
```

```
## 38
## 70
```

```
## DFBETAS ##
dfb <- dfbetas(model.BIC)
which(abs(dfb[,1]) > 2/sqrt(nrow(train)))
```

```
## 154  38  86 444 598 655 470 594 348  64  95 161 723 108 361 126 688 571 104 315
##  56  70  78 146 151 153 161 184 185 221 222 283 299 316 347 387 392 409 426 500
```

```
D <- cooks.distance(model.lasso)
which(D > qf(0.5, p+1, nrow(train)-p-1))
```

```
## named integer(0)
```

```
## DFFITS ##
dfits <- dffits(model.lasso)
which(abs(dfits) > 2*sqrt((p+1)/nrow(train)))
```

```
## 38
## 70
```

```
## DFBETAS ##
dfb <- dfbetas(model.lasso)
which(abs(dfb[,1]) > 2/sqrt(nrow(train)))
```

```
## 610 122 154  38 444 598 655 594 671  64  95 226 723 525 361 231 688 669 571 104
##  15  40  56  70 146 151 153 184 199 221 222 291 299 330 347 363 392 399 409 426
## 196 573 351  97 315
## 430 431 432 479 500
```

As observed, none of the models had any large influential points as determined by Cook's Distance. DFFITS and DFBETAs derived somewhat different results as well. For simplicity sake, we will keep every point in the train dataset. This is because lack of values in Cook's distance as well as inconsistent results from the other methods leads to inconclusive evidence of any major influential points. Finally, onto model validation!

Model Validation

Now that we have created a few models, we want to test the prediction performance between them to choose the "best one." We can first start with cross-validation. This will help us inform our thoughts on overfitting that we predict with every model other than BIC and LASSO. Cross-validation is a resampling method.

```r
# First create the "labels" from each model to perform ols/calibrate on them
full_labels <- attr(terms(model.full), "term.labels")
AIC_labels <- attr(terms(model.AIC), "term.labels")
BIC_labels <- attr(terms(model.BIC), "term.labels")
fun_labels <- attr(terms(model.fun), "term.labels")
lasso_labels <- attr(terms(model.lasso), "term.labels")

# Now use ols!
ols.full <- ols(BPSysAve ~ ., data = train[,which(colnames(train) %in%
               c(full_labels, "BPSysAve"))], x=T, y=T, model = T)
ols.aic <- ols(BPSysAve ~ ., data = train[,which(colnames(train) %in%
               c(AIC_labels, "BPSysAve"))], x=T, y=T, model = T)
ols.bic <- ols(BPSysAve ~ ., data = train[,which(colnames(train) %in%
               c(BIC_labels, "BPSysAve"))], x=T, y=T, model = T)
ols.fun <- ols(BPSysAve ~ ., data = train[,which(colnames(train) %in%
               c(fun_labels, "BPSysAve"))], x=T, y=T, model = T)
ols.lasso <- ols(BPSysAve ~ ., data = train[,which(colnames(train) %in%
               c(lasso_labels, "BPSysAve"))], x=T, y=T, model = T)
```

Before we do any interpretation, let us create all 5 calibration plots.

```r
# Finally we can calibrate. We can also check the prediction error using the test dataset.
full.cross <- calibrate(ols.full, method = "crossvalidation", B = 10)
## Calibration plot ##
pdf("full_cross.pdf", height = 6, width = 6)
plot(full.cross, las = 1, xlab = "Predicted BPSysAve", main =
       "Cross-Validation Calibration with Full Model")
```

```
##
## n=500   Mean absolute error=1.001   Mean squared error=2.8894
## 0.9 Quantile of absolute error=2.631
```

```r
dev.off()
```

```
## pdf
##   2
```

```r
# Prediction Error
pred.full <- predict(model.full, newdata = test[,which(colnames(train) %in%
                                       c(full_labels, "BPSysAve"))])
## Prediction error ##
pred.error.full <- mean((test$BPSysAve - pred.full)^2)
```

```r
aic.cross <- calibrate(ols.aic, method = "crossvalidation", B = 10)
## Calibration plot ##
pdf("aic_cross.pdf", height = 6, width = 6)
plot(aic.cross, las = 1, xlab = "Predicted BPSysAve", main =
       "Cross-Validation calibration with AIC Model")
```

```
##
## n=500   Mean absolute error=0.546   Mean squared error=0.42397
## 0.9 Quantile of absolute error=0.777
```

```r
dev.off()
```

```
## pdf
##   2
```

```r
# Prediction Error
pred.aic <- predict(model.AIC, newdata = test[,which(colnames(train) %in%
                                                      c(AIC_labels, "BPSysAve"))])
## Prediction error ##
pred.error.aic <- mean((test$BPSysAve - pred.aic)^2)
```

```r
bic.cross <- calibrate(ols.bic, method = "crossvalidation", B = 10)
## Calibration plot ##
pdf("bic_cross.pdf", height = 6, width = 6)
plot(bic.cross, las = 1, xlab = "Predicted BPSysAve", main =
       "Cross-Validation calibration with BIC Model")
```

```
##
## n=500   Mean absolute error=0.412   Mean squared error=0.3183
## 0.9 Quantile of absolute error=0.916
```

```r
dev.off()
```

```
## pdf
##   2
```

```r
# Prediction Error
pred.bic <- predict(model.BIC, newdata = test[,which(colnames(train) %in%
                                                      c(BIC_labels, "BPSysAve"))])
## Prediction error ##
pred.error.bic <- mean((test$BPSysAve - pred.bic)^2)
```

```r
fun.cross <- calibrate(ols.fun, method = "crossvalidation", B = 10)
## Calibration plot ##
pdf("fun_cross.pdf", height = 6, width = 6)
plot(fun.cross, las = 1, xlab = "Predicted BPSysAve", main =
       "Cross-Validation calibration with Test Model")
```

```
##
## n=500   Mean absolute error=0.639   Mean squared error=1.22713
## 0.9 Quantile of absolute error=1.317
```

```r
dev.off()
```

```
## pdf
##   2
```

```r
# Prediction Error
pred.fun <- predict(model.fun, newdata = test[,which(colnames(train) %in%
                                                      c(fun_labels, "BPSysAve"))])
## Prediction error ##
pred.error.fun <- mean((test$BPSysAve - pred.fun)^2)
pred.error.fun
```

```
## [1] 231.1259
```

```
lasso.cross <- calibrate(ols.lasso, method = "crossvalidation", B = 10)
## Calibration plot ##
pdf("lasso_cross.pdf", height = 6, width = 6)
plot(lasso.cross, las = 1, xlab = "Predicted BPSysAve", main =
       "Cross-Validation calibration with LASSO Model")
```

```
##
## n=500   Mean absolute error=0.755   Mean squared error=0.7373
## 0.9 Quantile of absolute error=1.03
```

```
dev.off()
```

```
## pdf
##   2
```

```
# Prediction Error
pred.lasso <- predict(model.lasso, newdata = test[,which(colnames(train) %in%
                                                      c(lasso_labels, "BPSysAve"))])
## Prediction error ##
pred.error.lasso <- mean((test$BPSysAve - pred.lasso)^2)
```

We can group together all 5 prediction errors to compare.

```
pred.error.all <- c(pred.error.full, pred.error.fun, pred.error.aic, pred.error.bic,
                    pred.error.lasso)
pred.error.all
```

```
## [1] 229.7480 231.1259 231.1610 234.2986 240.2261
```

We have in order the prediction error of the full model, then fun, AIC, BIC and finally Lasso model. We see that the fun model has the lowest prediction error, followed by AIC and BIC.

Let's look at the calibration charts. We easily see that the LASSO and Full model are the "worse models" in terms of cross-validation. The Fun model would be better due to conforming to the line more closely.

Choosing the Best Model

We've come to the point where we need to choose a model. Clearly the full model is not the one we want. This is because we clearly see there is multicollinearity within most of its parameters, and having very high p-values.

Look at Fun model without SmokeNow! Wow lower prediction error?

```
# Without SmokeNow!!!
model.fun2 <- lm(BPSysAve ~ Gender + Age + Height + Weight
                 + PhysActive, data = train)
summary(model.fun2)
```

```
##
## Call:
## lm(formula = BPSysAve ~ Gender + Age + Height + Weight + PhysActive,
##     data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -36.896 -10.410  -0.803   9.282  81.458
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   138.76345   17.24907   8.045 6.47e-15 ***
## Gendermale      6.32170    1.89164   3.342 0.000895 ***
## Age             0.39727    0.04367   9.097  < 2e-16 ***
## Height         -0.23464    0.10738  -2.185 0.029352 *
## Weight          0.03038    0.04073   0.746 0.456139
## PhysActiveYes  -1.80446    1.49107  -1.210 0.226790
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 16.19 on 494 degrees of freedom
## Multiple R-squared:  0.1918, Adjusted R-squared:  0.1836
## F-statistic: 23.45 on 5 and 494 DF,  p-value: < 2.2e-16
```

```
anova(model.fun2)
```

```
## Analysis of Variance Table
##
## Response: BPSysAve
##             Df Sum Sq Mean Sq  F value    Pr(>F)
## Gender       1   1937  1937.1   7.3867  0.006802 **
## Age          1  27030 27030.2 103.0736 < 2.2e-16 ***
## Height       1   1163  1163.1   4.4352  0.035710 *
## Weight       1    228   227.7   0.8684  0.351867
## PhysActive   1    384   384.1   1.4645  0.226790
## Residuals  494 129548   262.2
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
fun_labels <- attr(terms(model.fun2), "term.labels")
pred.fun <- predict(model.fun2, newdata = test[,which(colnames(train) %in%
                                                  c(fun_labels, "BPSysAve"))])
## Prediction error ##
pred.error.fun <- mean((test$BPSysAve - pred.fun)^2)
pred.error.fun
```

```
## [1] 231.0104
```