Math 158, Spring 2018
Jo Hardin
Shrinkage Methods – R code
Ridge Regression & LASSO

The Data

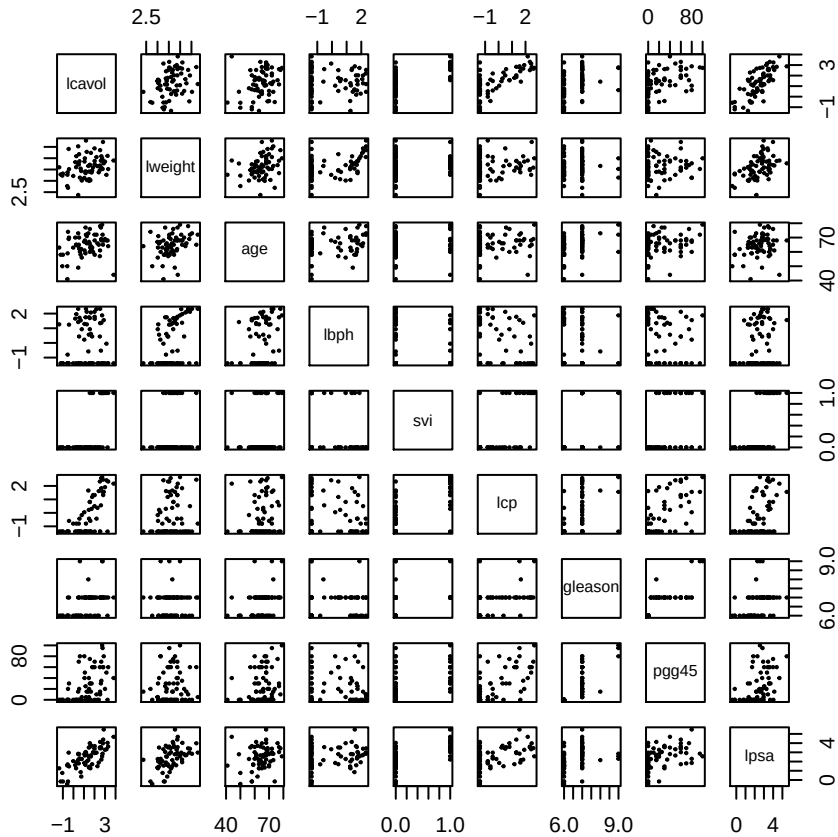
The following dataset is from Hastie, Tibshirani and Friedman (2009), from a studyby Stamey et al. (1989) of prostate cancer, measuring the correlation between the level of a prostate-specific antigen and some covariates. The covariates are

- lcavol : log-cancer volume
- lweight : log-prostate weight
- age : age of patient
- lbhp : log-amount of benign hyperplasia
- svi : seminal vesicle invasion
- lcp : log-capsular penetration
- gleason : Gleason Score, check `http://en.wikipedia.org/wiki/Gleason_Grading_System`
- pgg45 : percent of Gleason scores 4 or 5
- lpsa is the response variable, log-psa.

```
require(readr)
require(broom); require(dplyr)
require(glmnet)  # lars is where the LASSO functions live
pcancer <- read_delim("http://www-stat.stanford.edu/~hastie/ElemStatLearn/datasets/prostate.data",
     "\t", escape_double = FALSE, trim_ws = TRUE)
head(pcancer)

## # A tibble: 6 x 11
##      X1 lcavol lweight   age  lbph   svi   lcp gleason pgg45   lpsa train
##   <int>  <dbl>   <dbl> <int> <dbl> <int> <dbl>   <int> <int>  <dbl> <lgl>
## 1     1 -0.580    2.77    50 -1.39     0 -1.39       6     0 -0.431 T
## 2     2 -0.994    3.32    58 -1.39     0 -1.39       6     0 -0.163 T
## 3     3 -0.511    2.69    74 -1.39     0 -1.39       7    20 -0.163 T
## 4     4 -1.20     3.28    58 -1.39     0 -1.39       6     0 -0.163 T
## 5     5  0.751    3.43    62 -1.39     0 -1.39       6     0  0.372 T
## 6     6 -1.05     3.23    50 -1.39     0 -1.39       6     0  0.765 T

# They provide the breakdown of training / test observations
pr.train <- pcancer[which(pcancer$train),2:10]
pr.test <- pcancer[-which(pcancer$train),2:10]
```

Shrinkage methods can be helpful when covariates are correlated, is there strong collinearity with these data? As is often the case with biological data, there is some degree of correlation between the variables, here the collinearity does not seem to be exteme.

```
pairs(pr.train[,1:9], pch=19, cex=.25)
```




```
round(cor(pr.train[,1:9]),3)

##         lcavol lweight  age   lbph   svi    lcp gleason  pgg45 lpsa
## lcavol   1.000   0.300 0.29  0.063  0.59  0.692   0.426  0.483 0.73
## lweight  0.300   1.000 0.32  0.437  0.18  0.157   0.024  0.074 0.48
## age      0.286   0.317 1.00  0.287  0.13  0.173   0.366  0.276 0.23
## lbph     0.063   0.437 0.29  1.000 -0.14 -0.089   0.033 -0.030 0.26
## svi      0.593   0.181 0.13 -0.139  1.00  0.671   0.307  0.481 0.56
## lcp      0.692   0.157 0.17 -0.089  0.67  1.000   0.476  0.663 0.49
## gleason  0.426   0.024 0.37  0.033  0.31  0.476   1.000  0.757 0.34
## pgg45    0.483   0.074 0.28 -0.030  0.48  0.663   0.757  1.000 0.45
## lpsa     0.733   0.485 0.23  0.263  0.56  0.489   0.342  0.448 1.00
```

The full linear model:

```
pr.ls <- lm(lpsa ~ ., data=pr.train)
summary(pr.ls)$coef

##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.4292     1.5536    0.28  7.8e-01
## lcavol        0.5765     0.1074    5.37  1.5e-06
## lweight       0.6140     0.2232    2.75  7.9e-03
## age          -0.0190     0.0136   -1.40  1.7e-01
## lbph          0.1448     0.0705    2.06  4.4e-02
## svi           0.7372     0.2986    2.47  1.7e-02
## lcp          -0.2063     0.1105   -1.87  6.7e-02
## gleason      -0.0295     0.2011   -0.15  8.8e-01
## pgg45         0.0095     0.0054    1.74  8.8e-02
```

What if n is small?

No good! The coefficients cannot be estimated! (Here, 6 points were randomly selected, remember $p = 9$.)

```
set.seed(47)
fewpts = sample(c(TRUE, FALSE), 67, replace=TRUE, prob=c(.1,.9))
pr.ls.sub <- lm(lpsa~., data=pr.train, subset=fewpts)
summary(pr.ls.sub)

##
## Call:
## lm(formula = lpsa ~ ., data = pr.train, subset = fewpts)
##
## Residuals:
## ALL 6 residuals are 0: no residual degrees of freedom!
##
## Coefficients: (3 not defined because of singularities)
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -2.8264         NA      NA       NA
## lcavol        1.9668         NA      NA       NA
## lweight       0.5427         NA      NA       NA
## age          -0.0345         NA      NA       NA
## lbph          0.2128         NA      NA       NA
## svi               NA         NA      NA       NA
## lcp          -2.9234         NA      NA       NA
## gleason           NA         NA      NA       NA
## pgg45             NA         NA      NA       NA
##
## Residual standard error: NaN on 0 degrees of freedom
## Multiple R-squared:     1,Adjusted R-squared:   NaN
## F-statistic:  NaN on 5 and 0 DF,  p-value: NA
```

The model breaks down because $(X^tX)^{-1}$ is not invertible:

```
x.fewpts <- as.matrix(cbind(int = rep(1, dim(pr.train[fewpts,])[1]), pr.train[fewpts,-9]))
dim(x.fewpts)

## [1] 6 9

solve(t(x.fewpts) %*% x.fewpts)

## Error in solve.default(t(x.fewpts) %*% x.fewpts): Lapack routine dgesv: system is exactly
singular: U[6,6] = 0

x.full <- as.matrix(cbind(int = rep(1, dim(pr.train)[1]),pr.train[,-9]))
solve(t(x.full) %*% x.full)  # note that it is invertible w/all the data

##             int   lcavol  lweight      age     lbph      svi      lcp
## int      4.7573  0.06559 -3.6e-01 -1.0e-02  7.2e-02 -4.6e-02  0.00561
## lcavol   0.0656  0.02275 -8.2e-03 -3.1e-04 -9.0e-04 -1.6e-02 -0.01020
## lweight -0.3558 -0.00817  9.8e-02 -1.2e-03 -1.3e-02 -1.5e-02 -0.00021
## age     -0.0100 -0.00031 -1.2e-03  3.7e-04 -3.1e-04  5.7e-05  0.00022
## lbph     0.0722 -0.00090 -1.3e-02 -3.1e-04  9.8e-03  7.6e-03  0.00111
## svi     -0.0462 -0.01564 -1.5e-02  5.7e-05  7.6e-03  1.8e-01 -0.02166
## lcp      0.0056 -0.01020 -2.1e-04  2.2e-04  1.1e-03 -2.2e-02  0.02407
## gleason -0.4598 -0.00706  1.4e-02 -1.3e-03 -1.1e-03  1.2e-02  0.00245
## pgg45    0.0080  0.00012 -3.2e-05 -2.0e-06  3.4e-06 -4.1e-04 -0.00046
##         gleason    pgg45
## int     -0.4598  8.0e-03
## lcavol  -0.0071  1.2e-04
## lweight  0.0140 -3.2e-05
## age     -0.0013 -2.0e-06
## lbph    -0.0011  3.4e-06
## svi      0.0124 -4.1e-04
## lcp      0.0025 -4.6e-04
## gleason  0.0797 -1.4e-03
## pgg45   -0.0014  5.8e-05
```

AIC model building

Remembering AIC to model build going forward and backward, build two different model.

```
pr.fwd <- step(lm(lpsa ~ 1, data=pr.train),
               ~ lcavol + lweight + age + lbph + svi + lcp + gleason + pgg45,
               data=pr.train, direction="forward")

pr.bck <- step(pr.ls, direction="backward")
```

The AIC forward and backward models:

```
tidy(pr.fwd)

##          term estimate std.error statistic p.value
## 1 (Intercept)    -0.33     0.780     -0.42 6.8e-01
## 2      lcavol     0.51     0.093      5.46 8.9e-07
## 3     lweight     0.54     0.221      2.44 1.8e-02
## 4         svi     0.67     0.273      2.46 1.7e-02
## 5        lbph     0.14     0.070      1.99 5.1e-02

tidy(pr.bck)

##          term estimate std.error statistic p.value
## 1 (Intercept)   0.2591    1.0252      0.25 8.0e-01
## 2      lcavol   0.5739    0.1051      5.46 9.9e-07
## 3     lweight   0.6192    0.2186      2.83 6.3e-03
## 4         age  -0.0195    0.0131     -1.49 1.4e-01
## 5        lbph   0.1444    0.0698      2.07 4.3e-02
## 6         svi   0.7418    0.2945      2.52 1.4e-02
## 7         lcp  -0.2054    0.1094     -1.88 6.5e-02
## 8       pgg45   0.0089    0.0041      2.18 3.3e-02
```

Ridge Regression model building

The vignette for `glmnet` is at `http://web.stanford.edu/~hastie/glmnet/glmnet_alpha.html` and is very useful.

Note: all the logs, throughout, are natural logs.

The basic ridge regession model code is as follows:

```
lambda.grid =10^seq(5,-2, length =100)
# note the form of the regression function is glmnet(x,y,...)
# alpha = 0 gives RR, alpha=1 gives Lasso
pr.ridge <- glmnet(as.matrix(pr.train[,1:8]), pr.train$lpsa, alpha=0,
                   lambda = lambda.grid, standardize=TRUE)
```

Choosing λ

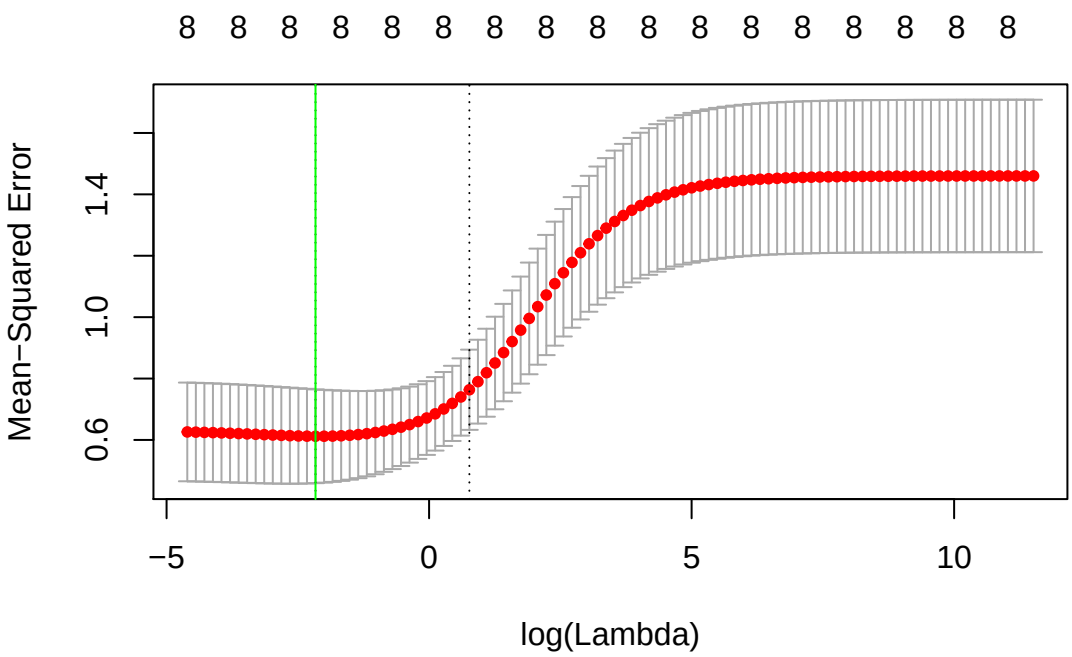
What happens to the coefficients and predictions when λ is large (8.5×10^4) verus λ being small (0.05)?

```
dim(coef(pr.ridge))  # 9 coef, 100 lambda values

## [1]   9 100

lambda.grid[2]

## [1] 84975

coef(pr.ridge)[,2] # note glmnet unstandardizes the variables

## (Intercept)      lcavol     lweight         age        lbph         svi
##     2.5e+00     1.0e-05     1.7e-05     5.2e-07     3.1e-06     2.3e-05
##         lcp     gleason       pgg45
##     6.0e-06     8.2e-06     2.6e-07

lambda.grid[90]

## [1] 0.051

coef(pr.ridge)[,90]  # note glmnet unstandardizes the variables

## (Intercept)      lcavol     lweight         age        lbph         svi
##      0.2007      0.5229      0.6079     -0.0164      0.1408      0.7007
##         lcp     gleason       pgg45
##     -0.1477      0.0029      0.0079
```

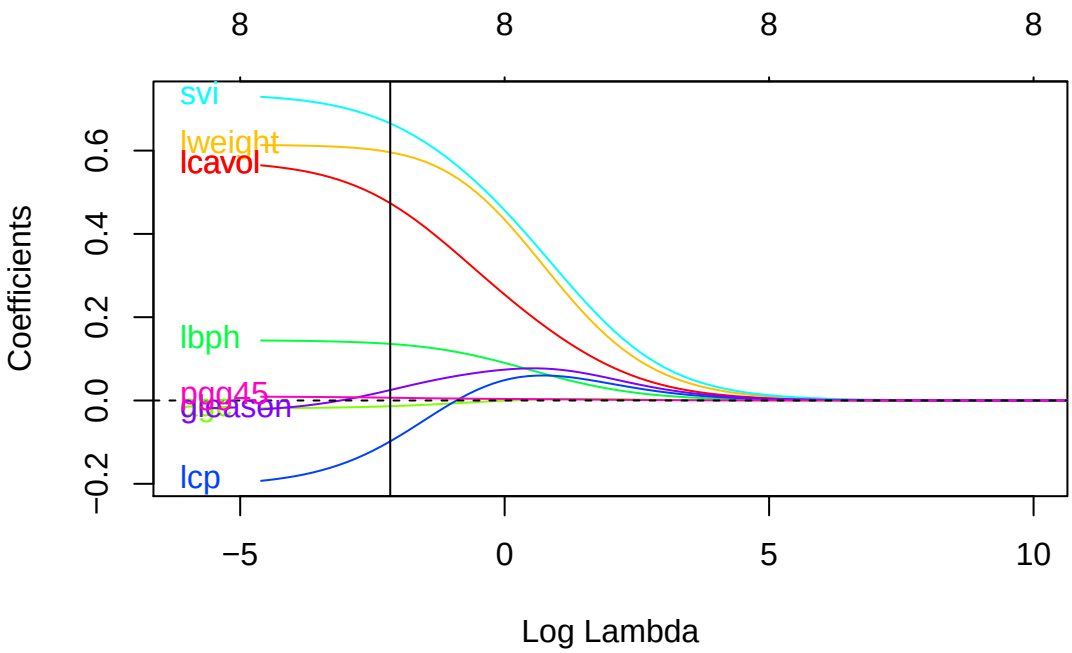
Each boxplot represents the MSE values for each left out sample according to the cross validation information. That is, given a particular λ value, the data is partitioned. Each hold out sample is predicted (and MSE calculated) given the model for the rest of the data. The boxplot describes those MSE values with the average MSE given by the red dot.

In the following figure, the lower dashed (green) vertical line gives the λ with the smallest cross validated MSE. The "other" line gives the most regularized (that is: smaller coefficients!) λ that is within one SE (a box whisker) of the minimum λ value.

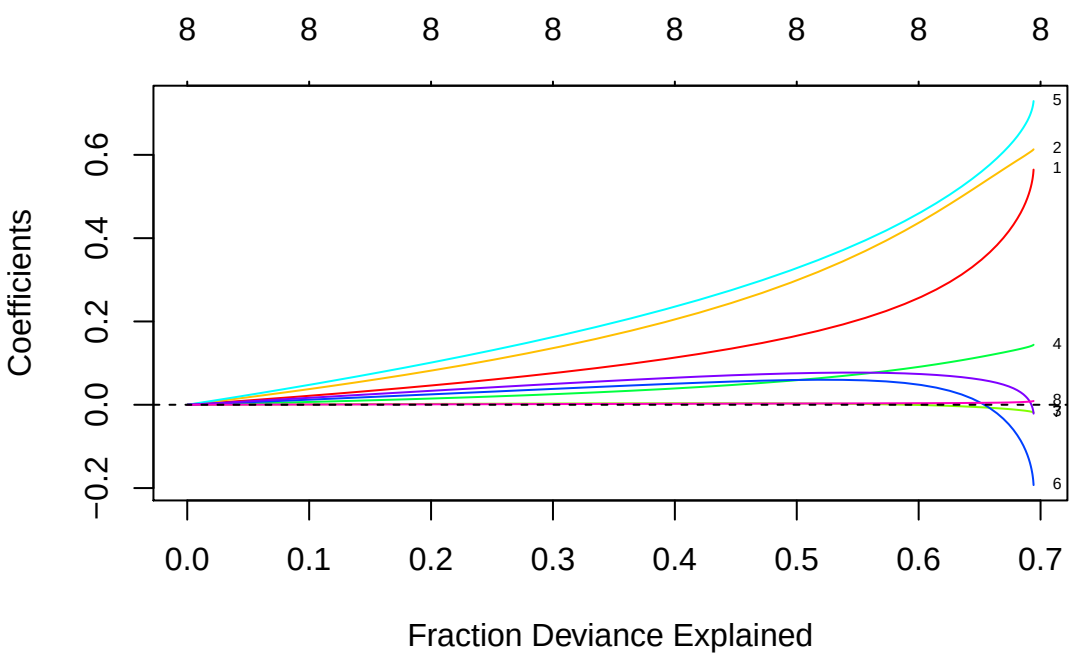
```
pr.ridge.cv <- cv.glmnet(as.matrix(pr.train[,1:8]), pr.train$lpsa,
                         alpha=0, lambda = lambda.grid, standardize=TRUE)
plot(pr.ridge.cv)
abline(v=log(pr.ridge.cv$lambda.min), col="green")
```



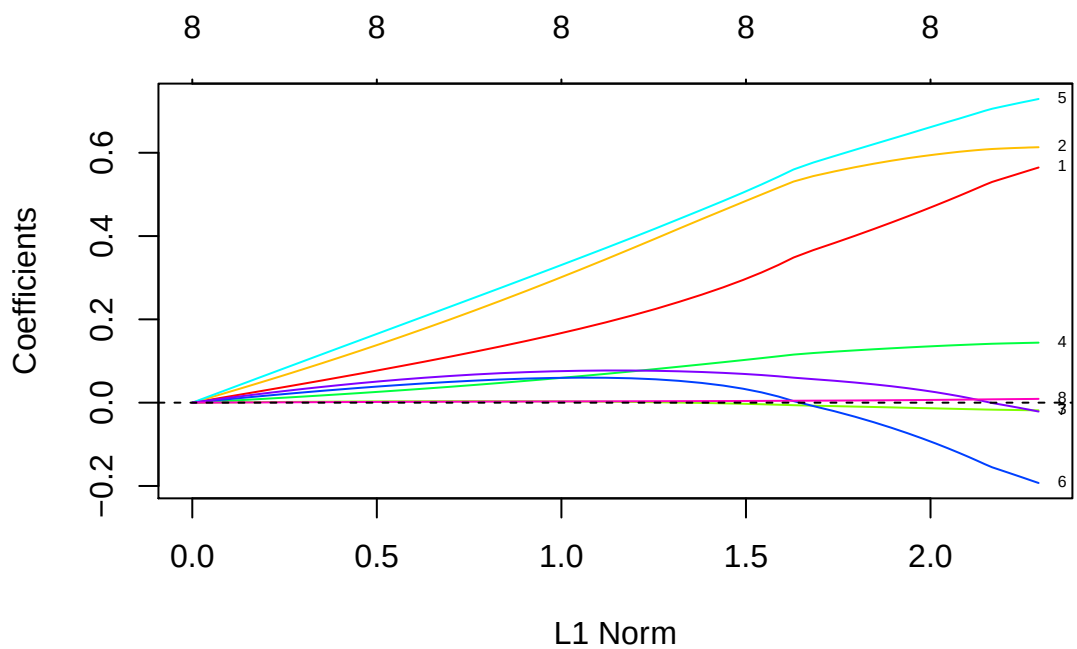

```
colors <- rainbow(8)
plot(pr.ridge, xvar="lambda", xlim=c(-6,10),col=colors)
pr.ridge.cv$lambda.min

## [1] 0.11

abline(v=log(pr.ridge.cv$lambda.min))
abline(h=0, lty=2)
text(rep(-6.5, 9), coef(pr.ridge)[-1,length(lambda.grid)], colnames(pr.train)[-9], pos=4, col=colors)
```




```
plot(pr.ridge, xvar="dev",col=colors, label=TRUE)
abline(h=0, lty=2)
```



```r
plot(pr.ridge, xvar="norm",col=colors, label=TRUE)
abline(h=0, lty=2)
```



The Ridge Model

Using the λ value for the minimum MSE in the cross validation, we output the coefficients / model associated with the best λ for the ridge regression model. Ridge regression does not do variable selection.

```r
coef(pr.ridge.cv, s = "lambda.min")

## 9 x 1 sparse Matrix of class "dgCMatrix"
##                    1
## (Intercept)  0.0372
## lcavol       0.4737
## lweight      0.5959
## age         -0.0138
## lbph         0.1359
## svi          0.6653
## lcp         -0.0981
## gleason      0.0251
## pgg45        0.0066
```

Lasso model building

```
# note the form of the regression function is glmnet(x,y,...)
# alpha = 0 gives RR, alpha=1 gives Lasso
pr.lasso <- glmnet(as.matrix(pr.train[,1:8]), pr.train$lpsa, alpha=1,
                   lambda = lambda.grid, standardize=TRUE)
```

Choosing λ

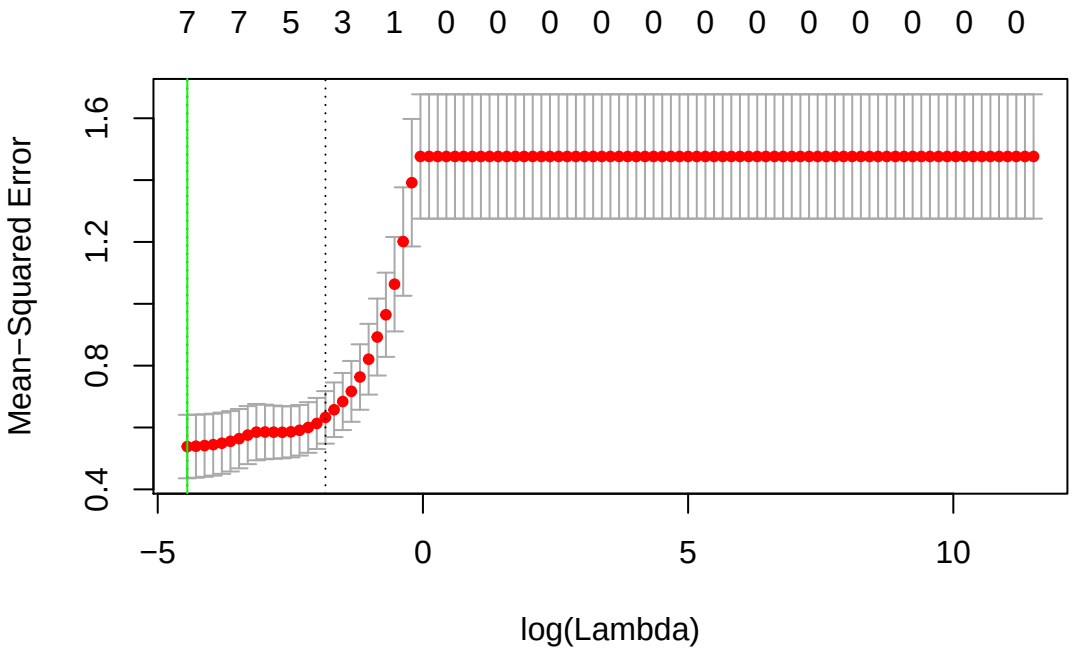
Again, the goal is to find the λ value which will minimize MSE. Doing this via cross validation avoids overfitting the model.

What happens to the coefficients and predictions when λ is large (8.5×10^4) verus λ being small (0.05)?

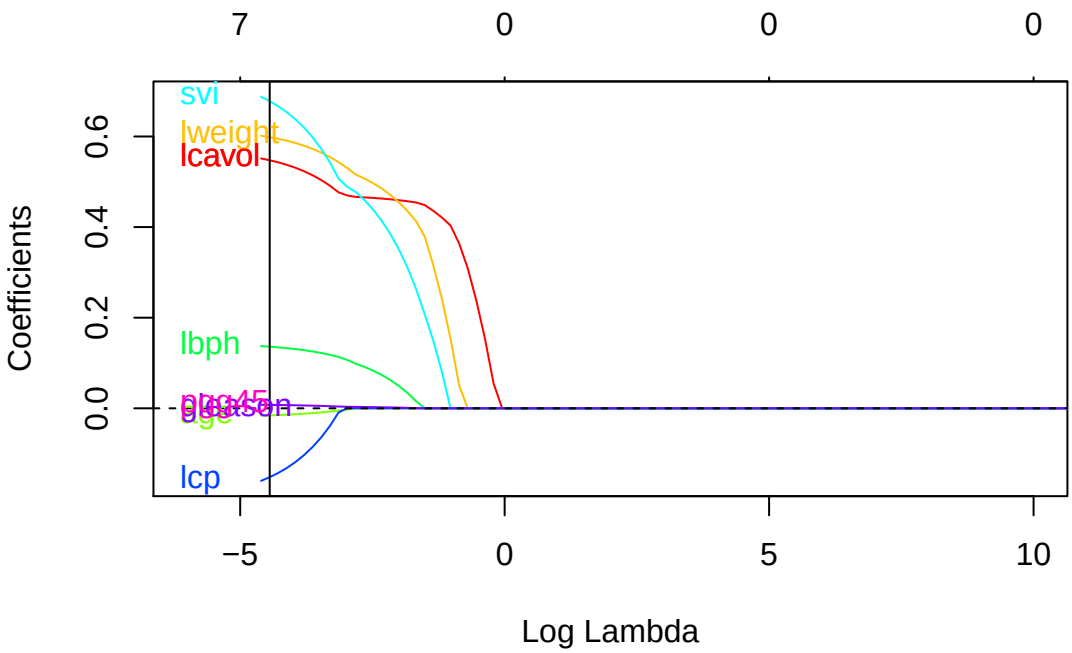
```
dim(coef(pr.lasso))  # 9 coef, 100 lambda values

## [1]   9 100

lambda.grid[2]

## [1] 84975

coef(pr.lasso)[,2] # note glmnet unstandardizes the variables

## (Intercept)      lcavol     lweight         age        lbph         svi
##         2.5         0.0         0.0         0.0         0.0         0.0
##         lcp     gleason       pgg45
##         0.0         0.0         0.0

lambda.grid[90]

## [1] 0.051

coef(pr.lasso)[,90]  # note glmnet unstandardizes the variables

## (Intercept)      lcavol     lweight         age        lbph         svi
##     -0.1228      0.4698      0.5306     -0.0027      0.1068      0.4888
##         lcp     gleason       pgg45
##      0.0000      0.0000      0.0034
```

In the following figure, the lower dashed (green) vertical line gives the λ with the smallest cross validated MSE. The "other" line gives the most regularized (that is: smaller coefficients!) λ that is within one SE (a box whisker) of the minimum λ value.

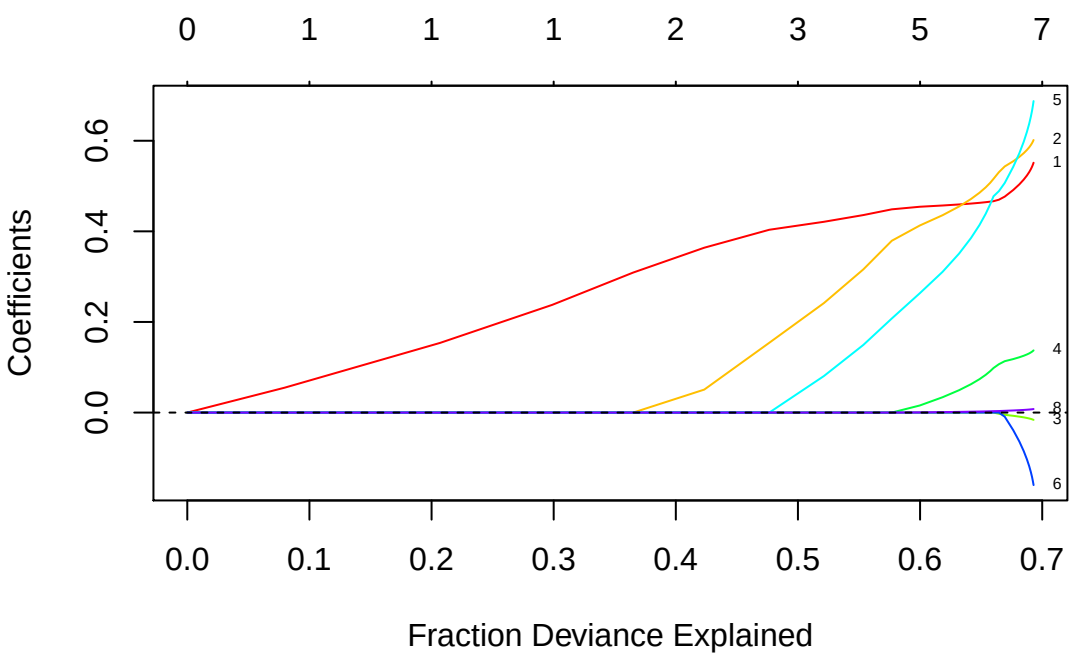
```
pr.lasso.cv <- cv.glmnet(as.matrix(pr.train[,1:8]), pr.train$lpsa,
                         alpha=1, lambda = lambda.grid, standardize=TRUE)
plot(pr.lasso.cv)
abline(v=log(pr.lasso.cv$lambda.min), col="green")
```



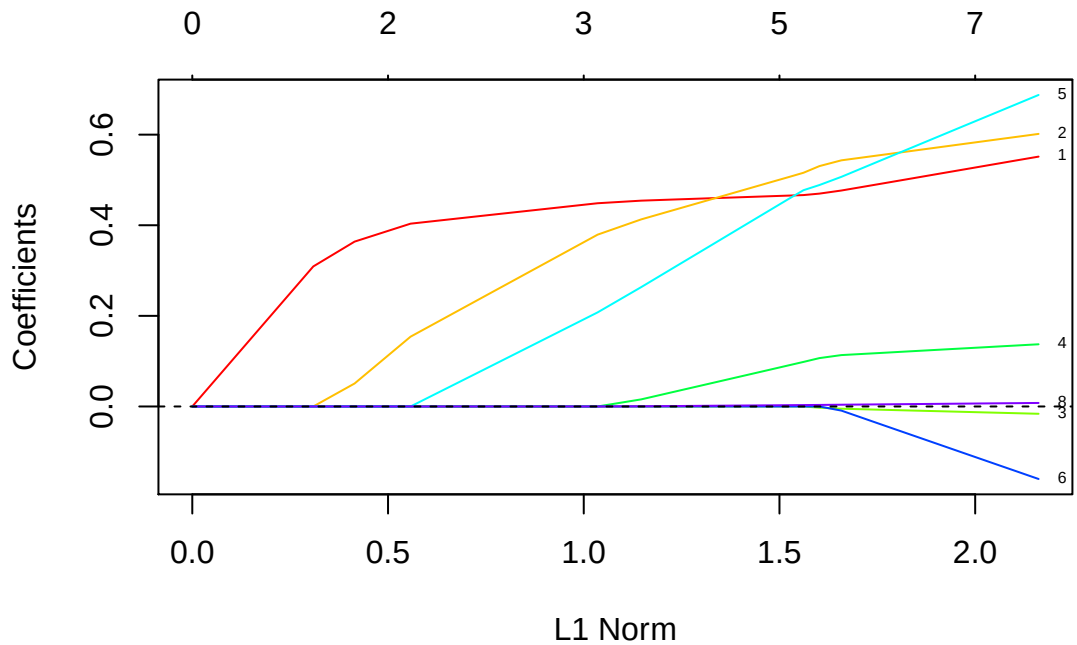

```
plot(pr.lasso, xvar="lambda", xlim=c(-6,10), col=colors)
pr.lasso.cv$lambda.min

## [1] 0.012

abline(v=log(pr.lasso.cv$lambda.min))
abline(h=0, lty=2)
text(rep(-6.5, 9), coef(pr.lasso)[-1,length(lambda.grid)], colnames(pr.train)[-9], pos=4, col=colors)
```




```
plot(pr.lasso, xvar="dev",col=colors, label=TRUE)
abline(h=0, lty=2)
```



```r
plot(pr.lasso, xvar="norm",col=colors, label=TRUE)
abline(h=0, lty=2)
```



The Lasso Model

Using the λ value for the minimum MSE in the cross validation, we output the coefficients / model associated with the best λ for the lasso model. Note that here (possibly because there wasn't much collinearity!), only one coefficient was set to zero. Lasso does do variable selection!

```r
coef(pr.lasso.cv, s = "lambda.min")

## 9 x 1 sparse Matrix of class "dgCMatrix"
##                    1
## (Intercept)  0.1756
## lcavol       0.5475
## lweight      0.5985
## age         -0.0155
## lbph         0.1360
## svi          0.6780
## lcp         -0.1520
## gleason      .
## pgg45        0.0076
```

Comparing Models

```
forw.pred <- predict(pr.fwd, newdata = pr.test)
back.pred <- predict(pr.bck, newdata = pr.test)
ridge.pred <- predict(pr.ridge.cv, newx = as.matrix(pr.test[,1:8]),
                      s = "lambda.min")
lasso.pred <- predict(pr.lasso.cv, newx = as.matrix(pr.test[,1:8]),
                      s = "lambda.min")

sum((forw.pred - pr.test$lpsa)^2)

## [1] 14

sum((back.pred - pr.test$lpsa)^2)

## [1] 15

sum((ridge.pred - pr.test$lpsa)^2)

## [1] 15

sum((lasso.pred - pr.test$lpsa)^2)

## [1] 15
```

Forward Won!