Problem Set 4: Linear Regression Models

Stat 154, Fall 2017, Prof. Sanchez

Self grade due date: Th Oct-26 (before midnight)

Instructions

Do not give raw computer output as your main answer to any question. Remember that providing a clear and reasonable justification of your answers is at least as important as getting the right answer.

Problem 1 (10 pts)

Multicollinearity is easy to detect in the two-predictor case; we need only look at the value of $r_{12} = cor(X_1, X_2)$. When there are more than two regressors, however, inspection of the r_{ij} is not sufficient.

For example, assume that we have four predictors X_1, X_2, X_3 and X_4, and correlation coefficients, r_{ij} are $r_{12} = r_{13} = r_{23} = 0$, with variances $\sigma_1^2 = \sigma_2^2 = \sigma_3^2$, and $X_4 = X_1 + X_2 + X_3$.

Show that $r_{14} = r_{24} = r_{34} = 0.577$

Notice that predictors X_1, X_2, and X_3 are uncorrelated since correlation coefficients $r_{12} = r_{13} = r_{23} = 0$.

The formula to calculate correlations with X_4 is

$$r_{j4} = \frac{cov(X_j, X_4)}{\sigma_j \times \sigma_4}$$

The variance σ_4^2 is

$$\sigma_4^2 = var(X_1 + X_2 + X_3) = \sigma_1^2 + \sigma_2^2 + \sigma_3^2 = 3\sigma_1^2$$

The correlation r_{14} is thus:

$$
\begin{aligned}
r_{14} &= \frac{cov(X_1, X_4)}{\sigma_1 \times \sigma_4} \\
&= \frac{cov(X_1, X_1 + X_2 + X_3)}{\sigma_1 \times \sqrt{3\sigma_1^2}} \\
&= \frac{cov(X_1, X_1)}{\sigma_1^2\sqrt{3}} \\
&= \frac{\sigma_1^2}{\sigma_1^2\sqrt{3}} \\
&= \frac{1}{\sqrt{3}} = 0.577
\end{aligned}
$$

The same procedure applies to r_{24} and r_{34}.

Problem 2 (10 pts)

In Partial Least Squares Regression, we obtain uncorrelated componentes $\mathbf{z_1}, \mathbf{z_2}, \ldots, \mathbf{z_h}$ that summarize variability in predictors $\mathbf{X}$ as well as capture variation in the response $\mathbf{y}$.

One of the properties of the original PLS regression algorithm is that it produces orthogonal components. Show that any two components $\mathbf{z_h}$ and $\mathbf{z_l}$ ($h \neq l$) are indeed orthogonal, that is:

$$\mathbf{z_h^T z_l} = 0 \quad \text{for} \quad h \neq l$$

Hint: The demonstration is done by recursivity.

We have $\mathbf{z_1^T z_2} = \mathbf{z_1^T}(\mathbf{X_1 w_2}) = 0$, because $\mathbf{z_1^T X_1} = 0$.

Assume that $\mathbf{z_1}, \ldots, \mathbf{z_h}$ are orthogonal, we claim that the vectors $\mathbf{z_1}, \ldots, \mathbf{z_h}, \mathbf{z_{h+1}}$ are orthogonal.

$$\mathbf{z_h^T z_{h+1}} = \mathbf{z_h^T}(\mathbf{X_h w_{h+1}}) = 0$$

because $\mathbf{z_h^T X_h} = 0$.

Thus:

$$
\begin{aligned}
\mathbf{z_{h-1}^T z_{h+1}} &= \mathbf{z_{h-1}^T}(\mathbf{X_h w_{h+1}}) \\
&= \mathbf{z_{h-1}^T}[\mathbf{X_{h-1}} - \mathbf{z_h p_h^T}]\mathbf{w_{h+1}} \\
&= [\mathbf{z_{h-1}^T X_{h-1}} - \mathbf{z_{h-1}^T z_h p_h^T}]\mathbf{w_{h+1}} \\
&= 0
\end{aligned}
$$

because $\mathbf{z}_{h-1}^\top \mathbf{X}_{h-1} = 0$ and $\mathbf{z}_{h-1}^\top \mathbf{z}_h = 0$ from the recursivity approach.

$$\begin{aligned}\mathbf{z}_{h-2}^\top \mathbf{z}_{h+1} &= \mathbf{z}_{h-2}^\top \mathbf{X}_h \mathbf{w}_{h+1} \\ &= \mathbf{z}_{h-2}^\top [\mathbf{X}_{h-2} - \mathbf{z}_{h-1}\mathbf{p}_{h-1}^\top - \mathbf{z}_h \mathbf{p}_h^\top]\mathbf{w}_{h+1} \\ &= 0\end{aligned}$$

because $\mathbf{z}_{h-2}^\top \mathbf{X}_{h-2} = 0$ and $\mathbf{z}_{h-2}^\top \mathbf{z}_{h-1} = 0$, and so on.

Problem 3 (100 pts)

Data set

The example *Prostate Cancer* is introduced in section 3.2.1, page 49 of ESL. The data is from a study by Stamey et al. (1989) in which the level of prostate-specific antigen and a number of clinical measures in men who were about to receive a radical prostatectomy. The variables are:

- `lcavol`: log cancer volume
- `lweight`: log prostate weight
- `age`: age of patient
- `lbph`: log of the amount of benign prostatic hyperplasia
- `svi`: seminal vesicle invasion
- `lcp`: log of capsular penetration
- `gleason`: Gleason score
- `pgg45`: percent of Gleason scores 4 or 5
- `lpsa`: log of prostate-specific antigen (response variable)

```
# read data
pros <- read.table('prostate.data', row.names = 1)
```

Models

You will apply the following methods (using the associated function, and package) to predict `lpsa` using the rest of the variables as predictors:

- Ordinary Leat Squares regression (OLS), with function `lm()`.
- Best subset regression, with function `regsubsets()` in `"leaps"`.
- Principal Components regression (PCR), with function `pcr()` in `"pls"`.
- Partial Least Squares regression (PLSR), with function `pls()` in `"pls"`.
- Ridge regression (RR), with function `glmnet()` in `"glmnet"`.
- Lasso regression (lasso), with function `glmnet()` in `"glmnet"`.

Take a look at chapter 6 in ISL, especially the computer lab section 6.5, to learn about the listed functions (and packages). You may also want to look at the following vignettes:

- https://cran.r-project.org/web/packages/pls/vignettes/pls-manual.pdf
- https://web.stanford.edu/~hastie/glmnet/glmnet_alpha.html

Training and Test sets

The data table contains an additional column `train` (logical values) indicating which observations form the *training* set (`TRUE`), and which observations form the *test* set (`FALSE`).

- Use the column `train` to split the data into a training set and a test set. There should be 67 training observations, and 30 test observations.
- You will use the training set to fit all the models, and perform a *model assessment* stage for each applied method.
- You will use the test set in the *model selection* stage to determine which method provides the best predictive performance.

```
# training & test sets
training <- pros$train
n <- sum(training)
test <- !training
```

```
# training data set
dat <- pros[training,-10]
```

Correlations of predictors, and preprocessing (10 pts)

Obtain the matrix of correlations of predictors. These correlations are like those displayed in table 3.1 (page 50).

Obtaining a matrix of correlations for the predictors is straightforward with the function `cor()`.

```
# matrix of correlations
round(cor(dat[ ,-9]), 3)
```

```
##         lcavol lweight   age   lbph    svi    lcp gleason  pgg45
## lcavol   1.000   0.300 0.286  0.063  0.593  0.692   0.426  0.483
## lweight  0.300   1.000 0.317  0.437  0.181  0.157   0.024  0.074
## age      0.286   0.317 1.000  0.287  0.129  0.173   0.366  0.276
## lbph     0.063   0.437 0.287  1.000 -0.139 -0.089   0.033 -0.030
## svi      0.593   0.181 0.129 -0.139  1.000  0.671   0.307  0.481
## lcp      0.692   0.157 0.173 -0.089  0.671  1.000   0.476  0.663
## gleason  0.426   0.024 0.366  0.033  0.307  0.476   1.000  0.757
## pgg45    0.483   0.074 0.276 -0.030  0.481  0.663   0.757  1.000
```

In case you are wondering whether it is possible to get a table of correlations "exactly" like table 3.2, here's one possibility in R (there are other ways to do this):

```
# table 3.1 (page 50)
# correlations (in a distance matrix object)
as.dist(round(cor(dat[ ,-9]), 3))
```

```
##         lcavol lweight    age   lbph    svi    lcp gleason
## lweight  0.300
## age      0.286   0.317
## lbph     0.063   0.437  0.287
## svi      0.593   0.181  0.129 -0.139
## lcp      0.692   0.157  0.173 -0.089  0.671
## gleason  0.426   0.024  0.366  0.033  0.307  0.476
## pgg45    0.483   0.074  0.276 -0.030  0.481  0.663   0.757
```

Preprocessing of predictors

Once you've split the data and selected the training set, you need to standardize the predictors (mean = 0, variance = 1). Confirm you get the following `summary()` statistics

The standardization of predictors can be done in various ways. Perhaps the quickest way is via `scale()`. For convenience, and to avoid scaling the response, I've decided to first mean-center all variables, and then scale just the predictors. You don't have to follow my approach. The important thing is to confirm that you obtained similar `summary()` stats to those displayed in the PDF of instructions.

```
# centering predictors
pred_means <- colMeans(dat)
pred_means[9] <- 0   # except the response
dat <- sweep(dat, 2, pred_means)

# standardizing predictors
pred_sdevs <- apply(dat, 2, sd)
pred_sdevs[9] <- 1   # except the response
dat <- sweep(dat, 2, pred_sdevs, FUN = "/")
```

- summary statistics for lcavol, lweight, age

```
     lcavol           lweight              age          
 Min.   :-2.1411   Min.   :-2.62526   Min.   :-3.16524  
 1st Qu.:-0.6641   1st Qu.:-0.62054   1st Qu.:-0.49935  
 Median : 0.1242   Median :-0.05755   Median : 0.03382  
 Mean   : 0.0000   Mean   : 0.00000   Mean   : 0.00000  
 3rd Qu.: 0.8334   3rd Qu.: 0.54029   3rd Qu.: 0.56700  
 Max.   : 2.0180   Max.   : 2.42189   Max.   : 1.89994  
```

- summary statistics for lbph, svi, lcp

```
      lbph                svi                lcp          
 Min.   :-0.99595   Min.   :-0.5331   Min.   :-0.8368  
 1st Qu.:-0.99595   1st Qu.:-0.5331   1st Qu.:-0.8368  
 Median :-0.08385   Median :-0.5331   Median :-0.4171  
 Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.0000  
 3rd Qu.: 1.00848   3rd Qu.:-0.5331   3rd Qu.: 0.8631  
 Max.   : 1.54057   Max.   : 1.8480   Max.   : 2.0496  
```

- summary statistics for gleason, pgg45

```
    gleason            pgg45        
 Min.   :-1.032   Min.   :-0.8965  
 1st Qu.:-1.032   1st Qu.:-0.8965  
 Median : 0.379   Median :-0.3846  
 Mean   : 0.000   Mean   : 0.0000  
 3rd Qu.: 0.379   3rd Qu.: 0.8099  
 Max.   : 3.200   Max.   : 2.5163  
```

Least Squares Model (10 pts)

The first linear model you will fit is an ordinary least squares regression. Regress the response `lpsa` on the standardized predictors (using the training data). See if you can reproduce the table 3.2 in ESL, page 48.

To most convenient way to fit an OLS regression is with the function `lm()`

```
# OLS
ols_reg <- lm(lpsa ~ ., data = dat)
ols_coeffs <- ols_reg$coefficients
ols_sum <- summary(ols_reg)
ols_sum
```

```
##
## Call:
## lm(formula = lpsa ~ ., data = dat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -1.64870 -0.34147 -0.05424  0.44941  1.48675 
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept)  2.45235    0.08702  28.182  < 2e-16 ***
```

```
## lcavol       0.71641    0.13350   5.366 1.47e-06 ***
## lweight      0.29264    0.10638   2.751  0.00792 **
## age         -0.14255    0.10212  -1.396  0.16806
## lbph         0.21201    0.10312   2.056  0.04431 *
## svi          0.30962    0.12539   2.469  0.01651 *
## lcp         -0.28901    0.15480  -1.867  0.06697 .
## gleason     -0.02091    0.14258  -0.147  0.88389
## pgg45        0.27735    0.15959   1.738  0.08755 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7123 on 58 degrees of freedom
## Multiple R-squared:  0.6944, Adjusted R-squared:  0.6522
## F-statistic: 16.47 on 8 and 58 DF,  p-value: 2.042e-12
```

My version of the table 3.2 (rounding to 2 decimal digits) looks like this. Notice that I'm computing the Z scores following the description of ESL, because I wanted to contrast the results against those in table 3.2

```r
# table 3.2, page 50 (some values don't match)
table_32 <- data.frame(
  term = rownames(ols_sum$coefficients),
  coefficient = round(ols_sum$coefficients[ ,1], 2),
  std_error = round(ols_sum$coefficients[ ,2], 2),
  z_score = round(ols_sum$coefficients[ ,1] / ols_sum$coefficients[ ,2], 2),
  row.names = 1:nrow(ols_sum$coefficients)
)
print(table_32, print.gap = 2)
```

```
##           term  coefficient  std_error  z_score
## 1  (Intercept)         2.45       0.09    28.18
## 2       lcavol         0.72       0.13     5.37
## 3      lweight         0.29       0.11     2.75
## 4          age        -0.14       0.10    -1.40
## 5         lbph         0.21       0.10     2.06
## 6          svi         0.31       0.13     2.47
## 7          lcp        -0.29       0.15    -1.87
## 8      gleason        -0.02       0.14    -0.15
## 9        pgg45         0.28       0.16     1.74
```

As I mention in the HW instructions, I have the suspicion that the first three coefficients reported in the book's table may be wrong. This seems to be confirmed by the output above. The larger differences are in `intercept`, `lcavol`, and `lweight`.

Best Subset Regression (10 pts)

Use the function `regsubsets()`, from the package `"leaps"`, to find the best subset regression.

```
# Best Subset regression
bset_reg <- regsubsets(lpsa ~ ., data = dat)
bset_sum <- summary(bset_reg)
bset_sum
```

```
## Subset selection object
## Call: regsubsets.formula(lpsa ~ ., data = dat)
## 8 Variables  (and intercept)
##         Forced in Forced out
## lcavol      FALSE      FALSE
## lweight     FALSE      FALSE
## age         FALSE      FALSE
## lbph        FALSE      FALSE
## svi         FALSE      FALSE
## lcp         FALSE      FALSE
## gleason     FALSE      FALSE
## pgg45       FALSE      FALSE
## 1 subsets of each size up to 8
## Selection Algorithm: exhaustive
##          lcavol lweight age lbph svi lcp gleason pgg45
## 1  ( 1 ) "*"    " "     " " " "  " " " " " " "     " "
## 2  ( 1 ) "*"    "*"     " " " "  " " " " " " "     " "
## 3  ( 1 ) "*"    "*"     " " " "  "*" " " " " "     " "
## 4  ( 1 ) "*"    "*"     " " "*"  "*" " " " " "     " "
## 5  ( 1 ) "*"    "*"     " " "*"  "*" " " " " "     "*"
## 6  ( 1 ) "*"    "*"     " " "*"  "*" "*" " " "     "*"
## 7  ( 1 ) "*"    "*"     "*" "*"  "*" "*" " " "     "*"
## 8  ( 1 ) "*"    "*"     "*" "*"  "*" "*" "*"     "*"
```

An asterisk indicates that a given variable is included in the corresponding model. For instance, the best two-variable model contains only `lcavol` and `lweight`. The `summary()` function also returns things like R^2, RSS, adjusted R^2, C_p, and BIC.

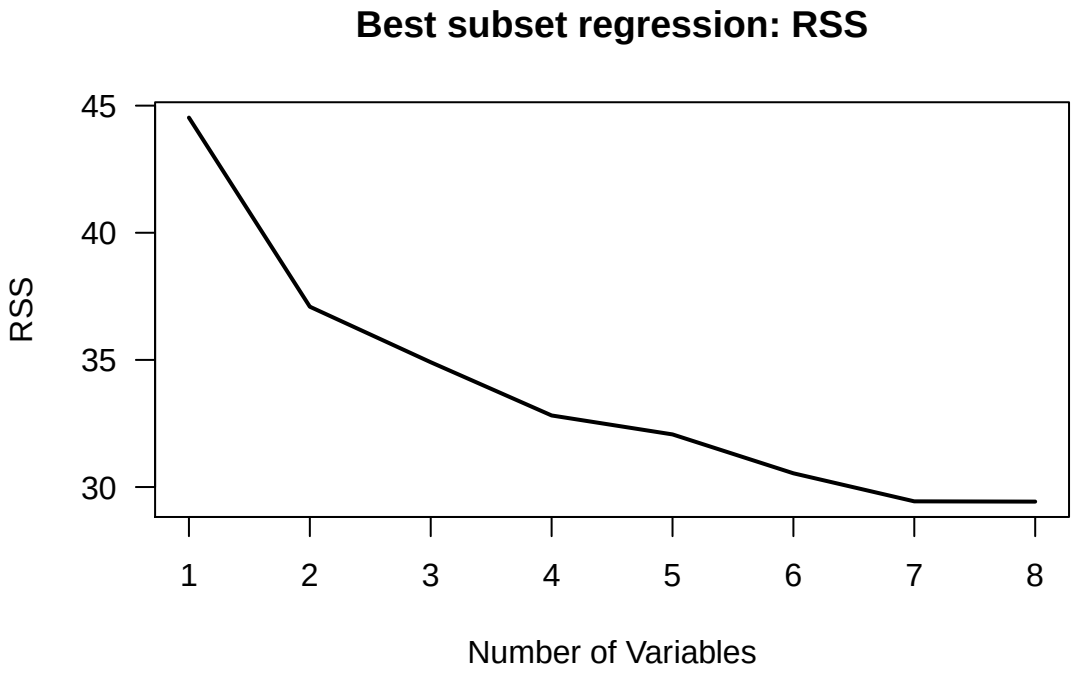
```
names(bset_sum)
```

```
## [1] "which"  "rsq"    "rss"    "adjr2"  "cp"     "bic"    "outmat" "obj"
```

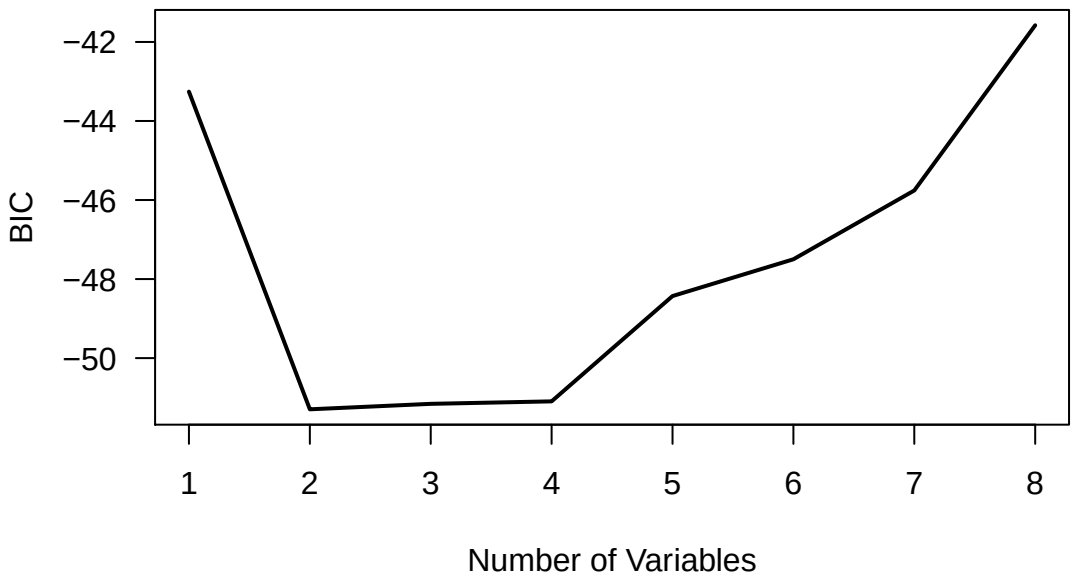
You can plot RSS, adjusted R^2, C_p, and BIC for all the models. In this case, let's plot the RSS and BIC:

```
plot(bset_sum$rss, ylab = "RSS", type = 'l', lwd = 2, las = 1,
     xlab = "Number of Variables",
```

```
     main = 'Best subset regression: RSS')
```




```
plot(bset_sum$bic, ylab = "BIC", type = 'l', lwd = 2, las = 1,
     xlab = "Number of Variables",
     main = 'Best subset regression: BIC')
```



To select the best model you may use the adjusted R^2, BIC, Cp, etc. The closest criterion that matches the output in ESL is BIC

```
which.min(bset_sum$bic)
```

```
## [1] 2
```

```
bset_coeffs <- coef(bset_reg, which.min(bset_sum$bic))
bset_coeffs
```

```
## (Intercept)      lcavol     lweight
##   2.4523451   0.7798589   0.3519101
```

Because the vector of best subset coefficients has shorter length than the number of predictors, I've decided to create an expanded vector full of zeros for those predictors not in the best subset. This will be used for the last table of coefficients for all regression methods.

```
bestset_coeffs <- rep(0, ncol(dat[,-9]))
bestset_coeffs[(names(dat[,-9]) %in% names(bset_coeffs))] <- bset_coeffs[-1]
bestset_coeffs <- c(bset_coeffs[1], bestset_coeffs)
names(bestset_coeffs) <- c("Intercept", names(dat[,-9]))
bestset_coeffs
```

```
## Intercept    lcavol   lweight       age      lbph       svi       lcp
## 2.4523451 0.7798589 0.3519101 0.0000000 0.0000000 0.0000000 0.0000000
##   gleason     pgg45
## 0.0000000 0.0000000
```

PC Regression (20 pts)

Fit PCR models with the training data, respectively, using ten-fold cross validation. Since the predictors are already standardized, you don't really need to set the argument `scale = TRUE`. Make sure to set a random seed so you can reproduce all results.

```
# PCR with ten-fold CV (using training data)
set.seed(123)
pc_reg <- pcr(lpsa ~ ., data = dat, scale = FALSE, validation = "CV")
```

Make a plot of *Profiles of Coefficients* for each method; the x-axis corresponds to the number of components, and the y-axis corresponds to the coefficients.

To obtain a plot of coefficient profiles, you typically need to extract the coefficients (which are in a 3-dim array). The code below retrieves the coefficients into a matrix object.

Ir order to make a graph with ggplot (you don't have to do this), the matrix of coeffs is melted via `melt()` from package `"reshape2"`. This creates a data frame that can be passed to `ggplot()`.

```
# PCR coefficients in matrix format
pcr_coeffs_mat <- apply(pc_reg$coefficients, 3, function(x) x)
colnames(pcr_coeffs_mat) <- paste0('comp', 1:(ncol(dat)-1))
rownames(pcr_coeffs_mat) <- rownames(pc_reg$coefficients)
pcr_coeffs_mat
```

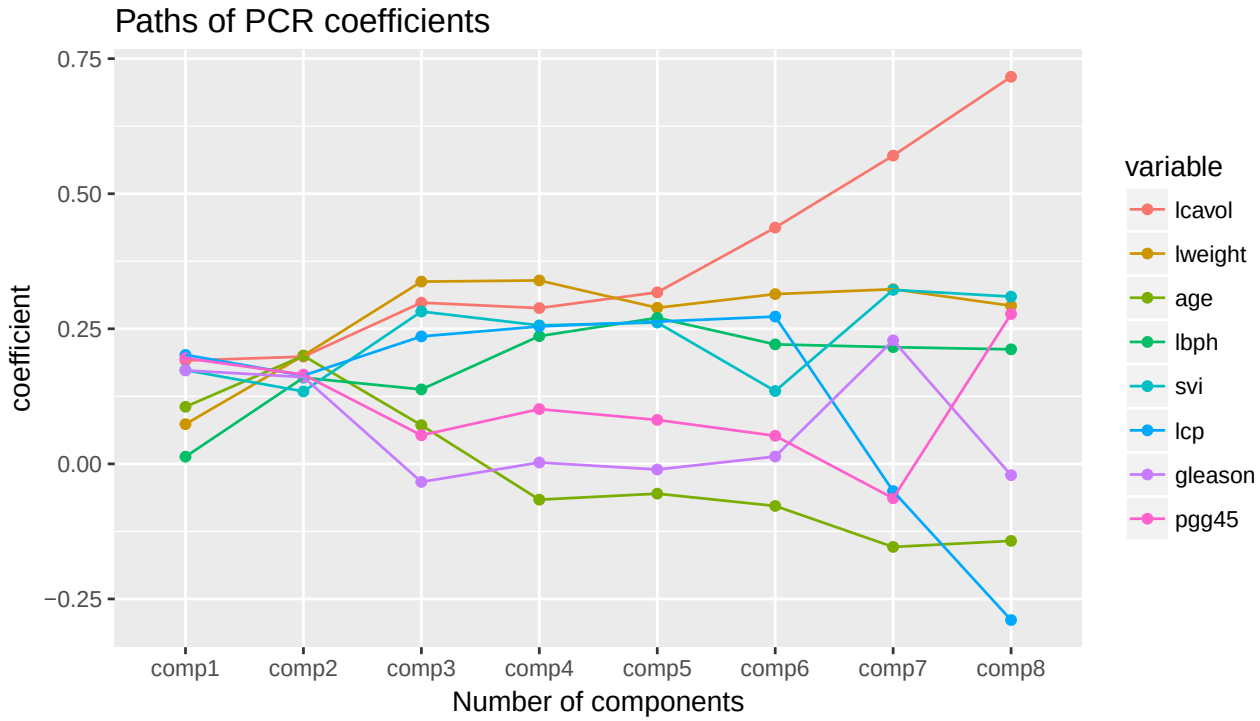
```
##              comp1     comp2       comp3        comp4       comp5
## lcavol  0.19120767 0.1984668  0.29836605  0.288375673  0.31745479
## lweight 0.07349303 0.2005002  0.33732251  0.339495295  0.28891140
## age     0.10565703 0.2000902  0.07156608 -0.066233524 -0.05513596
## lbph    0.01331490 0.1595677  0.13784771  0.236471613  0.27026315
## svi     0.17354994 0.1340809  0.28196567  0.256396248  0.26154340
## lcp     0.20172472 0.1634854  0.23588571  0.254376132  0.26331376
## gleason 0.17302293 0.1606244 -0.03341946  0.002551456 -0.01043395
## pgg45   0.19564717 0.1652961  0.05292742  0.101385484  0.08137749
##               comp6       comp7       comp8
## lcavol   0.43729893  0.57058087  0.71640701
## lweight  0.31423037  0.32328124  0.29264240
## age     -0.07787142 -0.15371952 -0.14254963
## lbph     0.22116755  0.21599970  0.21200760
## svi      0.13480517  0.32212005  0.30961953
## lcp      0.27260007 -0.05040169 -0.28900562
## gleason  0.01356923  0.22857290 -0.02091352
## pgg45    0.05198279 -0.06362645  0.27734595
```

```
# data frame of PCR coefficients (to be passed to ggplot)
pcr_coeffs_tbl <- melt(
  pcr_coeffs_mat,
  varnames = c('variable', 'component'),
  value.name = 'coefficient')
head(pcr_coeffs_tbl)
```

```
##   variable component coefficient
## 1   lcavol     comp1  0.19120767
## 2  lweight     comp1  0.07349303
## 3      age     comp1  0.10565703
## 4     lbph     comp1  0.01331490
## 5      svi     comp1  0.17354994
## 6      lcp     comp1  0.20172472
```

```
ggplot(data = pcr_coeffs_tbl,
       aes(x = component, y = coefficient,
           color = variable, group = variable)) +
  geom_point() +
  geom_path() +
  xlab('Number of components') +
```

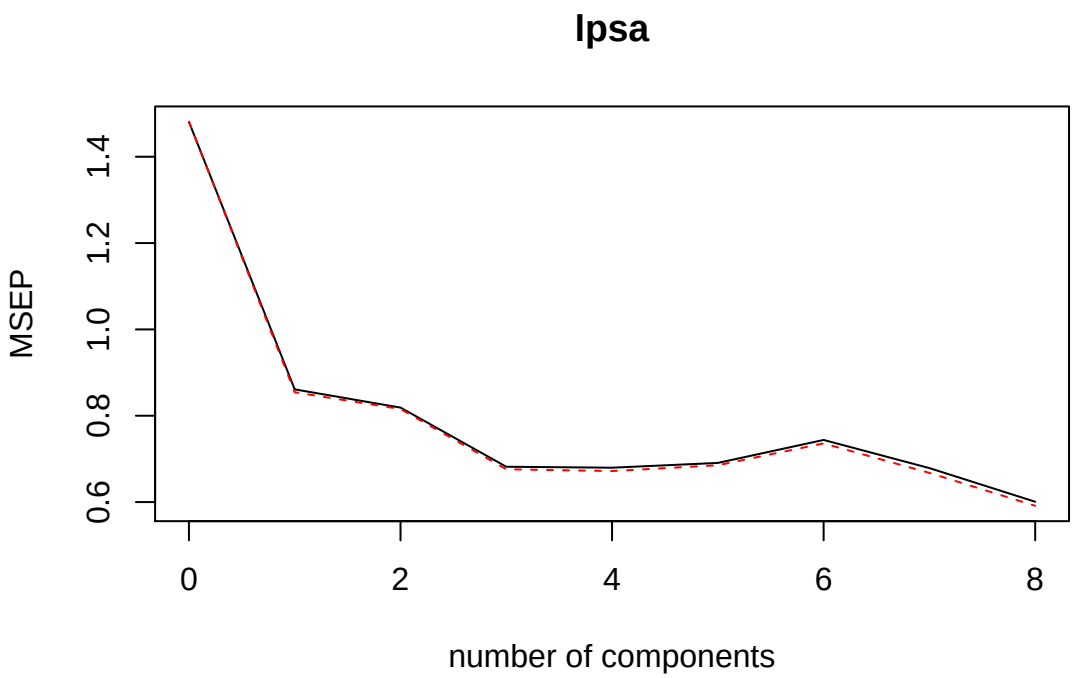
```
  ggtitle('Paths of PCR coefficients')
```



Notice the shrinking effect of the coefficients. As the number of PCs in the regression increases, so does the size of the coefficients.

Examine CV output

```
validationplot(pc_reg, val.type = "MSEP")
```



The above plot can also be obtained with:

```
# equivalent to validationplot() call
plot(MSEP(pc_reg))
```

Determine best number of principal components to be retained

```
pcr_mse <- MSEP(pc_reg)
pcr_cv_mse <- apply(pcr_mse$val, MARGIN = 3, function(x) x[1])
pcr_ncomp <- which.min(pcr_cv_mse[-1])
pcr_ncomp
```

```
## 8 comps
##       8
```

```
pc_reg$coefficients[,,pcr_ncomp]
```

```
##      lcavol     lweight         age        lbph         svi         lcp
##  0.71640701  0.29264240 -0.14254963  0.21200760  0.30961953 -0.28900562
##     gleason       pgg45
## -0.02091352  0.27734595
```

PLS Regression (20 pts)

Fit PLSR models with the training data, respectively, using ten-fold cross validation

```r
# PSLR with ten-fold CV (using training data)
set.seed(123)
pls_reg <- plsr(lpsa ~ ., data = dat, scale = FALSE, validation = "CV")
```

Path of coefficient profiles.

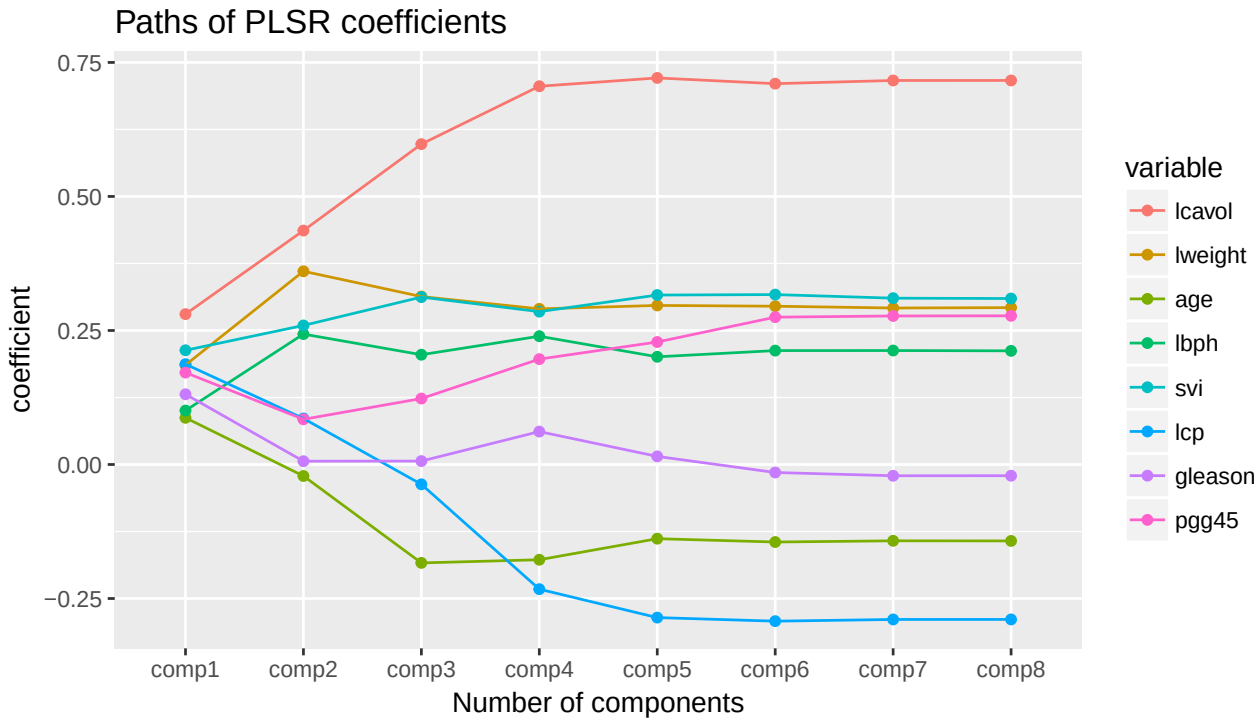
```r
# PLSR coefficients in matrix format
pls_coeffs_mat <- apply(pls_reg$coefficients, 3, function(x) x)
colnames(pls_coeffs_mat) <- paste0('comp', 1:(ncol(dat)-1))
rownames(pls_coeffs_mat) <- rownames(pls_reg$coefficients)
pls_coeffs_mat
```

```
##              comp1        comp2        comp3       comp4       comp5
## lcavol  0.28050274  0.436397115  0.597652312  0.70559130  0.72110274
## lweight 0.18564173  0.360459926  0.313302670  0.29036457  0.29673339
## age     0.08709522 -0.021442783 -0.183568300 -0.17768182 -0.13838310
## lbph    0.10059907  0.243273865  0.204788538  0.23927755  0.20089737
## svi     0.21306291  0.259381105  0.312294962  0.28512233  0.31612880
## lcp     0.18716753  0.085848230 -0.036982946 -0.23255263 -0.28556290
## gleason 0.13101175  0.006153702  0.006372098  0.06140048  0.01513561
## pgg45   0.17142167  0.084284153  0.123029999  0.19659086  0.22852247
##              comp6       comp7       comp8
## lcavol   0.7104094  0.71636186  0.71640701
## lweight  0.2952801  0.29192059  0.29264240
## age     -0.1446106 -0.14233890 -0.14254963
## lbph     0.2124677  0.21259719  0.21200760
## svi      0.3169434  0.31026935  0.30961953
## lcp     -0.2922292 -0.28905136 -0.28900562
## gleason -0.0149234 -0.02101011 -0.02091352
## pgg45    0.2748280  0.27706347  0.27734595
```

```r
# data frame of PLSR coefficients (to be passed to ggplot)
pls_coeffs_tbl <- melt(
  pls_coeffs_mat,
  varnames = c('variable', 'component'),
  value.name = 'coefficient')
head(pls_coeffs_tbl)
```

```
##   variable component coefficient
## 1   lcavol     comp1  0.28050274
## 2  lweight     comp1  0.18564173
## 3      age     comp1  0.08709522
## 4     lbph     comp1  0.10059907
## 5      svi     comp1  0.21306291
## 6      lcp     comp1  0.18716753
```

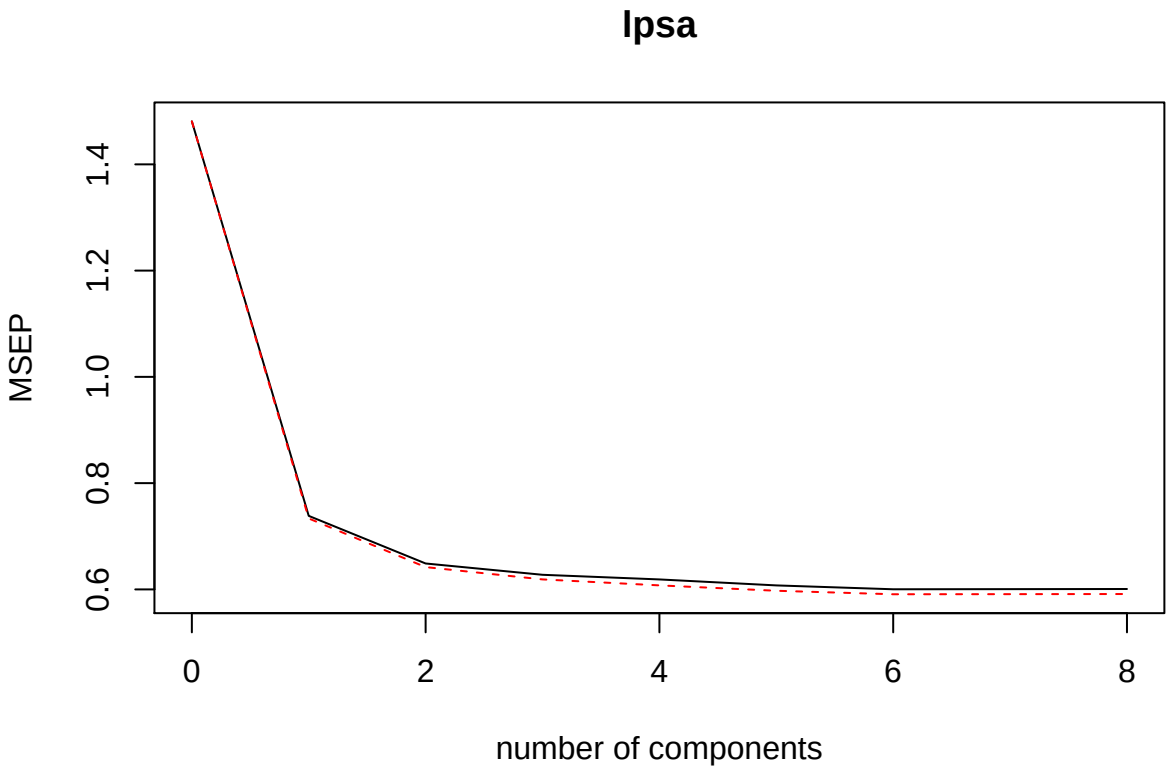
```
ggplot(data = pls_coeffs_tbl,
       aes(x = component, y = coefficient,
           color = variable, group = variable)) +
  geom_point() +
  geom_path() +
  xlab('Number of components') +
  ggtitle('Paths of PLSR coefficients')
```



Again, notice the shrinking of the coefficients. With few PLS components, the size of the coefficients is small; as we add more components to the regression equation, the size of the coefficients increases. Using all PLS components is equivalent to the OLS solution.

Examine CV output

```
validationplot(pls_reg, val.type = "MSEP")
```



Determine best number of PLS components to be retained

```
pls_mse <- MSEP(pls_reg)
pls_cv_mse <- apply(pls_mse$val, MARGIN = 3, function(x) x[1])
pls_ncomp <- which.min(pls_cv_mse[-1])
pls_ncomp
```

```
## 6 comps
##       6
```

```
pls_reg$coefficients[,,pls_ncomp]
```

```
##     lcavol    lweight        age       lbph        svi        lcp
##  0.7104094  0.2952801 -0.1446106  0.2124677  0.3169434 -0.2922292
##    gleason      pgg45
## -0.0149234  0.2748280
```

Ridge Regression (20 pts)

Fit Ridge regression models with the training data, respectively, using ten-fold cross validation

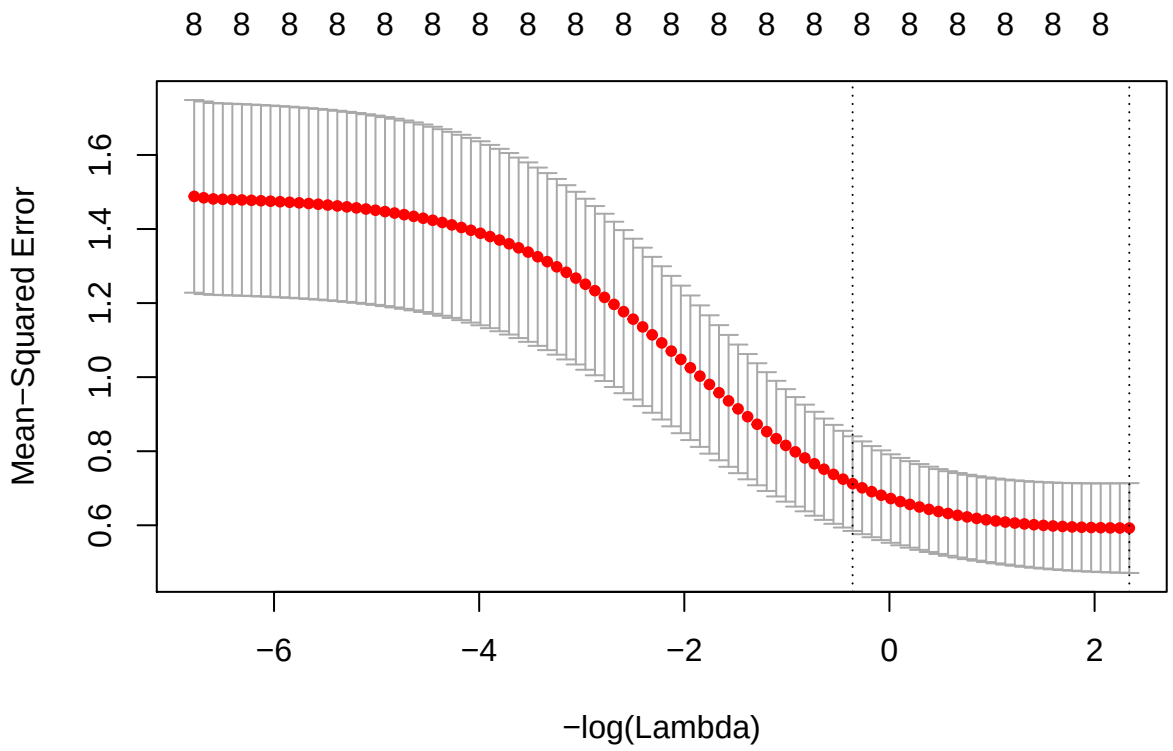
```
# RR with ten-fold CV (using training data)
x <- model.matrix(lpsa ~ ., data = dat)[ ,-1]
```

```r
y <- dat$lpsa

set.seed(123)
ridge_cv <- cv.glmnet(x, y, alpha = 0)
```

Examine CV output

```r
plot(ridge_cv, sign.lambda = -1)
```




```r
# best lambda from CV
ridge_lambda <- ridge_cv$lambda.min
ridge_lambda
```

```
## [1] 0.09645702
```

```r
coef(ridge_cv, s = "lambda.min")
```

```
## 9 x 1 sparse Matrix of class "dgCMatrix"
##                       1
## (Intercept)  2.45234509
## lcavol       0.60438317
## lweight      0.28576500
## age         -0.10858418
## lbph         0.20096586
## svi          0.28336365
## lcp         -0.15469409
```

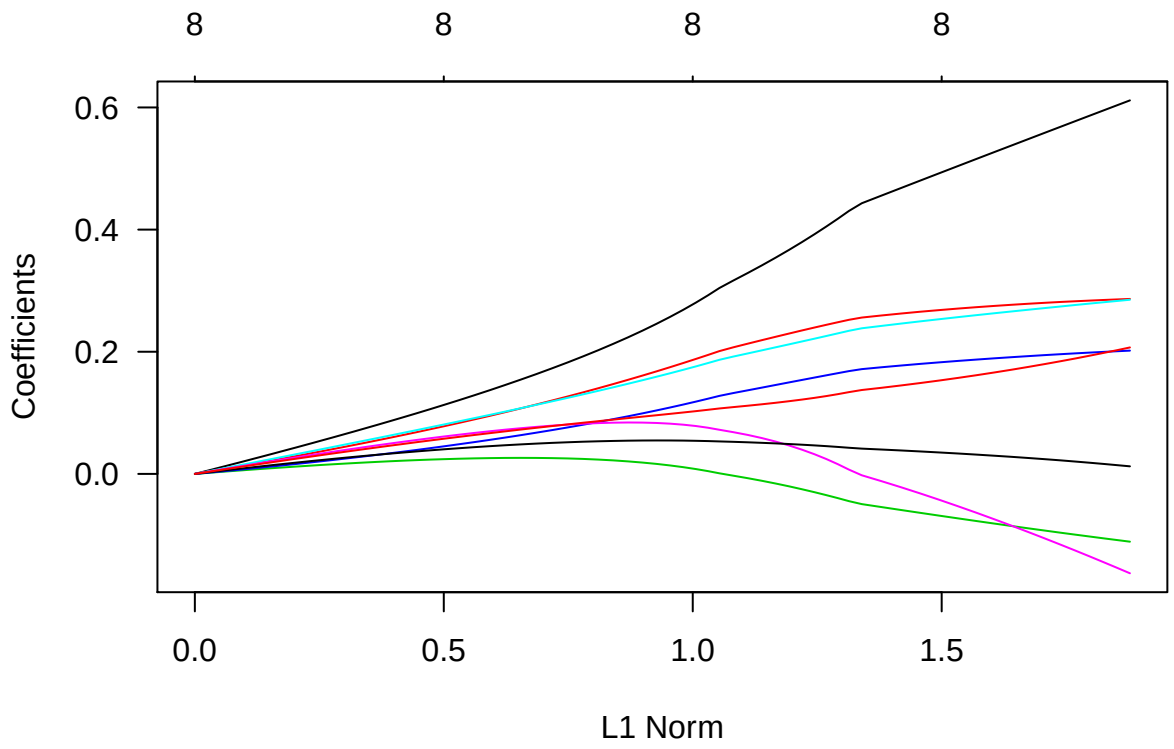
```
## gleason      0.01414138
## pgg45        0.20305366
```

Examine Paths plot

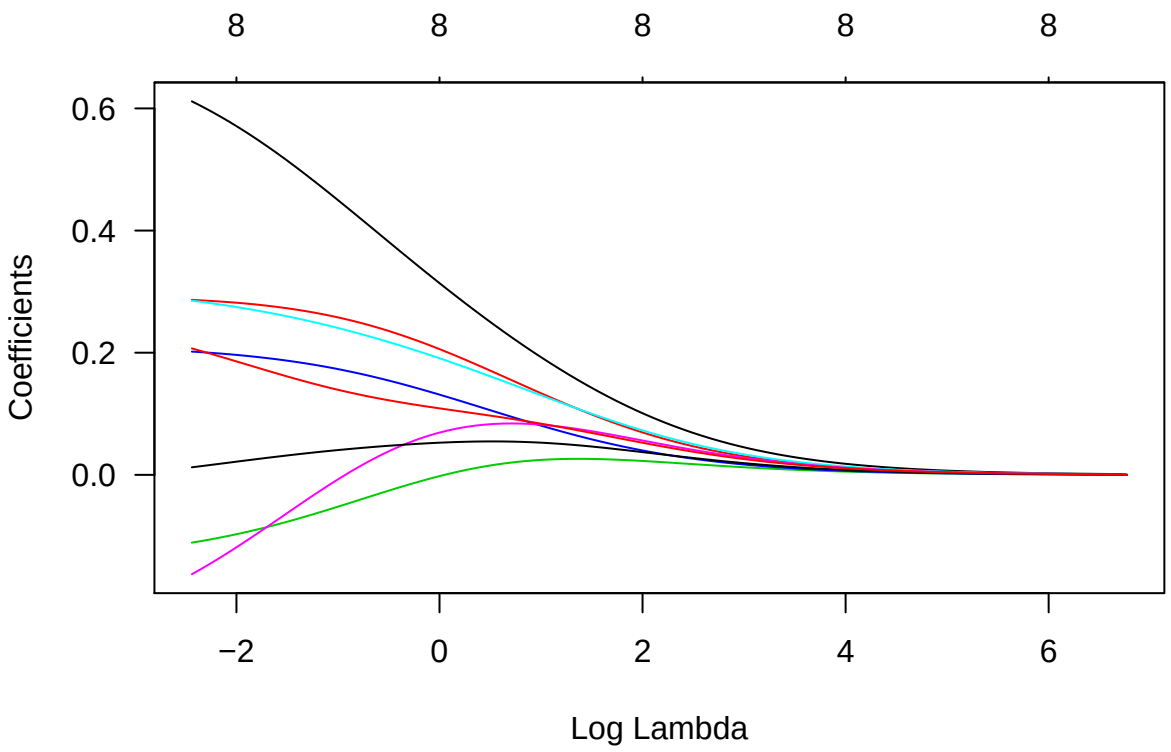
```
ridge_fit <- glmnet(x, y, alpha = 0, standardize = FALSE)
predict(ridge_fit, type = "coefficients", s = ridge_lambda)[1:9,]
```

```
## (Intercept)      lcavol     lweight         age        lbph         svi
##  2.45234509  0.60309861  0.28562935 -0.10815349  0.20080140  0.28304294
##         lcp     gleason       pgg45
## -0.15327237  0.01444703  0.20235376
```

```
plot(ridge_fit, las = 1)
```




```
plot(ridge_fit, xvar = "lambda", las = 1)
```



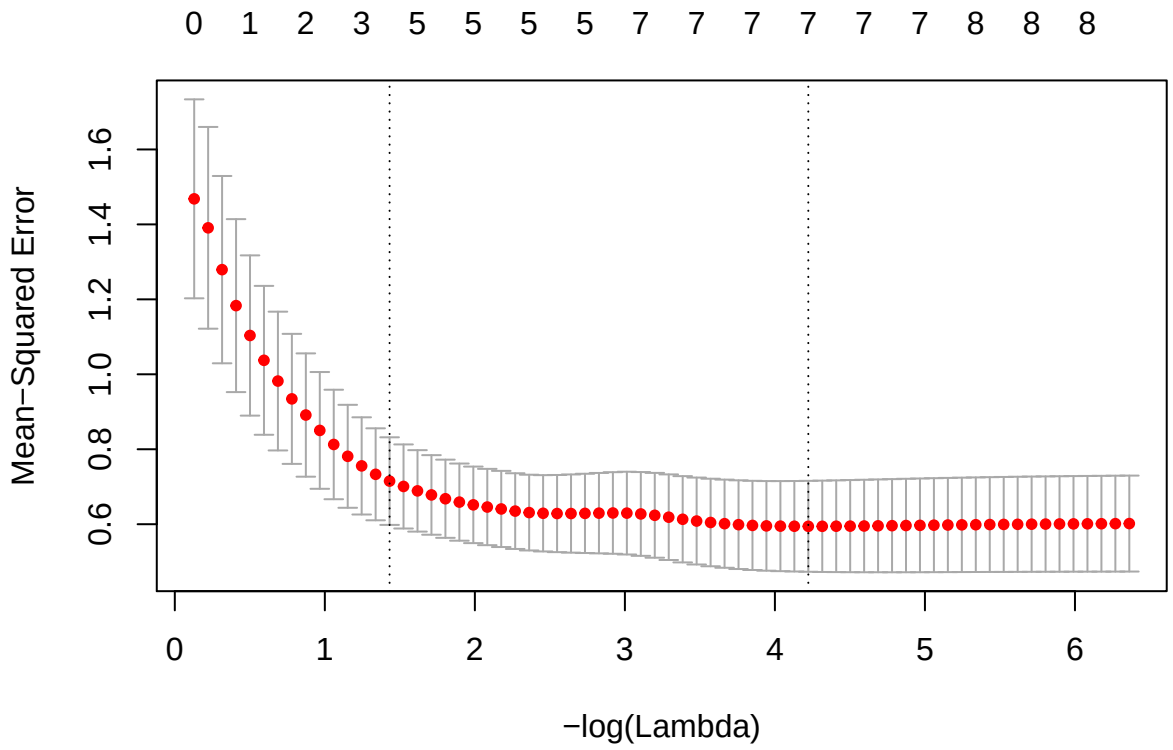
Lasso Regression (20 pts)

Fit Lasso models with the training data, respectively, using ten-fold cross validation

```
# Lasso with ten-fold CV (using training data)
set.seed(123)
lasso_cv <- cv.glmnet(x, y, alpha = 1)
```

Examine CV output

```
plot(lasso_cv, sign.lambda = -1)
```




```
# best lambda from CV
lasso_lambda <- lasso_cv$lambda.min
lasso_lambda
```

```
## [1] 0.01466061
```

```
coef(lasso_cv, s = "lambda.min")
```

```
## 9 x 1 sparse Matrix of class "dgCMatrix"
##                       1
## (Intercept)  2.4523451
## lcavol       0.6722784
## lweight      0.2828372
## age         -0.1092166
## lbph         0.1960157
## svi          0.2781206
## lcp         -0.1945150
## gleason      .
## pgg45        0.2116699
```

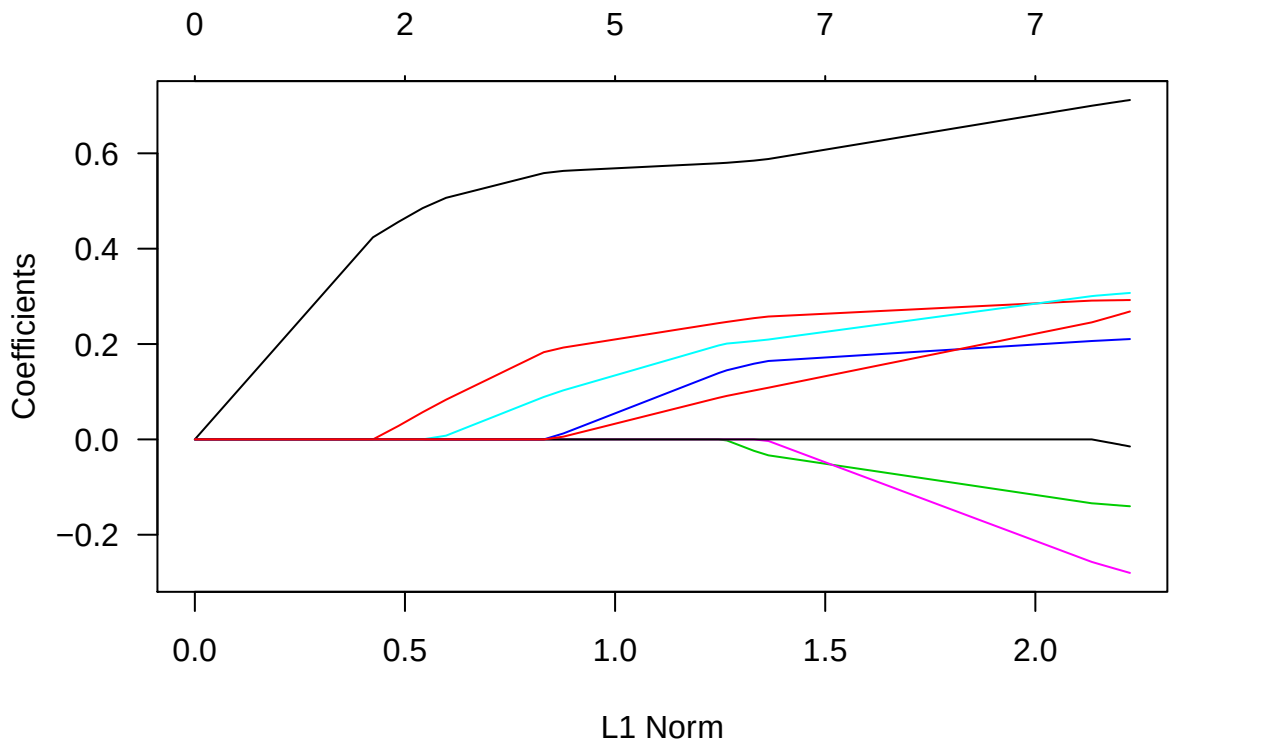
Examine Paths plot

```
lasso_fit <- glmnet(x, y, alpha = 1)
predict(lasso_fit, type = "coefficients", s = lasso_lambda)[1:9,]
```

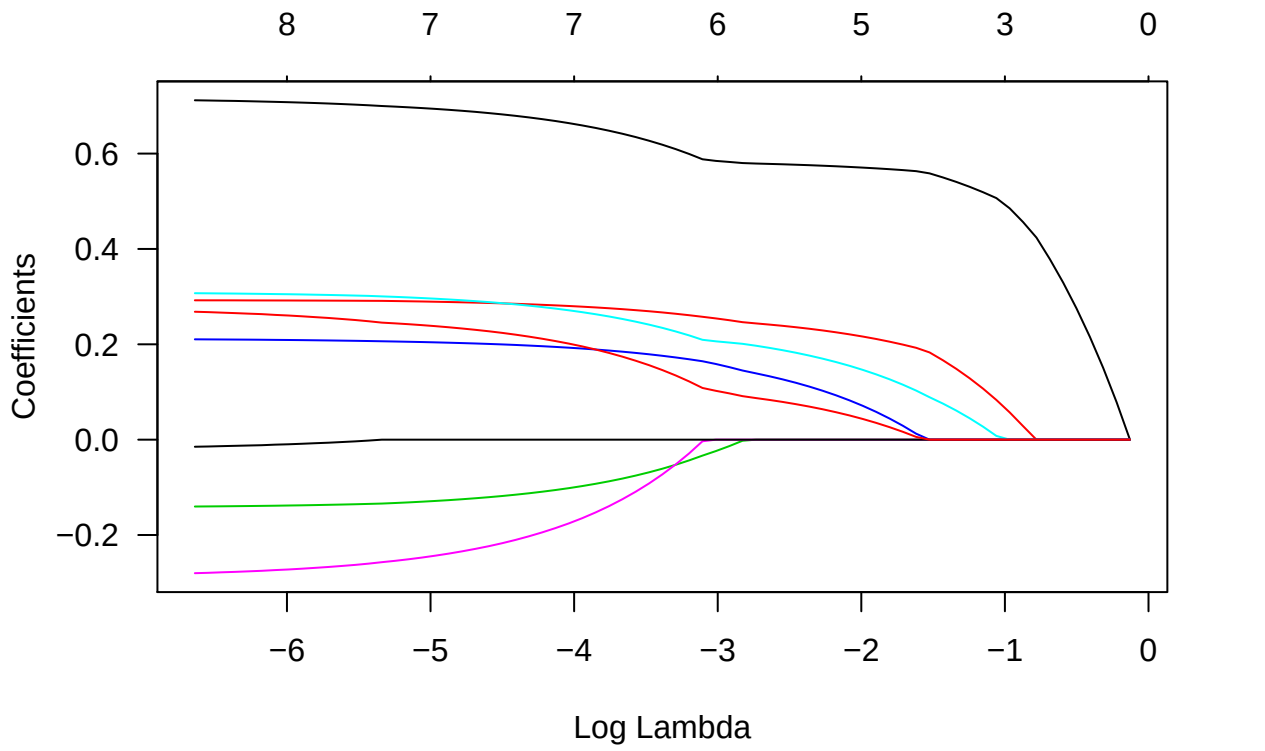
```
## (Intercept)      lcavol     lweight         age        lbph         svi
##   2.4523451   0.6722784   0.2828372  -0.1092166   0.1960157   0.2781206
```

```
##         lcp     gleason       pgg45
##  -0.1945150   0.0000000   0.2116699
```

```
plot(lasso_fit, las = 1)
```




```
plot(lasso_fit, xvar = "lambda", las = 1)
```



Model Selection (20 pts)

Test MSEs

```
Xtest <- scale(pros[test,1:8])
ytest <- pros$lpsa[test]

# ols regression
ols_test_yhat <- predict(ols_reg, data.frame(Xtest, ytest))
ols_test_mse <- mean((ytest - ols_test_yhat)^2)

# best subset
bset_fit <- lm(lpsa ~ lcavol + lweight, data = dat)
bset_test_yhat <- predict(bset_fit, data.frame(Xtest, ytest))
bset_test_mse <- mean((ytest - bset_test_yhat)^2)

# pcr
pcr_test_yhat <- predict(pc_reg, data.frame(Xtest), ncomp = pcr_ncomp)
pcr_test_mse <- mean((ytest - as.vector(pcr_test_yhat))^2)

# pls
pls_test_yhat <- predict(pls_reg, data.frame(Xtest), ncomp = pls_ncomp)
pls_test_mse <- mean((ytest - as.vector(pls_test_yhat))^2)

# ridge
ridge_mod <- glmnet(x, y, alpha = 0, lambda = ridge_lambda, nlambda = 1)
ridge_test_yhat <- predict(ridge_mod, newx = Xtest, s = ridge_lambda)
ridge_test_mse <- mean((ytest - as.vector(ridge_test_yhat))^2)

# lasso
lasso_mod <- glmnet(x, y, alpha = 1, lambda = lasso_lambda, nlambda = 1)
lasso_test_yhat <- predict(lasso_mod, newx = Xtest, s = lasso_lambda)
lasso_test_mse <- mean((ytest - as.vector(lasso_test_yhat))^2)

MSE <- c(
  'ols' = ols_test_mse,
  'subset' = bset_test_mse,
  'ridge' = ridge_test_mse,
  'lasso' = lasso_test_mse,
  'pcr' = pcr_test_mse,
  'pls' = pls_test_mse
```

```
)
MSE
```

```
##       ols    subset     ridge     lasso       pcr       pls
## 0.5491941 0.5483947 0.5171760 0.5179034 0.5491941 0.5493153
```

Table of Coefficients

```
coef_table <- data.frame(
  OLS = coef(ols_reg),
  BestSub = bestset_coeffs,
  Ridge = as.vector(coef.glmnet(ridge_fit, s = ridge_lambda)),
  Lasso = as.vector(coef.glmnet(lasso_fit, s = lasso_lambda)),
  PCR = as.vector(coef(pc_reg, ncomp = pcr_ncomp, intercept = TRUE)),
  PLS = as.vector(coef(pls_reg, ncomp = pls_ncomp, intercept = TRUE))
)

print(round(coef_table, 4), print.gap = 1)
```

```
##                 OLS BestSub   Ridge   Lasso     PCR     PLS
## (Intercept)  2.4523  2.4523  2.4523  2.4523  2.4523  2.4523
## lcavol       0.7164  0.7799  0.6031  0.6723  0.7164  0.7104
## lweight      0.2926  0.3519  0.2856  0.2828  0.2926  0.2953
## age         -0.1425  0.0000 -0.1082 -0.1092 -0.1425 -0.1446
## lbph         0.2120  0.0000  0.2008  0.1960  0.2120  0.2125
## svi          0.3096  0.0000  0.2830  0.2781  0.3096  0.3169
## lcp         -0.2890  0.0000 -0.1533 -0.1945 -0.2890 -0.2922
## gleason     -0.0209  0.0000  0.0144  0.0000 -0.0209 -0.0149
## pgg45        0.2773  0.0000  0.2024  0.2117  0.2773  0.2748
```