ISL - Chapter 6 Lab Tutorials
Linear Model Selection and Regularization

An introduction to Statistical Learning, with Applications in R
- G. James, D. Witten, T. Hastie, R. Tibshirani

Thu Nguyen

26 June, 2019

Contents

Main Contents:

1. Subset Selection
2. Shrinkage Methods
3. Dimension Reduction Methods
4. Considerations in High Dimensions

6.5. Lab 1: Subset Selection Methods

6.5.1. Best Subset Selection

```r
library(ISLR)
# Hitters dataset from ISLR, and remove NA from Salary column
Hitters <- na.omit(Hitters)
print(paste0('Dimension after removing NA: ', dim(Hitters)))
```

```
## [1] "Dimension after removing NA: 263" "Dimension after removing NA: 20"
```

`regsubsets()` Best subset selection, from library `leaps`

```r
# regsubsets(): Best subset selection, from library 'leaps'
library(leaps)
regfit.full <- regsubsets(Salary ~ ., data = Hitters)
summary(regfit.full)
```

```
## Subset selection object
## Call: regsubsets.formula(Salary ~ ., data = Hitters)
## 19 Variables  (and intercept)
##            Forced in Forced out
## AtBat          FALSE      FALSE
## Hits           FALSE      FALSE
## HmRun          FALSE      FALSE
## Runs           FALSE      FALSE
## RBI            FALSE      FALSE
## Walks          FALSE      FALSE
## Years          FALSE      FALSE
## CAtBat         FALSE      FALSE
## CHits          FALSE      FALSE
## CHmRun         FALSE      FALSE
## CRuns          FALSE      FALSE
## CRBI           FALSE      FALSE
## CWalks         FALSE      FALSE
## LeagueN        FALSE      FALSE
## DivisionW      FALSE      FALSE
## PutOuts        FALSE      FALSE
## Assists        FALSE      FALSE
## Errors         FALSE      FALSE
## NewLeagueN     FALSE      FALSE
## 1 subsets of each size up to 8
## Selection Algorithm: exhaustive
##          AtBat Hits HmRun Runs RBI Walks Years CAtBat CHits CHmRun CRuns
## 1  ( 1 ) " "   " "  " "   " "  " " " "   " "   " "    " "   " "    " "
## 2  ( 1 ) " "   "*"  " "   " "  " " " "   " "   " "    " "   " "    " "
## 3  ( 1 ) " "   "*"  " "   " "  " " " "   " "   " "    " "   " "    " "
## 4  ( 1 ) " "   "*"  " "   " "  " " " "   " "   " "    " "   " "    " "
## 5  ( 1 ) "*"   "*"  " "   " "  " " " "   " "   " "    " "   " "    " "
## 6  ( 1 ) "*"   "*"  " "   " "  " " "*"   " "   " "    " "   " "    " "
## 7  ( 1 ) " "   "*"  " "   " "  " " "*"   " "   "*"    "*"   "*"    " "
```

```
## 8  ( 1 ) "*"   "*"  " "   " "  " " "*"   " "   " "    " "   "*"    "*"
##           CRBI CWalks LeagueN DivisionW PutOuts Assists Errors NewLeagueN
## 1  ( 1 ) "*"  " "    " "     " "       " "     " "     " "    " "
## 2  ( 1 ) "*"  " "    " "     " "       " "     " "     " "    " "
## 3  ( 1 ) "*"  " "    " "     " "       "*"     " "     " "    " "
## 4  ( 1 ) "*"  " "    " "     "*"       "*"     " "     " "    " "
## 5  ( 1 ) "*"  " "    " "     "*"       "*"     " "     " "    " "
## 6  ( 1 ) "*"  " "    " "     "*"       "*"     " "     " "    " "
## 7  ( 1 ) " "  " "    " "     "*"       "*"     " "     " "    " "
## 8  ( 1 ) " "  "*"    " "     "*"       "*"     " "     " "    " "
```

By default: regsubsets() only include up to 8 variables, to increase, specify `nvmax = p`, with p variables

```r
regfit.full <- regsubsets(Salary ~ ., data = Hitters, nvmax = 19)
reg.summary <- summary(regfit.full)
mes <- 'Available attributes of model'
print(cat(mes, '\n', names(reg.summary), '\n'))
```

```
## Available attributes of model
##  which rsq rss adjr2 cp bic outmat obj
## NULL
```

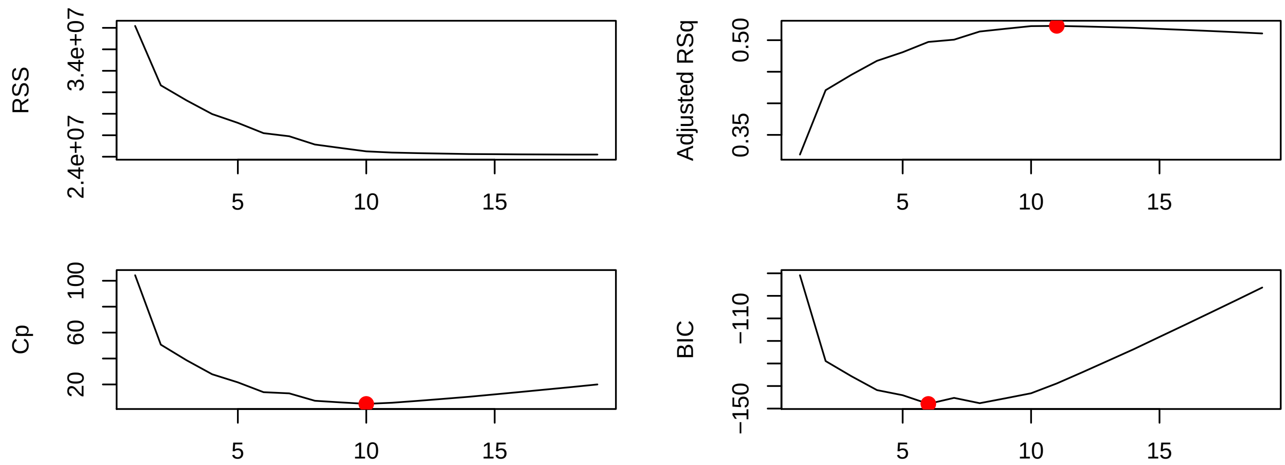
```r
# R^2
round(reg.summary$rsq, 4)
```

```
##  [1] 0.3215 0.4252 0.4514 0.4754 0.4908 0.5087 0.5141 0.5286 0.5346 0.5405
## [11] 0.5426 0.5436 0.5445 0.5452 0.5455 0.5458 0.5460 0.5461 0.5461
```

Interpretation: 1 var: $R^2 = .32$, 2 vars: $R^2 = .425$, … 19 vars: $R^2 = .546$

Plot

```r
# Plot
par(mfrow=c(2,2)); par(mar=c(3,5,1,1))
# RSS
plot(reg.summary$rss, xlab = 'Number of Variables', ylab = 'RSS', type = 'l')
# Adjusted R^2
plot(reg.summary$adjr2, xlab = 'Number of Variables', ylab = 'Adjusted RSq', type = 'l')
# Max(Adjusted R^2)
points(11, reg.summary$adjr2[11], col = 'red', cex = 2, pch = 20)

# C_p
plot(reg.summary$cp, xlab = 'Number of Variables', ylab = 'Cp', type = 'l')
points(10, reg.summary$cp[10], col = 'red', cex = 2, pch = 20)

# BIC
plot(reg.summary$bic, xlab = 'Number of Variables', ylab = 'BIC', type = 'l')
points(6, reg.summary$bic[6], col = 'red', cex = 2, pch = 20)
```




```r
attr.names <- c('Max(Adjusted R^2)', 'Min(Cp)', 'Min(BIC')
attr.values <- c(which.max(reg.summary$adjr2), which.min(reg.summary$cp), which.min(reg.summary$bic))
data.frame(Attributes = attr.names, Number_of_Variables = attr.values)
```

```
##          Attributes Number_of_Variables
## 1 Max(Adjusted R^2)                  11
## 2           Min(Cp)                  10
## 3           Min(BIC                   6
```

```r
# Plot with labelled best variables
# par(mfrow=c(1,1))
# plot(regfit.full, scale = 'r2')
# plot(regfit.full, scale = 'adjr2')
# plot(regfit.full, scale = 'Cp')
# plot(regfit.full, scale = 'bic')
```

Interpretation:

- The top row of each plot contains a black square for each variable selected according to the optimal model. For instance, we see that several models share a `BIC` close to -150. However, the model with the lowest `BIC` is the 6-variable model that contains only `AtBat`, `Hits`, `Walks`, `CRBI`, `DivisionW`, and `PutOuts`.

```r
mes <- 'Coefficients of model with 6 var'
print(cat(mes, '\n', round(coef(regfit.full, 6),4), '\n'))
```

```
## Coefficients of model with 6 var
##  91.5118 -1.8686 7.6044 3.6976 0.643 -122.9515 0.2643
## NULL
```

```r
round(coef(regfit.full, 6),4)
```

```
## (Intercept)       AtBat        Hits       Walks        CRBI   DivisionW
##     91.5118     -1.8686      7.6044      3.6976      0.6430   -122.9515
##     PutOuts
##      0.2643
```

6.5.2. Forward and Backward Stepwise Selection

to specify method: `method = 'forward/backward'`

Forward

```
regfit.fwd <- regsubsets(Salary ~ ., data = Hitters, nvmax = 19, method = 'forward')
# summary(regfit.fwd)
```

Backward

```
regfit.bwd <- regsubsets(Salary ~ ., data = Hitters, nvmax = 19, method = 'backward')
# summary(regfit.bwd)
```

Comment:

- For best models with 1-6 vars, both are the same
- 7 vars: Forward and Backward return different models

Full model

```
round(coef(regfit.full, 7),2)
```

```
## (Intercept)        Hits       Walks      CAtBat       CHits      CHmRun
##       79.45        1.28        3.23       -0.38        1.50        1.44
##   DivisionW     PutOuts
##     -129.99        0.24
```

Forward Stepwise model

```
round(coef(regfit.fwd, 7),2)
```

```
## (Intercept)       AtBat        Hits       Walks        CRBI      CWalks
##      109.79       -1.96        7.45        4.91        0.85       -0.31
##   DivisionW     PutOuts
##     -127.12        0.25
```

Backward Stepwise model

```
round(coef(regfit.bwd, 7),2)
```

```
## (Intercept)       AtBat        Hits       Walks       CRuns      CWalks
##      105.65       -1.98        6.76        6.06        1.13       -0.72
##   DivisionW     PutOuts
##     -116.17        0.30
```

6.5.3. Choosing among Models using Validation Set Approach and CV

```
# Split into train and test sets
set.seed(1)
train <- sample(c(TRUE, FALSE), nrow(Hitters), rep = TRUE)
test <- (!train)
# Best subset model
regfit.best <- regsubsets(Salary ~ ., data = Hitters[train,], nvmax = 19)
# Validation set test error
test.mat <- model.matrix(Salary ~ ., data = Hitters[test,])
# Test set MSE
val.errors <- c()
for (i in 1:19) {
  coefi <- coef(regfit.best, id = i)
  pred <- test.mat[, names(coefi)]%*%coefi
  val.errors[i] <- round(mean((Hitters$Salary[test] - pred)^2), 0)
}
```

MSE on test set for different number of variables, arranged by row.

```
matrix(val.errors, ncol = 5, byrow = T)
```

```
##        [,1]   [,2]   [,3]   [,4]   [,5]
## [1,] 220968 169157 178518 163426 168418
## [2,] 171271 162377 157909 154056 148162
## [3,] 151156 151742 152214 157359 158541
## [4,] 158743 159973 159860 160106 220968
```

Model with the best MSE

```
# Model with min(MSE)
mes <- 'The number of variables giving lowest MSE on test set: '
print(paste0(mes, which.min(val.errors)))
```

```
## [1] "The number of variables giving lowest MSE on test set: 10"
```

```
coef(regfit.best, which.min(val.errors))
```

```
## (Intercept)       AtBat        Hits       Walks      CAtBat       CHits
## -80.2751499  -1.4683816   7.1625314   3.6430345  -0.1855698   1.1053238
##      CHmRun      CWalks     LeagueN   DivisionW     PutOuts
##   1.3844863  -0.7483170  84.5576103 -53.0289658   0.2381662
```

`predict.regsubsets()` function, and 10-fold Cross-Validation

```
# predict fn, by hand
predict.regsubsets <- function(object, newdata, id, ...) {
  form <- as.formula(object$call[[2]])
  mat <- model.matrix(form, newdata)
  coefi <- coef(object, id = id)
  xvars <- names(coefi)
  mat[,xvars]%*%coefi
```

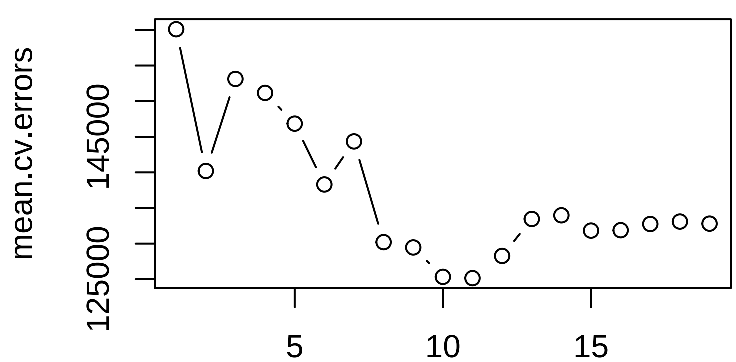
```
}

# 10-fold Cross-Validation
k <- 10
set.seed(1)
folds <- sample(1:k, nrow(Hitters), replace = TRUE)
cv.errors <- matrix(NA, k, 19, dimnames = list(NULL, paste(1:19)))
```

Average MSE on test set for each number of variables after 10-fold CV

```
# Steps: prediction -> test set errors
# Loop 1: j for each fold
for (j in 1:k) {
  best.fit <- regsubsets(Salary ~ ., data = Hitters[folds != j,], nvmax = 19)
  # Loop 2: i for each variable
  for (i in 1:19) {
    pred <- predict(best.fit, Hitters[folds == j,], id = i)
    cv.errors[j,i] <- mean( (Hitters$Salary[folds == j] - pred)^2 )
  }
}
mean.cv.errors <- round(apply(cv.errors, 2, mean),0)
mean.cv.errors
```

```
##      1      2      3      4      5      6      7      8      9     10
## 160093 140197 153117 151159 146841 138303 144346 130208 129460 125335
##     11     12     13     14     15     16     17     18     19
## 125154 128274 133461 133975 131826 131883 132751 133096 132805
```

```
par(mfrow = c(1,1)); par(mar=c(2,4,1,1))
plot(mean.cv.errors, type = 'b')
```



Comment: based on plot, best model has 11 variables

Coefficients for the model

```
reg.best <- regsubsets(Salary ~ ., data = Hitters, nvmax = 19)
round(coef(reg.best, 11),2)
```

```
## (Intercept)       AtBat        Hits       Walks      CAtBat       CRuns
##      135.75       -2.13        6.92        5.62       -0.14        1.46
##        CRBI      CWalks     LeagueN   DivisionW     PutOuts     Assists
##        0.79       -0.82       43.11     -111.15        0.29        0.27
```

6.6. Lab 2: Ridge Regression and the Lasso

`glmnet()` from package `glmnet`, for Ridge Regression: `alpha = 0`, Lasso: `alpha = 1`, also, by default, `glmnet()` standardizes the variables automatically.

```
library(glmnet)
# Rmb to remove na/missing values
# Convert from data.frame to matrix, also, categorical var -> dummy var
x <- model.matrix(Salary ~ ., Hitters)[,-1]
y <- Hitters$Salary
```

6.6.1. Ridge Regression

```
# grid for range of values for lambda
grid <- 10^seq(10, -2, length = 100)
# Ridge Regression model
ridge.mod <- glmnet(x, y, alpha = 0, lambda = grid)
# Result is a 20x100 matrix of predictors' coef. depending on lambda
mes <- 'Dimension of result model:'
print(paste(mes, dim(coef(ridge.mod))[1], 'x', dim(coef(ridge.mod))[2]))
```

```
## [1] "Dimension of result model: 20 x 100"
```

```
mes <- 'Available attributes from the model:'
print(cat(mes, '\n', names(ridge.mod), '\n'))
```

```
## Available attributes from the model:
##  a0 beta df dim lambda dev.ratio nulldev npasses jerr offset call nobs
## NULL
```

Example:

```
# Ex: at index 50, lambda = 11,498
lambda <- ridge.mod$lambda[50]
# l^2 norm
l2 <- round(sqrt(sum(coef(ridge.mod)[-1,50]^2)),2)
```

$$\text{At } \lambda = 11,497.57, l_2 \text{ norm } = 6.36, \text{ and the coefficients are:}$$

```
coef(ridge.mod)[,50]
```

```
##   (Intercept)         AtBat          Hits         HmRun          Runs
## 407.356050200   0.036957182   0.138180344   0.524629976   0.230701523
##           RBI         Walks         Years        CAtBat         CHits
##   0.239841459   0.289618741   1.107702929   0.003131815   0.011653637
##        CHmRun         CRuns          CRBI        CWalks       LeagueN
##   0.087545670   0.023379882   0.024138320   0.025015421   0.085028114
##     DivisionW       PutOuts       Assists        Errors    NewLeagueN
##  -6.215440973   0.016482577   0.002612988  -0.020502690   0.301433531
```

```r
# Ex: at index 50, lambda = 11,498
lambda <- ridge.mod$lambda[60]
# l^2 norm
l2 <- round(sqrt(sum(coef(ridge.mod)[-1,60]^2)),2)
```

Compare against different λ:

$$\text{At } \lambda = 705.48, l_2 \text{ norm } = 57.11, \text{ and the coefficients are:}$$

```r
coef(ridge.mod)[,60]
```

```
##  (Intercept)        AtBat         Hits        HmRun         Runs
##  54.32519950   0.11211115   0.65622409   1.17980910   0.93769713
##          RBI        Walks        Years       CAtBat        CHits
##   0.84718546   1.31987948   2.59640425   0.01083413   0.04674557
##       CHmRun        CRuns         CRBI       CWalks      LeagueN
##   0.33777318   0.09355528   0.09780402   0.07189612  13.68370191
##    DivisionW      PutOuts      Assists       Errors   NewLeagueN
## -54.65877750   0.11852289   0.01606037  -0.70358655   8.61181213
```

Predict

```r
# predict()
predict(ridge.mod, s = 50, type = 'coefficients')[1:20,]
```

```
##   (Intercept)         AtBat          Hits         HmRun          Runs
##  4.876610e+01 -3.580999e-01  1.969359e+00 -1.278248e+00  1.145892e+00
##           RBI         Walks         Years        CAtBat         CHits
##  8.038292e-01  2.716186e+00 -6.218319e+00  5.447837e-03  1.064895e-01
##        CHmRun         CRuns          CRBI        CWalks       LeagueN
##  6.244860e-01  2.214985e-01  2.186914e-01 -1.500245e-01  4.592589e+01
##     DivisionW       PutOuts       Assists        Errors    NewLeagueN
## -1.182011e+02  2.502322e-01  1.215665e-01 -3.278600e+00 -9.496680e+00
```

```r
# Split into train and test sets
set.seed(1)
train <- sample(1:nrow(x), nrow(x)/2)
test <- (-train)
y.test <- y[test]
```

Models

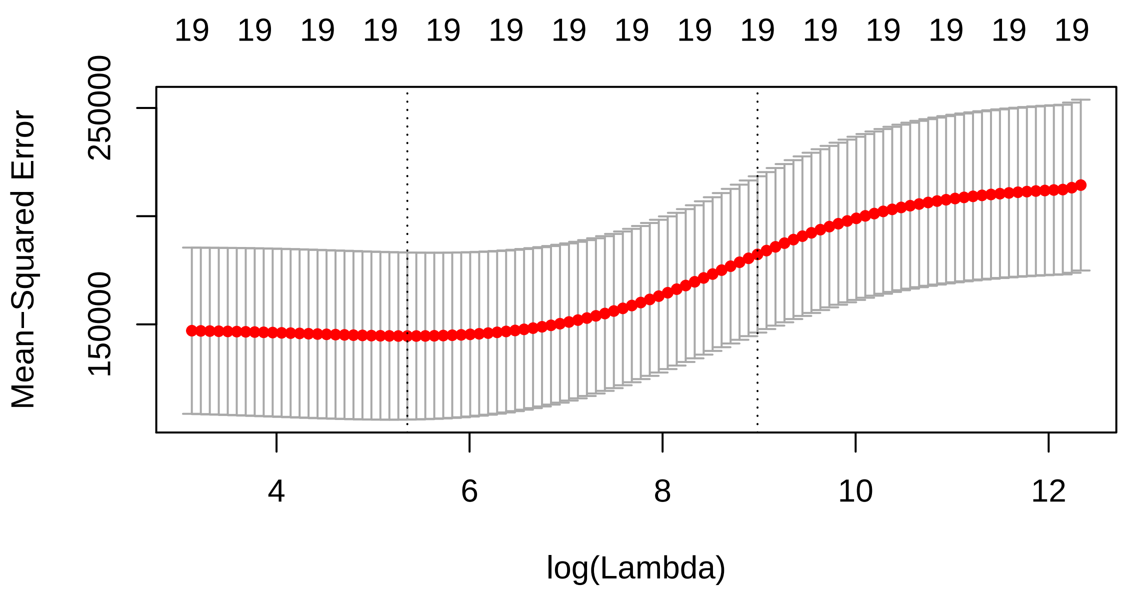
```r
ridge.mod <- glmnet(x[train,], y[train], alpha = 0, lambda = grid, thresh = 1e-12)
# lambda = s = 4, to specify newdata: newx = x[test,]
ridge.pred <- predict(ridge.mod, s = 4, newx = x[test,])
# MSE
mse.1 <- mean((ridge.pred - y.test)^2)
# MSE for model of only the intercept and no var.: horizontal line y = ybar
mse.2 <- mean((mean(y[train]) - y.test)^2)
# lambda = 10^10 aka very large
ridge.pred <- predict(ridge.mod, s = 1e10, newx = x[test,])
```

```r
# MSE
mse.3 <- mean((ridge.pred - y.test)^2)
# lambda = 0 <=> Least Squares Regression <=> lm()
ridge.pred <- predict(ridge.mod, s = 0, newx = x[test,])
# MSE
mse.4 <- mean((ridge.pred - y.test)^2)

attr.names <- c('lambda = 4', 'Intercept only', 'lambda = 10^10', 'lambda = 0')
attr.values <- c(mse.1, mse.2, mse.3, mse.4)
data.frame(Models = attr.names, MSE = attr.values)
```

```
##           Models      MSE
## 1     lambda = 4 101036.8
## 2 Intercept only 193253.1
## 3 lambda = 10^10 193253.1
## 4     lambda = 0 114723.6
```

Cross-Validation

```r
par(mar=c(4,4,2,1))
# Cross-Validation: cv.glmnet()
set.seed(1)
cv.out <- cv.glmnet(x[train,], y[train], alpha = 0)
plot(cv.out)
```




```r
# Best lambda <=> lambda with min(MSE)
bestlam <- round(cv.out$lambda.min,2)
# predict() and test set MSE based on best lambda
ridge.pred <- predict(ridge.mod, s = bestlam, newx = x[test,])
mse <- round(mean((ridge.pred - y.test)^2),0)
```

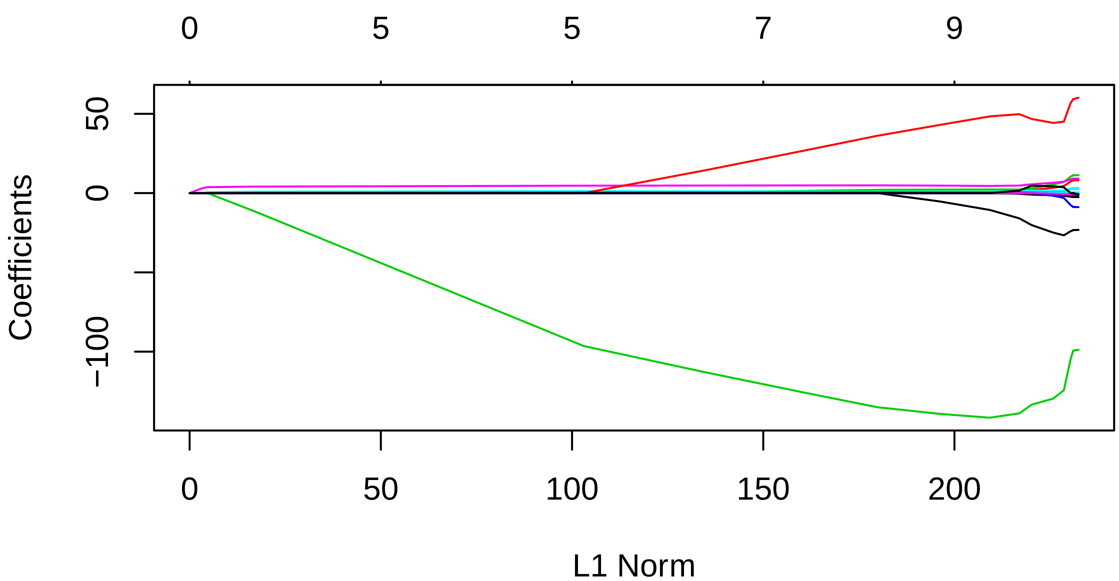
From Cross-Validation under **Ridge Regression**, λ with the lowest MSE on test set is 211.74, with $MSE = 9.6016 \times 10^4$.

```r
# Model on the entire dataset
out <- glmnet(x, y, alpha = 0)
predict(out, type = 'coefficients', s = bestlam)[1:20,]
```

```
##  (Intercept)        AtBat         Hits        HmRun         Runs
##   9.88487135   0.03143885   1.00883177   0.13926743   1.11320858
##          RBI        Walks        Years       CAtBat        CHits
##   0.87318972   1.80410571   0.13071797   0.01113977   0.06489859
##       CHmRun        CRuns         CRBI       CWalks      LeagueN
##   0.45158636   0.12900078   0.13737743   0.02908517  27.18236859
##    DivisionW      PutOuts      Assists       Errors   NewLeagueN
## -91.63431942   0.19149294   0.04254564  -1.81245301   7.21203276
```

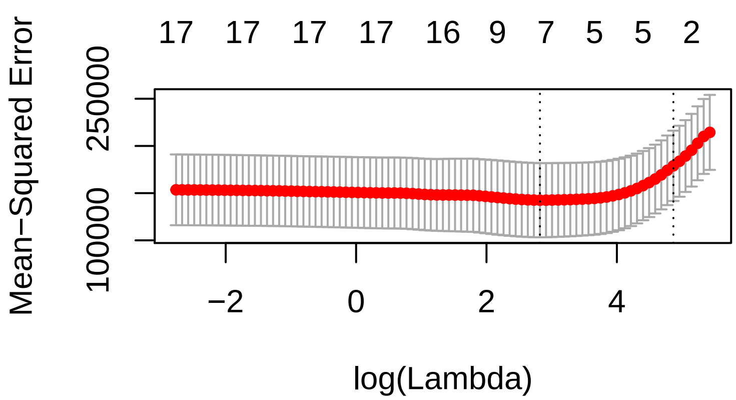
6.6.2. The Lasso

```r
par(mar=c(4,4,2,1))
# glmnet() with alpha = 1
lasso.mod <- glmnet(x[train,], y[train], alpha = 1, lambda = grid)
plot(lasso.mod)
```



Cross-Validation: `cv.glmnet()`

```r
# Cross-Validation: cv.glmnet()
par(mar=c(4,4,2,1))
set.seed(1)
cv.out <- cv.glmnet(x[train,], y[train], alpha = 1)
plot(cv.out)
```




```r
# Best lambda <=> lambda with min(MSE)
bestlam <- cv.out$lambda.min
# predict() and test set MSE based on best lambda
lasso.pred <- predict(lasso.mod, s = bestlam, newx = x[test,])
mse <- mean((lasso.pred - y.test)^2)
```

From Cross-Validation under **Lasso**, λ with the lowest MSE on test set is 16.7801585, with $MSE = 1.0074345 \times 10^5$.

```r
# Model on the entire dataset
out <- glmnet(x, y, alpha = 1, lambda = grid)
predict(out, type = 'coefficients', s = bestlam)[1:20,]
```

```
##  (Intercept)        AtBat         Hits        HmRun         Runs
##   18.5394844    0.0000000    1.8735390    0.0000000    0.0000000
##          RBI        Walks        Years       CAtBat        CHits
##    0.0000000    2.2178444    0.0000000    0.0000000    0.0000000
##       CHmRun        CRuns         CRBI       CWalks      LeagueN
##    0.0000000    0.2071252    0.4130132    0.0000000    3.2666677
##    DivisionW      PutOuts      Assists       Errors   NewLeagueN
## -103.4845458    0.2204284    0.0000000    0.0000000    0.0000000
```

Comment:

- Lasso and Ridge regression models' MSE are similar
- Lasso model: 12 vars have coef = 0 => Use fewer vars => Easier to interpret

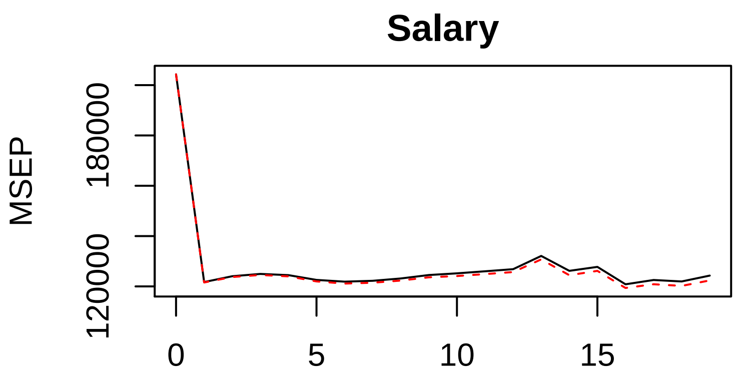
---

6.7. Lab 3: PCR and PLS Regression

6.7.1. Principal Components Regression

pcr() from pls package, > scale = TRUE to normalize data, > validation = 'CV' for 10-fold Cross Validation by default.

```
# `pcr()`` fn from `pls`` package
library(pls)
par(mar=c(2,4,2,1))
set.seed(2)
pcr.fit <- pcr(Salary ~ ., data = Hitters, scale = TRUE, validation = 'CV')
# summary(pcr.fit)
# Plot: `validationplot()`, to specify MSE: `val.type = 'MSEP'`
validationplot(pcr.fit, val.type = 'MSEP')
```



Comment:

- printed are RMSE, to get MSE = RMSE^2
- CV MSE is smallest when $M = 16$, not much different from $M = 19 \iff$ no reduction
- summary() shows Percentage of Variance Explained

```
par(mar=c(2,4,2,1))
# Model from train and test sets
set.seed(1)
pcr.fit <- pcr(Salary ~ ., data = Hitters, subset = train, scale = TRUE, validation = 'CV')
validationplot(pcr.fit, val.type = 'MSEP')
```

```r
# Prediction, based on M = 7
pcr.pred <- predict(pcr.fit, x[test,], ncomp = 7)
mean((pcr.pred - y.test)^2)
```

```
## [1] 96556.22
```

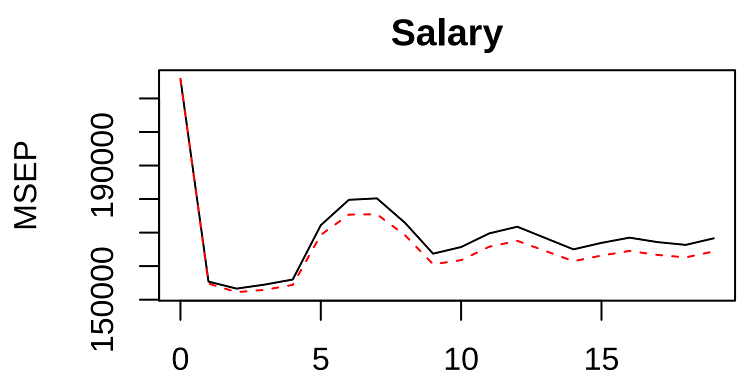
Comment: MSE is competitive vs. Ridge Regression and the Lasso, However, model from PCR is more difficult to interpret.

```r
# Fit model on full data set, using M = 7
pcr.fit <- pcr( y ~ x, scale = TRUE, ncomp = 7)
# summary(pcr.fit)
```

6.7.2. Partial Least Squares

`plsr()` from `pls` package

```r
par(mar=c(2,4,2,1))
set.seed(1)
pls.fit <- plsr(Salary ~ ., data = Hitters, subset = train, scale = TRUE, validation = 'CV')
#summary(pls.fit)
# Plot
validationplot(pls.fit, val.type = 'MSEP')
```




```r
pls.pred <- predict(pls.fit, x[test,], ncomp = 2)
mean((pls.pred - y.test)^2)
```

```
## [1] 101417.5
```

Comment: MSE is comparable but slightly higher than Ridge Regression, the Lasso, and PCR

```r
# Fit model on full data set, using M = 2
pls.fit <- plsr(Salary ~ ., data = Hitters, scale = TRUE, ncomp = 2)
summary(pls.fit)
```

```
## Data:    X dimension: 263 19
##  Y dimension: 263 1
## Fit method: kernelpls
## Number of components considered: 2
```

```
## TRAINING: % variance explained
##         1 comps  2 comps
## X         38.08    51.03
## Salary    43.05    46.40
```

Comment: PLSR model with 2 components explains 46.40% variance in Salary while PCR needs 7 components to explain \$46.69 %%

Reason: PCR only attemps to maximize variance explained in the predictors while PLSR searches for Directions that explain the variance in both predictors and response.