6 Linear Model Selection and Regularization

Andrew Liang

10/7/2020

Notes

Subset Selection

Best Subset

1. Fit M_0, the null model, with no predictors. (only predicts sample mean for each observation).
2. For $k = 1, 2, \ldots, p$:
   - Fit all $\binom{p}{k}$ models that contain exactly k predictors
   - Choose the best among the $\binom{p}{k}$ models and call it M_k. Best is defined as having smallest RSS, or equivalently largest R^2
3. Select single best model among $M_0, \ldots, M_p$ using CV prediction error, $C_p(AIC)$, BIC, or adjusted R^2

- Suffers from computational limitations, as the number of possible models grows rapidly as p increases (2^p models)

Forward Stepwise Selection

1. Fit M_0, the null model, with no predictors.
2. For $k = 0, \ldots, p-1$:
   - Consider all $p-k$ models that augment the predictors in M_k with one additional predictor
   - Choose best among $p-k$ models (M_{k+1})
3. Select single best model among $M_0, \ldots, M_p$ using CV prediction error, $C_p(AIC)$, BIC, or adjusted R^2

- Much less computationally expensive compared to best subset
- However, not guaranteed to find best subset model
- Can be applied in high-dimensional setting ($n < p$)

Backward Stepwise Selection

1. Fit M_p, the full model, with all predictors.
2. For $k = p, p-1, \ldots, 1$:
   - Consider all k models that contain all but one of the predictors in M_k, for a total of $k-1$ predictors
   - Choose best among k models (M_{k-1})
3. Select single best model among $M_0, \ldots, M_p$ using CV prediction error, $C_p(AIC)$, BIC, or adjusted R^2

- Also not guaranteed to find best model
- REQUIRES that n is larger than p

Best subset, forward, and backward selection generally give similar but not identical models

Choosing the Optimal Model

Techniques for adjusting the training error for the model size are available

1. C_p

   - for a fitted least squares model containing d predictors and the variance of the error $\hat{\sigma}^2$, C_p estiamte of test MSE is:
   $$C_p = \frac{1}{n}(RSS + 2d\hat{\sigma}^2)$$
   - penalty increases as number of predictors in model increases
   - choose model with lowest C_p value

2. AIC

   - defined for models fit by maximum likelihood (least squares)

   $$AIC = \frac{1}{n\hat{\sigma}^2}(RSS + 2d\hat{\sigma}^2)$$

   - proportional to C_p

3. BIC (similar to C_p and AIC, but from a Bayesian POV)

   $$BIC = \frac{1}{n\hat{\sigma}^2}(RSS + 2log(n)d\hat{\sigma}^2)$$

   - replaces $2d\hat{\sigma}^2$ with $log(n)d\hat{\sigma}^2$
   - since $log(n) > 2$ for any $n > 7$, BIC generally places heavier penalty on models with many predictors

4. Adjusted R^2

   $$AdjustedR^2 = 1 - \frac{RSS/(n-d-1)}{TSS/(n-1)}$$

   - unlike previous penalties, we want to choose model with highest adjusted R^2
   - despite popularity, is not as statistically motivated as the previous penalties

Shrinkage Methods

- fit model using all predictors and regularizes coefficients/shrinks coefficients towards zero
  - reduces variance

Ridge Regression

wants to minimize:
$$\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \lambda\sum_{j=1}^{p}\beta_j^2 = RSS + \lambda\sum_{j=1}^{p}\beta_j^2$$

- $\lambda\sum_{j=1}^{n}\beta_j^2$ is the shrinkage penalty
- $\lambda \geq 0$ is the tuning parameter
  - as $\lambda \to \infty$, the model coefficients approaches zero (except for model intercept β_0)
- selecting λ value is important (can use CV)
- best to apply ridge after predictors have been standardized (due to potential scaling issues):

$$\tilde{x_{ij}} = \frac{x_{ij}}{\sqrt{(\frac{1}{n}\sum_{i=1}^{n}(x_{ij} - \overline{x}_j)^2)}}$$

- important to note that all the predictors will still be included in the model; only the magnitude of the coefficients is affected

The Lasso

- similar to ridge, but has the ability to exclude predictors in final model (better for interpretability)

wants to minimize:
$$\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \lambda\sum_{j=1}^{p}|\beta_j| = RSS + \lambda\sum_{j=1}^{p}|\beta_j|$$

- λ penalty has the effect of forcing some of the coefficient estimates to be zero when λ is sufficiently large

Ridge vs Lasso

- generally, ridge performs better when response is a function of many predictors, with all coefficients roughly the same size
- generally, lasso performs better when only a relatively small number of predictors have substantial coefficients, and remaining variables are very small coefficients
- both perform shrinkage, whereas ridge shrinks the coefficients by the same proportion, whereas lasso shrinks all coefficients toward 0 by the same amount, and sufficiently small coefficients are shrunken all the way to 0

Dimension Reduction Methods

- idea is to transform the predictors then fit a least squares model let $Z_1, Z_2, \cdots, Z_M$ represent $M < p$ linear combinations of original p predictors:

$$Z_M = \sum_{j=1}^{p} \phi_{jm} X_j$$

for some constants $\phi_{1m}, \phi_{2m}, \cdots, \phi_{pm}$, then we fit the linear regression model:

$$y_i = \theta_0 + \sum_{m=1}^{M} \theta_m z_{im} + \epsilon_i$$

* dimension of the problem has been reduced from $p+1$ to $M+1$ * can often outperform least squares IF the choice of $Z_1, Z_2, \cdots, Z_M$ is chosen wisely

Principal Components Analysis (PCA)

- dimension reduction technique in which the *first principle component* direction of the data is that along which the observations *vary the most* (have highest variance)
  - is a vector that defines a line that minimizes perpendicular distances between each point and the line (distance represents the projection of the point onto that line)
- PCA scores for the 1st component is defined as:

$$Z_{j1} = \sum_{j=1}^{p} \beta_j (X_j - \overline{X}_j)$$

* can calculate up to p distinct pcincipal components * 2nd PC is a linear combination of variables that is uncorrelated with Z_1, or equivalently must be perpendicular/orthogonal to Z_1 * first component will always contain the most info

Principal Components Regression Approach (PCR)

- involved using $Z_1, Z_2, \cdots, Z_M$ as predictors in linear regression
- assume that the directions in which $X_1, \ldots, X_p$ *show the most variation are the directions that are associated with* Y
- will be better than the original linear model with $X_1, \ldots, X_p$ as predictors if PCR assumptions are met
- performs better when the first few principal components are sufficient to capture most of variation in the predictors and their relationships with the response
- since PCR is a lienar combination of all p of the *original* features, it is not a feature selection method
- number of components M usually chosen by CV
- usually recommended to standardize predictors using method from ridge if these predictors aren't on the same scale
- example of an *unsupervised* method

Partial Least Squares (PLS)

- a supervised method similar to PCA where it is dimension reduction
- same process as PCR, but also uses response Y to find directions that help explain both response and predictors
  - places highest weight on variables strongly correlated with Y
- often performs no better than PCR or ridge

Considerations in High Dimensional Data

- when $p \geq n$, linear regression/logistic regression should not be performed
- C_p, AIC, BIC unfortunately are not appropriate in high dimensional settings, as estimating $\hat{\sigma}^2$ is problematic
- 3 important points:

  1. regularization/shrinkage is very important in high-dimensional settings
  2. appropriate tuning parameter selection key for good predictive performance
  3. test error tends to increase as dimensionality increases, unless the additional predictors are truly associated with response

- adding new features is a truly a double-edged sword, depending whether or not they are truly associated with Y
- should *never* use sum of squared errors, p-values, R^2 statistics as evidence of model fit in high dimensional setting

Applied

Subset Selection

```r
library(ISLR)
names(Hitters)
```

```
##  [1] "AtBat"     "Hits"      "HmRun"     "Runs"      "RBI"       "Walks"    
##  [7] "Years"     "CAtBat"    "CHits"     "CHmRun"    "CRuns"     "CRBI"     
## [13] "CWalks"    "League"    "Division"  "PutOuts"   "Assists"   "Errors"   
## [19] "Salary"    "NewLeague"
```

```r
summary(Hitters)
```

```
##      AtBat            Hits         HmRun            Runs       
##  Min.   : 16.0   Min.   :  1   Min.   : 0.00   Min.   :  0.00  
##  1st Qu.:255.2   1st Qu.: 64   1st Qu.: 4.00   1st Qu.: 30.25  
##  Median :379.5   Median : 96   Median : 8.00   Median : 48.00  
##  Mean   :380.9   Mean   :101   Mean   :10.77   Mean   : 50.91  
##  3rd Qu.:512.0   3rd Qu.:137   3rd Qu.:16.00   3rd Qu.: 69.00  
##  Max.   :687.0   Max.   :238   Max.   :40.00   Max.   :130.00  
##                                                                
##       RBI             Walks            Years            CAtBat       
##  Min.   :  0.00   Min.   :  0.00   Min.   : 1.000   Min.   :   19.0  
##  1st Qu.: 28.00   1st Qu.: 22.00   1st Qu.: 4.000   1st Qu.:  816.8  
##  Median : 44.00   Median : 35.00   Median : 6.000   Median : 1928.0  
##  Mean   : 48.03   Mean   : 38.74   Mean   : 7.444   Mean   : 2648.7  
##  3rd Qu.: 64.75   3rd Qu.: 53.00   3rd Qu.:11.000   3rd Qu.: 3924.2  
##  Max.   :121.00   Max.   :105.00   Max.   :24.000   Max.   :14053.0  
##                                                                      
##      CHits            CHmRun           CRuns             CRBI        
##  Min.   :   4.0   Min.   :  0.00   Min.   :   1.0   Min.   :   0.00  
##  1st Qu.: 209.0   1st Qu.: 14.00   1st Qu.: 100.2   1st Qu.:  88.75  
##  Median : 508.0   Median : 37.50   Median : 247.0   Median : 220.50  
```

```
##  Mean   : 717.6   Mean   : 69.49   Mean   : 358.8   Mean   : 330.12
##  3rd Qu.:1059.2   3rd Qu.: 90.00   3rd Qu.: 526.2   3rd Qu.: 426.25
##  Max.   :4256.0   Max.   :548.00   Max.   :2165.0   Max.   :1659.00
##
##      CWalks        League  Division    PutOuts          Assists
##  Min.   :   0.00   A:175   E:157    Min.   :   0.0   Min.   :  0.0
##  1st Qu.:  67.25   N:147   W:165    1st Qu.: 109.2   1st Qu.:  7.0
##  Median : 170.50                    Median : 212.0   Median : 39.5
##  Mean   : 260.24                    Mean   : 288.9   Mean   :106.9
##  3rd Qu.: 339.25                    3rd Qu.: 325.0   3rd Qu.:166.0
##  Max.   :1566.00                    Max.   :1378.0   Max.   :492.0
##
##      Errors          Salary       NewLeague
##  Min.   : 0.00   Min.   :  67.5   A:176
##  1st Qu.: 3.00   1st Qu.: 190.0   N:146
##  Median : 6.00   Median : 425.0
##  Mean   : 8.04   Mean   : 535.9
##  3rd Qu.:11.00   3rd Qu.: 750.0
##  Max.   :32.00   Max.   :2460.0
##                  NA's   :59
```

```r
Hitters <- na.omit(Hitters) #omit na rows

library(leaps) #for subset selection

# best subset function
regfit.full <- regsubsets(Salary~., data = Hitters) #default up to 8 variables
summary(regfit.full)
```

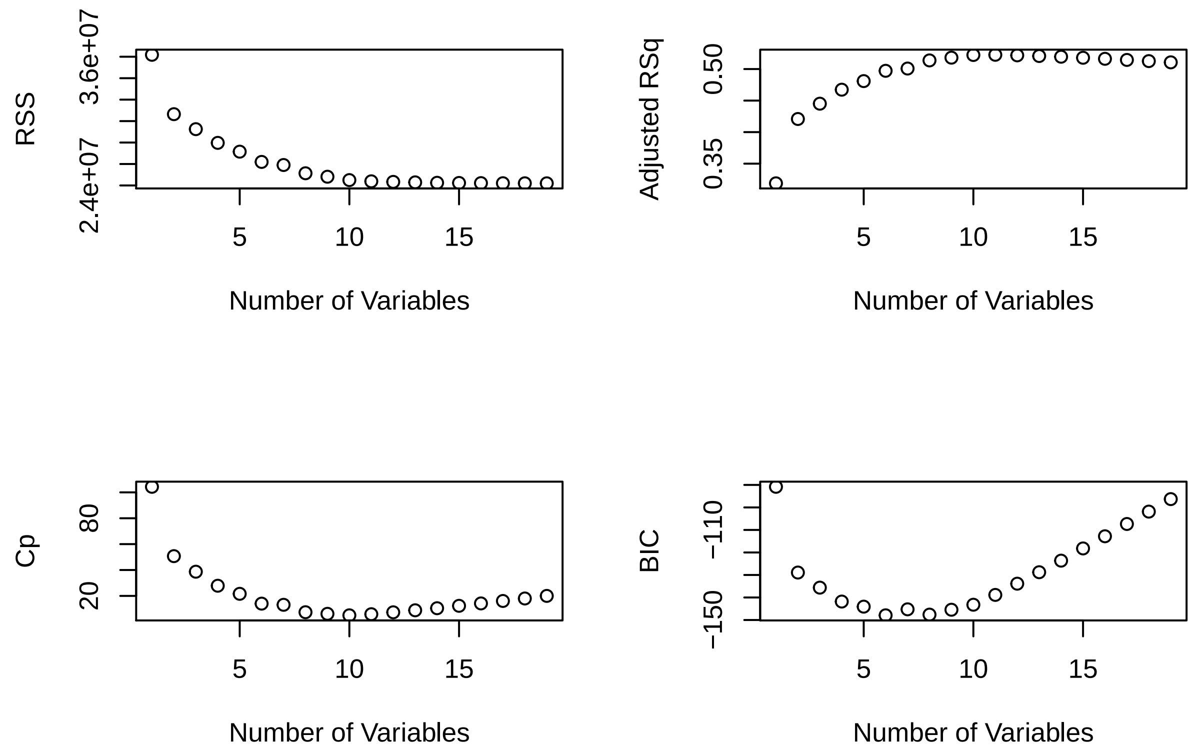
```
## Subset selection object
## Call: regsubsets.formula(Salary ~ ., data = Hitters)
## 19 Variables  (and intercept)
##            Forced in Forced out
## AtBat          FALSE      FALSE
## Hits           FALSE      FALSE
## HmRun          FALSE      FALSE
## Runs           FALSE      FALSE
## RBI            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 CRBI
## 1  ( 1 ) " "   " "  " "   " "  " " " "   " "   " "    " "   " "    " "   "*"
## 2  ( 1 ) " "   "*"  " "   " "  " " " "   " "   " "    " "   " "    " "   "*"
## 3  ( 1 ) " "   "*"  " "   " "  " " " "   " "   " "    " "   " "    " "   "*"
## 4  ( 1 ) " "   "*"  " "   " "  " " " "   " "   " "    " "   " "    " "   "*"
## 5  ( 1 ) "*"   "*"  " "   " "  " " " "   " "   " "    " "   " "    " "   "*"
## 6  ( 1 ) "*"   "*"  " "   " "  " " "*"   " "   " "    " "   " "    " "   "*"
## 7  ( 1 ) " "   "*"  " "   " "  " " "*"   " "   "*"    "*"   "*"    " "   " "
## 8  ( 1 ) "*"   "*"  " "   " "  " " "*"   " "   " "    " "   "*"    "*"   " "
##          CWalks LeagueN DivisionW PutOuts Assists Errors NewLeagueN
## 1  ( 1 ) " "    " "     " "       " "     " "     " "    " "
## 2  ( 1 ) " "    " "     " "       " "     " "     " "    " "
## 3  ( 1 ) " "    " "     " "       "*"     " "     " "    " "
## 4  ( 1 ) " "    " "     "*"       "*"     " "     " "    " "
## 5  ( 1 ) " "    " "     "*"       "*"     " "     " "    " "
## 6  ( 1 ) " "    " "     "*"       "*"     " "     " "    " "
## 7  ( 1 ) " "    " "     "*"       "*"     " "     " "    " "
## 8  ( 1 ) "*"    " "     "*"       "*"     " "     " "    " "
```

```r
regfit.full <- regsubsets(Salary~., data = Hitters, nvmax = 19) #set max # of variables to 19
reg.summary <- summary(regfit.full)
reg.summary$names #list of accuracy/penalty measurements
```

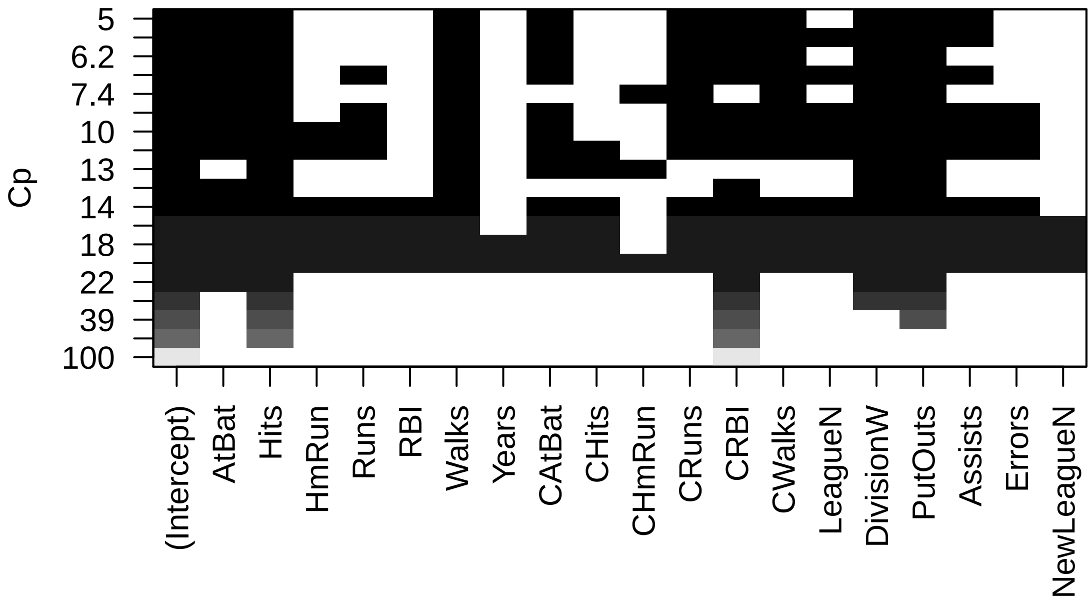
```
## NULL
```

```r
par(mfrow = c(2,2))
plot(reg.summary$rss, xlab = "Number of Variables", ylab = "RSS", type = "p")
plot(reg.summary$adjr2, xlab = "Number of Variables", ylab = "Adjusted RSq", type ="p")
plot(reg.summary$cp, xlab = "Number of Variables", ylab = "Cp", type ="p")
plot(reg.summary$bic, xlab = "Number of Variables", ylab = "BIC", type ="p")
```




```
#seems that the number of variables that best fit the model is around 10

par(mfrow = c(1,1))
plot(regfit.full, scale = "Cp") #shows Cp values for all combinations
```




```r
coef(regfit.full, 10) #coefficients for the 10 variables in model
```

```
##   (Intercept)        AtBat         Hits        Walks       CAtBat        CRuns
##   162.5354420   -2.1686501    6.9180175    5.7732246   -0.1300798    1.4082490
##          CRBI       CWalks    DivisionW      PutOuts      Assists
##     0.7743122   -0.8308264 -112.3800575    0.2973726    0.2831680
```

```r
#10 variables include AtBat, Hits, Walks, CAtBat, CRuns, CRBI, CWalks, DivisionW, PutOuts, Assists
```

```r
#Forward and Backward Selection

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

```
## Subset selection object
## Call: regsubsets.formula(Salary ~ ., data = Hitters, nvmax = 19, method = "forward")
## 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 19
## Selection Algorithm: forward
##           AtBat Hits HmRun Runs RBI Walks Years CAtBat CHits CHmRun CRuns CRBI
## 1  ( 1 )  " "   " "  " "   " "  " " " "   " "   " "    " "   " "    " "   "*"
## 2  ( 1 )  " "   "*"  " "   " "  " " " "   " "   " "    " "   " "    " "   "*"
## 3  ( 1 )  " "   "*"  " "   " "  " " " "   " "   " "    " "   " "    " "   "*"
## 4  ( 1 )  " "   "*"  " "   " "  " " " "   " "   " "    " "   " "    " "   "*"
## 5  ( 1 )  "*"   "*"  " "   " "  " " " "   " "   " "    " "   " "    " "   "*"
## 6  ( 1 )  "*"   "*"  " "   " "  " " "*"   " "   " "    " "   " "    " "   "*"
## 7  ( 1 )  "*"   "*"  " "   " "  " " "*"   " "   " "    " "   " "    " "   "*"
## 8  ( 1 )  "*"   "*"  " "   " "  " " "*"   " "   " "    " "   " "    "*"   "*"
## 9  ( 1 )  "*"   "*"  " "   " "  " " "*"   " "   "*"    " "   " "    "*"   "*"
## 10  ( 1 ) "*"   "*"  " "   " "  " " "*"   " "   "*"    " "   " "    "*"   "*"
## 11  ( 1 ) "*"   "*"  " "   " "  " " "*"   " "   "*"    " "   " "    "*"   "*"
## 12  ( 1 ) "*"   "*"  " "   "*"  " " "*"   " "   "*"    " "   " "    "*"   "*"
## 13  ( 1 ) "*"   "*"  " "   "*"  " " "*"   " "   "*"    " "   " "    "*"   "*"
## 14  ( 1 ) "*"   "*"  "*"   "*"  " " "*"   " "   "*"    " "   " "    "*"   "*"
## 15  ( 1 ) "*"   "*"  "*"   "*"  " " "*"   " "   "*"    "*"   " "    "*"   "*"
## 16  ( 1 ) "*"   "*"  "*"   "*"  "*" "*"   " "   "*"    "*"   " "    "*"   "*"
## 17  ( 1 ) "*"   "*"  "*"   "*"  "*" "*"   " "   "*"    "*"   " "    "*"   "*"
## 18  ( 1 ) "*"   "*"  "*"   "*"  "*" "*"   "*"   "*"    "*"   " "    "*"   "*"
## 19  ( 1 ) "*"   "*"  "*"   "*"  "*" "*"   "*"   "*"    "*"   "*"    "*"   "*"
##           CWalks LeagueN DivisionW PutOuts Assists Errors NewLeagueN
## 1  ( 1 )  " "    " "     " "       " "     " "     " "    " "
## 2  ( 1 )  " "    " "     " "       " "     " "     " "    " "
## 3  ( 1 )  " "    " "     " "       "*"     " "     " "    " "
## 4  ( 1 )  " "    " "     "*"       "*"     " "     " "    " "
## 5  ( 1 )  " "    " "     "*"       "*"     " "     " "    " "
## 6  ( 1 )  " "    " "     "*"       "*"     " "     " "    " "
## 7  ( 1 )  "*"    " "     "*"       "*"     " "     " "    " "
## 8  ( 1 )  "*"    " "     "*"       "*"     " "     " "    " "
## 9  ( 1 )  "*"    " "     "*"       "*"     " "     " "    " "
## 10  ( 1 ) "*"    " "     "*"       "*"     "*"     " "    " "
## 11  ( 1 ) "*"    "*"     "*"       "*"     "*"     " "    " "
## 12  ( 1 ) "*"    "*"     "*"       "*"     "*"     " "    " "
## 13  ( 1 ) "*"    "*"     "*"       "*"     "*"     "*"    " "
## 14  ( 1 ) "*"    "*"     "*"       "*"     "*"     "*"    " "
## 15  ( 1 ) "*"    "*"     "*"       "*"     "*"     "*"    " "
## 16  ( 1 ) "*"    "*"     "*"       "*"     "*"     "*"    " "
## 17  ( 1 ) "*"    "*"     "*"       "*"     "*"     "*"    "*"
## 18  ( 1 ) "*"    "*"     "*"       "*"     "*"     "*"    "*"
```

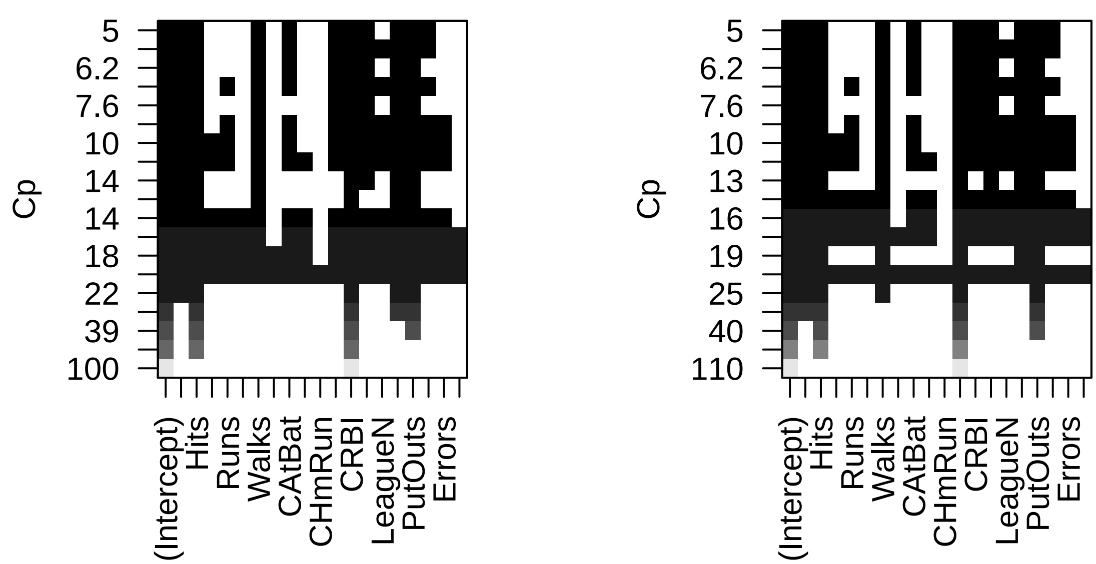
```
## 19  ( 1 ) "*"    "*"     "*"       "*"     "*"     "*"    "*"
```

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

```
## Subset selection object
## Call: regsubsets.formula(Salary ~ ., data = Hitters, nvmax = 19, method = "backward")
## 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 19
## Selection Algorithm: backward
##           AtBat Hits HmRun Runs RBI Walks Years CAtBat CHits CHmRun CRuns CRBI
## 1  ( 1 )  " "   " "  " "   " "  " " " "   " "   " "    " "   " "    "*"   " "
## 2  ( 1 )  " "   "*"  " "   " "  " " " "   " "   " "    " "   " "    "*"   " "
## 3  ( 1 )  " "   "*"  " "   " "  " " " "   " "   " "    " "   " "    "*"   " "
## 4  ( 1 )  "*"   "*"  " "   " "  " " " "   " "   " "    " "   " "    "*"   " "
## 5  ( 1 )  "*"   "*"  " "   " "  " " "*"   " "   " "    " "   " "    "*"   " "
## 6  ( 1 )  "*"   "*"  " "   " "  " " "*"   " "   " "    " "   " "    "*"   " "
## 7  ( 1 )  "*"   "*"  " "   " "  " " "*"   " "   " "    " "   " "    "*"   " "
## 8  ( 1 )  "*"   "*"  " "   " "  " " "*"   " "   " "    " "   " "    "*"   "*"
## 9  ( 1 )  "*"   "*"  " "   " "  " " "*"   " "   "*"    " "   " "    "*"   "*"
## 10  ( 1 ) "*"   "*"  " "   " "  " " "*"   " "   "*"    " "   " "    "*"   "*"
## 11  ( 1 ) "*"   "*"  " "   " "  " " "*"   " "   "*"    " "   " "    "*"   "*"
## 12  ( 1 ) "*"   "*"  " "   "*"  " " "*"   " "   "*"    " "   " "    "*"   "*"
## 13  ( 1 ) "*"   "*"  " "   "*"  " " "*"   " "   "*"    " "   " "    "*"   "*"
## 14  ( 1 ) "*"   "*"  "*"   "*"  " " "*"   " "   "*"    " "   " "    "*"   "*"
## 15  ( 1 ) "*"   "*"  "*"   "*"  " " "*"   " "   "*"    "*"   " "    "*"   "*"
## 16  ( 1 ) "*"   "*"  "*"   "*"  "*" "*"   " "   "*"    "*"   " "    "*"   "*"
## 17  ( 1 ) "*"   "*"  "*"   "*"  "*" "*"   " "   "*"    "*"   " "    "*"   "*"
## 18  ( 1 ) "*"   "*"  "*"   "*"  "*" "*"   "*"   "*"    "*"   " "    "*"   "*"
## 19  ( 1 ) "*"   "*"  "*"   "*"  "*" "*"   "*"   "*"    "*"   "*"    "*"   "*"
##           CWalks LeagueN DivisionW PutOuts Assists Errors NewLeagueN
## 1  ( 1 )  " "    " "     " "       " "     " "     " "    " "
## 2  ( 1 )  " "    " "     " "       " "     " "     " "    " "
## 3  ( 1 )  " "    " "     " "       "*"     " "     " "    " "
```

```
## 4  ( 1 ) " "    " "     " "       "*"     " "     " "    " "
## 5  ( 1 ) " "    " "     " "       "*"     " "     " "    " "
## 6  ( 1 ) " "    " "     "*"       "*"     " "     " "    " "
## 7  ( 1 ) "*"    " "     "*"       "*"     " "     " "    " "
## 8  ( 1 ) "*"    " "     "*"       "*"     " "     " "    " "
## 9  ( 1 ) "*"    " "     "*"       "*"     " "     " "    " "
## 10  ( 1 ) "*"    " "     "*"       "*"     "*"     " "    " "
## 11  ( 1 ) "*"    "*"     "*"       "*"     "*"     " "    " "
## 12  ( 1 ) "*"    "*"     "*"       "*"     "*"     " "    " "
## 13  ( 1 ) "*"    "*"     "*"       "*"     "*"     "*"    " "
## 14  ( 1 ) "*"    "*"     "*"       "*"     "*"     "*"    " "
## 15  ( 1 ) "*"    "*"     "*"       "*"     "*"     "*"    " "
## 16  ( 1 ) "*"    "*"     "*"       "*"     "*"     "*"    " "
## 17  ( 1 ) "*"    "*"     "*"       "*"     "*"     "*"    "*"
## 18  ( 1 ) "*"    "*"     "*"       "*"     "*"     "*"    "*"
## 19  ( 1 ) "*"    "*"     "*"       "*"     "*"     "*"    "*"
```

```r
#comparing forward and backward
par(mfrow = c(1,2))
plot(regfit.fwd, scale = "Cp")
plot(regfit.bwd, scale = "Cp")
```




```r
#in 7 variable model, selected variables are different
coef(regfit.full, 7)
```

```
##  (Intercept)         Hits        Walks       CAtBat        CHits       CHmRun
```

```
##   79.4509472    1.2833513    3.2274264   -0.3752350    1.4957073    1.4420538
##    DivisionW      PutOuts
## -129.9866432    0.2366813
```

```r
coef(regfit.fwd, 7)
```

```
##  (Intercept)        AtBat         Hits        Walks         CRBI       CWalks
##  109.7873062   -1.9588851    7.4498772    4.9131401    0.8537622   -0.3053070
##    DivisionW      PutOuts
## -127.1223928    0.2533404
```

```r
coef(regfit.bwd, 7)
```

```
##  (Intercept)        AtBat         Hits        Walks        CRuns       CWalks
##  105.6487488   -1.9762838    6.7574914    6.0558691    1.1293095   -0.7163346
##    DivisionW      PutOuts
## -116.1692169    0.3028847
```

```r
#choosing models using Validation Set and CV
set.seed(1)
train <- sample(c(TRUE,FALSE), nrow(Hitters), replace = TRUE)
test <- !train

#perform best subset on train
regfit.best <- regsubsets(Salary ~., data = Hitters[train,], nvmax = 19)

test.mat <- model.matrix(Salary ~ ., data = Hitters[test,])
val.errors <- rep(NA,19)

for(i in 1:19){
  coefi <- coef(regfit.best, id = i) #extract coefficients from regfit.best for each model of size i
  pred <- test.mat[,names(coefi)] %*% coefi #gives us the predicted value for each observation
  val.errors[i] <- mean((Hitters$Salary[test] - pred)^2) #MSE for each model of size i
}

val.errors
```

```
##  [1] 164377.3 144405.5 152175.7 145198.4 137902.1 139175.7 126849.0 136191.4
##  [9] 132889.6 135434.9 136963.3 140694.9 140690.9 141951.2 141508.2 142164.4
## [17] 141767.4 142339.6 142238.2
```

```r
which.min(val.errors) #7 variables gives us the lowest test MSE
```

```
## [1] 7
```

```r
coef(regfit.best,7)
```

```
##  (Intercept)        AtBat         Hits        Walks        CRuns       CWalks
##   67.1085369   -2.1462987    7.0149547    8.0716640    1.2425113   -0.8337844
##    DivisionW      PutOuts
## -118.4364998    0.2526925
```

```r
#function for predicting subset selection
predict.regsubsets = function(object, newdata, id, ...) {
  form = as.formula(object$call[[2]])
  mat = model.matrix(form, newdata)
  coefi = coef(object, id = id)
  mat[, names(coefi)] %*% coefi
}
```

```r
#Using k-fold validations

k <- 10
set.seed(1)
folds <- sample(1:k, nrow(Hitters),replace = T) #sample numbers 1 to 10 of length = dataset
table(folds)
```

```
## folds
##  1  2  3  4  5  6  7  8  9 10
## 28 22 24 24 30 26 25 24 31 29
```

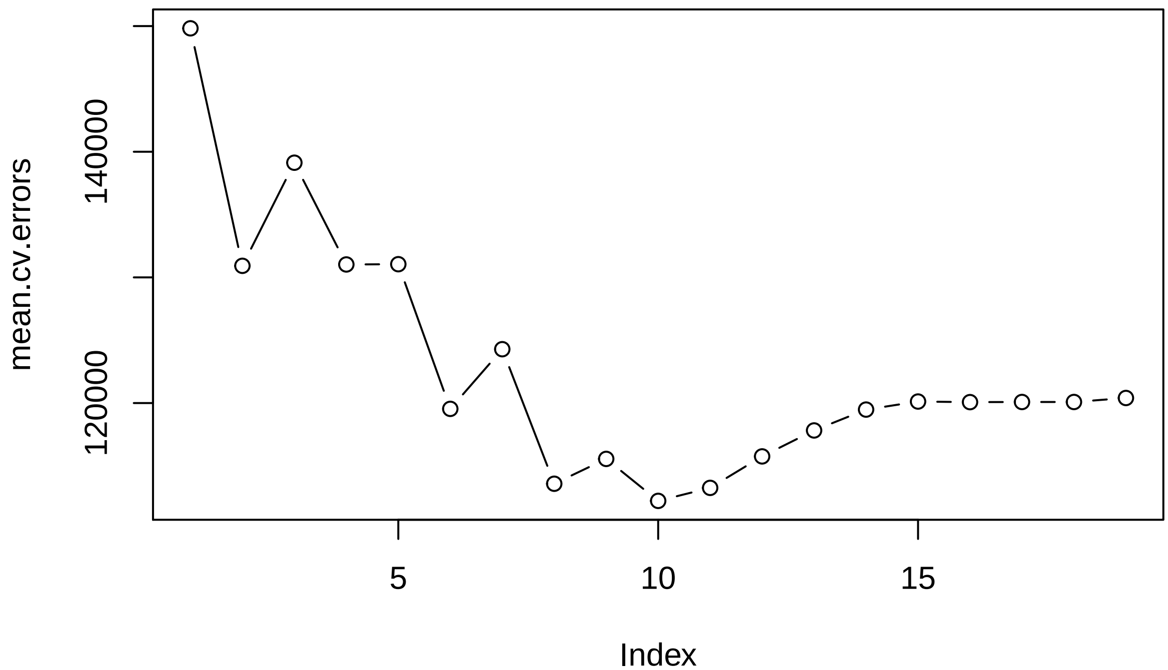
```r
cv.errors <- matrix(NA,k,19, dimnames = list(NULL, paste(1:19))) #matrix to store results

for(j in 1:k){
  best.fit <- regsubsets(Salary~., data = Hitters[folds != j,], nvmax = 19)
  for(i in 1:19){
    pred <- predict(best.fit, Hitters[folds == j,], id = i)
    #(i,j)th element corresponds to test MSE for ith CV for the best j-variable model
    cv.errors[j,i] <- mean((Hitters$Salary[folds == j] - pred)^2)
  }
}

mean.cv.errors <- apply(cv.errors,2,mean) #get a vector of the avg jth validation error for the jth mod
mean.cv.errors
```

```
##        1        2        3        4        5        6        7        8
## 149821.1 130922.0 139127.0 131028.8 131050.2 119538.6 124286.1 113580.0
##        9       10       11       12       13       14       15       16
## 115556.5 112216.7 113251.2 115755.9 117820.8 119481.2 120121.6 120074.3
##       17       18       19
## 120084.8 120085.8 120403.5
```

```r
par(mfrow = c(1,1))
plot(mean.cv.errors, type = "b") #selects a 10 variable model
```




```r
#perform best subset with 10 variables
reg.best <- regsubsets(Salary~., data = Hitters, nvmax = 19)
coef(reg.best,10)
```

```
##  (Intercept)        AtBat         Hits        Walks       CAtBat        CRuns
##  162.5354420   -2.1686501    6.9180175    5.7732246   -0.1300798    1.4082490
##         CRBI       CWalks    DivisionW      PutOuts      Assists
##    0.7743122   -0.8308264 -112.3800575    0.2973726    0.2831680
```

Ridge and Lasso

```r
#Ridge Regression

x <- model.matrix(Salary~., data = Hitters)[,-1] #create matrix of values for all predictors
#also transforms qualitative variables into dummy variables
y <- Hitters$Salary
```

```r
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.0-2
```

```r
grid <- 10^seq(10,-2, length = 100) #lambda values from 10^10 to 10^-2
ridge.mod <- glmnet(x,y,alpha = 0, lambda = grid) #alpha = 0 for ridge, 1 for lasso
ridge.mod$lambda[50]
```

```
## [1] 11497.57
```

```r
coef(ridge.mod)[,50] #Ridge coefficients for lambda = 11498
```

```
##   (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
ridge.mod$lambda[60]
```

```
## [1] 705.4802
```

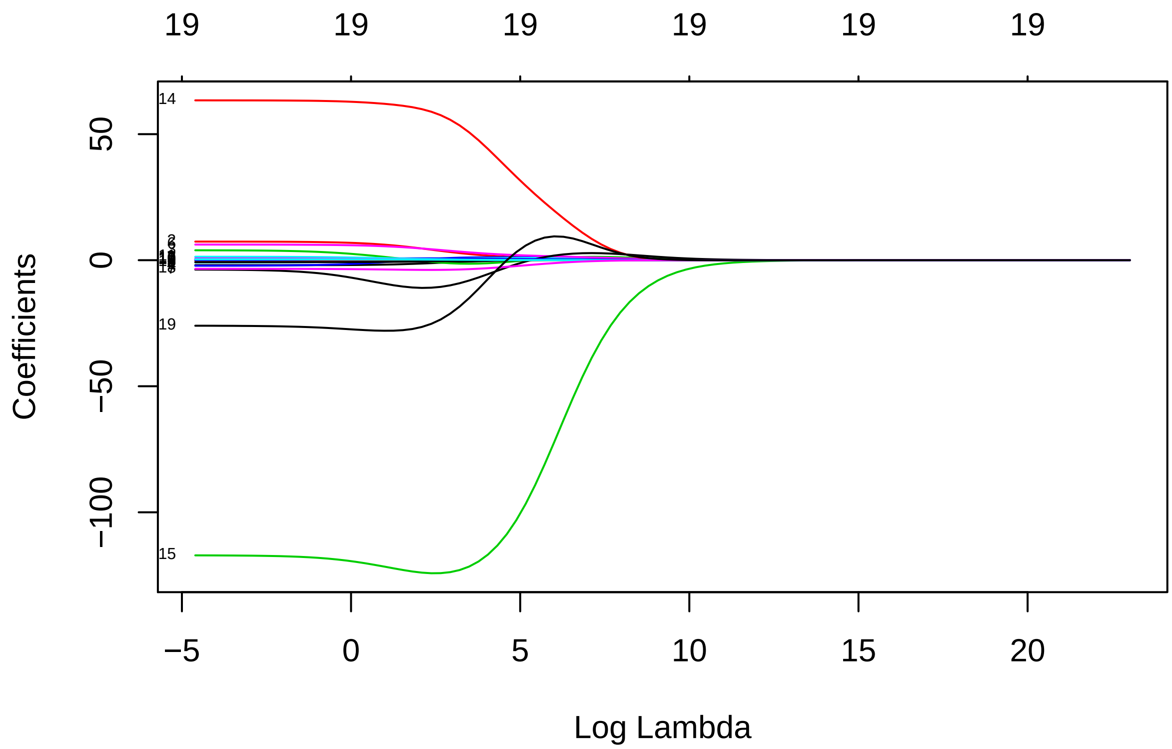
```r
coef(ridge.mod)[,60] #Ridge coefficients for lambda = 705
```

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

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

```
##   (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
plot(ridge.mod, xvar = "lambda", label = T) #plot of coefficients against lambda values
```




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

#fit ridge model on train
ridge.mod <- glmnet(x[train,],y[train], alpha = 0, lambda = grid, thresh = 1e-12)
ridge.pred <- predict(ridge.mod, s = 4, newx = x[test,])
mean((ridge.pred - y.test)^2) #evaluate test MSE with lambda = 4
```

```
## [1] 142199.2
```

MSE is 142199

```r
ridge.pred <- predict(ridge.mod, s=0, newx=x[test,]) #fitting ridge with lambda = 0
mean((ridge.pred - y.test)^2)
```

```
## [1] 167789.8
```

```r
lm(y~x, subset = train)
```

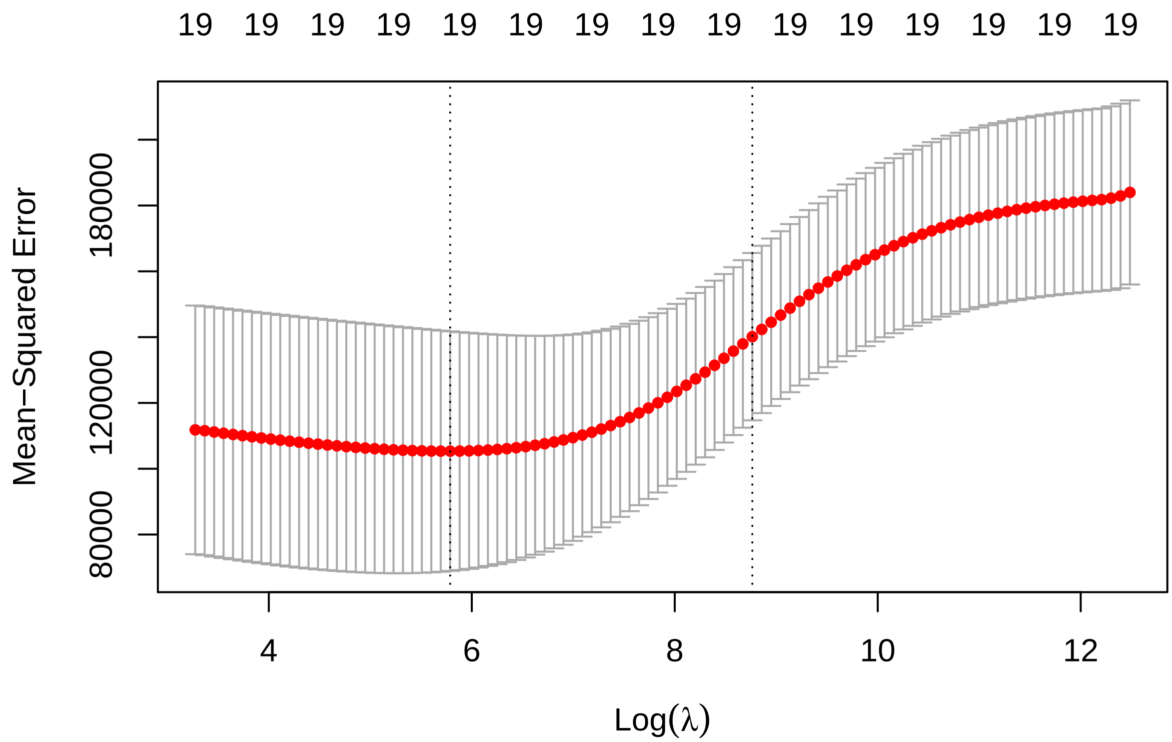
```
##
```

```
## Call:
## lm(formula = y ~ x, subset = train)
##
## Coefficients:
## (Intercept)       xAtBat        xHits       xHmRun        xRuns         xRBI
##    274.0145      -0.3521      -1.6377       5.8145       1.5424       1.1243
##      xWalks       xYears      xCAtBat       xCHits      xCHmRun       xCRuns
##      3.7287     -16.3773      -0.6412       3.1632       3.4008      -0.9739
##       xCRBI      xCWalks     xLeagueN   xDivisionW     xPutOuts     xAssists
##     -0.6005       0.3379     119.1486    -144.0831       0.1976       0.6804
##     xErrors  xNewLeagueN
##     -4.7128     -71.0951
```

```r
predict(ridge.mod,s=0, type="coefficients")[1:20,] #comparing ridge to original linear model
```

```
##  (Intercept)         AtBat          Hits         HmRun          Runs           RBI
##  274.2089049    -0.3699455    -1.5370022     5.9129307     1.4811980     1.0772844
##        Walks         Years        CAtBat         CHits        CHmRun         CRuns
##    3.7577989   -16.5600387    -0.6313336     3.1115575     3.3297885    -0.9496641
##         CRBI        CWalks       LeagueN     DivisionW       PutOuts       Assists
##   -0.5694414     0.3300136   118.4000592  -144.2867510     0.1971770     0.6775088
##       Errors    NewLeagueN
##   -4.6833775   -70.1616132
```

```r
set.seed(1)
cv.ridge <- cv.glmnet(x[train,],y[train], alpha = 0) #CV with default 10 folds
plot(cv.ridge)
```




```
bestlam <- cv.ridge$lambda.min
bestlam
```

```
## [1] 326.0828
```

Seems like the lambda with the lowest test MSE is 326

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

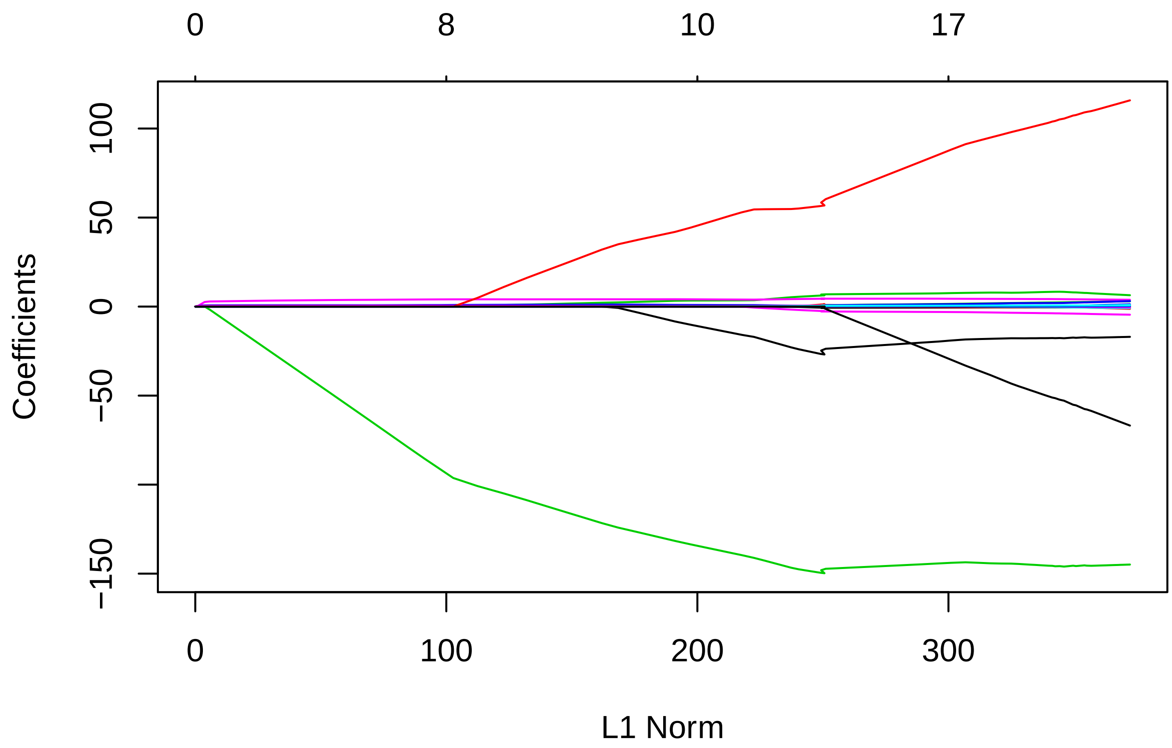
The test MSE is 139856

Refitting ridge regression model using lambda chosen by CV

```
out <- glmnet(x,y, alpha = 0)
predict(out, type = "coefficients", s = bestlam)[1:20]
```

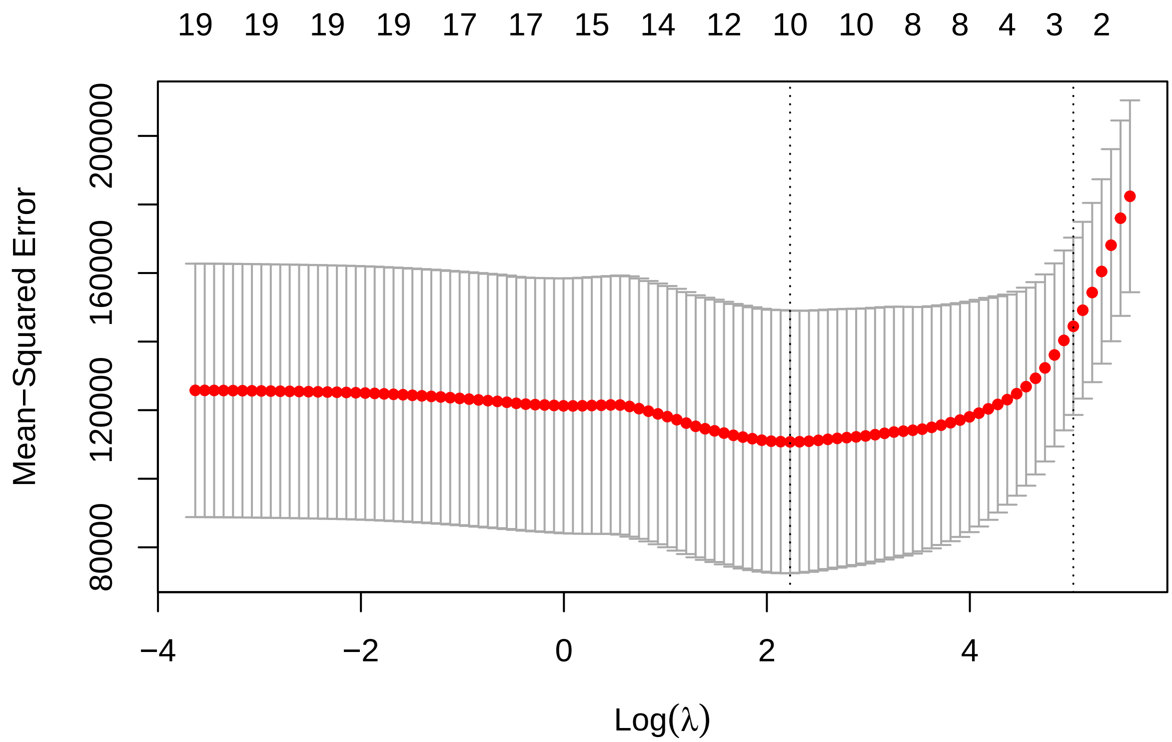
```
##  [1]  15.44383135   0.07715547   0.85911581   0.60103107   1.06369007
##  [6]   0.87936105   1.62444616   1.35254780   0.01134999   0.05746654
## [11]   0.40680157   0.11456224   0.12116504   0.05299202  22.09143189
## [16] -79.04032637   0.16619903   0.02941950  -1.36092945   9.12487767
```

```
#Lasso
lasso.mod <- glmnet(x[train,],y[train],alpha = 1)
plot(lasso.mod)
```



Same process as before, using alpha = 1.

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




```
bestlam <- cv.out$lambda.min
bestlam
```

```
## [1] 9.286955
```

Seems like the best lambda value is around 9

```
lasso.pred <- predict(lasso.mod, s = bestlam, newx = x[test,])
testMSElasso <- mean((lasso.pred - y.test)^2)
```

test MSE for CV lambda is 143668

```
out <- glmnet(x,y, alpha = 1,lambda = grid)
lasso.coef <- predict(out, type = "coefficients", s = bestlam)[1:20,]
lasso.coef
```

```
##   (Intercept)         AtBat          Hits         HmRun          Runs
##    1.27479059   -0.05497143    2.18034583    0.00000000    0.00000000
##           RBI         Walks         Years        CAtBat         CHits
##    0.00000000    2.29192406   -0.33806109    0.00000000    0.00000000
##        CHmRun         CRuns          CRBI        CWalks       LeagueN
##    0.02825013    0.21628385    0.41712537    0.00000000   20.28615023
##     DivisionW       PutOuts       Assists        Errors    NewLeagueN
## -116.16755870    0.23752385    0.00000000   -0.85629148    0.00000000
```

We can see that some variables have been shrunken down to 0. In this case, it seems that test MSE actually performed better with the Ridge than Lasso, but the Lasso model is notably more sparse, making it easier for interpretation

##PCR and PLS

```
library(pls)
```

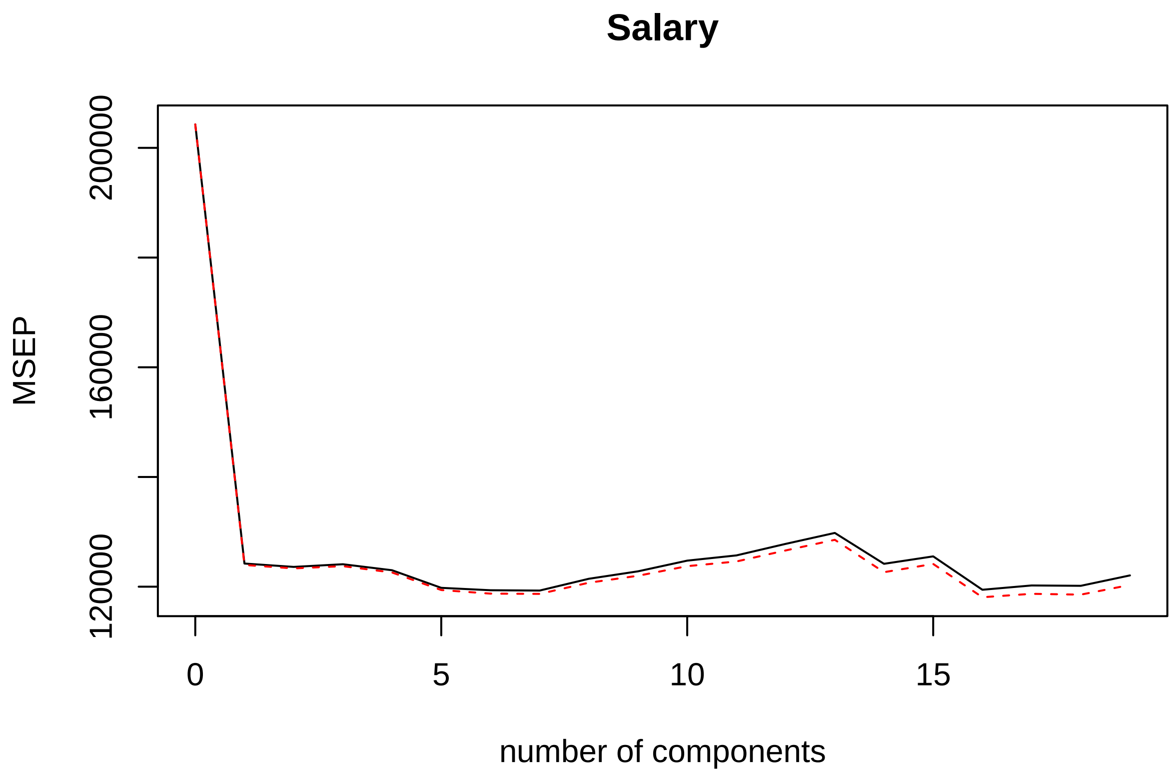
```
##
## Attaching package: 'pls'

## The following object is masked from 'package:stats':
##
##     loadings
```

```
set.seed(1)
pcr.fit <- pcr(Salary~., data = Hitters, scale = T, validation = "CV")
#scale standardizes each predictor, validation = CV computes 10-fold CV
summary(pcr.fit)
```

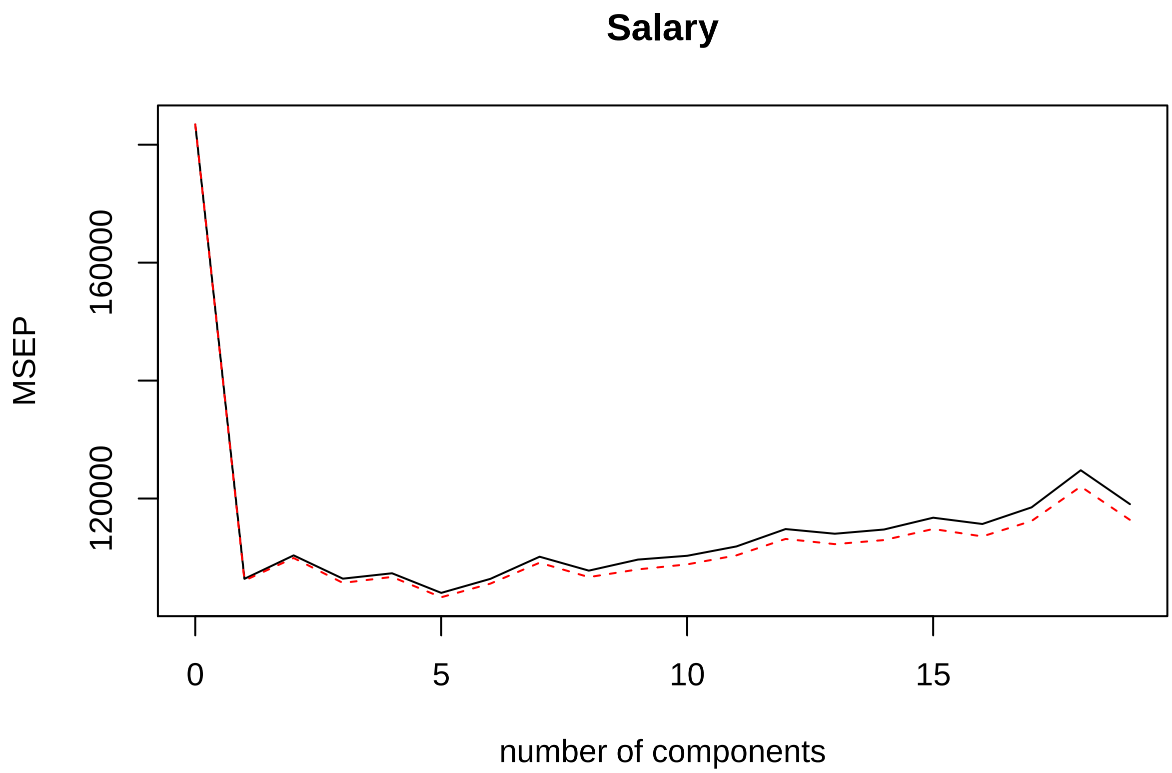
```
## Data:    X dimension: 263 19
##  Y dimension: 263 1
## Fit method: svdpc
## Number of components considered: 19
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV             452    352.5    351.6    352.3    350.7    346.1    345.5
## adjCV          452    352.1    351.2    351.8    350.1    345.5    344.6
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV       345.4    348.5    350.4     353.2     354.5     357.5     360.3
## adjCV    344.5    347.5    349.3     351.8     353.0     355.8     358.5
##        14 comps  15 comps  16 comps  17 comps  18 comps  19 comps
## CV        352.4     354.3     345.6     346.7     346.6     349.4
## adjCV     350.2     352.3     343.6     344.5     344.3     346.9
##
## TRAINING: % variance explained
##         1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X         38.31    60.16    70.84    79.03    84.29    88.63    92.26    94.96
## Salary    40.63    41.58    42.17    43.22    44.90    46.48    46.69    46.75
##         9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
## X         96.28     97.26     97.98     98.65     99.15     99.47     99.75
## Salary    46.86     47.76     47.82     47.85     48.10     50.40     50.55
##         16 comps  17 comps  18 comps  19 comps
## X          99.89     99.97     99.99    100.00
## Salary     53.01     53.85     54.61     54.61
```

```
#validation plot with CV MSE
validationplot(pcr.fit, val.type = "MSEP")
```



Seems that MSE is lowest at 16, but CV error is roughly the same at 1 PC score

```r
#train/test
set.seed(1)
pcr.fit <- pcr(Salary~., data = Hitters, subset = train, scale = T,
               validation = "CV")
validationplot(pcr.fit, val.type = "MSEP")
```



lowest CV error occurs when m = 5, so we now compute the test MSE

```
pcr.pred <- predict(pcr.fit, x[test,], ncomp = 5)
testMSEpcr <- mean((pcr.pred - y.test)^2)
```

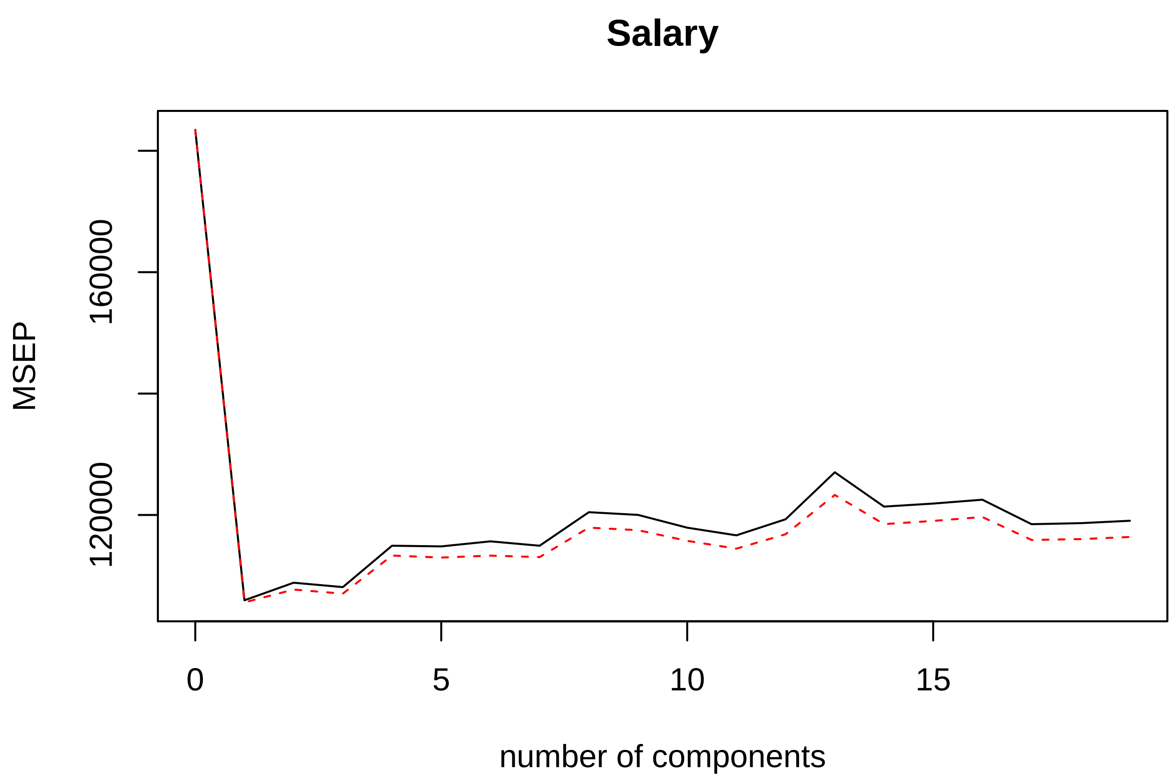
Seems like the test MSE is competitive with Lasso and Ridge

```
#PLS
set.seed(1)
pls.fit <- plsr(Salary~., data = Hitters, subset = train, scale = T,
                validation = "CV")
summary(pls.fit)
```

```
## Data:    X dimension: 131 19
##  Y dimension: 131 1
## Fit method: kernelpls
## Number of components considered: 19
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV           428.3    325.5    329.9    328.8    339.0    338.9    340.1
## adjCV        428.3    325.0    328.2    327.2    336.6    336.1    336.6
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV       339.0    347.1    346.4     343.4     341.5     345.4     356.4
```

```
## adjCV    336.2    343.4    342.8     340.2     338.3     341.8     351.1
##        14 comps  15 comps  16 comps  17 comps  18 comps  19 comps
## CV        348.4     349.1     350.0     344.2     344.5     345.0
## adjCV     344.2     345.0     345.9     340.4     340.6     341.1
##
## TRAINING: % variance explained
##          1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X         39.13    48.80    60.09    75.07    78.58    81.12    88.21    90.71
## Salary    46.36    50.72    52.23    53.03    54.07    54.77    55.05    55.66
##          9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
## X         93.17     96.05     97.08     97.61     97.97     98.70     99.12
## Salary    55.95     56.12     56.47     56.68     57.37     57.76     58.08
##          16 comps  17 comps  18 comps  19 comps
## X          99.61     99.70     99.95    100.00
## Salary     58.17     58.49     58.56     58.62
```

```r
validationplot(pls.fit, val.type = "MSEP")
```



Lowest CV error seems to be M = 1

```r
pls.pred <- predict(pls.fit,x[test,],ncomp = 1)
testMSEpls <- mean((pls.pred - y.test)^2)
```

Seems like the MSE is a little higher than the other methods

```r
pls.fit <- plsr(Salary~., data = Hitters, scale = T,
                ncomp = 1)
summary(pls.fit)
```

```
## Data:    X dimension: 263 19
##  Y dimension: 263 1
## Fit method: kernelpls
## Number of components considered: 1
## TRAINING: % variance explained
##         1 comps
## X         38.08
## Salary    43.05
```

Box graph of all the test MSEs using the different methods

```r
library(ggplot2)
alltestMSE <- c(testMSEridge, testMSElasso, testMSEpcr, testMSEpls)
barplot(alltestMSE,
        names.arg = c("Ridge,Lasso,PCR,PLS"),
        cex.names = 0.8,
        args.legend = alltestMSE,
        xlab = "Type of Regularization",
        ylab = "Test MSE")
```