Model Selection, Ridge and Lass Regression

Ryan Zhang

October 19, 2015

- Multiple Regression Model

$$Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p + \epsilon$$

- Despite the model is simple in form, there are couple of advantages of it
  1. Good interpretability, if multicolinearity is not that bad
  2. Often time good predictive power
- Solved via optimization

$$\min_{\beta} RSS = \Sigma(Y - \beta^T X)^2$$

Example Data

```
library(ISLR); Hitters=na.omit(Hitters); str(Hitters)
```

```
## 'data.frame':    263 obs. of  20 variables:
##  $ AtBat     : int  315 479 496 321 594 185 298 323 401 574 ...
##  $ Hits      : int  81 130 141 87 169 37 73 81 92 159 ...
##  $ HmRun     : int  7 18 20 10 4 1 0 6 17 21 ...
##  $ Runs      : int  24 66 65 39 74 23 24 26 49 107 ...
##  $ RBI       : int  38 72 78 42 51 8 24 32 66 75 ...
##  $ Walks     : int  39 76 37 30 35 21 7 8 65 59 ...
##  $ Years     : int  14 3 11 2 11 2 3 2 13 10 ...
##  $ CAtBat    : int  3449 1624 5628 396 4408 214 509 341 5206 4631 ...
##  $ CHits     : int  835 457 1575 101 1133 42 108 86 1332 1300 ...
##  $ CHmRun    : int  69 63 225 12 19 1 0 6 253 90 ...
##  $ CRuns     : int  321 224 828 48 501 30 41 32 784 702 ...
##  $ CRBI      : int  414 266 838 46 336 9 37 34 890 504 ...
##  $ CWalks    : int  375 263 354 33 194 24 12 8 866 488 ...
##  $ League    : Factor w/ 2 levels "A","N": 2 1 2 2 1 2 1 2 1 1 ...
##  $ Division  : Factor w/ 2 levels "E","W": 2 2 1 1 2 1 2 2 1 1 ...
##  $ PutOuts   : int  632 880 200 805 282 76 121 143 0 238 ...
##  $ Assists   : int  43 82 11 40 421 127 283 290 0 445 ...
##  $ Errors    : int  10 14 3 4 25 7 9 19 0 22 ...
##  $ Salary    : num  475 480 500 91.5 750 ...
##  $ NewLeague: Factor w/ 2 levels "A","N": 2 1 2 2 1 1 1 2 1 1 ...
##  - attr(*, "na.action")=Class 'omit'  Named int [1:59] 1 16 19 23 31 33 37 3
##   .. ..- attr(*, "names")= chr [1:59] "-Andy Allanson" "-Billy Beane" "-Bruc
```

Why Not Use Them All?

```
summary(lm(Salary~., data = Hitters))$coefficients
```

```
##                 Estimate  Std. Error    t value     Pr(>|t|)
## (Intercept)  163.1035878 90.77853560  1.7967197 0.0736219581
## AtBat         -1.9798729  0.63397803 -3.1229361 0.0020076676
## Hits           7.5007675  2.37753415  3.1548517 0.0018082442
## HmRun          4.3308829  6.20144741  0.6983665 0.4856158216
## Runs          -2.3762100  2.98075530 -0.7971839 0.4261224977
## RBI           -1.0449620  2.60087649 -0.4017730 0.6882042223
## Walks          6.2312863  1.82850381  3.4078607 0.0007662281
## Years         -3.4890543 12.41218587 -0.2810991 0.7788735944
## CAtBat        -0.1713405  0.13523684 -1.2669660 0.2063803985
## CHits          0.1339910  0.67455233  0.1986369 0.8427129019
## CHmRun        -0.1728611  1.61723721 -0.1068867 0.9149670939
## CRuns          1.4543049  0.75045769  1.9378906 0.0537950850
## CRBI           0.8077088  0.69261896  1.1661662 0.2446905326
## CWalks        -0.8115709  0.32808251 -2.4736793 0.0140574151
## LeagueN       62.5994230 79.26140140  0.7897845 0.4304236438
## DivisionW   -116.8492456 40.36695165 -2.8946760 0.0041407553
## PutOuts        0.2818925  0.07744057  3.6401141 0.0003329325
## Assists        0.3710692  0.22119878  1.6775373 0.0947231832
## Errors        -3.3607605  4.39163222 -0.7652646 0.4448566263
## NewLeagueN   -24.7623251 79.00262945 -0.3134367 0.7542177622
```

Why Not Use Them All?

```
summary(lm(Salary~., data = Hitters))$adj.r.squared
```

```
## [1] 0.510627
```

```
summary(lm(Salary~., data = Hitters))$r.squared
```

```
## [1] 0.5461159
```

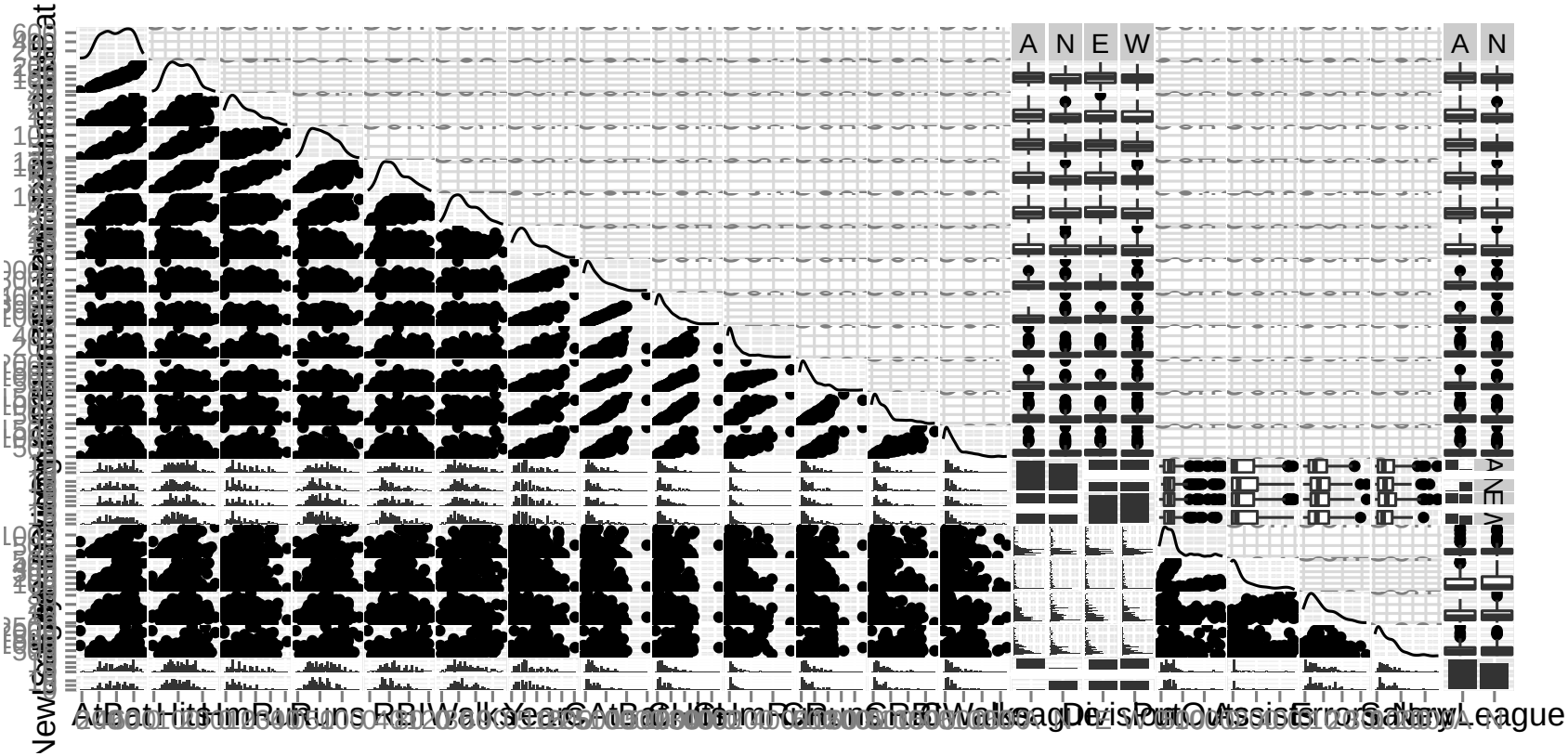
Why Not Use Them All?

1. Hard to interpret
2. Possible overfitting

Correlation Matrix

- Warning: Extremely Slow on Large Dataset. Don't try this

```
library(GGally); ggpairs(Hitters);
```



Best Single Variable Regression Model?

```
p <- ncol(Hitters)- 1; features <- names(Hitters)[names(Hitters)!="Salary"]
for (i in 1:p){formula = paste("Salary~",features[i], sep = "")
    print(paste("Model with",features[i],":",
                summary(lm(formula,Hitters))$adj.r.squared))}
```

```
## [1] "Model with AtBat : 0.152609785028896"
## [1] "Model with Hits : 0.189341408884815"
## [1] "Model with HmRun : 0.114287681032377"
## [1] "Model with Runs : 0.173125199128788"
## [1] "Model with RBI : 0.198954247362752"
## [1] "Model with Walks : 0.193941585811811"
## [1] "Model with Years : 0.157309651772855"
## [1] "Model with CAtBat : 0.274047553167146"
## [1] "Model with CHits : 0.298624699034669"
## [1] "Model with CHmRun : 0.272776429861305"
## [1] "Model with CRuns : 0.313987839436556"
## [1] "Model with CRBI : 0.3188502805785"
## [1] "Model with CWalks : 0.237013464878359"
## [1] "Model with League : -0.0036266653179247"
## [1] "Model with Division : 0.0333723755416829"
## [1] "Model with PutOuts : 0.0868029591562989"
## [1] "Model with Assists : -0.0031819417121548"
## [1] "Model with Errors : -0.0038021382946513 7"
## [1] "Model with NewLeague : -0.0038233526807756"
```

Best Single Variable Regression Model?

```
adjR <- vector()
for (i in 1:p){formula = paste("Salary~",features[i], sep = "")
    adjR <- c(adjR, summary(lm(formula,Hitters))$adj.r.squared)
    names(adjR)[i] <-features[i] }
adjR
which.max(adjR)
```

```
##        AtBat         Hits        HmRun         Runs          RBI
##  0.152609785  0.189341409  0.114287681  0.173125199  0.198954247
##        Walks        Years       CAtBat        CHits       CHmRun
##  0.193941586  0.157309652  0.274047553  0.298624699  0.272776430
##        CRuns         CRBI       CWalks       League     Division
##  0.313987839  0.318850281  0.237013465 -0.003626666  0.033372376
##      PutOuts      Assists       Errors    NewLeague
##  0.086802959 -0.003181942 -0.003802138 -0.003823353
## CRBI
##   12
```

Best Single Variable Regression Model?

```
summary(lm(Salary~CRBI , data = Hitters))
```

```
##
## Call:
## lm(formula = Salary ~ CRBI, data = Hitters)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1099.27  -203.45   -97.43   146.37  1847.22
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 274.58039   32.85537   8.357 3.85e-15 ***
## CRBI          0.79095    0.07113  11.120  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 372.3 on 261 degrees of freedom
## Multiple R-squared:  0.3215, Adjusted R-squared:  0.3189
## F-statistic: 123.6 on 1 and 261 DF,  p-value: < 2.2e-16
```

Best Two Variables Regression Model?

```
step2features <- features[features!="CRBI"]
for (i in 1:(p-1)){formula = paste("Salary~CRBI+",step2features[i], sep = "")
    print(paste("Model with CRBI and",step2features[i],":",
                summary(lm(formula,Hitters))$adj.r.squared))}
```

```
## [1] "Model with CRBI and AtBat : 0.393048154127622"
## [1] "Model with CRBI and Hits : 0.420802390670369"
## [1] "Model with CRBI and HmRun : 0.340262535446806"
## [1] "Model with CRBI and Runs : 0.4140631538282"
## [1] "Model with CRBI and RBI : 0.378754407657895"
## [1] "Model with CRBI and Walks : 0.395602362096423"
## [1] "Model with CRBI and Years : 0.347741967372335"
## [1] "Model with CRBI and CAtBat : 0.317974140011281"
## [1] "Model with CRBI and CHits : 0.317656784129349"
## [1] "Model with CRBI and CHmRun : 0.316240202289396"
## [1] "Model with CRBI and CRuns : 0.322931410535473"
## [1] "Model with CRBI and CWalks : 0.317212740125855"
## [1] "Model with CRBI and League : 0.316453949718148"
## [1] "Model with CRBI and Division : 0.34899966381821"
## [1] "Model with CRBI and PutOuts : 0.377975049237213"
## [1] "Model with CRBI and Assists : 0.32277000662758"
## [1] "Model with CRBI and Errors : 0.319904555609653"
## [1] "Model with CRBI and NewLeague : 0.316566469366743"
```

Best Two Variables Regression Model?

```
adjR <- vector()
for (i in 1:(p-1)){formula = paste("Salary~CRBI+",step2features[i], sep = "")
    adjR <- c(adjR, summary(lm(formula,Hitters))$adj.r.squared)
    names(adjR)[i] <-step2features[i] }
adjR
```

```
##     AtBat      Hits     HmRun      Runs       RBI     Walks     Years
## 0.3930482 0.4208024 0.3402625 0.4140632 0.3787544 0.3956024 0.3477420
##    CAtBat     CHits    CHmRun     CRuns    CWalks    League  Division
## 0.3179741 0.3176568 0.3162402 0.3229314 0.3172127 0.3164539 0.3489997
##   PutOuts   Assists    Errors NewLeague
## 0.3779750 0.3227700 0.3199046 0.3165665
```

```
which.max(adjR)
```

```
## Hits
##    2
```

Best Two Variables Regression Model?

```
for (i in 1:(p-1)){formula = paste("Salary~CRBI+",step2features[i], sep = "")
    print(round(summary(lm(formula,Hitters))$coefficients[,"Pr(>|t|)"],4))}
```

```
## (Intercept)        CRBI       AtBat
##       0.468       0.000       0.000
## (Intercept)        CRBI        Hits
##      0.3924      0.0000      0.0000
## (Intercept)        CRBI       HmRun
##      0.0000      0.0000      0.0023
## (Intercept)        CRBI        Runs
##      0.9478      0.0000      0.0000
## (Intercept)        CRBI         RBI
##      0.1039      0.0000      0.0000
## (Intercept)        CRBI       Walks
##      0.1725      0.0000      0.0000
## (Intercept)        CRBI       Years
##       0e+00       0e+00       5e-04
## (Intercept)        CRBI      CAtBat
##      0.0000      0.0000      0.4156
## (Intercept)        CRBI       CHits
##      0.0000      0.0043      0.4617
## (Intercept)        CRBI      CHmRun
##      0.0000      0.0000      0.9515
## (Intercept)        CRBI       CRuns
##      0.0000      0.0359      0.1099
```

Best Two Variables Regression Model?

```
pvals <- vector()
for (i in 1:(p-1)){formula = paste("Salary~CRBI+",step2features[i], sep = "")
    pvals <- c(pvals, summary(lm(formula,Hitters))$coefficients[,"Pr(>|t|)"][3]
pvals
```

```
##        AtBat         Hits        HmRun         Runs          RBI
## 2.681060e-08 5.275361e-11 2.310752e-03 2.450010e-10 6.085303e-07
##        Walks        Years       CAtBat        CHits       CHmRun
## 1.525909e-08 4.668818e-04 4.156463e-01 4.616584e-01 9.515485e-01
##        CRuns       CWalks      LeagueN    DivisionW      PutOuts
## 1.099006e-01 5.413451e-01 7.708606e-01 3.570835e-04 7.204345e-07
##      Assists       Errors   NewLeagueN
## 1.142958e-01 2.370372e-01 7.209896e-01
```

```
which.min(pvals)
```

```
## Hits
##    2
```

Best Two Variables Regression Model?

```
summary(lm(Salary~CRBI+Hits , data = Hitters))
```

```
##
## Call:
## lm(formula = Salary ~ CRBI + Hits, data = Hitters)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -942.47 -183.72  -37.85   96.55 2167.16
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -47.95590   55.98245  -0.857    0.392
## CRBI          0.68990    0.06723  10.262  < 2e-16 ***
## Hits          3.30084    0.48177   6.851 5.28e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 343.3 on 260 degrees of freedom
## Multiple R-squared:  0.4252, Adjusted R-squared:  0.4208
## F-statistic: 96.17 on 2 and 260 DF,  p-value: < 2.2e-16
```

Best Three Variables Regression Model?

```
step3features <- step2features[step2features!="Hits"]
for (i in 1:(p-2)){formula = paste("Salary~CRBI+Hits+",step3features[i], sep =
    print(paste("Model with CRBI, Hits and",step3features[i],":",
                summary(lm(formula,Hitters))$adj.r.squared))}
```

```
## [1] "Model with CRBI, Hits and AtBat : 0.434156864525245"
## [1] "Model with CRBI, Hits and HmRun : 0.418607334318472"
## [1] "Model with CRBI, Hits and Runs : 0.42073270214376"
## [1] "Model with CRBI, Hits and RBI : 0.418587670531636"
## [1] "Model with CRBI, Hits and Walks : 0.43339797127514"
## [1] "Model with CRBI, Hits and Years : 0.423402319314451"
## [1] "Model with CRBI, Hits and CAtBat : 0.420150613673043"
## [1] "Model with CRBI, Hits and CHits : 0.418648784758709"
## [1] "Model with CRBI, Hits and CHmRun : 0.418656992840453"
## [1] "Model with CRBI, Hits and CRuns : 0.421082083944684"
## [1] "Model with CRBI, Hits and CWalks : 0.419015137095381"
## [1] "Model with CRBI, Hits and League : 0.422529025913789"
## [1] "Model with CRBI, Hits and Division : 0.442836811949443"
## [1] "Model with CRBI, Hits and PutOuts : 0.445075316286119"
## [1] "Model with CRBI, Hits and Assists : 0.419415709357749"
## [1] "Model with CRBI, Hits and Errors : 0.420459081240245"
## [1] "Model with CRBI, Hits and NewLeague : 0.420813816519391"
```

Best Three Variables Regression model?

```
adjR <- vector()
for (i in 1:(p-2)){formula = paste("Salary~CRBI+Hits+",step3features[i], sep =
    adjR <- c(adjR, summary(lm(formula,Hitters))$adj.r.squared)
    names(adjR)[i] <-step3features[i] }
adjR
which.max(adjR)
```

```
##     AtBat     HmRun      Runs       RBI     Walks     Years    CAtBat
## 0.4341569 0.4186073 0.4207327 0.4185877 0.4333980 0.4234023 0.4201506
##     CHits    CHmRun     CRuns    CWalks    League  Division   PutOuts
## 0.4186488 0.4186570 0.4210821 0.4190151 0.4225290 0.4428368 0.4450753
##   Assists    Errors NewLeague
## 0.4194157 0.4204591 0.4208138
## PutOuts
##      14
```

This is Called Stepwise

- Tedious to do by hand already
- Criterials for entering a variable and possibly througing out a variable?
- Default criterial in SPSS is:
  1. new entering variable results a p value < 0.05
  2. if old variable p value > 0.1 then through it out
  3. if no enter no leaving variables, then stop.
- This criterial tend to favor interpretability

Criterial for choosing most predictive model

- If all we want is predictive power, we may ignore p values and focus on other types of measures
  1. adjusted $R^2 = 1 - \frac{n-1}{n-d-1}(1 - R^2)$:
     - The higher the better
  2. $C_p = \frac{1}{n}(RSS - 2d\hat{\sigma}^2)$
     - The lower the better
  3. $AIC = -2logL + 2d$ where L is the maximum likelihood
     - The lower the better
     - can be prove that $AIC \sim C_p$
  4. $BIC = \frac{1}{n}(RSS + log(n)d\hat{\sigma}^2)$
     - The lower the better
     - BIC tend to choose smaller model than AIC/Cp

Best Subset Method

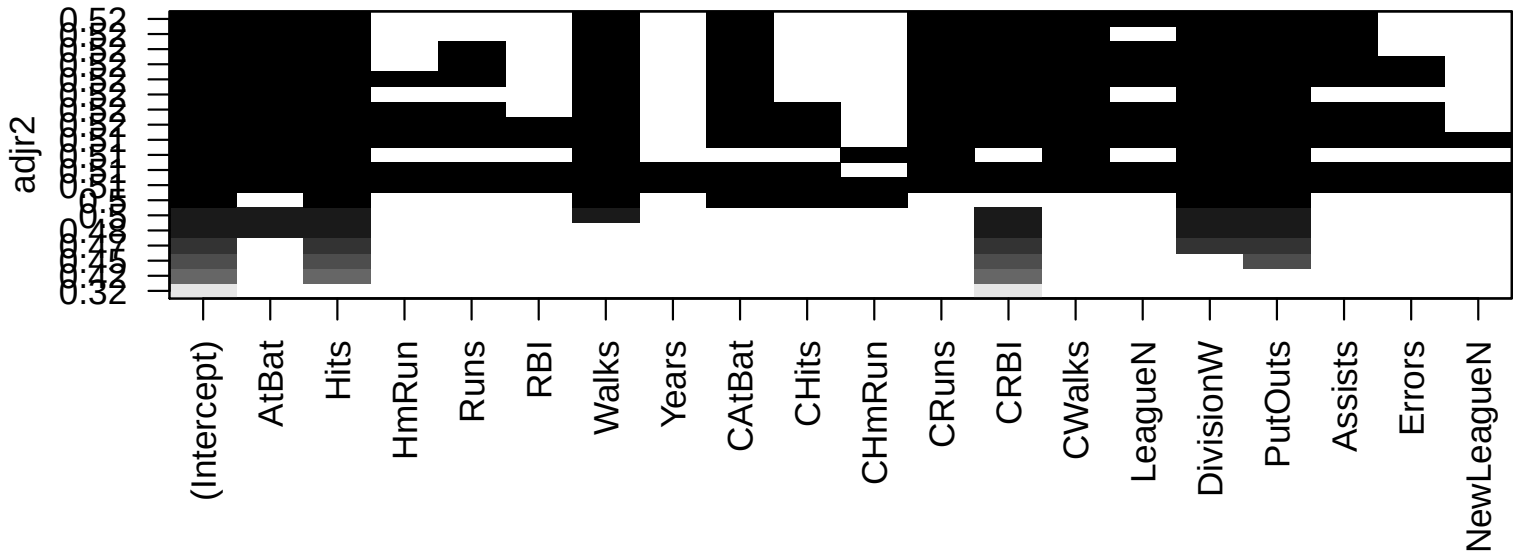
- Consider all possible models...
- 2^p is stupidly large number
- This is a case where we say we are doomed by dimensionality
- Stepwise method try at most $\frac{1}{2}p^2$ models
- huge reduction in the searching space

```
2^19
19^2
2^40
40^2
```

```
## [1] 524288
## [1] 361
## [1] 1.099512e+12
## [1] 1600
```

Best Subset Method

```
library(leaps)
bestSubsetRegression = regsubsets(Salary~.,nvmax=p,data=Hitters)
plot(bestSubsetRegression,scale="adjr2")
```



Best Subset Method

```
summary(bestSubsetRegression)$adjr2
```

```
##  [1] 0.3188503 0.4208024 0.4450753 0.4672734 0.4808971 0.4972001 0.5007849
##  [8] 0.5137083 0.5180572 0.5222606 0.5225706 0.5217245 0.5206736 0.5195431
## [15] 0.5178661 0.5162219 0.5144464 0.5126097 0.5106270
```

```
summary(bestSubsetRegression)$cp
```

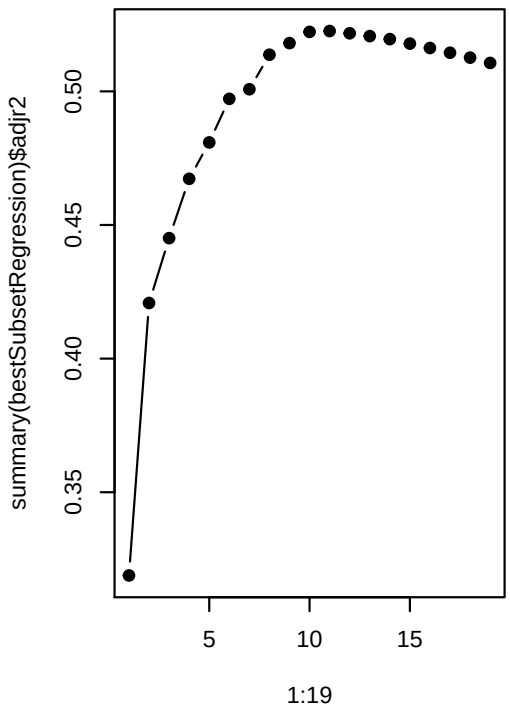
```
##  [1] 104.281319  50.723090  38.693127  27.856220  21.613011  14.023870
##  [7]  13.128474   7.400719   6.158685   5.009317   5.874113   7.330766
## [13]   8.888112  10.481576  12.346193  14.187546  16.087831  18.011425
## [19]  20.000000
```

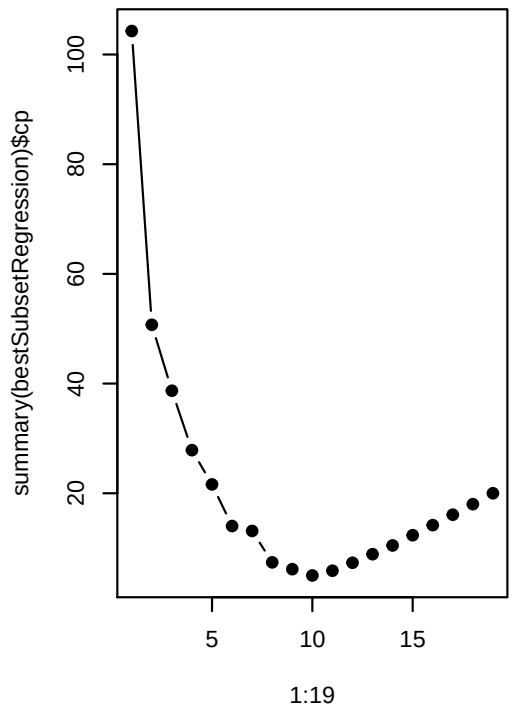
```
summary(bestSubsetRegression)$bic
```

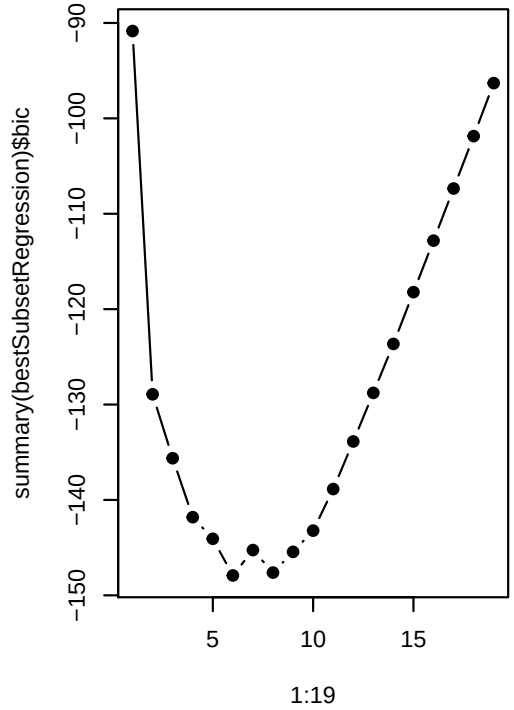
```
##  [1]  -90.84637 -128.92622 -135.62693 -141.80892 -144.07143 -147.91690
##  [7] -145.25594 -147.61525 -145.44316 -143.21651 -138.86077 -133.87283
## [13] -128.77759 -123.64420 -118.21832 -112.81768 -107.35339 -101.86391
## [19]  -96.30412
```

Best Subset Method

```
plot(1:19,summary(bestSubsetRegression)$adjr2, pch = 19, type="b")
plot(1:19,summary(bestSubsetRegression)$cp, pch = 19, type="b")
plot(1:19,summary(bestSubsetRegression)$bic, pch = 19, type="b")
```








```
par(mfrow = c(1,1))
```

Stepwise Method

- Ok, we know best subset method is impossible when p is large
- We instead search for models with restrictions
- Three types
  1. Forward: start with model with only intercept
  2. Backward: start with model with all variables
  3. Both:
- Trade off for this: We might not identify the "best" model given by best subset method

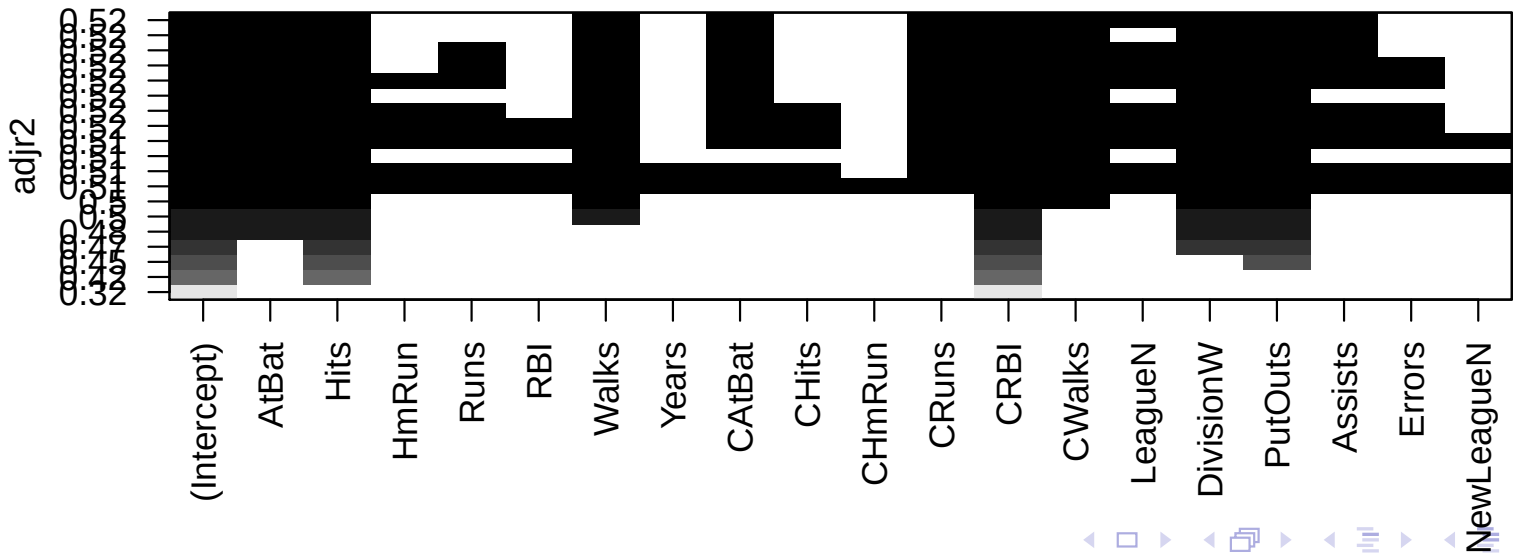
Forward Stepwise Method

- Try backward selection yourself
- the best 11 variables model has the highest adjusted R^2

```
forwardSelection =regsubsets(Salary~.,data=Hitters,nvmax=p,method="forward")
which.max(summary(forwardSelection)$adjr2)
```

```
## [1] 11
```

```
plot(forwardSelection,scale="adjr2")
```



Forward Selection

```
features[summary(forwardSelection)$which[11,2:(p+1)]]
```

```
##  [1] "AtBat"    "Hits"     "Walks"    "CAtBat"   "CRuns"    "CRBI"
##  [7] "CWalks"   "League"   "Division" "PutOuts"  "Assists"
```

```
model_11 <- lm(Salary~AtBat+Hits+Walks+CAtBat+CRuns+CRBI+CWalks+
               League+Division+PutOuts+Assists, data = Hitters)
coef(forwardSelection,11)
```

```
##  (Intercept)        AtBat         Hits        Walks       CAtBat
##  135.7512195   -2.1277482    6.9236994    5.6202755   -0.1389914
##        CRuns         CRBI       CWalks      LeagueN    DivisionW
##    1.4553310    0.7852528   -0.8228559   43.1116152 -111.1460252
##      PutOuts      Assists
##    0.2894087    0.2688277
```

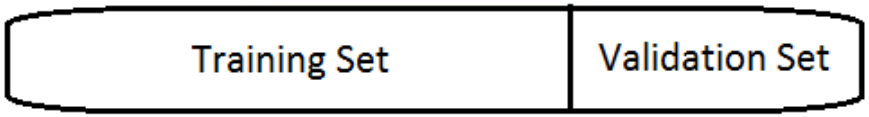
```
summary(model_11)$coefficients
```

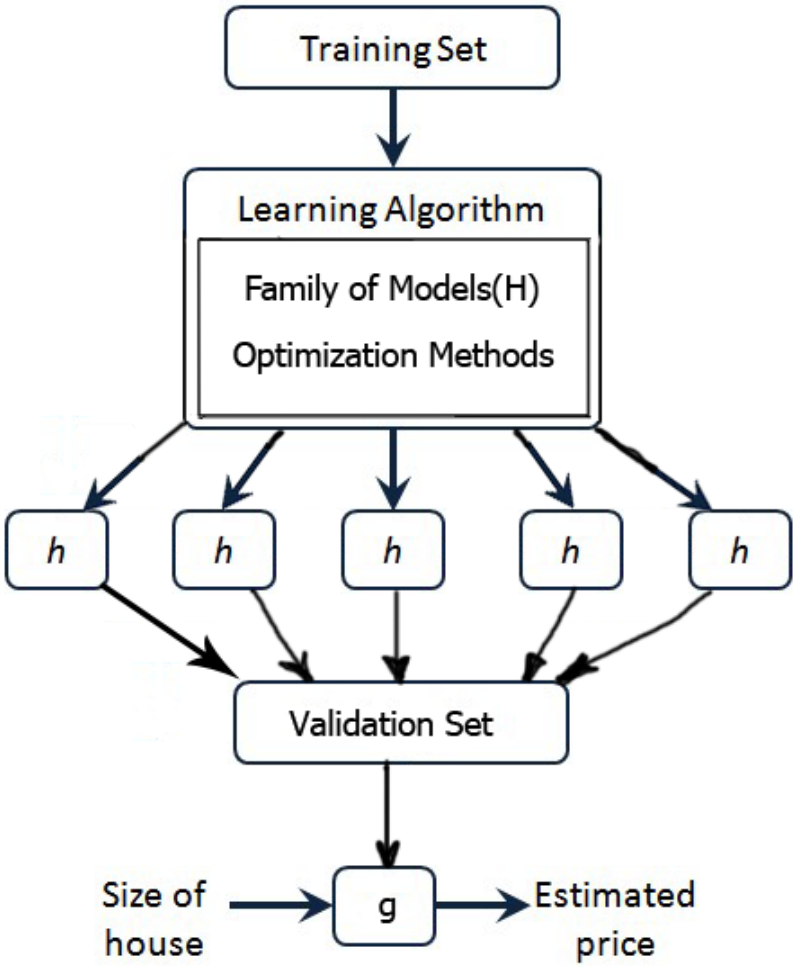
```
##                Estimate  Std. Error   t value     Pr(>|t|)
## (Intercept)  135.7512195 71.34622501  1.902711 5.822267e-02
## AtBat         -2.1277482  0.53746460 -3.958862 9.811456e-05
## Hits           6.9236994  1.64612222  4.206066 3.618766e-05
## Walks          5.6202755  1.59064429  3.533333 4.883226e-04
## CAtBat        -0.1389914  0.05608979 -2.478017 1.387004e-02
## CRuns          1.4553310  0.39270334  3.705930 2.591328e-04
```

Estimate Out of Sample Error

- adjusted R^2, C_p, BIC, AIC are measure of in sample error adjusted to be used to inference the out of sample error.
- Can we directly estimate the out of sample error?
- Simplist way is use a validation set

Revisit the Machine Learning Diagram





Creating a validation set

- Let's use a 70% split

```
library(caTools)
set.seed(0306)
split <- sample.split(Hitters$Salary, SplitRatio = 7/10)
trainingSet <- Hitters[split,]
validationSet <- Hitters[!split,]
```

Model Selection Using a Validation Set

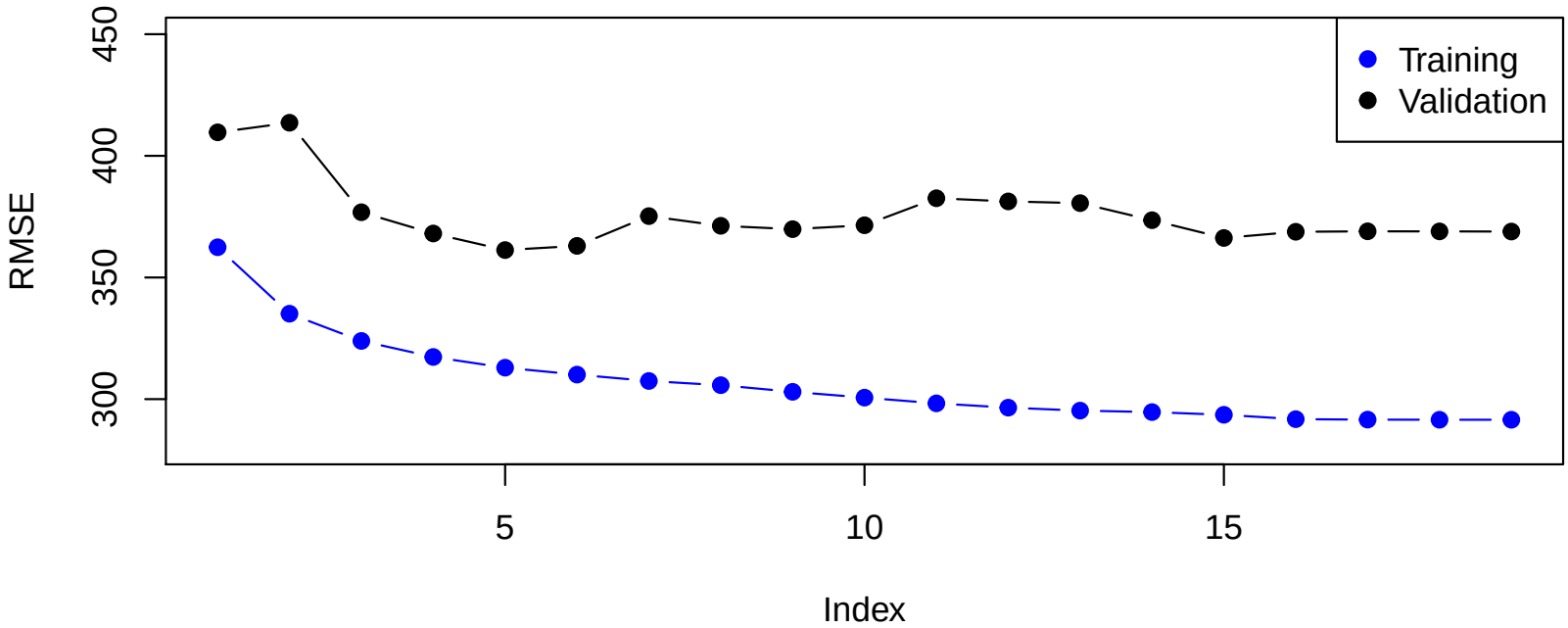
- We look at the RMSE on the validation set
- And compare with RMSE on the training set

```
trainingForward = regsubsets(Salary~.,data=trainingSet,
                             nvmax=p,method="forward")
validationErrors = rep(NA,19)
validationX = model.matrix(Salary~., data = validationSet)
for(i in 1:p){
  coefi=coef(trainingForward, id=i)
  pred=validationX[,names(coefi)]%*%coefi
  validationErrors[i] = sqrt(mean((validationSet$Salary-pred)^2))
}
```

Training Error V.S. Validation Error

- We look at the RMSE on the validation set
- And compare with RMSE on the training set

```
plot(validationErrors,ylab="RMSE",pch=19,type="b", ylim = c(280,450))
points(sqrt(trainingForward$rss[-1]/180), col = 'blue', pch = 19, type ="b")
legend("topright",legend=c("Training","Validation"),col=c("blue","black"),pch=1
```



Model Selection Via Validation

- This is Wrong!

```
which.min(validationErrors)
```

```
## [1] 5
```

```
coef(trainingForward, 5)
```

```
##  (Intercept)        AtBat         Hits         CRBI    DivisionW
##  149.1089360   -1.3564279    6.1951195    0.5913093 -123.9405534
##      PutOuts
##    0.3928374
```

- This is Correct

```
coef(forwardSelection,5)
```

```
##  (Intercept)        AtBat         Hits         CRBI    DivisionW
##   97.7684116   -1.4401428    7.1753197    0.6882079 -129.7319386
##      PutOuts
##    0.2905164
```

1. OLS Regression:
$$\min_{\beta} RSS$$
2. Ridge Regression:
$$\min_{\beta}(RSS + \lambda \Sigma_i {\beta_i}^2)$$
3. Lasso Regression:
$$\min_{\beta}(RSS + \lambda \Sigma_i |\beta_i|$$

- Don't select a variable in the model is equivalent to set the coefficient of that varaiable to be zero(or very samll).
- We want RSS to be small, we also don't want some coefficients to be small
- We add a shrinkage penalty to the cost(objective) function
- Just optimize with the two goals together...

Tuning Parameter

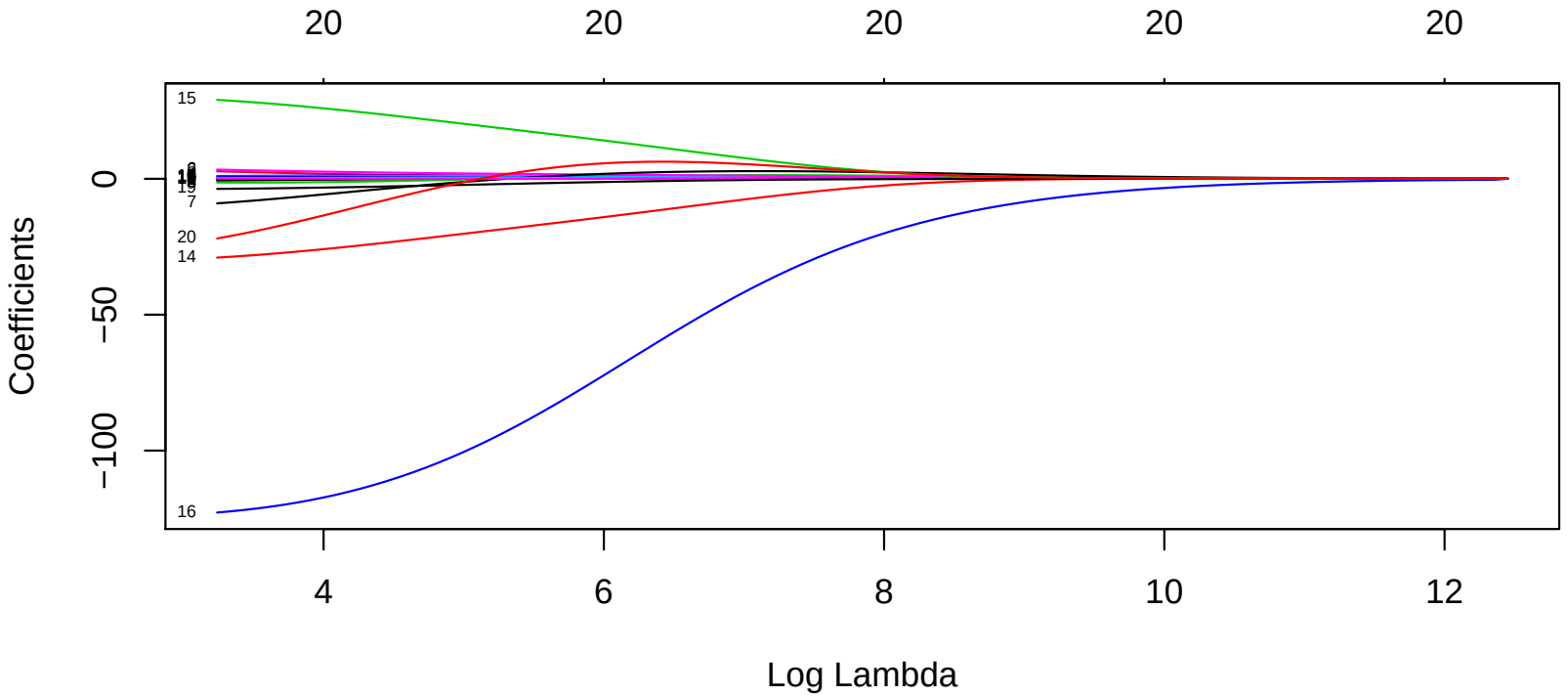
$$\min_{\beta}(RSS + \lambda \Sigma_i {\beta_i}^2)$$

$$\min_{\beta}(RSS + \lambda \Sigma_i |\beta_i|)$$

+ The λ is a tuning parameter
+ $\lambda = 0$ it is just OLS Regression
+ $\lambda = \infty$ it is just a horizontal line

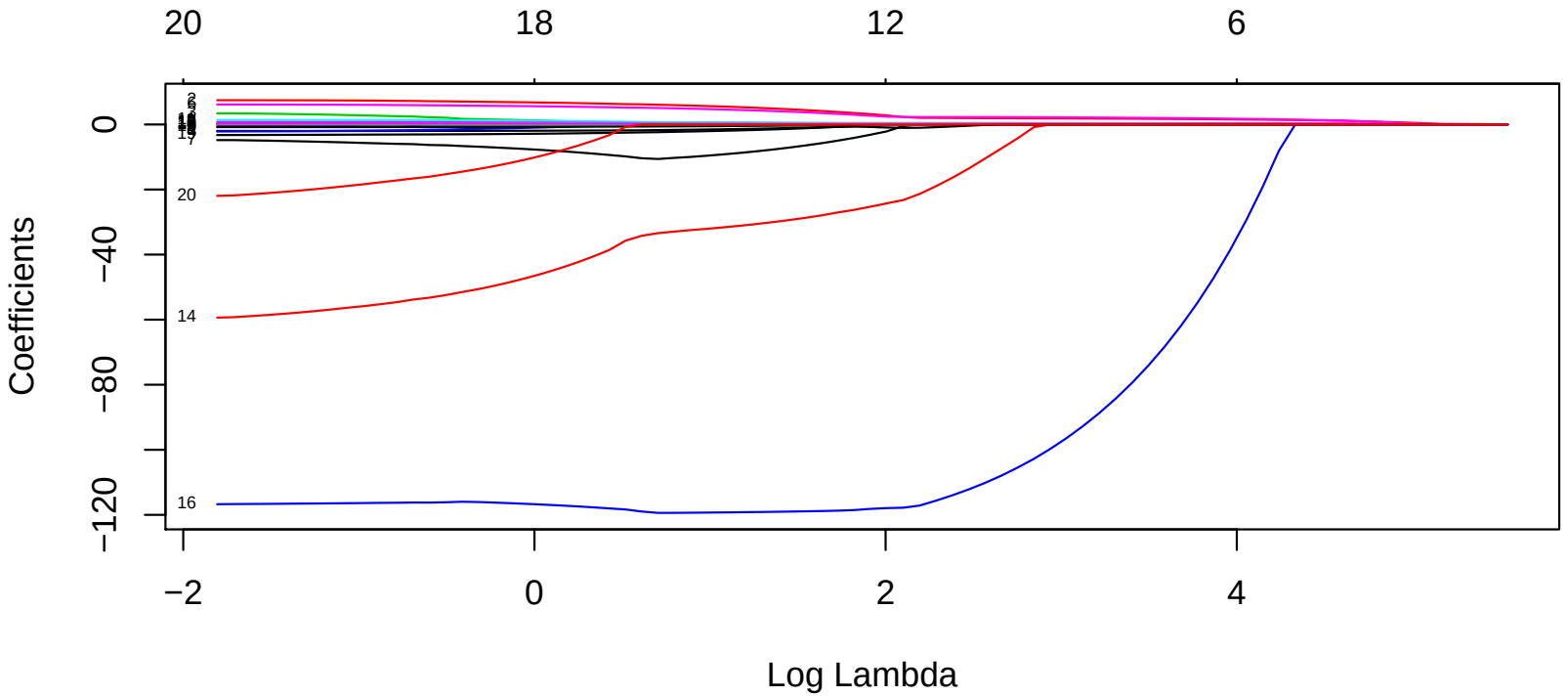
Ridge Example

```
library(glmnet);y=Hitters$Salary
X=model.matrix(Salary~.-1,data=Hitters)
ridgeRegression = glmnet(X,y,alpha=0, standardize = T)
plot(ridgeRegression,xvar="lambda",label=TRUE)
```



Lasso Example

```
lassoRegression = glmnet(X,y,alpha=1, standardize = T)
plot(lassoRegression,xvar="lambda",label=TRUE)
```



```
lassoRegression$lambda[1]
```

```
## [1] 255.2821
```

```
as.matrix(lassoRegression$beta)[,1]
```

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

```
##       AtBat        Hits       HmRun        Runs         RBI       Walks
##           0           0           0           0           0           0
##       Years      CAtBat       CHits      CHmRun       CRuns        CRBI
##           0           0           0           0           0           0
##      CWalks     LeagueA     LeagueN  DivisionW     PutOuts     Assists
##           0           0           0           0           0           0
##      Errors NewLeagueN
##           0           0
```

```
lassoRegression$lambda[2]
```

```
## [1] 232.6035
```

```
which(lassoRegression$beta[,2] > 0)
```

```
## CRBI
##   12
```

```
lassoRegression$beta[,2]
```

```
##       AtBat        Hits      HmRun        Runs         RBI       Walks
## 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
##       Years     CAtBat      CHits     CHmRun      CRuns        CRBI
## 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.07026614
##      CWalks    LeagueA    LeagueN  DivisionW    PutOuts    Assists
## 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
##      Errors NewLeagueN
## 0.00000000 0.00000000
```

Lasso Model Selection

```
lassoRegression$lambda[3]
```

```
## [1] 211.9397
```

```
which(lassoRegression$beta[,3] > 0)
```

```
## CRuns  CRBI
##    11    12
```

```
lassoRegression$beta[,3]
```

```
##       AtBat        Hits       HmRun        Runs         RBI       Walks
## 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
##       Years      CAtBat       CHits      CHmRun       CRuns        CRBI
## 0.00000000 0.00000000 0.00000000 0.00000000 0.01515244 0.11961370
##      CWalks    LeagueA    LeagueN  DivisionW    PutOuts    Assists
## 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
##     Errors NewLeagueN
## 0.00000000 0.00000000
```

Lasso Model Selection

```
lassoRegression$lambda[10]
```

```
## [1] 110.5055
```

```
which(lassoRegression$beta[,10] > 0)
```

```
##  Hits Walks CRuns  CRBI
##     2     6    11    12
```

```
lassoRegression$beta[,10]
```

```
##      AtBat       Hits      HmRun       Runs        RBI      Walks
##  0.0000000  1.0036044  0.0000000  0.0000000  0.0000000  0.9857987
##      Years     CAtBat      CHits     CHmRun      CRuns       CRBI
##  0.0000000  0.0000000  0.0000000  0.0000000  0.1094218  0.2911570
##     CWalks    LeagueA    LeagueN  DivisionW    PutOuts    Assists
##  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000
##     Errors NewLeagueN
##  0.0000000  0.0000000
```

```
lassoRegression$lambda[15]
```

```
## [1] 69.40069
```

```
which(lassoRegression$beta[,15] > 0)
```

```
##    Hits   Walks   CRuns    CRBI PutOuts
##       2       6      11      12      17
```

```
lassoRegression$beta[,15]
```

```
##       AtBat        Hits       HmRun        Runs         RBI       Walks
##  0.00000000  1.42342566  0.00000000  0.00000000  0.00000000  1.58214111
##       Years      CAtBat       CHits      CHmRun       CRuns        CRBI
##  0.00000000  0.00000000  0.00000000  0.00000000  0.16027975  0.33667715
##      CWalks     LeagueA     LeagueN   DivisionW     PutOuts     Assists
##  0.00000000  0.00000000  0.00000000 -8.06171262  0.08393604  0.00000000
##      Errors  NewLeagueN
##  0.00000000  0.00000000
```

How to Select λ ?

- We will talk about it next week...