Penalized Regression Methods

Lecture 7 – Mark Segal – Epidemiology and Biostatistics

Recap: multiple linear regression

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

Frequently used to describe relationship between continuous outcome *Y* and multiple predictors X_j

Coefficients β_j are estimated by the least squares criterion: minimize the residual sum-of-squares:

$$\text{RSS} = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2$$

Resulting LS estimates have minimum variance amongst the class of all unbiased estimates

Recap: multiple linear regression

Let $y \in \mathbb{R}^n$ be a fn of multiple predictors $X_1, \ldots, X_p \in \mathbb{R}^n$
Collect predictors into columns of matrix $X \in \mathbb{R}^{n \times p}$
Assume $X_1, \ldots, X_p$ are linearly independent: $\text{rank}(X) = p$
Model: $y = X\beta^* + \epsilon$
where $\beta^* = (\beta_1^*, \ldots, \beta_p^*) \in \mathbb{R}^p$ are the true coefficients and the errors $\epsilon = (\epsilon_1, \ldots, \epsilon_n) \in \mathbb{R}^n \sim (0, \sigma^2 I)$

Appealing to the (7 if) Gauss-Markov theorem estimate the coefficients $\hat{\beta}$ by least squares:

$$\hat{\beta} = \underset{\beta \in \mathbb{R}^p}{\text{argmin}} ||y - X\beta||_2^2$$

RSS

with solution

$$\hat{\beta} = (X^T X)^{-1} X^T y \qquad \left(\hat{\boldsymbol{\beta}}^{\text{OLS}}\right)$$

Recap: multiple linear regression

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

So, we have minimum variance unbiased estimates

End of story, right?

Wrong! What can go wrong?

Shortcomings (remedies)

- Non-linearity (smoothing, additive models)
- Irrelevant, noisy predictors (variable selection)
- Collinearity, interpretability (variable selection)
- Dependent, heterogeneous errors (GLS)
- Predictive performance, even when underlying model is linear (regularization)
- $p > n$ (dimension reduction, regularization)

Variable Selection Strategies

- Best subsets regression
  - try all models of all sizes (1,…p); pick best fit
  - computationally burdensome: 2^p candidates
- Forward stepwise regression
  - fit sequentially: enter best variable at each step
  - stop using thresholds or picking overall best fit
- Best fit determination requires estimating test error
  - modify training error to account for overfitting
  - use cross-validation or independent test data

```
> library(ISLR)
> fix(Hitters)
> names(Hitters)
 [1] "AtBat"     "Hits"      "HmRun"     "Runs"      "RBI"
 [6] "Walks"     "Years"     "CAtBat"    "CHits"     "CHmRun"
[11] "CRuns"     "CRBI"      "CWalks"    "League"    "Division"
[16] "PutOuts"   "Assists"   "Errors"    "Salary"    "NewLeague"
> dim(Hitters)
[1] 322  20
> sum(is.na(Hitters$Salary))
[1] 59
```

```
> Hitters=na.omit(Hitters)
> dim(Hitters)
[1] 263  20
> sum(is.na(Hitters))
[1] 0
```

There are many strategies for handling missing data with the pattern, extent, missingness mechanism and planned analysis method all being germane considerations.

```
> library(leaps)
> regfit.full=regsubsets(Salary~.,Hitters)
> summary(regfit.full)
Subset selection object
Call: regsubsets.formula(Salary ~ ., Hitters)
19 Variables  (and intercept)
...
1 subsets of each size up to 8
Selection Algorithm: exhaustive
          AtBat Hits HmRun Runs RBI Walks Years CAtBat CHits
1  ( 1 ) " "   " "  " "   " "  " " " "   " "   " "    " "
2  ( 1 ) " "   "*"  " "   " "  " " " "   " "   " "    " "
3  ( 1 ) " "   "*"  " "   " "  " " " "   " "   " "    " "
4  ( 1 ) " "   "*"  " "   " "  " " " "   " "   " "    " "
5  ( 1 ) "*"   "*"  " "   " "  " " " "   " "   " "    " "
6  ( 1 ) "*"   "*"  " "   " "  " " " "*" " "   " "    " "
7  ( 1 ) " "   "*"  " "   " "  " " " "*" " "   "*"    "*"
8  ( 1 ) "*"   "*"  " "   " "  " " " "*" " "   " "    " "
          CHmRun CRuns CRBI CWalks LeagueN DivisionW PutOuts
1  ( 1 ) " "    " "   "*"  " "   " "     " "       " "
2  ( 1 ) " "    " "   "*"  " "   " "     " "       " "
3  ( 1 ) " "    " "   "*"  " "   " "     " "       "*"
4  ( 1 ) " "    " "   "*"  " "   " "     "*"       "*"
5  ( 1 ) " "    " "   "*"  " "   " "     "*"       "*"
6  ( 1 ) " "    " "   "*"  " "   " "     "*"       "*"
7  ( 1 ) "*"    " "   " "  " "   " "     "*"       "*"
8  ( 1 ) "*"    "*"   " "  "*"   " "     "*"       "*"
```

```
> regfit.full=regsubsets(Salary~.,data=Hitters,nvmax=19)
> reg.summary=summary(regfit.full)
```

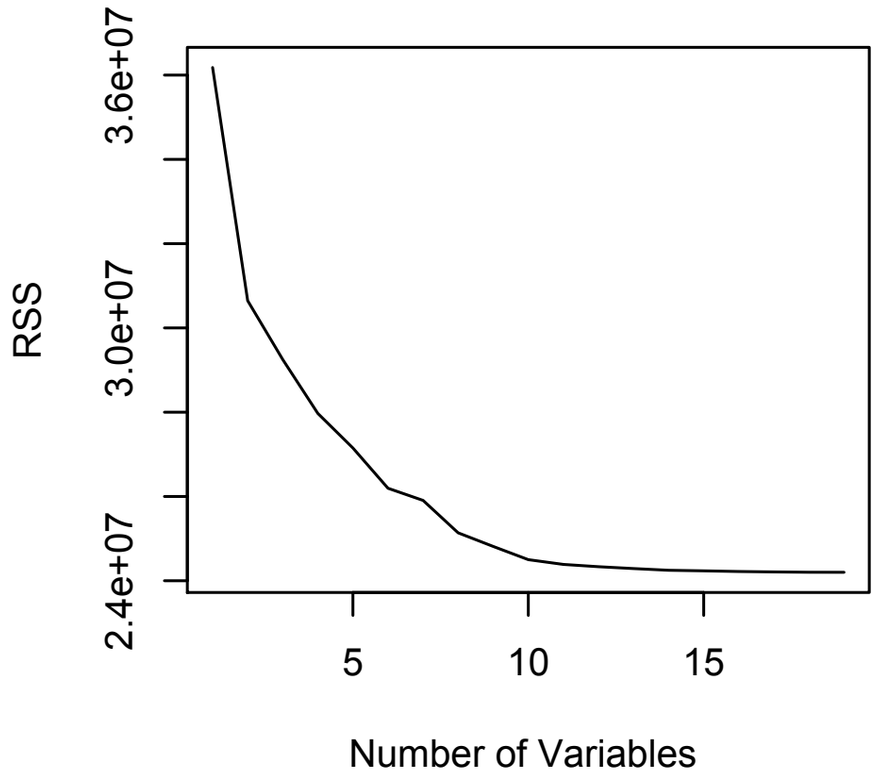
The `summary()` function also returns R^2, RSS, adjusted R^2, C_p, and BIC. We can examine these to try to select the *best* overall model.

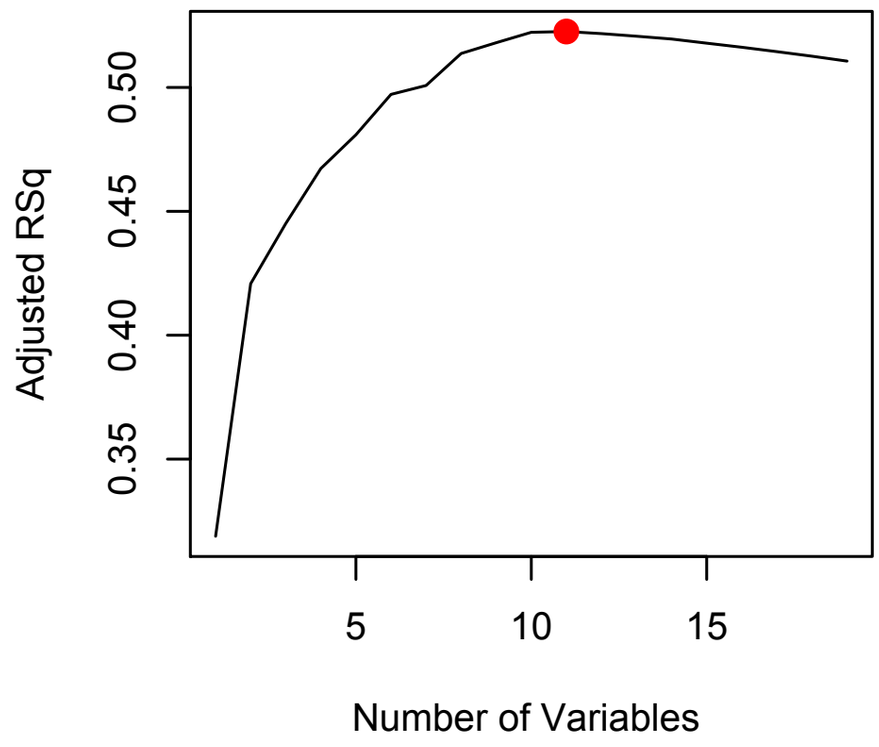
```
> names(reg.summary)
[1] "which"  "rsq"    "rss"    "adjr2"  "cp"     "bic"
[7] "outmat" "obj"
```

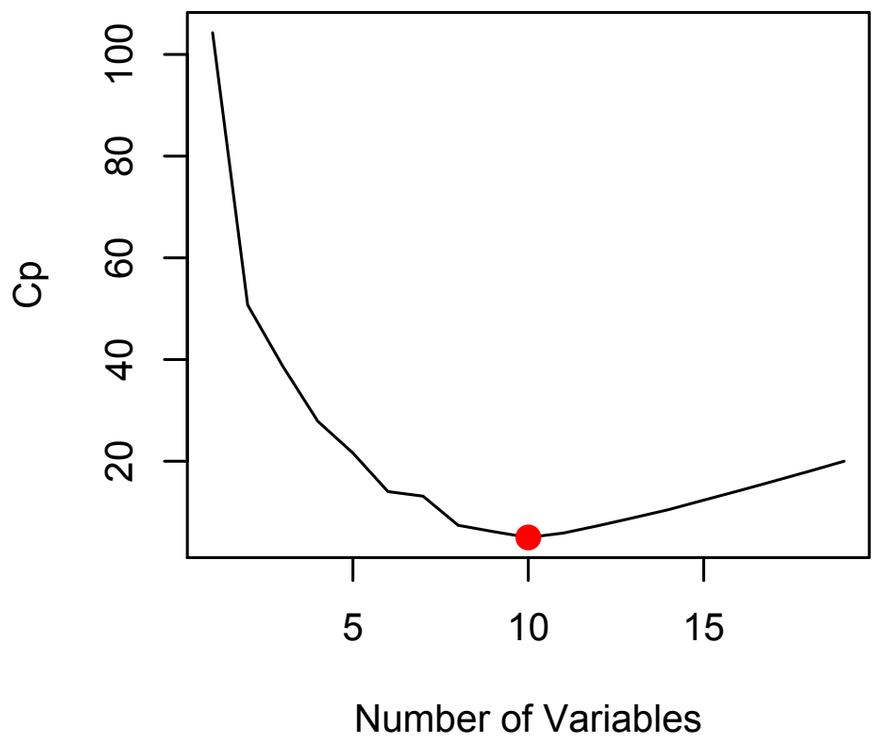
```
> par(mfrow=c(2,2))
> plot(reg.summary$rss,xlab="Number of Variables",ylab="RSS",
    type="l")
> plot(reg.summary$adjr2,xlab="Number of Variables",
    ylab="Adjusted RSq",type="l")
```

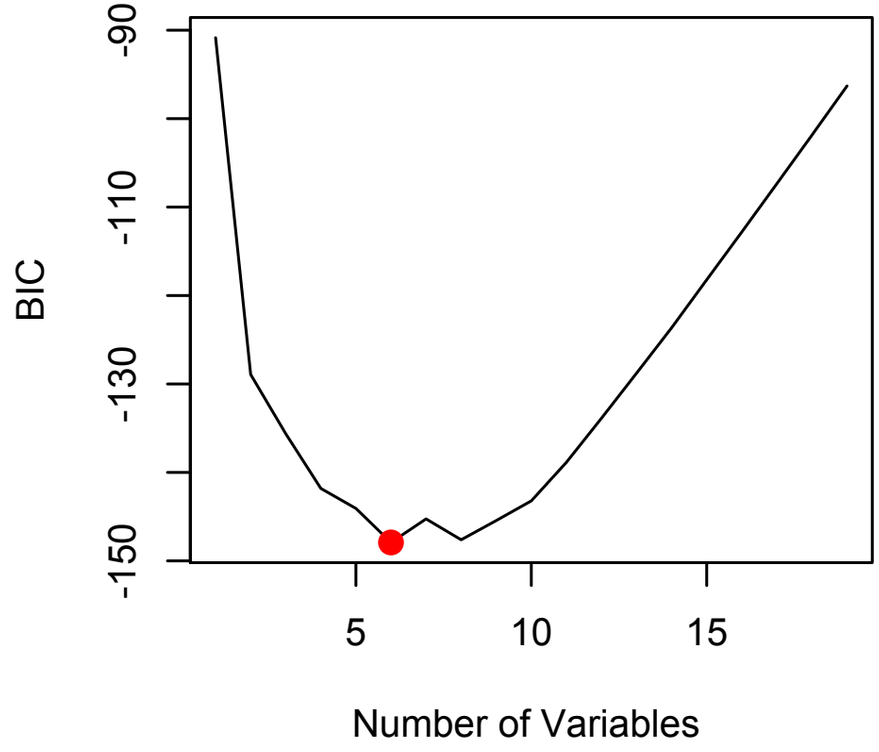
```
> which.max(reg.summary$adjr2)
[1] 11
> points(11,reg.summary$adjr2[11], col="red",cex=2,pch=20)
```

```
> plot(reg.summary$cp,xlab="Number of Variables",ylab="Cp",
    type='l')
> which.min(reg.summary$cp)
[1] 10
> points(10,reg.summary$cp[10],col="red",cex=2,pch=20)
```


RSS
2.4e+07
3.0e+07
3.6e+07
5
10
15
Number of Variables


Adjusted RSq
0.35
0.40
0.45
0.50
5
10
15
Number of Variables


Cp
20
40
60
80
100
5
10
15
Number of Variables


BIC
-150
-130
-110
-90
5
10
15
Number of Variables

```
> regfit.fwd=regsubsets(Salary~.,data=Hitters,nvmax=19,
    method="forward")
> summary(regfit.fwd)
> regfit.bwd=regsubsets(Salary~.,data=Hitters,nvmax=19,
    method="backward")
> summary(regfit.bwd)
```

```
Selection Algorithm: forward
          AtBat Hits HmRun Runs RBI Walks Years CAtBat CHits CHmRun CRuns CRBI CWalks LeagueN DivisionW PutOuts Assists Errors NewLeagu
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 ) "*"   "*"  "*"   "*"  "*" "*"   "*"   "*"    "*"   "*"    "*"   "*"  "*"    "*"     "*"       "*"     "*"     "*"    "*"
```

```
> coef(regfit.full,7)
(Intercept)        Hits       Walks      CAtBat       CHits
     79.451       1.283       3.227      -0.375       1.496
     CHmRun   DivisionW     PutOuts
      1.442    -129.987       0.237
> coef(regfit.fwd,7)
(Intercept)       AtBat        Hits       Walks        CRBI
    109.787      -1.959       7.450       4.913       0.854
     CWalks   DivisionW     PutOuts
     -0.305    -127.122       0.253
> coef(regfit.bwd,7)
(Intercept)       AtBat        Hits       Walks       CRuns
    105.649      -1.976       6.757       6.056       1.129
     CWalks   DivisionW     PutOuts
     -0.716    -116.169       0.303
```

Ridge regression

Ridge regression is like least squares but shrinks the estimated coefficients towards zero. Given a response vector $y \in \mathbb{R}^n$ and a predictor matrix $X \in \mathbb{R}^{n \times p}$, the ridge regression coefficients are defined as

$$\begin{aligned}\hat{\beta}^{\text{ridge}} &= \underset{\beta \in \mathbb{R}^p}{\text{argmin}} \sum_{i=1}^{n} (y_i - x_i^T \beta)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \\ &= \underset{\beta \in \mathbb{R}^p}{\text{argmin}} \underbrace{\|y - X\beta\|_2^2}_{\text{RSS}} + \lambda \underbrace{\|\beta\|_2^2}_{\text{Loss}} \qquad \text{Penalty}\end{aligned}$$

Here $\lambda \geq 0$ is a tuning parameter, which controls the strength of the penalty term. Note that:

- When $\lambda = 0$, we get the linear regression estimate
- When $\lambda = \infty$, we get $\hat{\beta}^{\text{ridge}} = 0$
- For λ in between, we are balancing two ideas: fitting a linear model of y on X, and shrinking the coefficients

- The variance of the ridge regression estimate is

$$\mathrm{Var}(\widehat{\boldsymbol{\beta}}) = \sigma^2 \mathbf{W}\mathbf{X}^T\mathbf{X}\mathbf{W},$$

  where $\mathbf{W} = (\mathbf{X}^T\mathbf{X} + n\lambda\mathbf{I})^{-1}$
- Meanwhile, the bias is

$$\mathrm{Bias}(\widehat{\boldsymbol{\beta}}) = -n\lambda\mathbf{W}\boldsymbol{\beta}$$

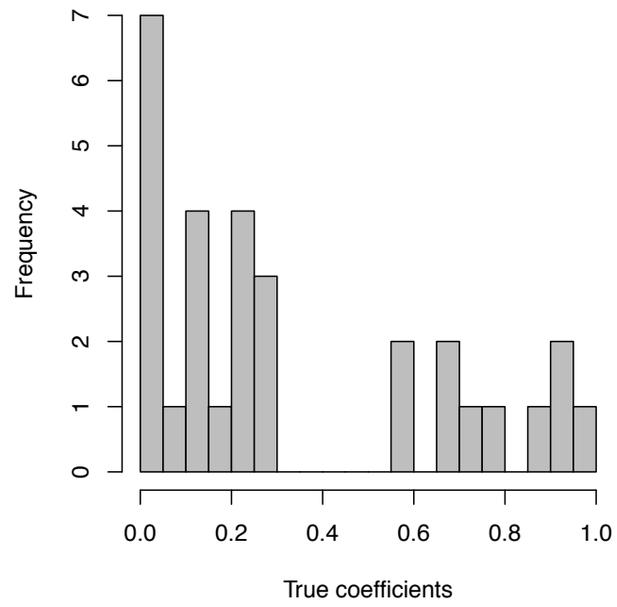
- Both bias and variance contribute to overall accuracy, as measured by mean squared error (MSE):

$$\begin{aligned}\mathrm{MSE}(\widehat{\boldsymbol{\beta}}) &= \mathbb{E}\left\|\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}\right\|^2 \\ &= \sum_j \mathrm{Var}(\widehat{\boldsymbol{\beta}}_j) + \sum_j \mathrm{Bias}(\widehat{\boldsymbol{\beta}}_j)^2\end{aligned}$$

- **Theorem:** There always exists a value λ such that

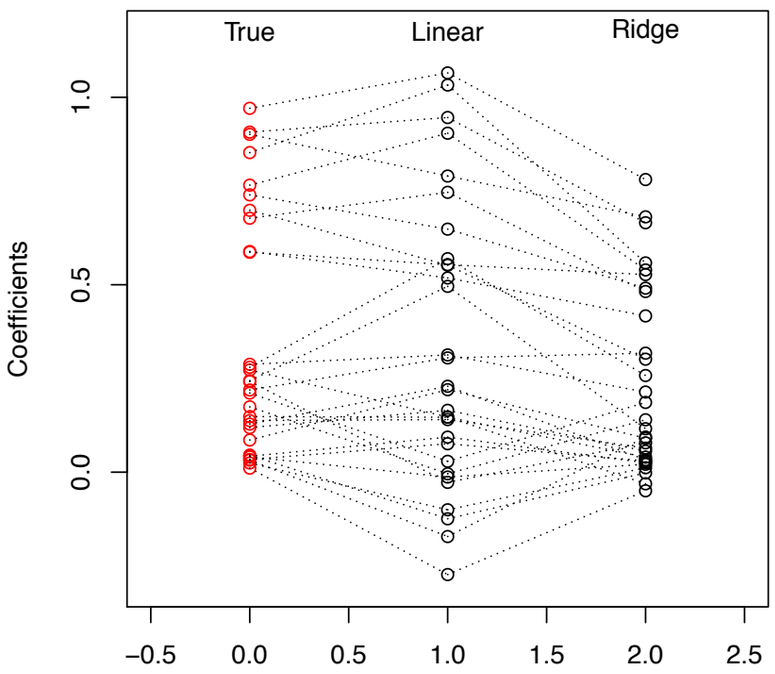
$$\mathrm{MSE}\left(\widehat{\boldsymbol{\beta}}_\lambda\right) < \mathrm{MSE}\left(\widehat{\boldsymbol{\beta}}^{\mathrm{OLS}}\right)$$

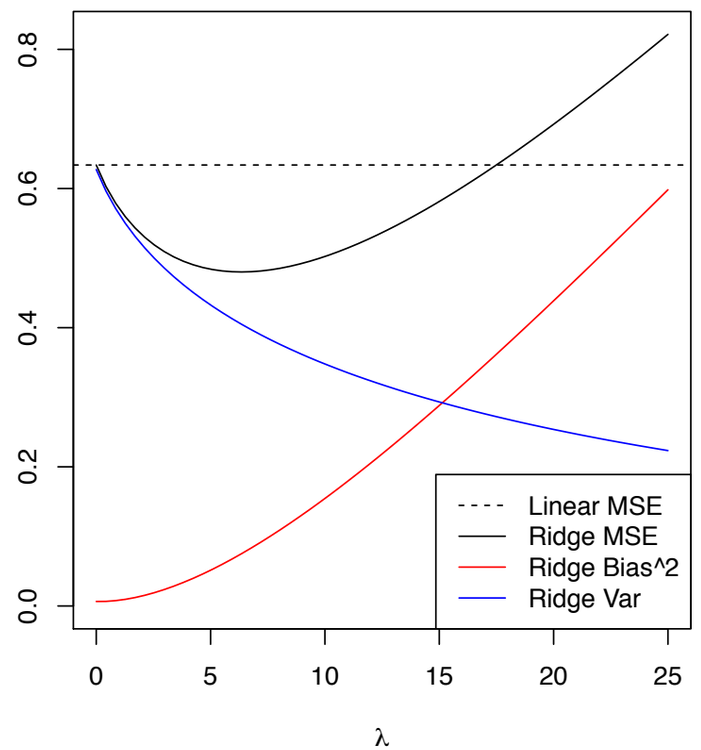
Example: simulation with $n = 50$ and $p = 30$. The entries of the predictor matrix $X \in \mathbb{R}^{50\times 30}$ are all i.i.d. $N(0, 1)$, so overall the variables have low correlation



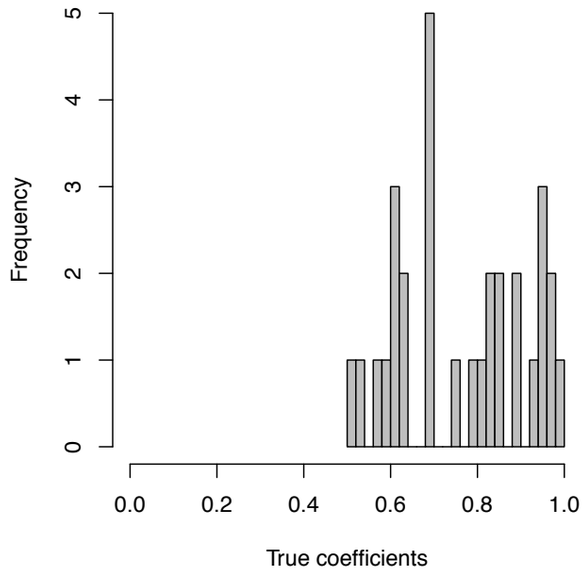
Linear regression:
Squared bias ≈ 0.006
Variance ≈ 0.627
Pred. error $\approx 1 + 0.006 + 0.627$
≈ 1.633

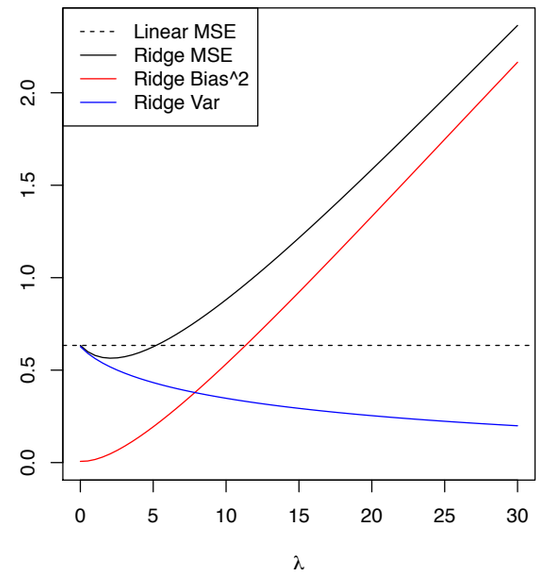
Ridge regression, at its best:
Squared bias ≈ 0.077
Variance ≈ 0.403
Pred. error $\approx 1 + 0.077 + 0.403$
≈ 1.48



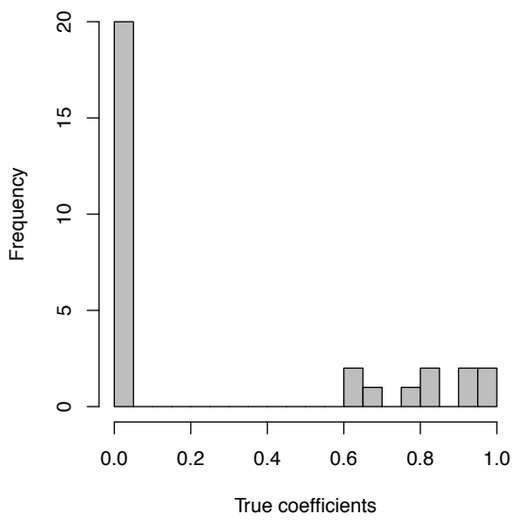


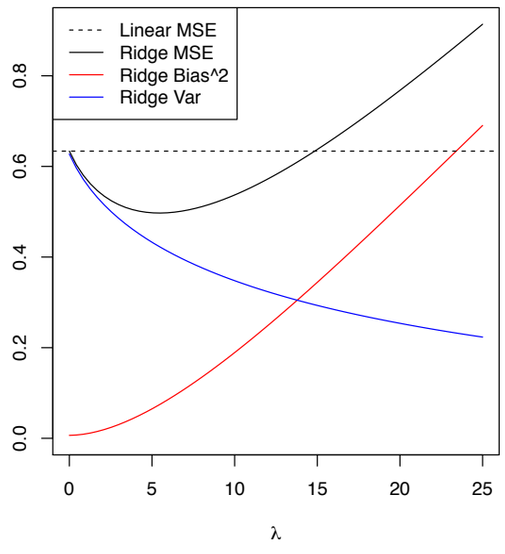
Same setup as our last example: $n = 50$, $p = 30$, and $\sigma^2 = 1$. Except now the true coefficients are all moderately large (between 0.5 and 1). Histogram:

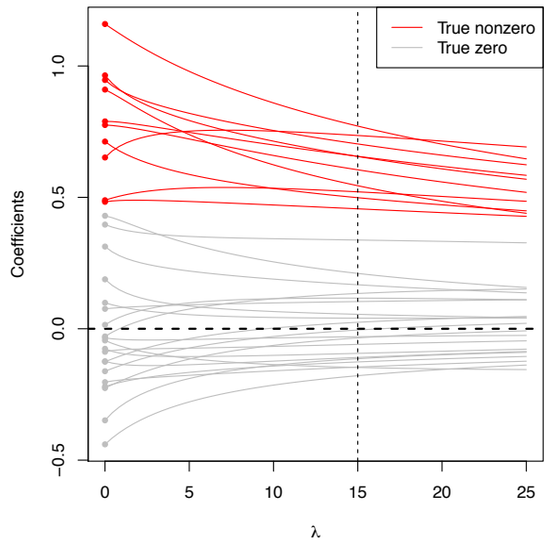




Same general setup as our running example: $n = 50$, $p = 30$, and $\sigma^2 = 1$. Now, the true coefficients: 10 are large (between 0.5 and 1) and 20 are exactly 0. Histogram:







```
> x=model.matrix(Salary~.,Hitters)[,-1]
> y=Hitters$Salary
> library(glmnet)
> grid=10^seq(10,-2,length=100)
> ridge.mod=glmnet(x,y,alpha=0,lambda=grid)
```

glmnet automatically prescribes a range of lambda values: here we specify a range from 10^{10} to 10^{-2} so as to cover scenarios from full LS fit to null (intercept only) model

alpha = 0 specifies ridge regression; alpha = 1 lasso

covariates standardized by default; can turn off
x must be a numeric matrix (not a data.frame)

possible to force selected covariates into the model without their coefficients being subject to shrinkage

Impact of penalty on coefficients

```
> ridge.mod$lambda[50]
[1] 11498
> coef(ridge.mod)[,50]
(Intercept)       AtBat        Hits       HmRun        Runs
    407.356       0.037       0.138       0.525       0.231
        RBI       Walks       Years      CAtBat       CHits
      0.240       0.290       1.108       0.003       0.012
     CHmRun       CRuns        CRBI      CWalks     LeagueN
      0.088       0.023       0.024       0.025       0.085
  DivisionW     PutOuts     Assists      Errors  NewLeagueN
     -6.215       0.016       0.003      -0.021       0.301
> sqrt(sum(coef(ridge.mod)[-1,50]^2))
[1] 6.36
```

```
> ridge.mod$lambda[60]
[1] 705
> coef(ridge.mod)[,60]
(Intercept)       AtBat        Hits       HmRun        Runs
     54.325       0.112       0.656       1.180       0.938
        RBI       Walks       Years      CAtBat       CHits
      0.847       1.320       2.596       0.011       0.047
     CHmRun       CRuns        CRBI      CWalks     LeagueN
      0.338       0.094       0.098       0.072      13.684
  DivisionW     PutOuts     Assists      Errors  NewLeagueN
    -54.659       0.119       0.016      -0.704       8.612
> sqrt(sum(coef(ridge.mod)[-1,60]^2))
[1] 57.1
```

Impact of penalty on MSE

```
> set.seed(1)
> train=sample(1:nrow(x), nrow(x)/2)
> test=(-train)
> y.test=y[test]
```

```
> 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)
[1] 101037
```

s : lambda value for predict fn

```
> ridge.pred=predict(ridge.mod,s=1e10,newx=x[test,])
> mean((ridge.pred-y.test)^2)
[1] 193253
```

very large lambda : just intercept

```
> ridge.pred=predict(ridge.mod,s=0,newx=x[test,],exact=T)
> mean((ridge.pred-y.test)^2)
[1] 114783
```

lambda=0: least squares solution

How to determine extent of penalization? cv.glmnet

```
> cv.out=cv.glmnet(x[train,],y[train],alpha=0)
> bestlam=cv.out$lambda.min
> bestlam
[1] 212
```

```
> ridge.pred=predict(ridge.mod,s=bestlam,newx=x[test,])
> mean((ridge.pred-y.test)^2)
[1] 96016
```

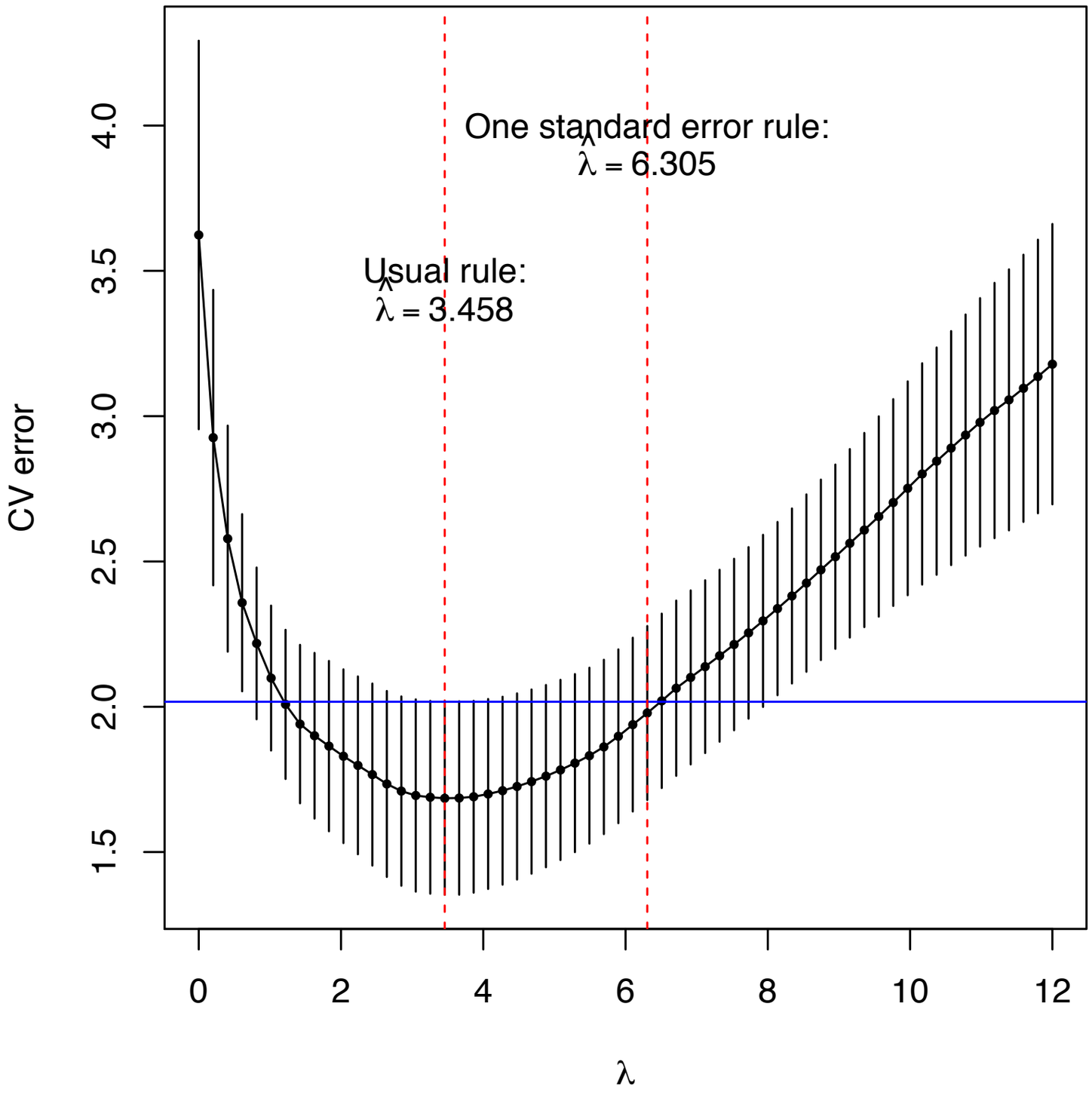
The one standard error rule is an alternative rule for choosing the value of the tuning parameter, as opposed to the usual rule

$$\hat{\theta} = \underset{\theta \in \{\theta_1, \ldots \theta_m\}}{\operatorname{argmin}} \ \mathrm{CV}(\theta)$$

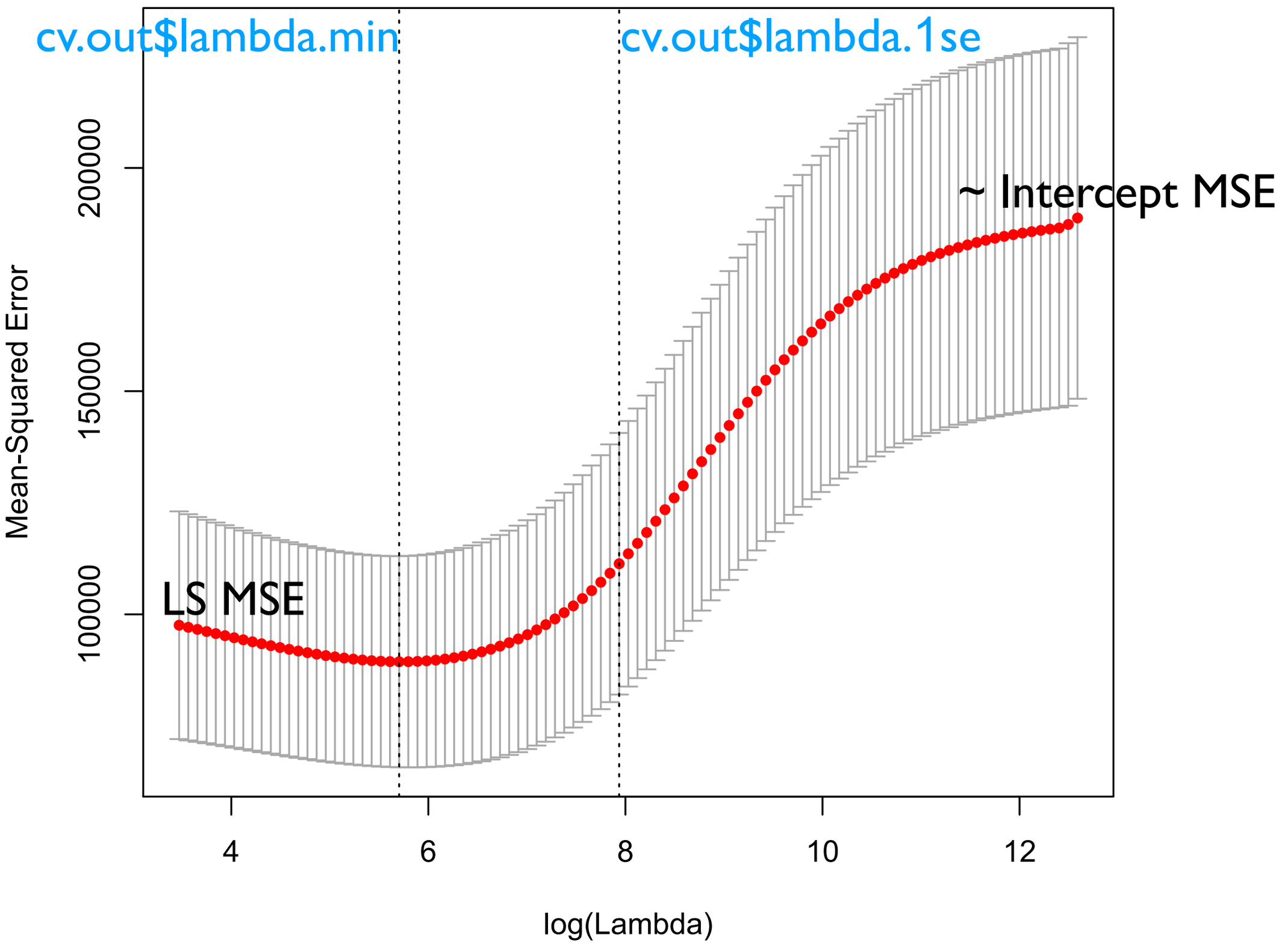
We first find the usual minimizer $\hat{\theta}$ as above, and then move θ in the direction of increasing regularization (this may be increasing or decreasing θ, depending on the parametrization) as much as we can, such that the cross-validation error curve is still within one standard error of $\mathrm{CV}(\hat{\theta})$. In other words, we maintain

$$\mathrm{CV}(\theta) \leq \mathrm{CV}(\hat{\theta}) + \mathrm{SE}(\hat{\theta})$$

Idea: "All else equal (up to one standard error), go for the simpler (more regularized) model"


One standard error rule:
λ = 6.305
Usual rule:
λ = 3.458
CV error
4.0
3.5
3.0
2.5
2.0
1.5
0
2
4
6
8
10
12
λ

number of covariates in model : ridge retains all



The Lasso:
least absolute selection and shrinkage operator

The lasso coefficients $\hat{\beta}^{\text{lasso}}$ minimize the quantity

$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}|\beta_j| \;= \text{RSS} + \lambda\sum_{j=1}^{p}|\beta_j|$$

RSS

The only difference between the lasso problem and ridge regression is that the latter uses a (squared) ℓ_2 penalty $\|\beta\|_2^2$, while the former uses an ℓ_1 penalty $\|\beta\|_1$. But even though these problems look similar, their solutions behave very differently

The tuning parameter λ controls the strength of the penalty, and (like ridge regression) we get $\hat{\beta}^{\text{lasso}} =$ the linear regression estimate when $\lambda = 0$, and $\hat{\beta}^{\text{lasso}} = 0$ when $\lambda = \infty$

For λ in between these two extremes, we are balancing two ideas: fitting a linear model of y on X, and shrinking the coefficients. But the nature of the ℓ_1 penalty causes some coefficients to be shrunken to zero exactly

This is what makes the lasso substantially different from ridge regression: it is able to perform variable selection in the linear model. As λ increases, more coefficients are set to zero (less variables are selected), and among the nonzero coefficients, more shrinkage is employed

Another formulation of the lasso

$$\underset{\beta}{\text{minimize}} \left\{ \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 \right\} \quad \text{subject to} \quad \sum_{j=1}^{p} |\beta_j| \leq s$$

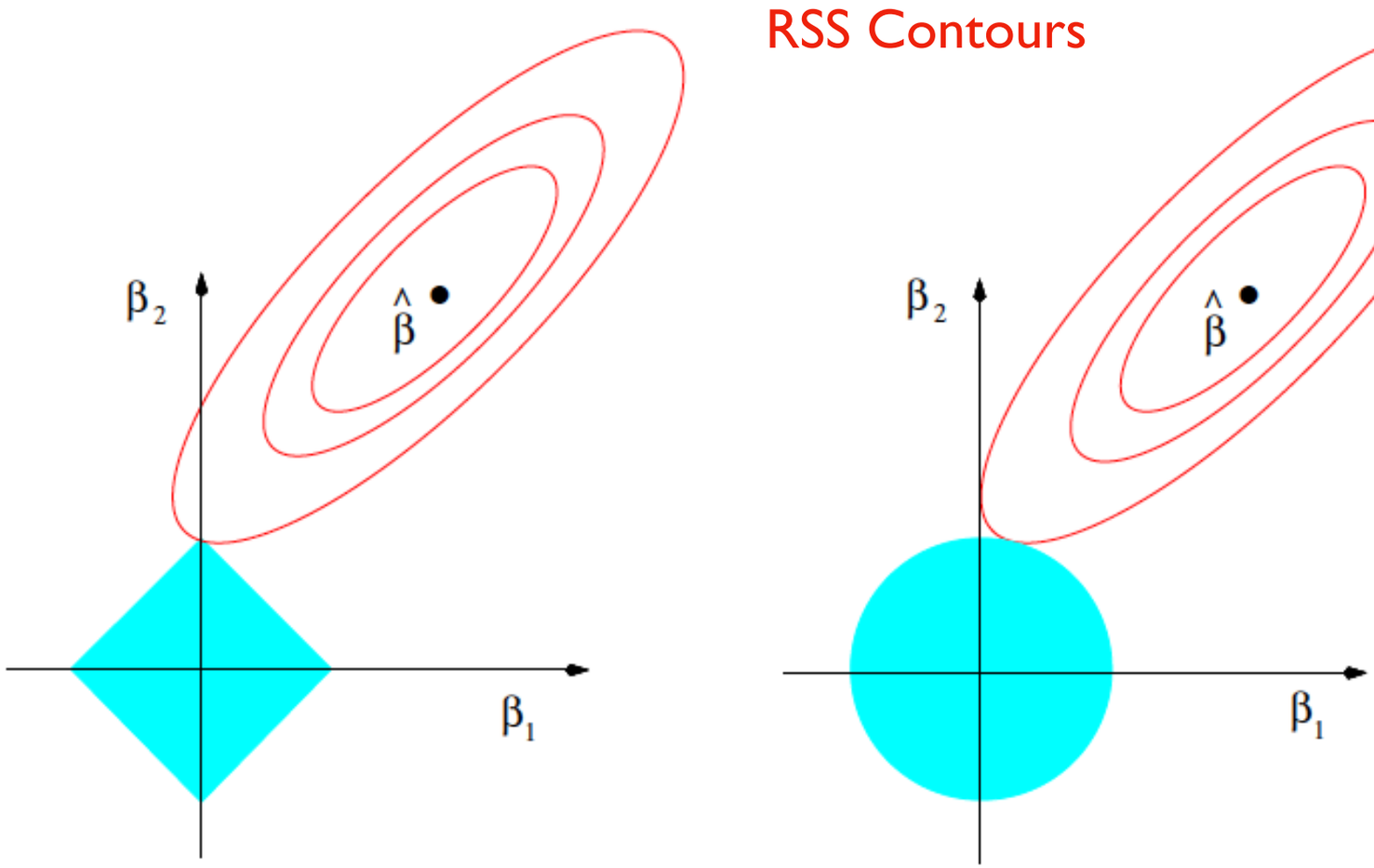
RSS

For every value of λ there is some value of *s* such that the above constrained minimization will give the same lasso solution

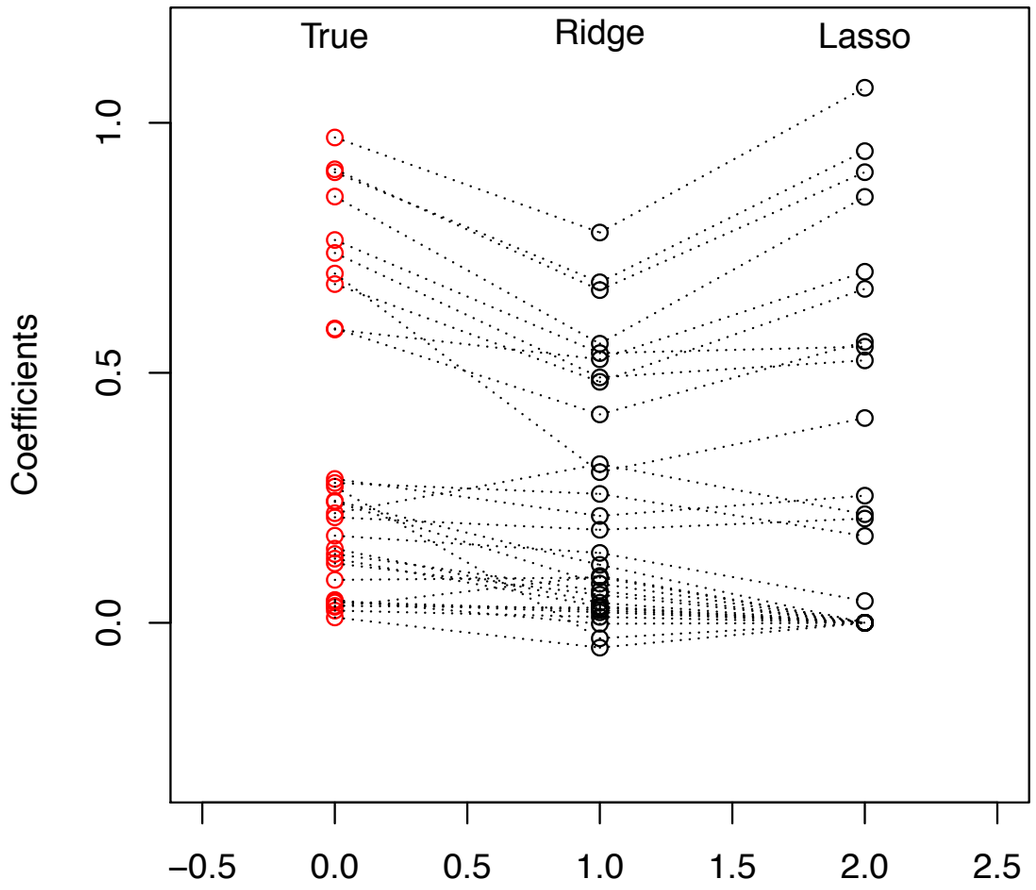
When $p = 2$ the lasso coefficients have the smallest RSS amongst those points lying within the diamond defined by $|\beta_1| + |\beta_2| \leq s$

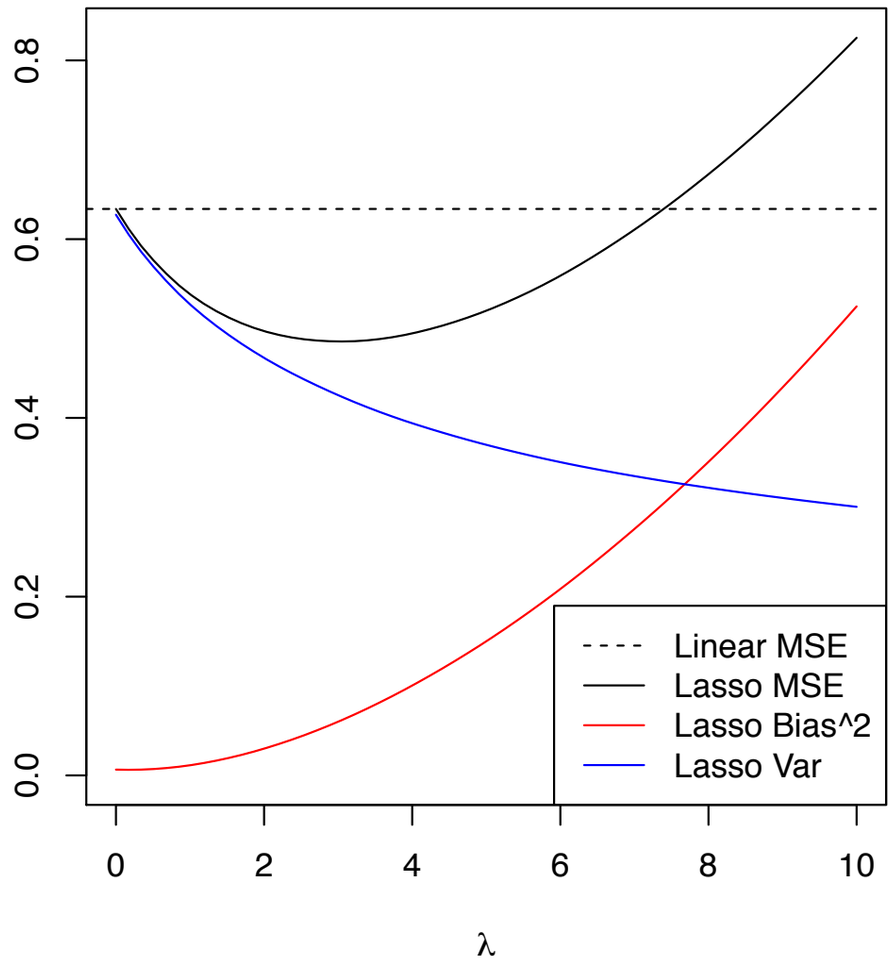
Similarly, the ridge coefficients have the smallest RSS amongst those points lying within the circle defined by $\beta_1^2 + \beta_2^2 \leq s$

Why does the lasso give zero coefficients?



Example: $n = 50$, $p = 30$; true coefficients: 10 large, 20 small





- To understand the effect of the ridge penalty on the estimator $\widehat{\boldsymbol{\beta}}$, it helps to consider the special case of an orthonormal design matrix ($\mathbf{X}^T\mathbf{X}/n = \mathbf{I}$)
- In this case,

$$\widehat{\beta}_J = \frac{\widehat{\beta}_J^{\text{OLS}}}{1+\lambda}$$

- This illustrates the essential feature of ridge regression: *shrinkage*; i.e., the primary effect of applying ridge penalty is to shrink the estimates toward zero

- **Result:** In the orthonormal case, the lasso estimate is

$$\widehat{\beta}_j(\lambda) = \begin{cases} z_j - \lambda, & \text{if } z_j > \lambda, \\ 0, & \text{if } |z_j| \leq \lambda, , \\ z_j + \lambda, & \text{if } z_j < -\lambda \end{cases}$$

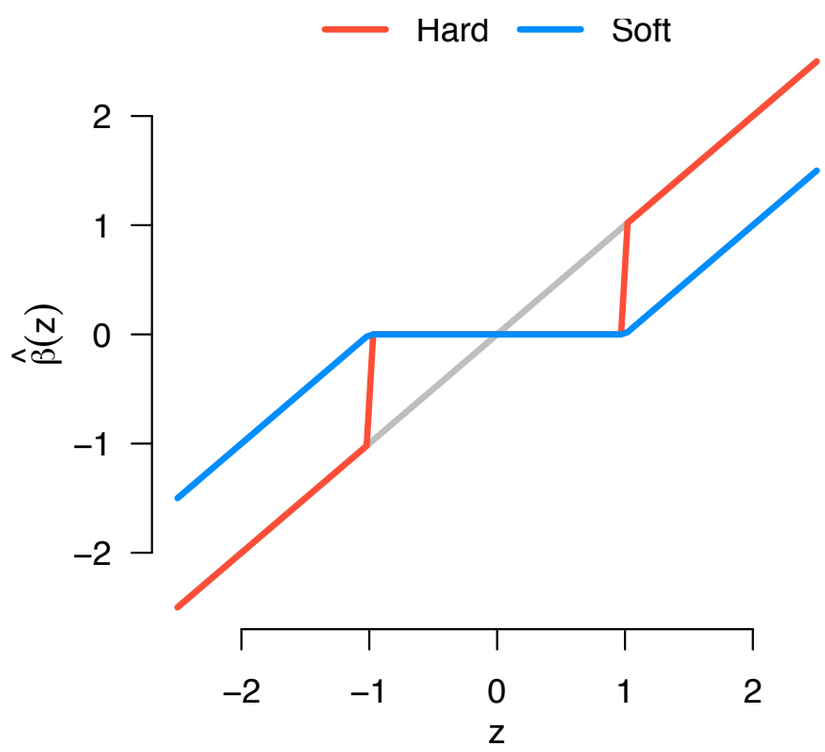
  where $z_j = \mathbf{x}_j^T\mathbf{y}/n$ is the OLS solution

- The result on the previous slide can be written more compactly as

$$\widehat{\beta}_j(\lambda) = S(z_j|\lambda),$$

where the function $S(\cdot|\lambda)$ is known as the *soft thresholding operator*

- By comparison, the "hard" thresholding operator is $H(z, \lambda) = zI\{|z| > \lambda\}$, where $I(S)$ is the indicator function for set S



- The multivariate version of hard thresholding is ℓ_0 penalization, in which we minimize the objective function

$$\frac{1}{2n}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda\|\boldsymbol{\beta}\|_0,$$

  where $\|\boldsymbol{\beta}\|_0 = \sum_j I(\beta_j \neq 0)$
- For the orthonormal case, the solution is given by $\widehat{\beta}_j = H(\widehat{\beta}_j^{OLS}, \sqrt{2\lambda})$
- Estimating $\boldsymbol{\beta}$ in this manner is equivalent to subset selection, and model selection criteria such as AIC and BIC are simply special cases corresponding to different λ values

- Thus, the lasso can be thought of as a "soft" relaxation of ℓ_0 penalized regression
- This relaxation has two important benefits:
  - Estimates are continuous with respect to both λ and the data
  - The lasso objective function is convex
- These facts allow optimization of ℓ_1-penalized regression to proceed very efficiently, as we will see; in comparison, ℓ_0-penalized regression is computationally infeasible when p is large

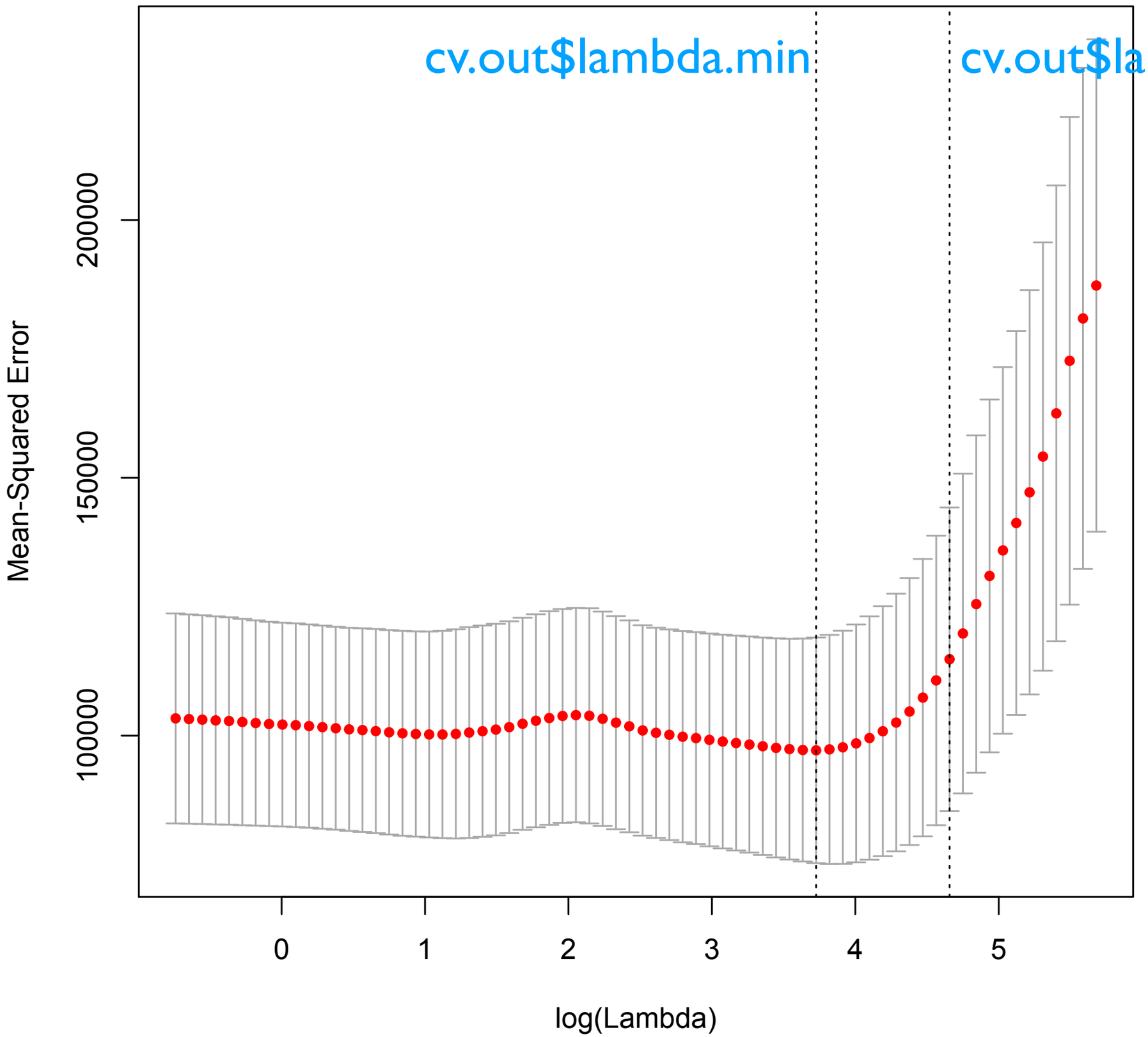
- To get around the difficulty of finding the best possible subset, a common approach is to employ the greedy algorithm known as forward selection
- Like forward selection, the lasso will allow more variables to enter the model as λ is lowered
- However, the lasso performs a continuous version of variable selection and is less greedy about allowing selected variables into the model

number of covariates in model : lasso selects!

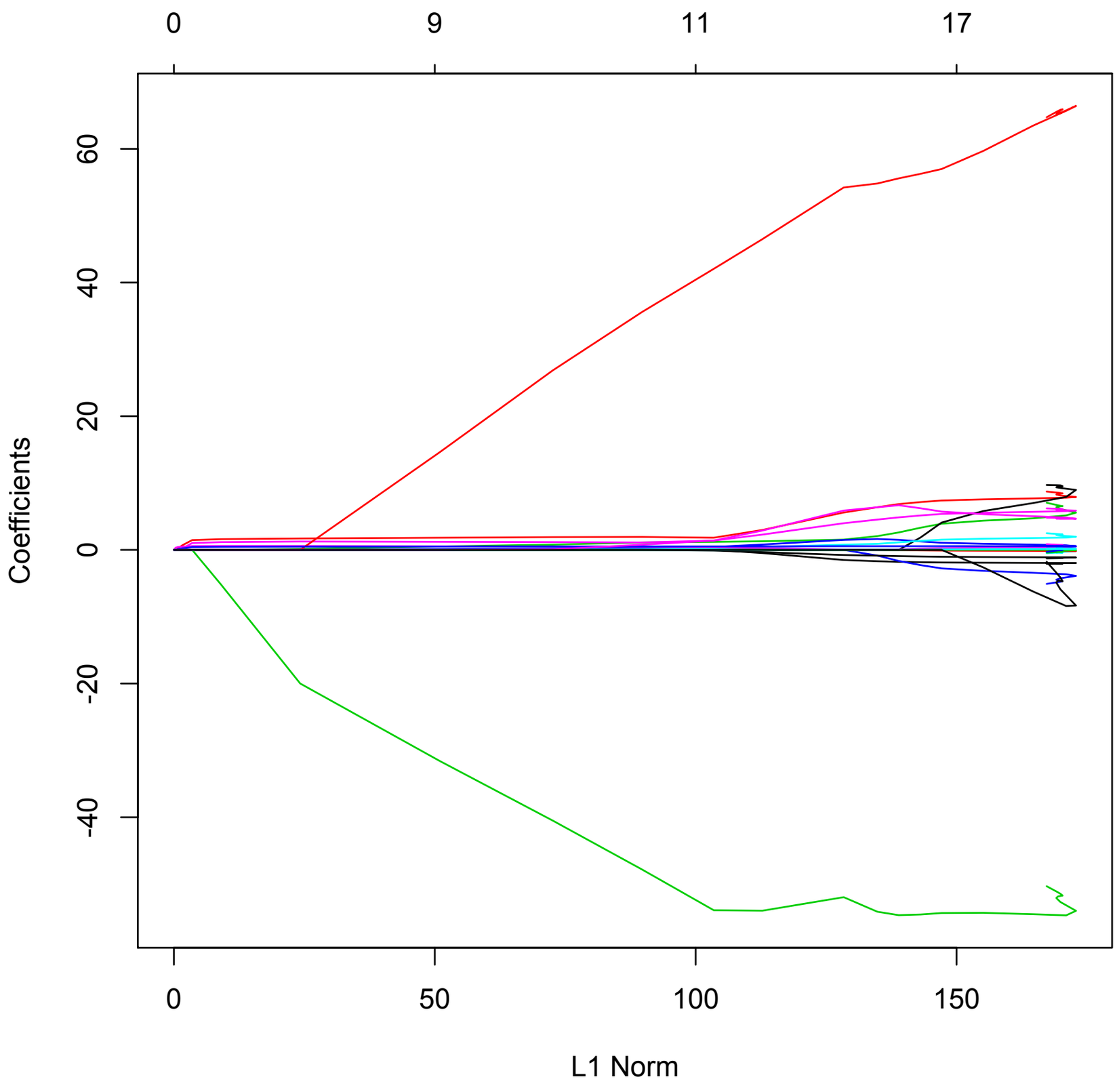
17 17 16 13 12 12 11 10 9 9 7 5 4 4 4 3 1 1 0

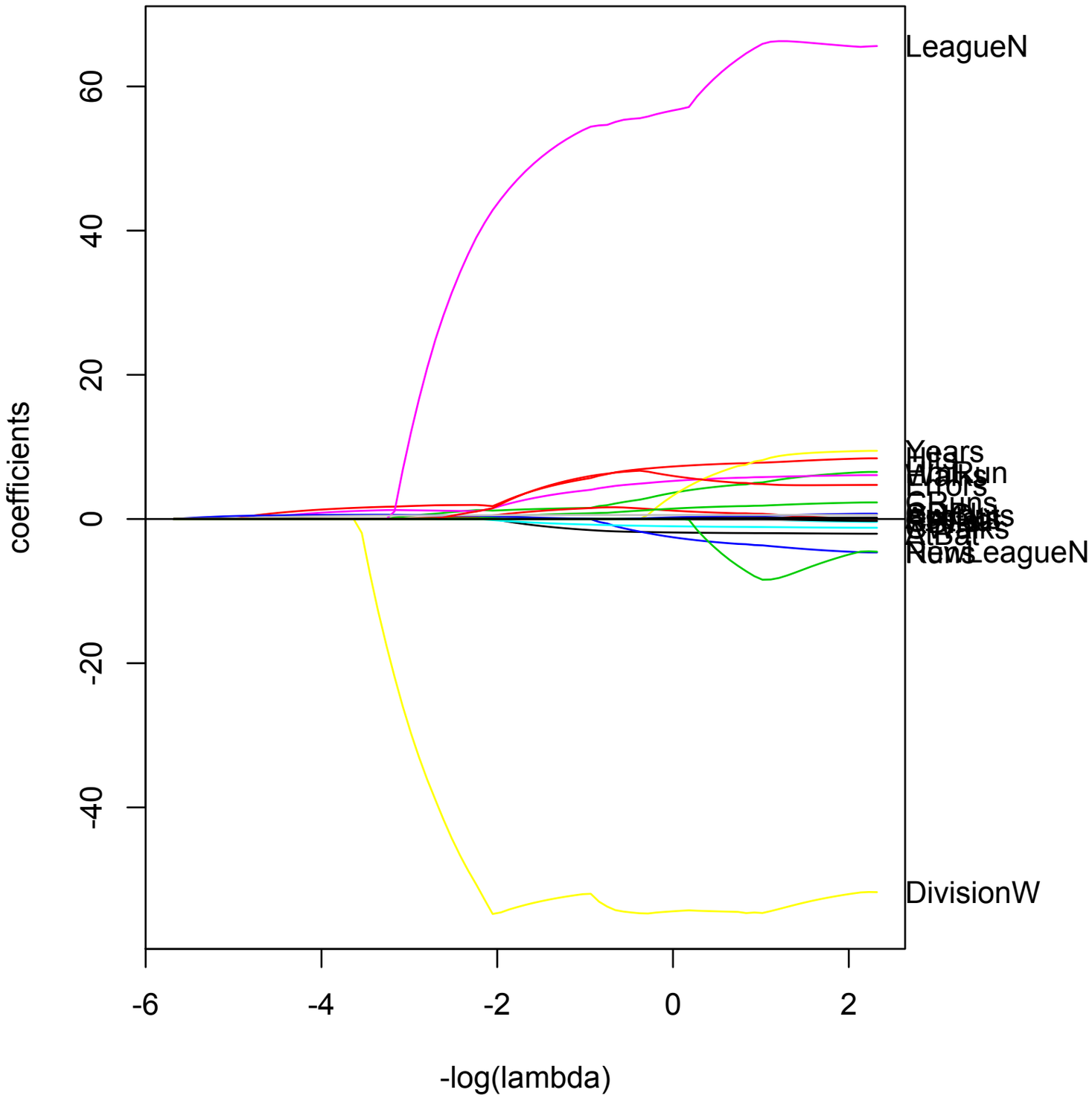
cv.out$lambda.min

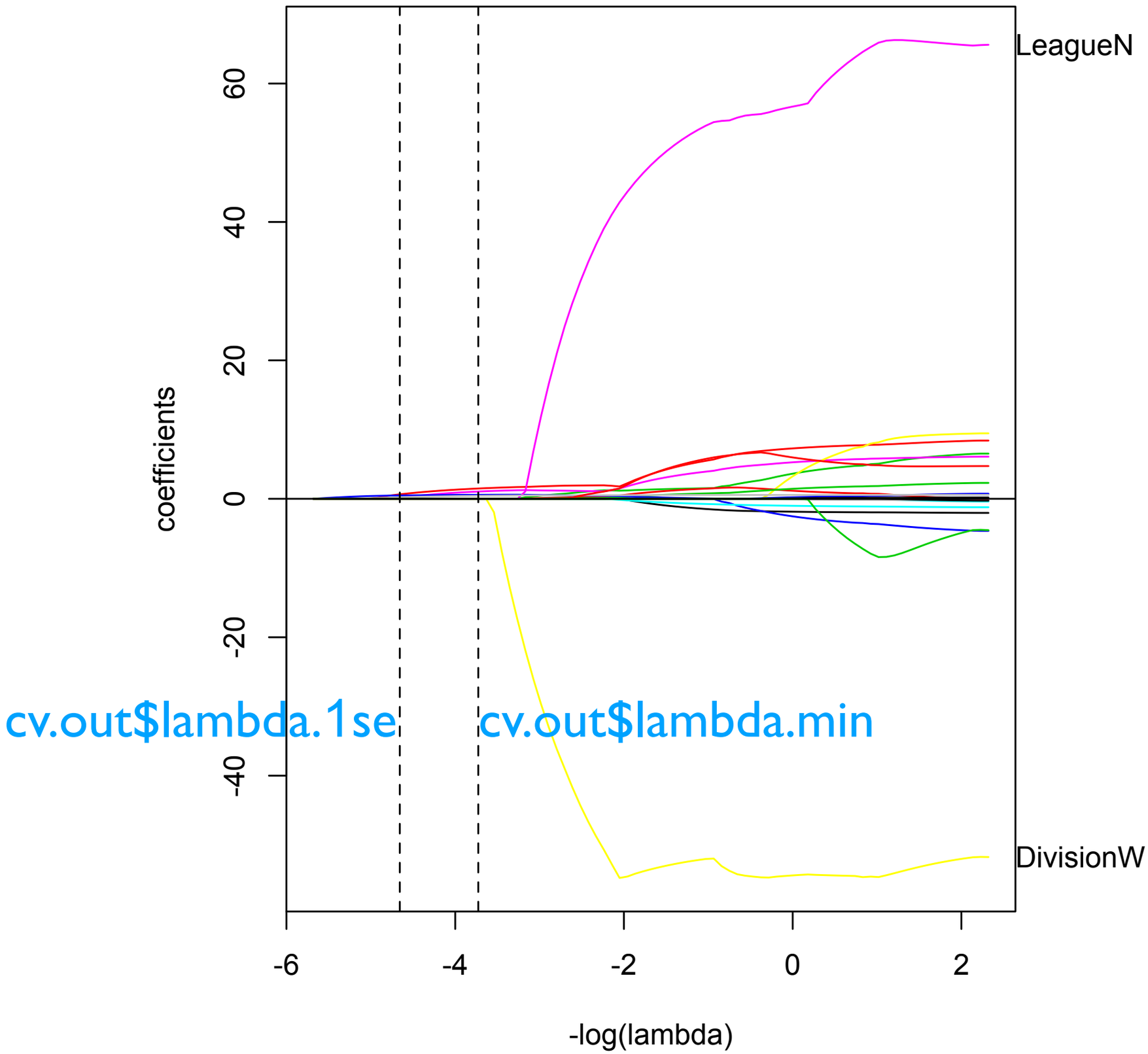
cv.out$lambda.1se



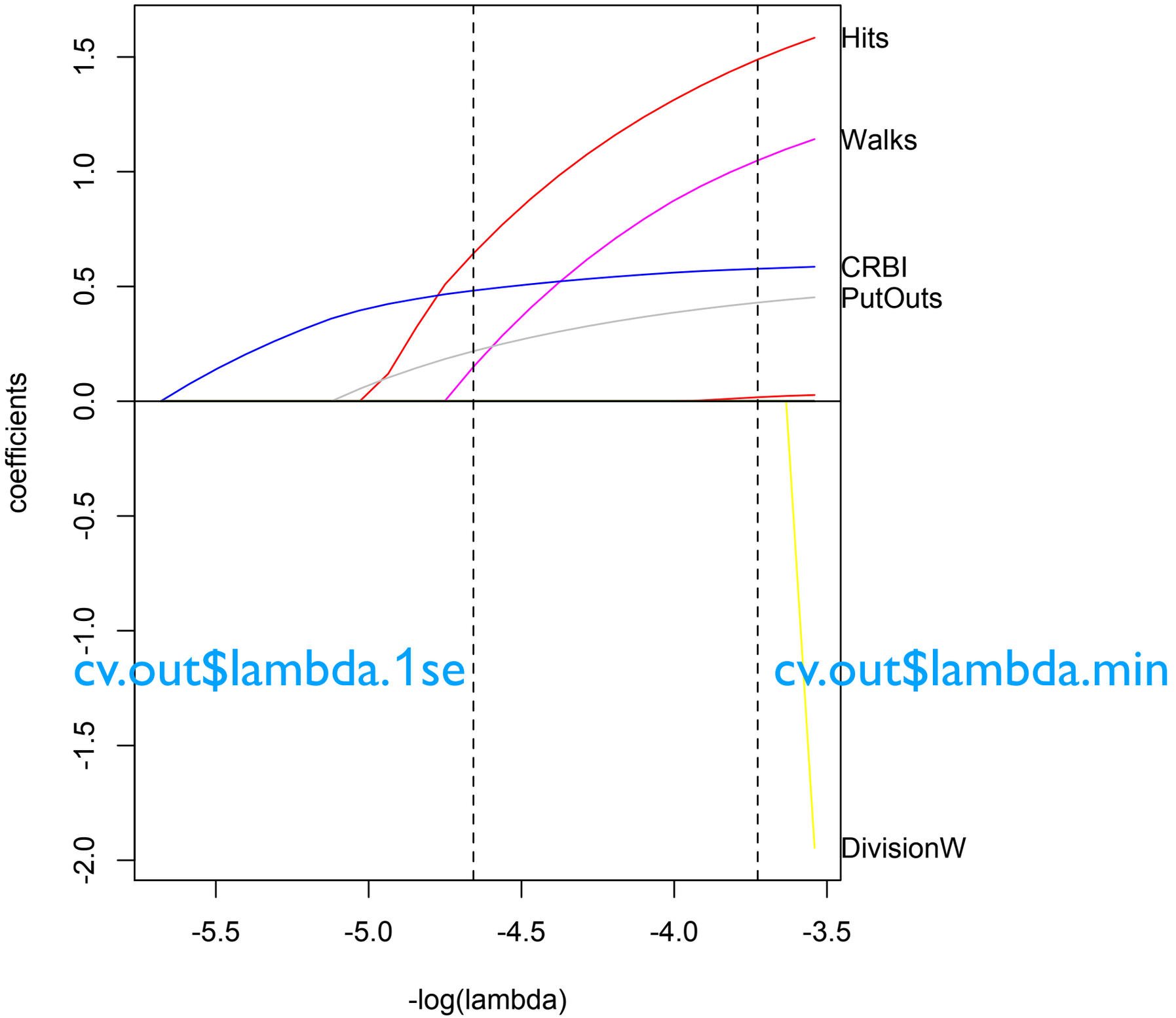
```
> plot(lasso.mod)
```




coefficients
60
40
20
0
-20
-40
-6
-4
-2
0
2
-log(lambda)
LeagueN
Years
DivisionW

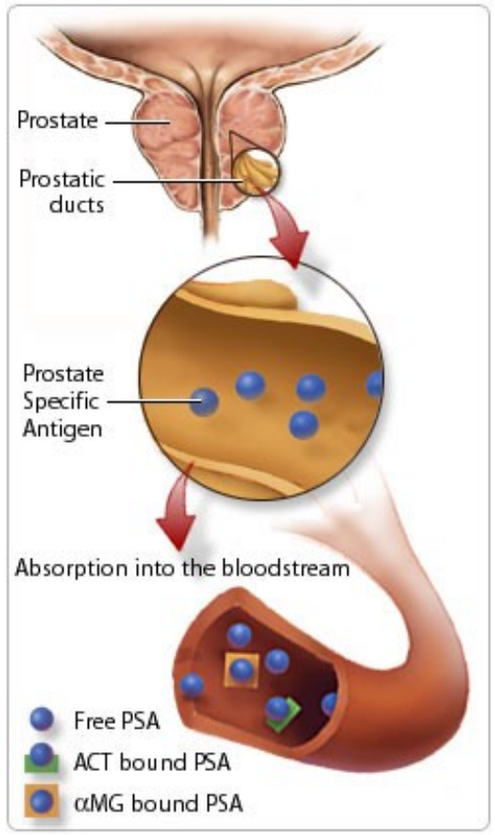

LeagueN
DivisionW
coefficients
60
40
20
0
-20
-40
-6
-4
-2
0
2
-log(lambda)
cv.out$lambda.1se
cv.out$lambda.min

glmnet argument dfmax: limits the maximum number of covariates



Recall prostate cancer example:

E.g., suppose that we are studying the level of prostate-specific antigen (PSA), which is often elevated in men who have prostate cancer. We look at $n = 97$ men with prostate cancer, and $p = 8$ clinical measurements.[1] We are interested in identifying a small number of predictors, say 2 or 3, that drive PSA

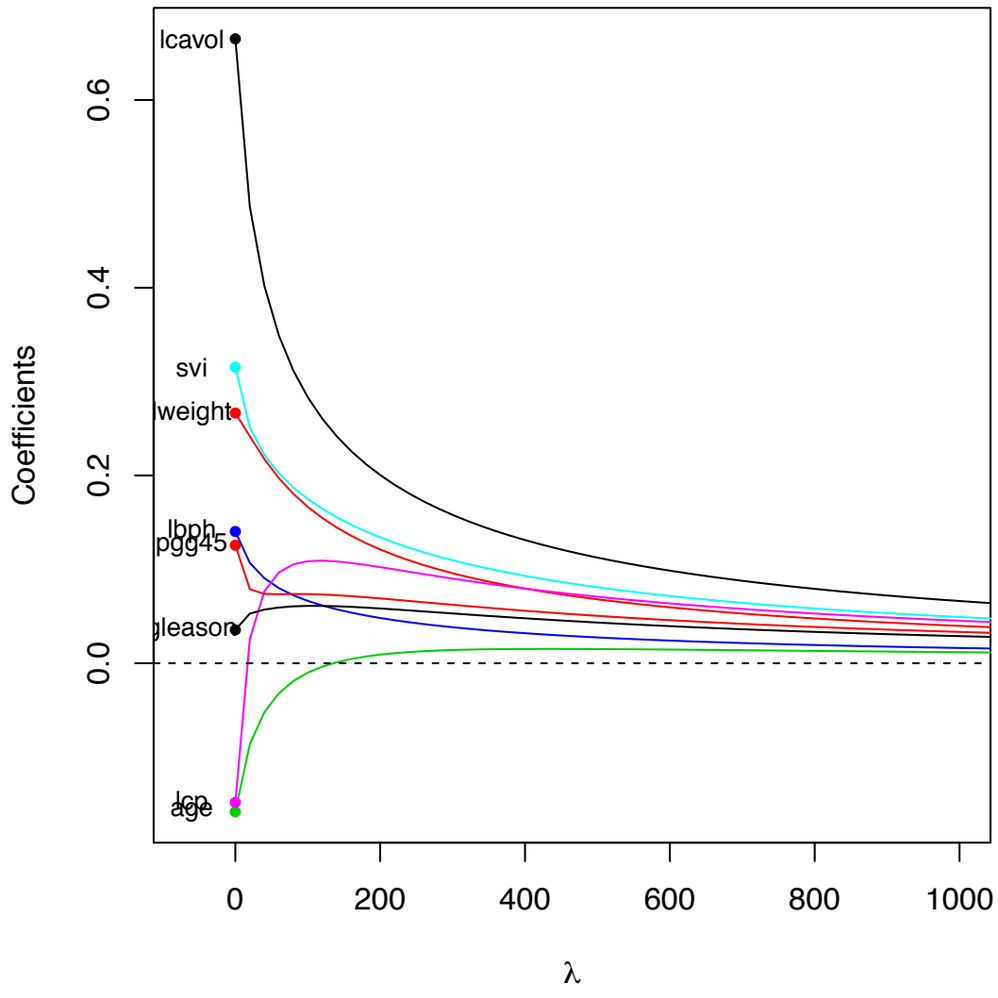


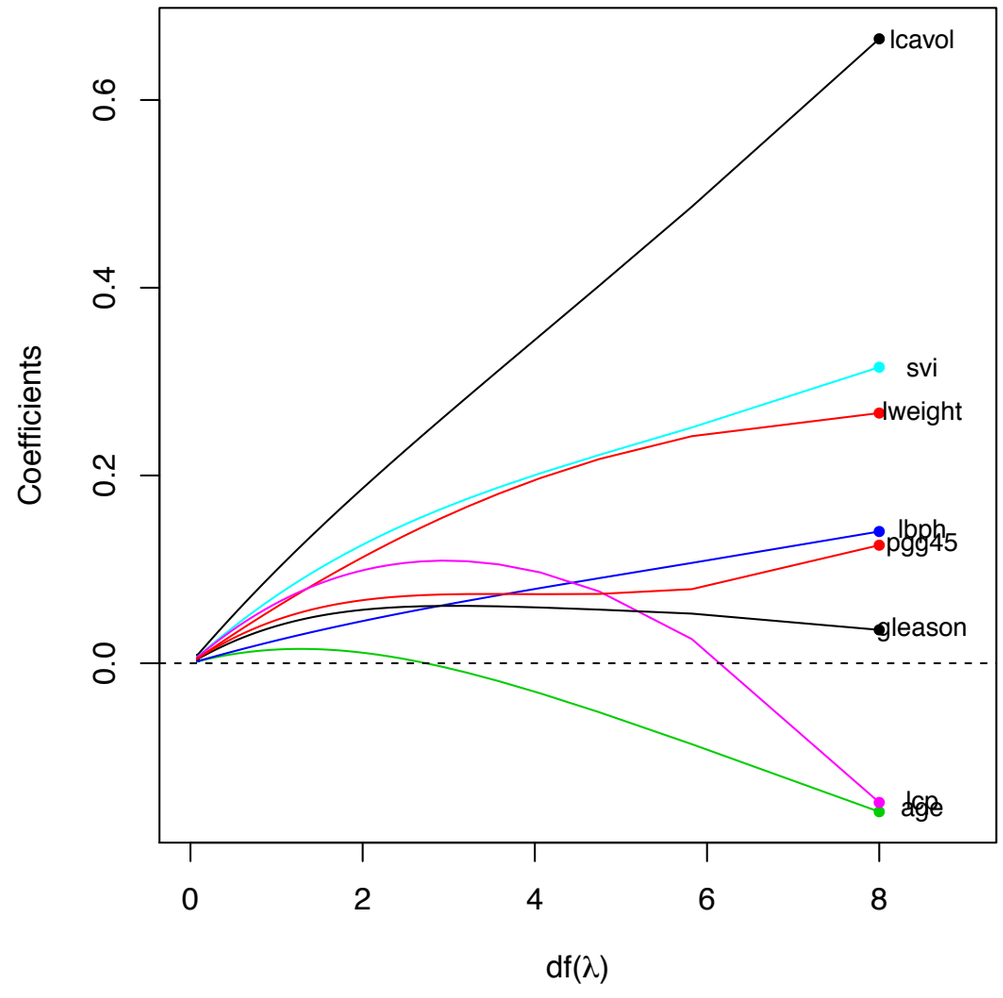
[2]

[1] Data from Stamey et al. (1989), "Prostate specific antigen in the diag..."

[2] Figure from `http://www.mens-hormonal-health.com/psa-score.html`

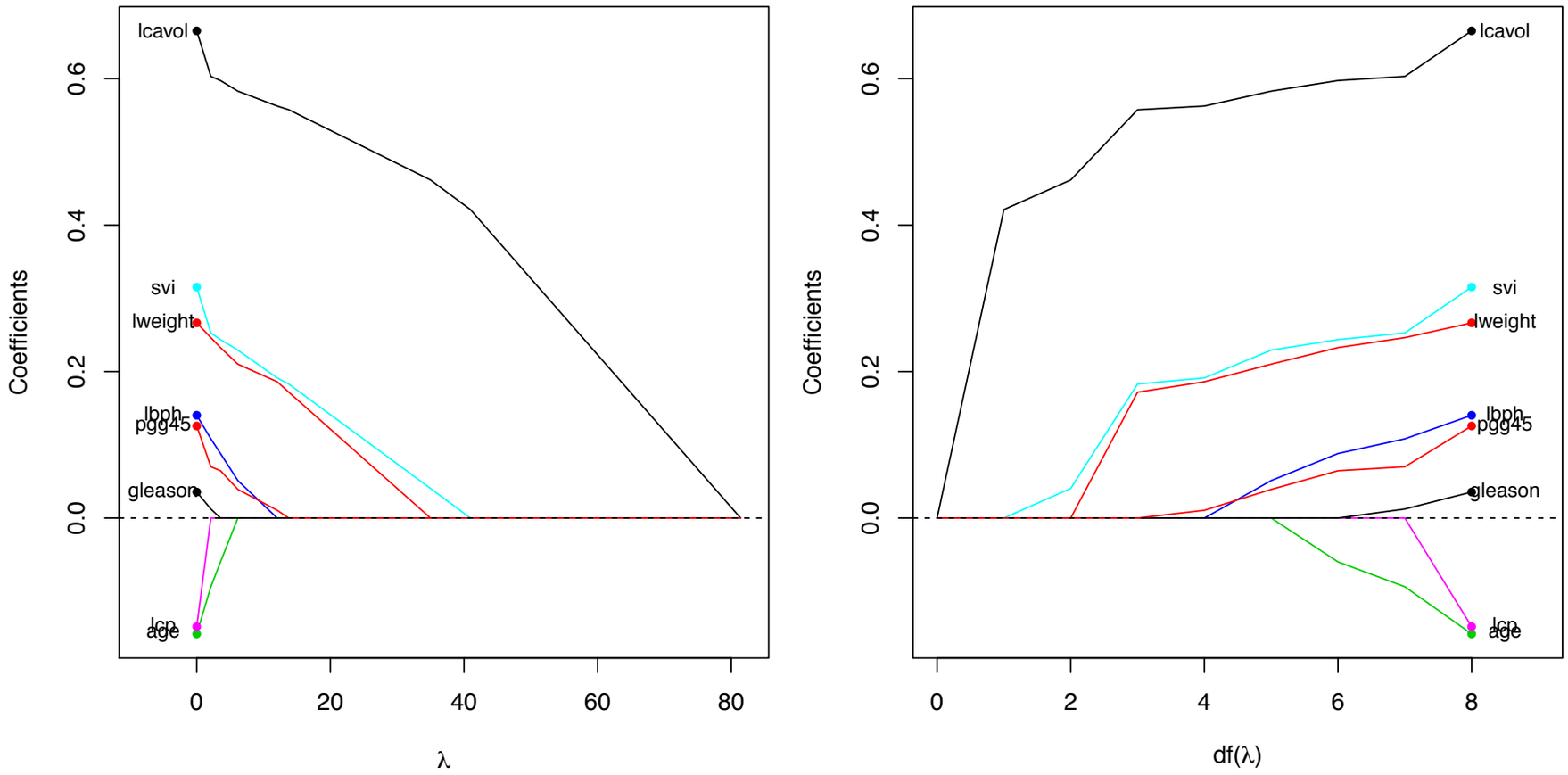
We perform ridge regression over a wide range of λ values (after centering and scaling). The resulting coefficient profiles:





This doesn't give us a clear answer to our question ...

Now the lasso coefficient paths:



We might report the first 3 coefficients to enter the model: lcavol (the log cancer volume), svi (seminal vesicle invasion), and lweight (the log prostate weight)

Critical issue: how much penalization to employ?

Both standard cross-validation and the "1SE rule" select df ~ 3

Broadly speaking, the degrees of freedom of an estimate describes its effective number of parameters

More precisely, given data $y \in \mathbb{R}^n$ from the model

$$y_i = \mu_i + \epsilon_i, \quad i = 1, \ldots n$$

where $\mathrm{E}[\epsilon_i] = 0$, $\mathrm{Var}(\epsilon_i) = \sigma^2$, $\mathrm{Cov}(\epsilon_i, \epsilon_j) = 0$, suppose that we estimate y by $\hat{y}$. The degrees of freedom of the estimate $\hat{y}$ is

$$\mathrm{df}(\hat{y}) = \frac{1}{\sigma^2} \sum_{i=1}^{n} \mathrm{Cov}(\hat{y}_i, y_i)$$

Higher correlation between fitted value and data the more adaptive the estimate and so higher the degrees of freedom

Let $X \in \mathbb{R}^{n \times p}$ be a fixed matrix of predictors[3]

- For linear regression, $\hat{y} = X\hat{\beta}^{\mathrm{linear}}$, we have $\mathrm{df}(\hat{y}) = p$
- For ridge regression, $\hat{y} = X\hat{\beta}^{\mathrm{ridge}}$, where

$$\hat{\beta}^{\mathrm{ridge}} = \underset{\beta \in \mathbb{R}^p}{\mathrm{argmin}} \ \|y - X\beta\|_2^2 + \lambda \|\beta\|_2^2$$

  we have $\mathrm{df}(\hat{y}) = \mathrm{trace}\Big(X(X^T X + \lambda I)^{-1} X^T\Big)$

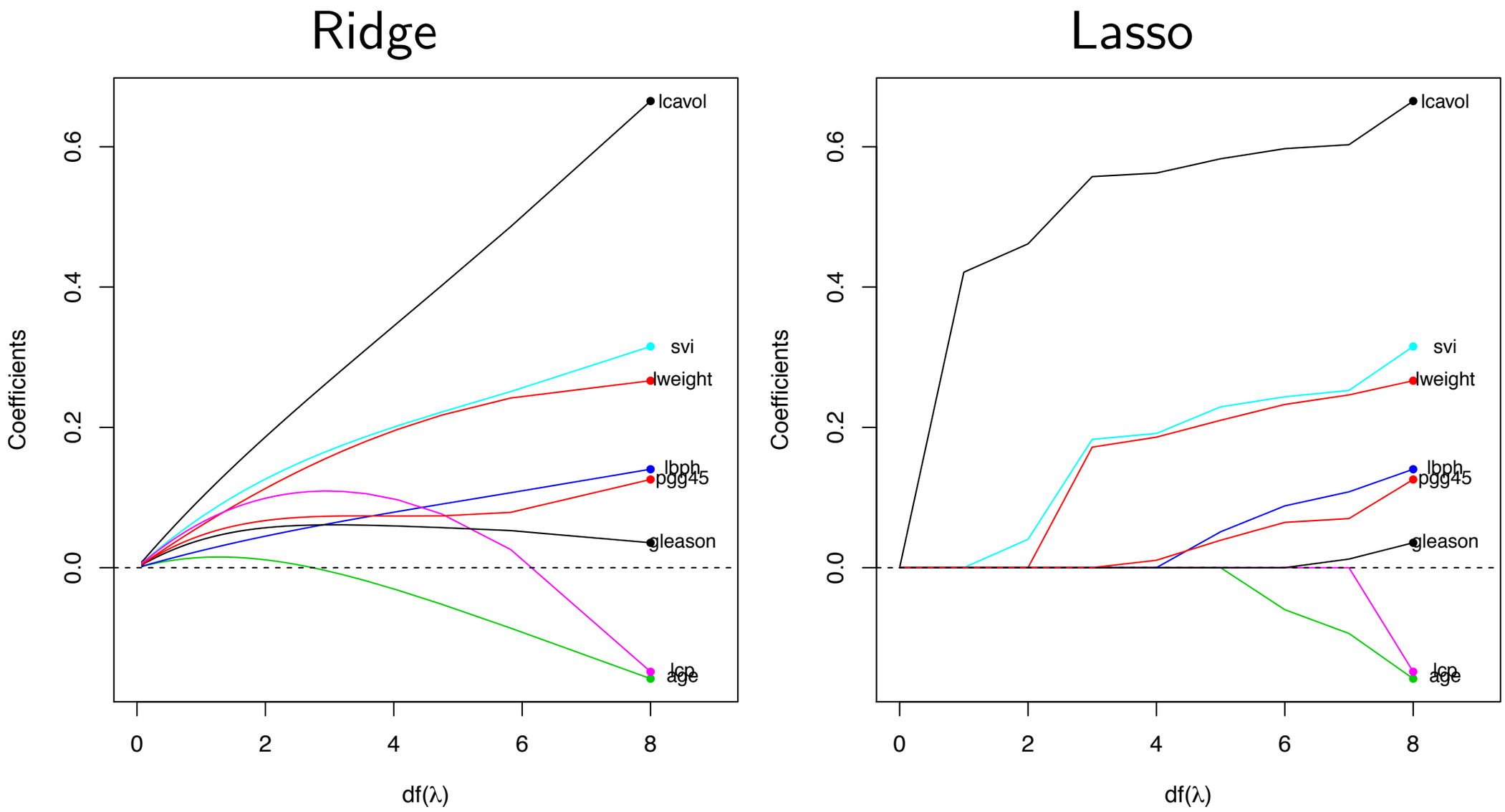
- For the lasso, $\hat{y} = X\hat{\beta}^{\mathrm{lasso}}$, where

$$\hat{\beta}^{\mathrm{lasso}} = \underset{\beta \in \mathbb{R}^p}{\mathrm{argmin}} \ \|y - X\beta\|_2^2 + \lambda \|\beta\|_1$$

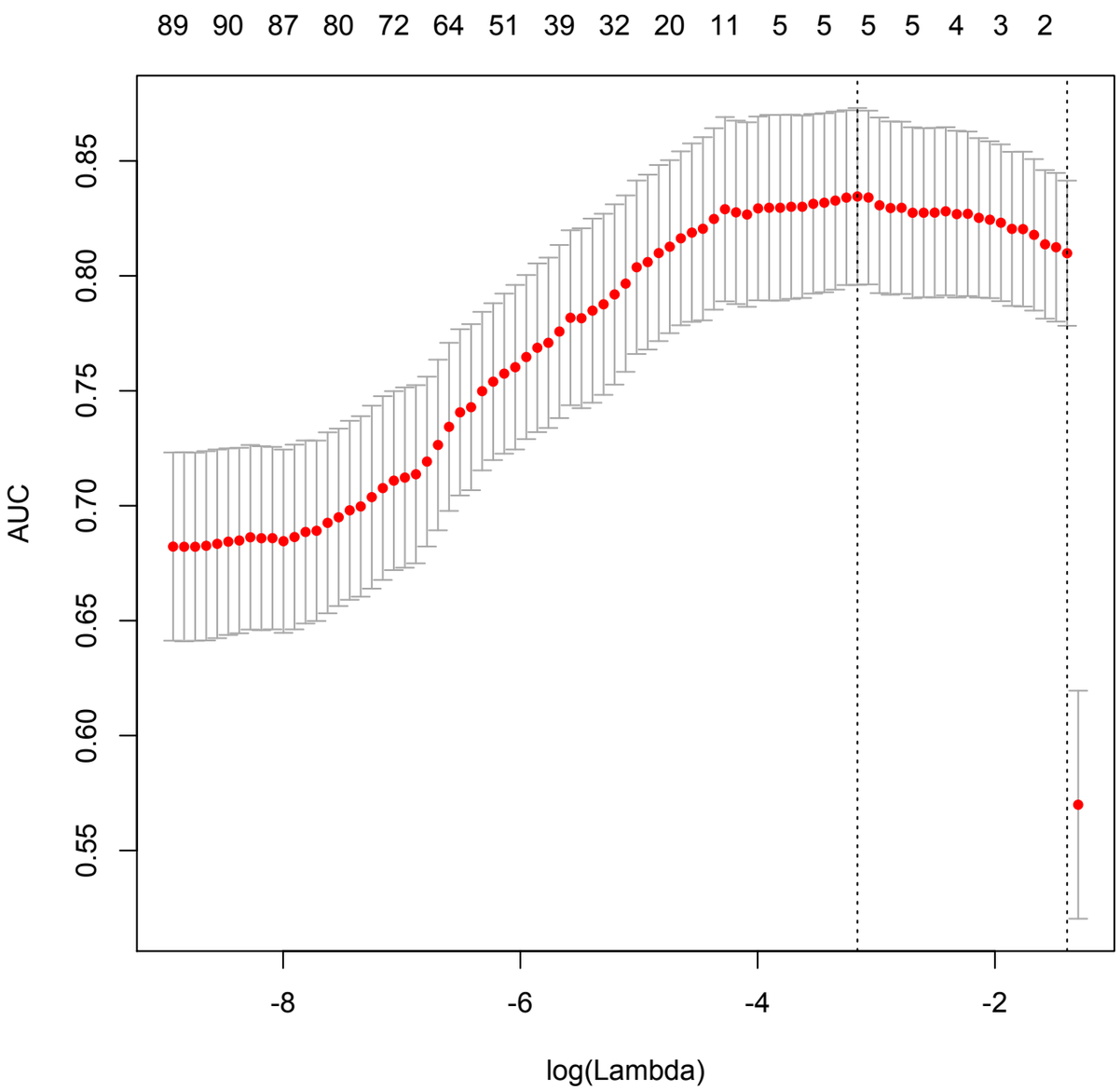
  we have $\mathrm{df}(\hat{y}) = \mathrm{E}[\text{number of nonzero coefficients in } \hat{\beta}^{\mathrm{lasso}}]$

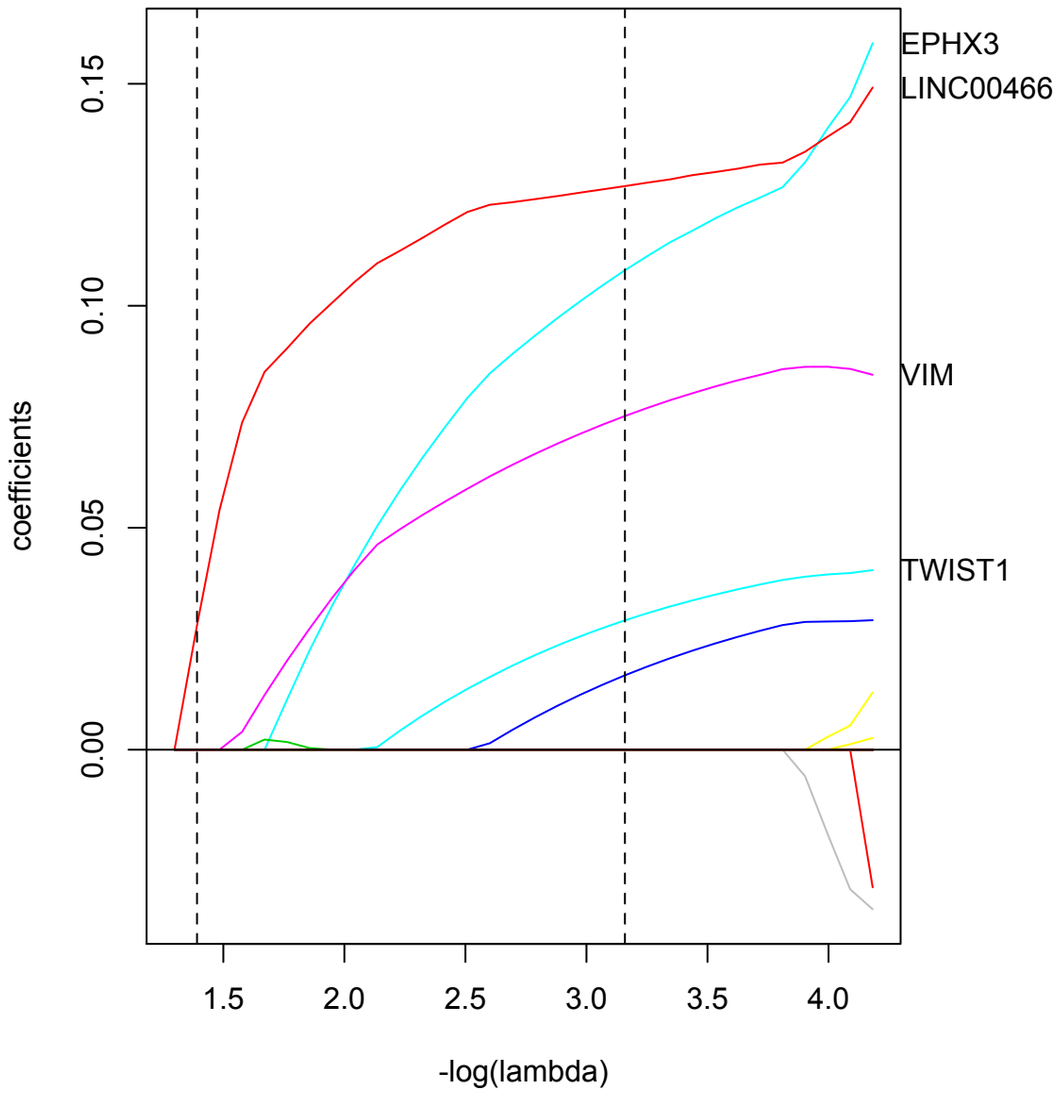
One usage of degrees of freedom is to put two different estimates on equal footing

E.g., comparing ridge and lasso for the prostate cancer data set



- glmnet handles a variety of outcome types
  - penalized logistic regression for binary outcomes
  - penalized Cox regression for survival outcomes
- achieved by generalizing RSS loss function





The Elastic Net penalty

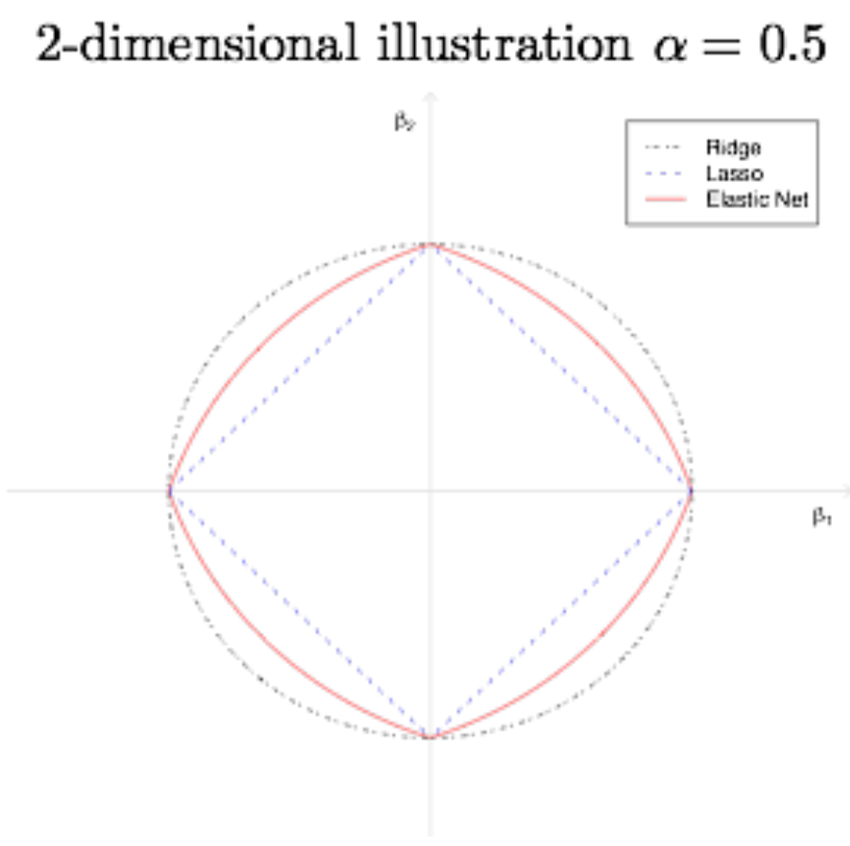
$$P_\lambda(\boldsymbol{\beta}) = \lambda_1 \|\boldsymbol{\beta}\|_1 + \frac{\lambda_2}{2} \|\boldsymbol{\beta}\|_2^2,$$

which consists of two terms, a lasso term plus a ridge term

- Because the ridge penalty is strictly convex, the elastic net solution $\widehat{\boldsymbol{\beta}}$ is unique provided that $\lambda_2 > 0$
- A common reparameterization of the elastic net is to express the regularization parameters in terms of λ, which controls the overall degree of regularization, and α, which controls the balance between the lasso and ridge penalties:

$$\lambda_1 = \alpha\lambda$$
$$\lambda_2 = (1-\alpha)\lambda$$

- This reparameterization is useful in practice, as it allows one to fix α and then select a single tuning parameter λ, which is more straightforward than attempting to select λ_1 and λ_2 separately



The Grouping Effect Property

- The property states that highly correlated features will have similar estimated coefficients, which seems intuitively reasonable

- The shrinkage and grouping effects produced by the elastic net are an effective way of dealing with these correlated predictors

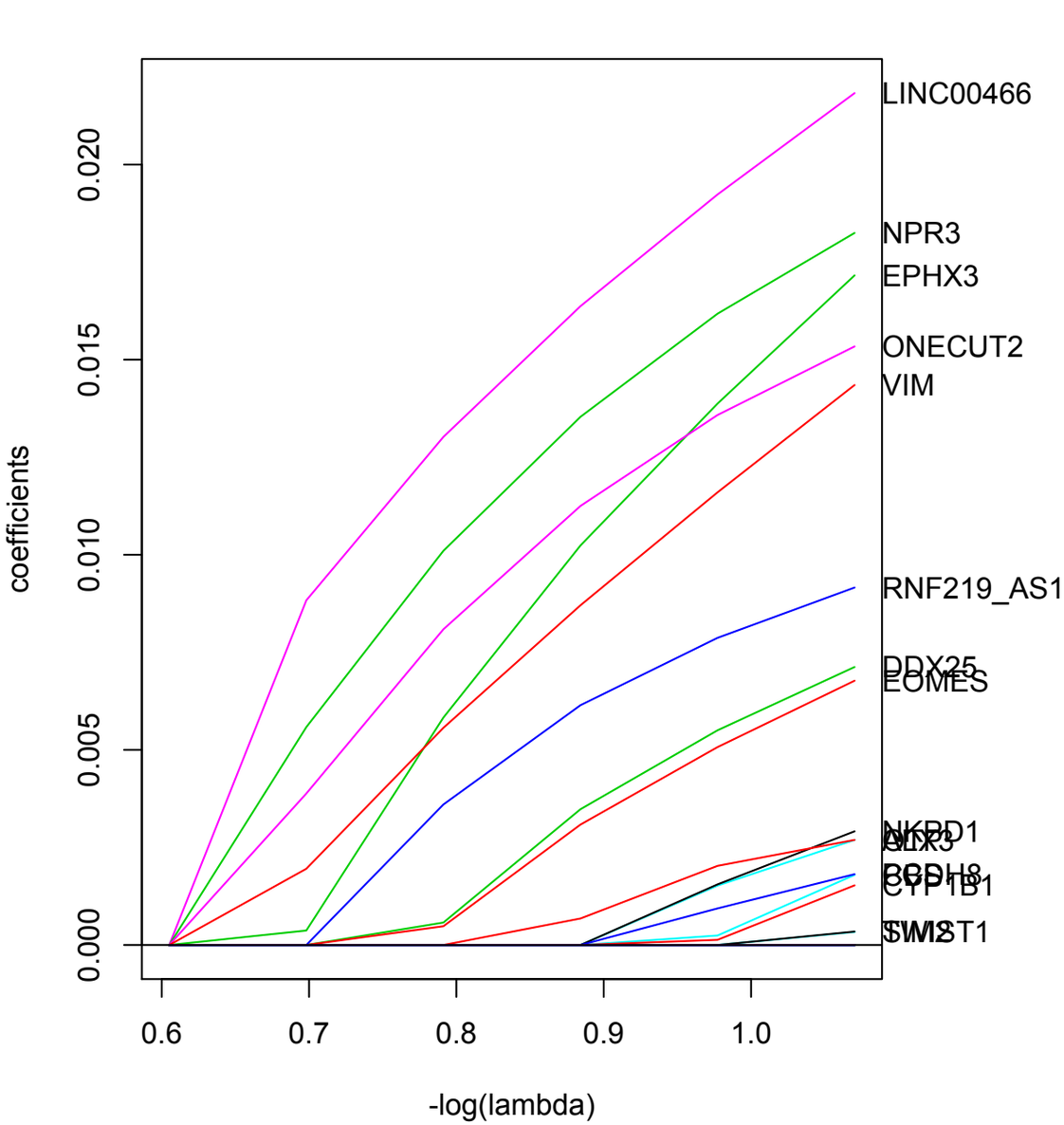
- The property can be described formally in terms of an upper bound on the difference between two coefficients as it relates to the correlation between the predictors:

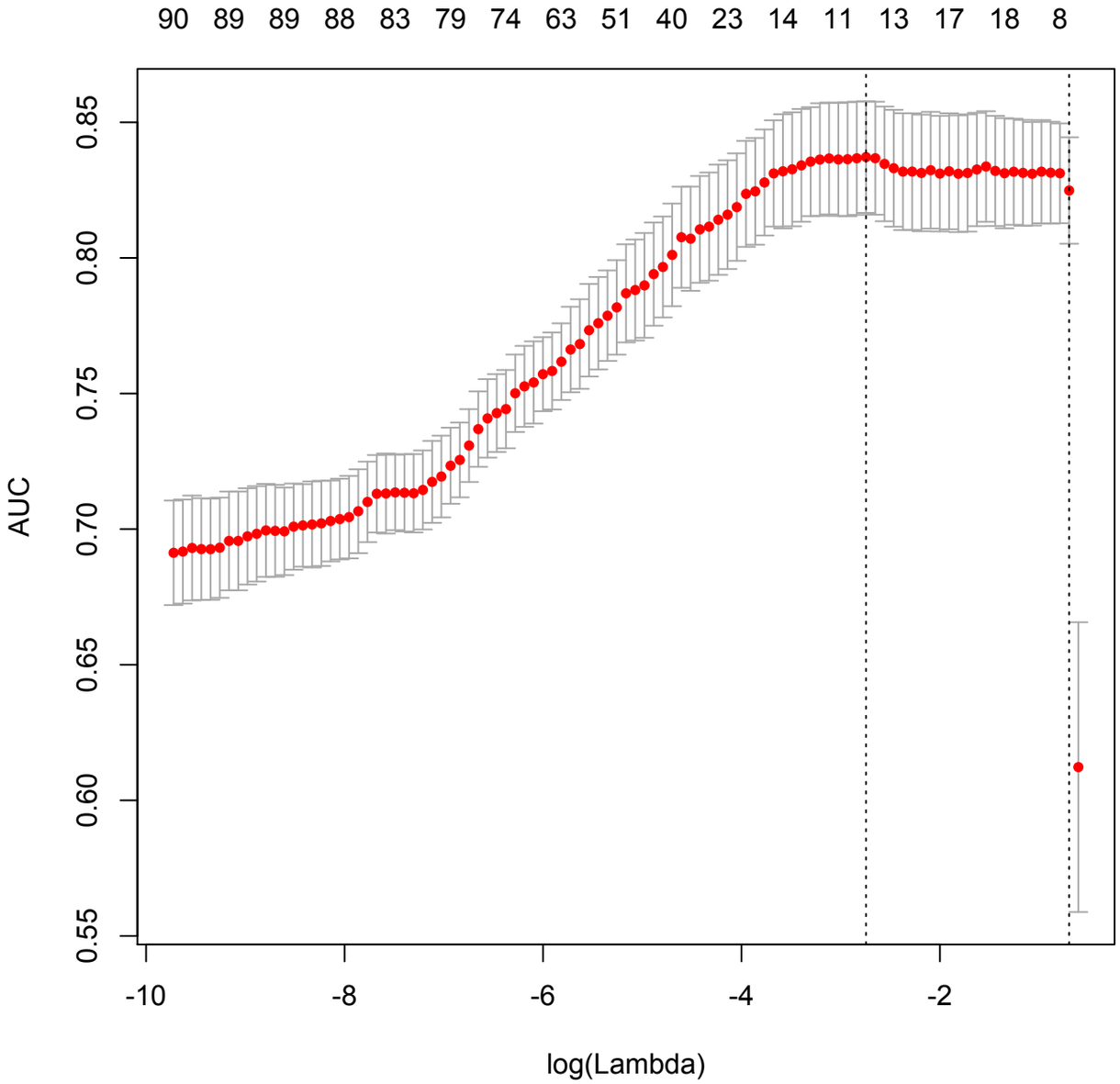
$$|\widehat{\beta}_j - \widehat{\beta}_k| \leq \frac{\|\mathbf{y}\|\sqrt{2(1-\rho_{jk})}}{\lambda_2\sqrt{n}}$$

  where ρ_{jk} is the sample correlation between $\mathbf{x}_j$ and $\mathbf{x}_k$

- Note, in particular, that as $\rho_{jk} \to 1$, the difference between $\widehat{\beta}_j$ and $\widehat{\beta}_k$ goes to zero

- In practice, the grouping effect is often one of the strongest motivations for applying an elastic net penalty
- For example, in gene expression studies, genes that have similar functions, or that work together in a pathway to accomplish a certain function, are often correlated
- It is often reasonable to assume, then, that if the function is relevant to the response we are analyzing, the coefficients will be similar across the correlated group



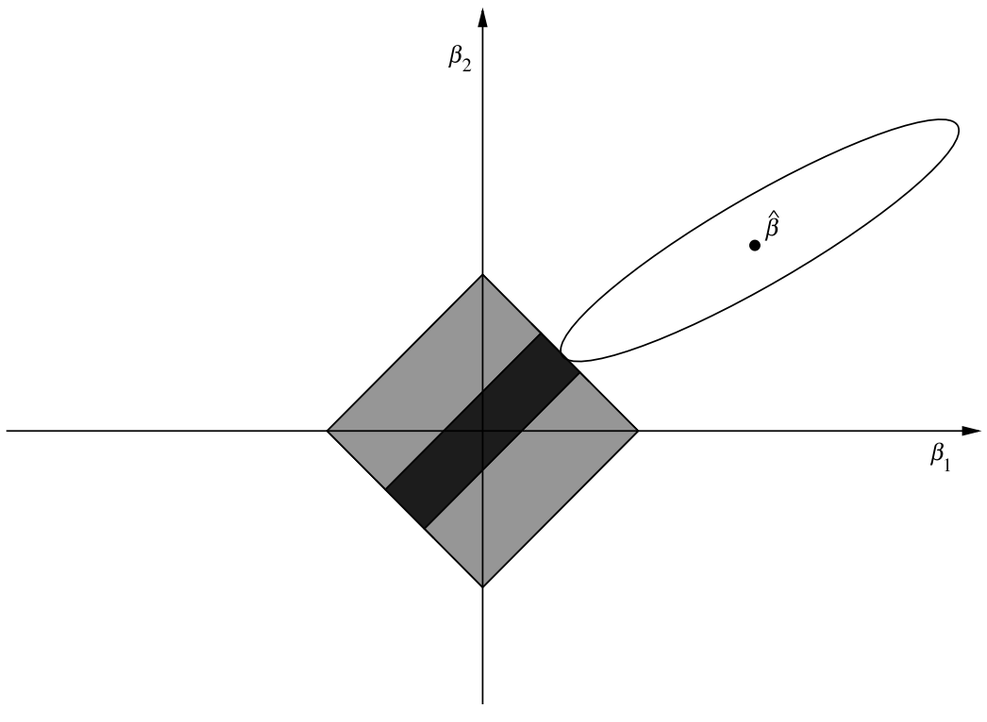


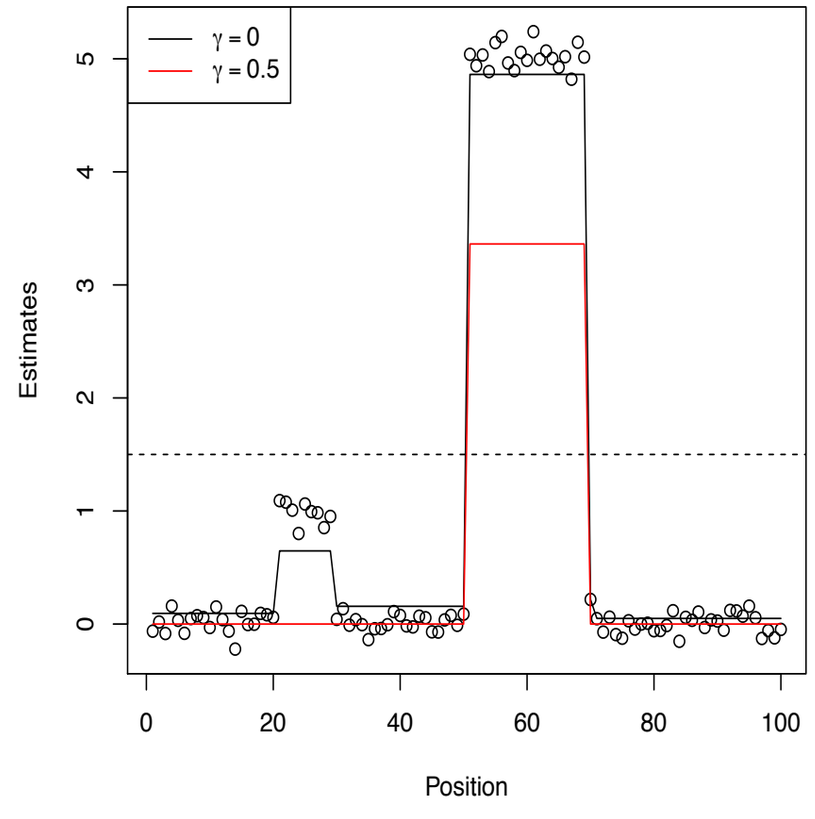
Refinements and Extensions

Fused lasso

$$\hat{\beta} = \underset{\beta \in \mathbb{R}^n}{\operatorname{argmin}} \ \frac{1}{2} \sum_{i=1}^{n} (y_i - \beta_i)^2 + \lambda \sum_{i=1}^{n-1} |\beta_{i+1} - \beta_i|$$

$$\hat{\beta} = \underset{\beta \in \mathbb{R}^n}{\operatorname{argmin}} \ \frac{1}{2} \sum_{i=1}^{n} (y_i - \beta_i)^2 + \lambda \sum_{(i,j) \in E} |\beta_i - \beta_j| + \gamma \cdot \lambda \sum_{i=1}^{n} |\beta_i|$$





Group lasso

- In many regression problems predictors are not distinct but arise from common underlying factors
- Obvious example: representing a categorical factor by a group of indicator functions
- We denote $\mathbf{X}$ as being composed of J groups $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_J$, with K_j denoting the size of group j; i.e., $\sum_j K_j = p$
- As usual, we are interested in estimating a vector of coefficients $\boldsymbol{\beta}$ using a loss function L which quantifies the discrepancy between the observations $\mathbf{y}$ and the linear predictors $\boldsymbol{\eta} = \mathbf{X}\boldsymbol{\beta} = \sum_j \mathbf{X}_j \boldsymbol{\beta}_j$, where $\boldsymbol{\beta}_j$ represents the coefficients belonging to the jth group
- Covariates that do not belong to a group may be thought of as a group of one

Group lasso

$$\mathbf{Q}(\boldsymbol{\beta}|\mathbf{X},\mathbf{y}) = \mathbf{L}(\boldsymbol{\beta}|\mathbf{X},\mathbf{y}) + \sum_j \lambda_j \left\|\boldsymbol{\beta}_j\right\|$$

- Instead of penalizing magnitude of individual coefficients penalize the magnitude of a group of coefficients
- To ensure that the same degree of penalization is applied to large and small groups set $\lambda_j = \lambda\sqrt{K_j}$