Statistical Learning

Lecture 06a

ANU - RSFAS

Last Updated: Mon Mar 21 14:54:14 2022

Linear Model Selection and Regularization

- Recall our famous (and friendly) linear model

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

- In the lectures that follow, we consider some approaches for extending the linear model framework. In the lectures covering Chapter 7 of the text, we generalize the linear model in order to accommodate non-linear, but still additive, relationships.

- In the lectures covering Chapter 8 we consider even more general non-linear models.

In Praise of Linear Models!

- Despite its simplicity, the linear model has distinct advantages in terms of its interpretability and often shows good predictive performance.
- Hence we discuss in this lecture some ways in which the simple linear model can be improved, by replacing ordinary least squares fitting with some alternative fitting procedures.

Why consider alternatives to least squares?

- Prediction Accuracy: especially when $p > n$, to control the variance.
- Model Interpretability: By removing irrelevant features — that is, by setting the corresponding coefficient estimates to zero — we can obtain a model that is more easily interpreted.
- We will present some approaches for automatically performing feature/covariate selection.

Three Classes of Methods

- Subset Selection: We identify a subset of the p predictors that we believe to be related to the response. We then fit a model using least squares on the reduced set of variables. **This material was discussed in your previous regression course see Chapter 6 of ISL for a review.**

- Shrinkage: We fit a model involving all p predictors, but the estimated coefficients are shrunken towards zero relative to the least squares estimates. This shrinkage (also known as regularization) has the effect of reducing variance and can also perform variable selection.

- Dimension Reduction: We project the p predictors into a M-dimensional subspace, where $M < p$. This is achieved by computing M different linear combinations, or projections, of the variables. Then these M projections are used as predictors to fit a linear regression model by least squares.

Shrinkage Methods

- Ridge regression and Lasso
- The subset selection methods use least squares to fit a linear model that contains a subset of the predictors.
- As an alternative, we can fit a model containing all p predictors using a technique that constrains or regularizes the coefficient estimates.
  - Or equivalently, that shrinks the coefficient estimates towards zero.
- Shrinking the coefficient estimates can significantly reduce their variance.

Ridge regression

- The ridge regression minimizes:

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

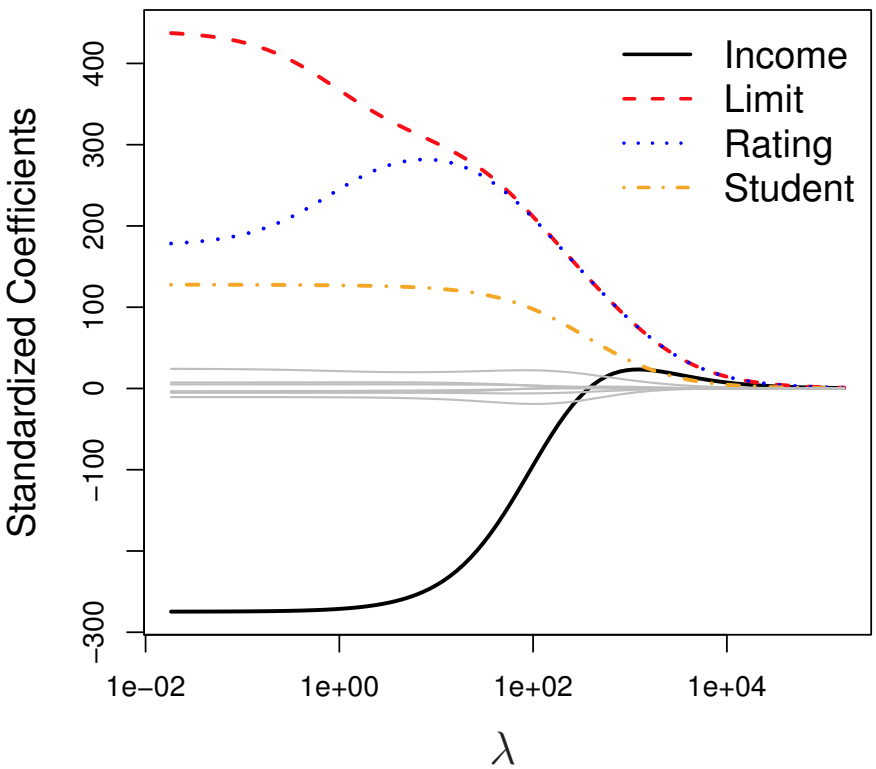
- Where $\lambda \geq 0$ is a tuning parameter, to be determined separately.
- We have standard regression plus a penalty.

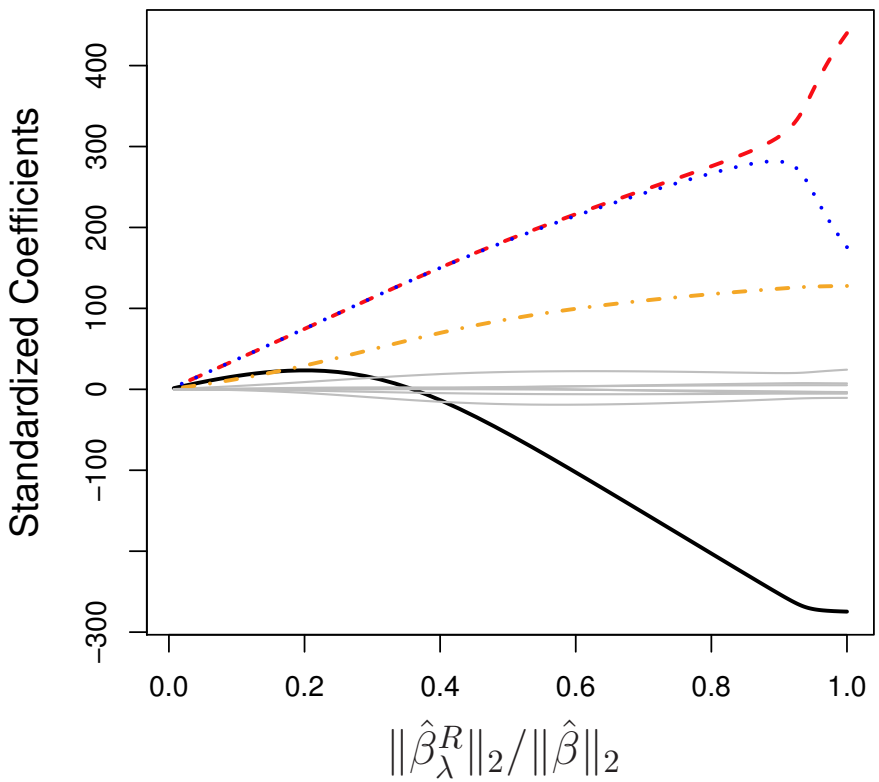
Ridge regression

- As with standard regression fit via least squares, ridge regression seeks coefficient estimates that fit the data well, by making the RSS small.
- The penalty, $\lambda \sum_{j=1}^{p} \beta_j^2$, is small when $\beta_1, \ldots, \beta_p$ are close to zero.
  - So it shrinks the estimates of the βs towards zero!
- The tuning parameter λ serves to control the relative impact of these two terms on the regression coefficient estimates.
- Selecting a good value for λ is critical; cross-validation is used to do this.

Credit Balance Data

- Recall: The response Y is the amount left on the credit card after a monthly payment.





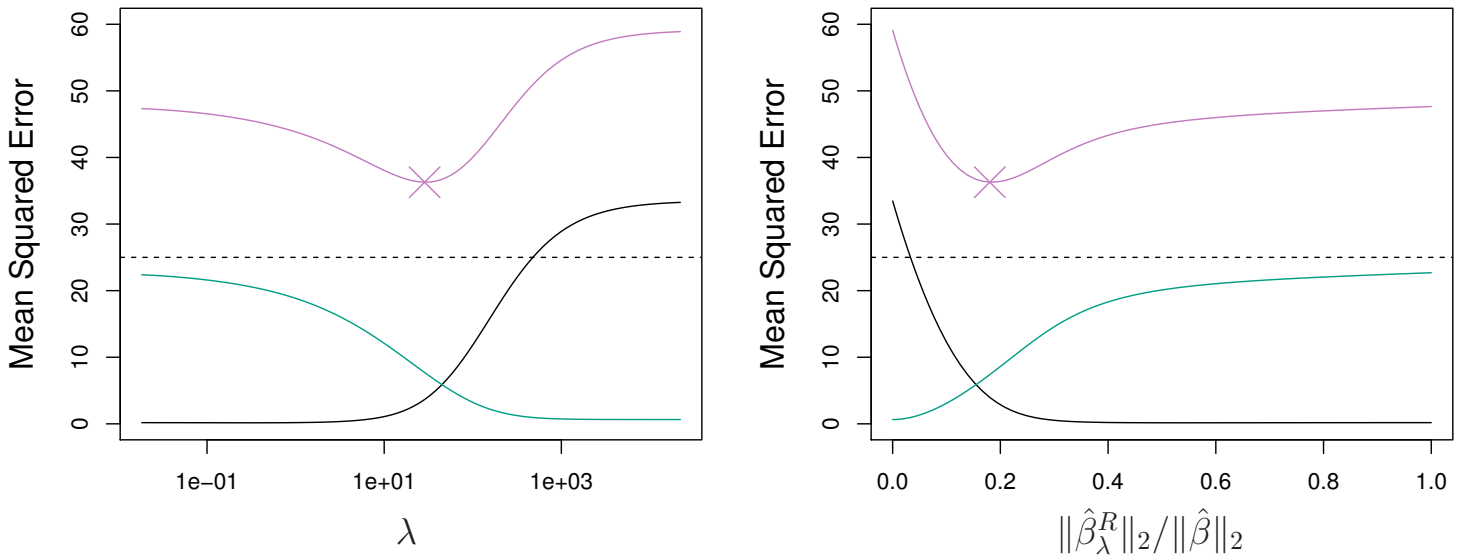
- Each curve corresponds to the ridge regression coefficient estimate for one of the ten variables, plotted as a function of λ.
- $\hat{\beta}_\lambda^R$ are the ridge regression estimates.
- $\hat{\beta}$ are the standard regression estimates (fit via least squares).
- $||\beta||_2$ denotes the ℓ_2-norm:

$$||\beta||_2 = \sqrt{\sum_{j=1}^{p} \beta^2}$$

- Standard regression coefficients are scale equivariant.
  - multiplying X_j by a constant c simply leads to a scaling of the least squares coefficient estimates by a factor of $1/c$.
- In contrast, the ridge regression coefficient estimates can change substantially when multiplying a given predictor by a constant.
  - Due to the sum of squared coefficients term in the penalty part of the ridge regression objective function.
- Therefore, it is best to apply ridge regression after standardizing the predictors, using the formula:

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

Why Does Ridge Regression Improve Over Least Squares?



- Simulated data with $n = 50$ observations, $p = 45$ predictors, all having nonzero coefficients.
- Squared bias (black), variance (green), and test mean squared error (purple) for the ridge regression predictions.

The Lasso

- Ridge regression does have one obvious disadvantage:
  - Unlike best subset selection, ridge regression will include all p predictors in the final model.
- The Lasso overcomes this difficulty.
- The lasso coefficients, $\hat{\beta}^L_\lambda$, minimize the quantity:

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

- Instead of basing the penalty on an ℓ_2 norm (without the square-root) we have an ℓ_1 penalty:

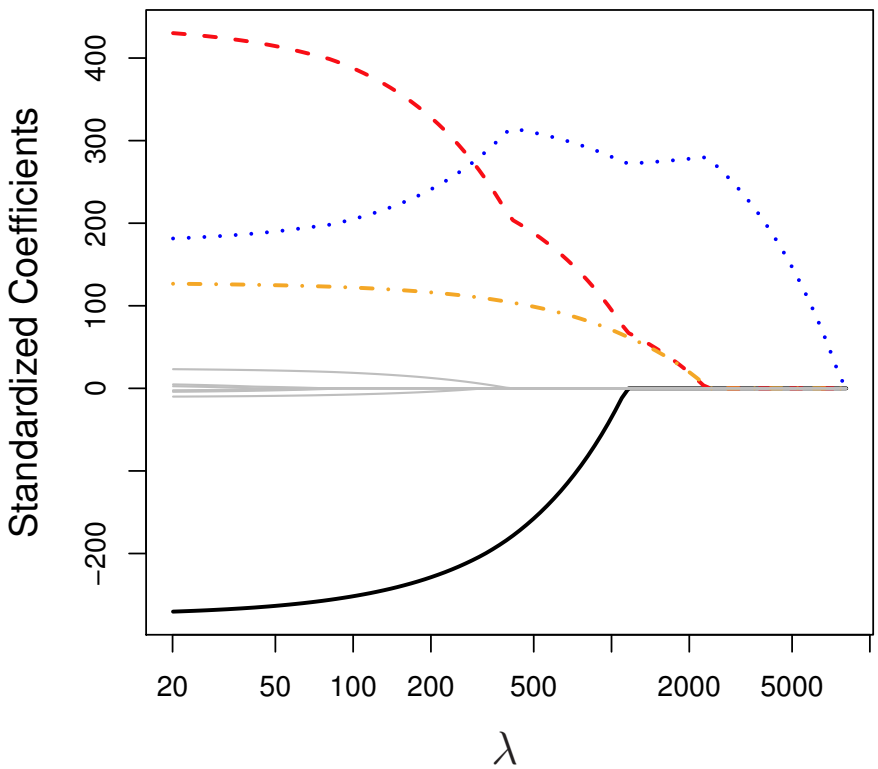
$$||\beta||_1 = \sum_{j=1}^{p}|\beta_j|$$

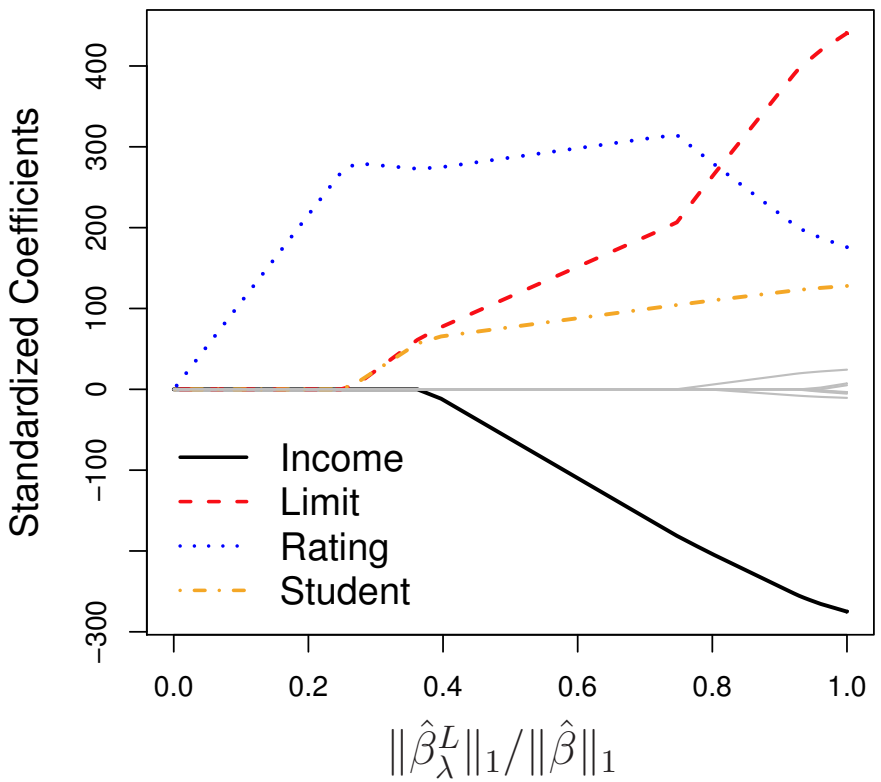
The Lasso

- As with ridge regression, the lasso shrinks the coefficient estimates towards zero.
- The ℓ_1 penalty has the effect of forcing some of the coefficient estimates to be exactly equal to zero, when the tuning parameter is sufficiently large.
- Thus, the lasso performs variable selection.
- We say that the lasso yields sparse models — that is, models that involve only a subset of the variables.
- As in ridge regression, selecting a good value of λ for the lasso is critical; cross-validation is again the method of choice.

Credit Balance Data - Lasso

- Recall: The response Y is the amount left on the credit card after a monthly payment.





The Variable Selection Property of the Lasso

- Why is it that the lasso, unlike ridge regression, results in coefficient estimates that are exactly equal to zero?
- Ridge:

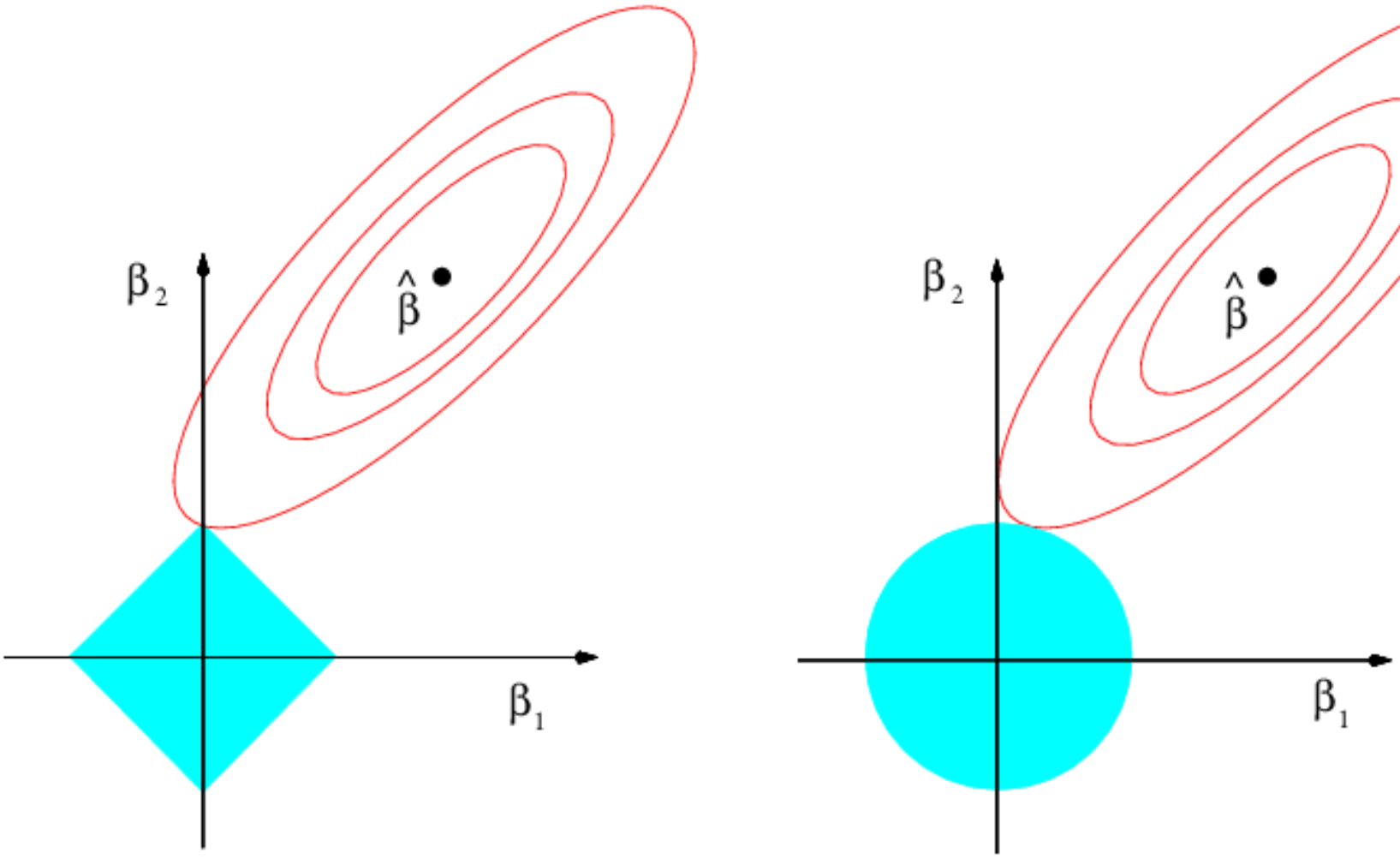
$$\text{minimize}_\beta \ \mathit{RSS} \text{ such that } \sum_{j=1}^{p} \beta_j^2 \leq s$$

- Lasso

$$\text{minimize}_\beta \ \mathit{RSS} \text{ such that } \sum_{j=1}^{p} \left|\beta_j\right| \leq s$$

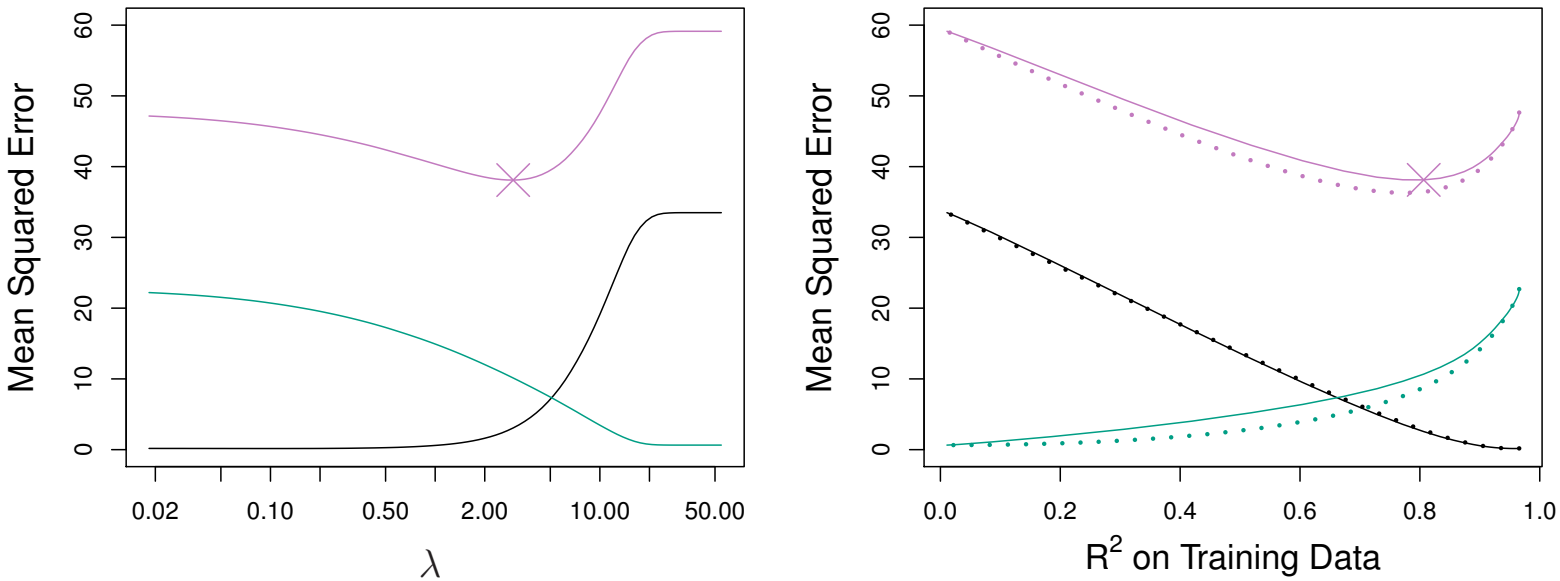
- Best subset

$$\text{minimize}_\beta \ \mathit{RSS} \text{ such that } \sum_{j=1}^{p} \mathbb{I}(\beta_j \neq 0) \leq s$$



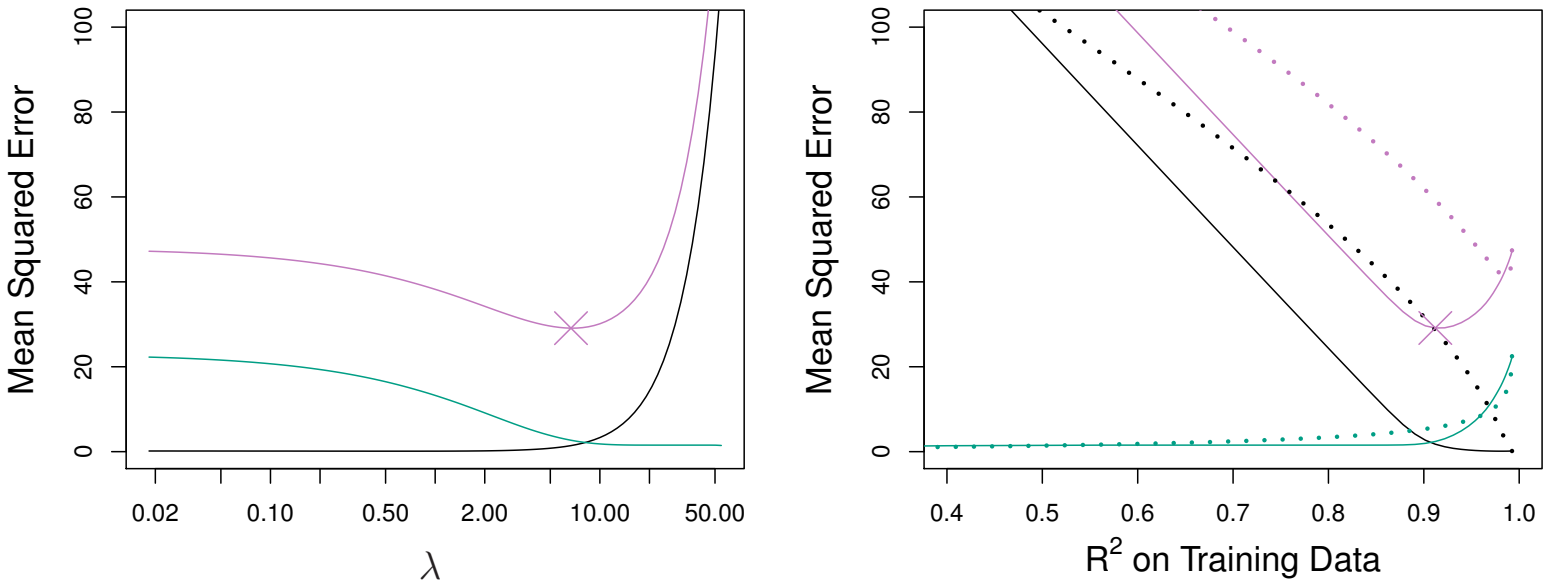
- Left is lasso, right is ridge.

Comparing the Lasso and Ridge Regression



- Simulated data with $n = 50$ observations, $p = 45$ predictors, all having nonzero coefficients.
- Squared bias (black), variance (green), and test MSE (purple)
- Left: Lasso
- Right: Lasso (solid) and ridge (dashed)

Comparing the Lasso and Ridge Regression



- Simulated data with $n = 50$ observations, $p = 45$ predictors, only 2 having nonzero coefficients.
- Squared bias (black), variance (green), and test MSE (purple)
- Left: Lasso
- Right: Lasso (solid) and ridge (dashed)

Thoughts

- These two examples illustrate that neither ridge regression nor the lasso will universally dominate the other.
- One might expect the lasso to perform better when the response is a function of only a relatively small number of predictors.
- The number of predictors that is related to the response is never known a priori for real data sets.
- A technique such as cross-validation can be used in order to determine which approach is better on a particular data set.

Example - Diabetes Data

- We are interested in modeling a measure Y of disease progression taken one year after baseline measurements.
- We have 10 baseline measurements: *age*, *sex*, *bmi*, *map*, *tc*, *ldl*, *hdl*, *tch*, *ltg*, *glu*.
- We are interested in potential non-linearities and interactions.
  - We include X_j^2 terms (except for gender) — 9 additional covariates.
  - And $X_i X_j$ interaction terms — $\binom{10}{2} = 45$ additional covariates.
  - Total of $10 + 9 + 45 = 64$ covariates
- All of the variables have been centered and scaled so that Y and the columns of X all have mean zero and variance one.
- To evaluate the models, we will randomly divide the 442 diabetes subjects into 342 training samples and 100 test samples

References

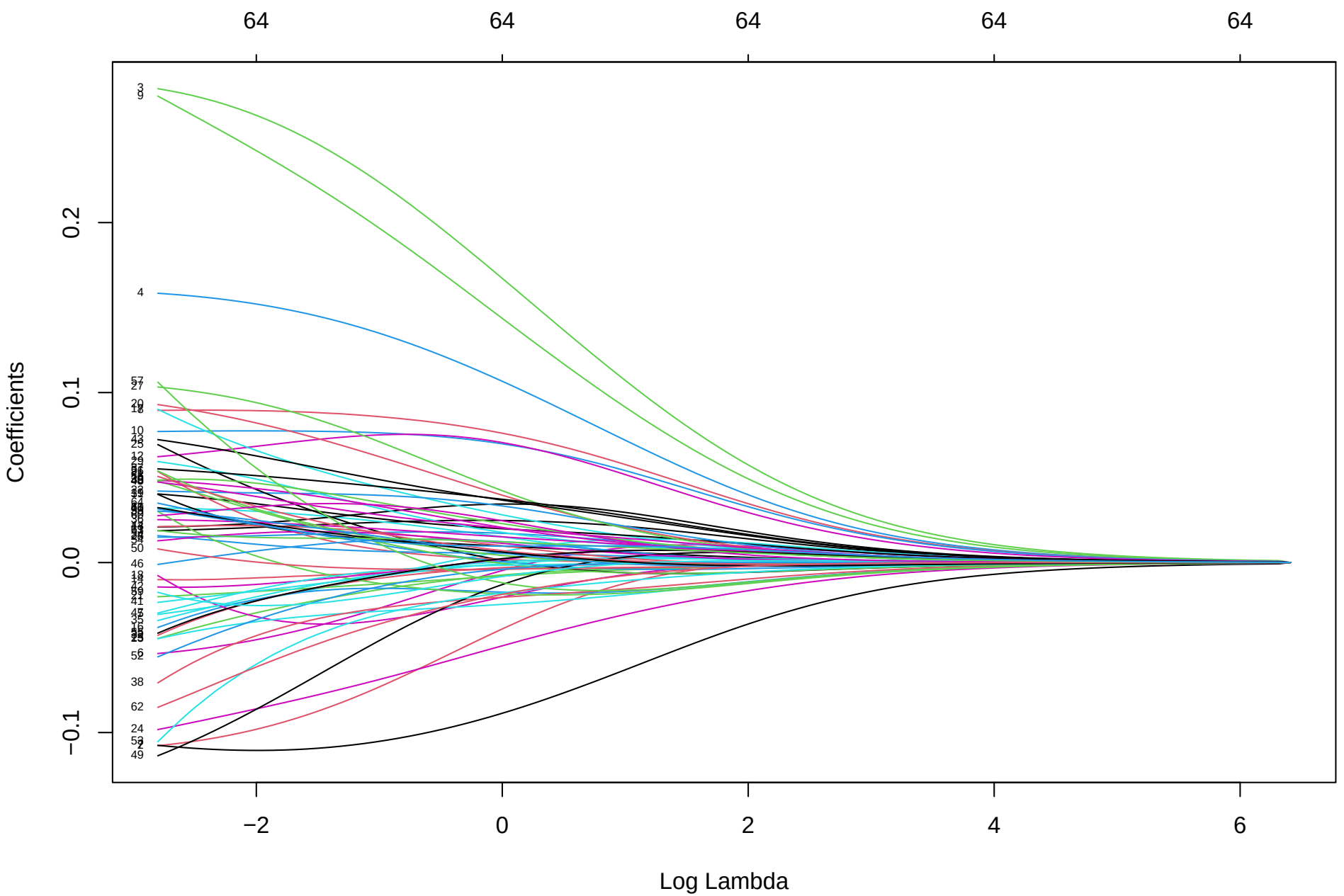
- Efron, Hastie, Johnstone and Tibshirani (2003) "Least Angle Regression" (with discussion) Annals of Statistics.
- Also discussed in Hoff (2009) *A First Course in Bayesian Statistical Methods* — Chapter 9.
- We will ignore model hierarchy here.

Ridge Regression

```
diabTrain <- read.table("diabTrain.dat", header=T)
diabTest <- read.table("diabTest.dat", header=T)

X <- as.matrix(diabTrain[,-1])
y <- diabTrain[,1]

X.test <- as.matrix(diabTest[,-1])
y.test <- diabTest[,1]
```

```
library(glmnet)
mod.ridge <- glmnet(X, y, alpha=0, standardize=FALSE)
```

```
plot(mod.ridge, label=TRUE, xvar="lambda")
```



- For each λ we get a set of coefficients.
- We have $p + 1 = 65$ regression coefficients.
- The algorithm was minimized for 100 different λs.
- You can set the λs if you wish.

```
dim(coef(mod.ridge))
```

```
## [1]  65 100
```

```
mod.ridge$lambda[1]
```

```
## [1] 607.572
```

```
mod.ridge$lambda[100]
```

```
## [1] 0.0607572
```

- We can examine the coefficients for a particular λ.

```
mod.ridge$lambda[50]
```

```
## [1] 6.36502
```

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

```
##   (Intercept)           age           sex           bmi           map
## -6.428003e-03  1.513519e-02 -1.975724e-03  6.376687e-02  4.411090e-02
##            tc           ldl           hdl           tch           ltg
##  1.218477e-02  1.001135e-02 -3.984414e-02  3.796454e-02  5.462064e-02
##           glu         age.2         bmi.2         map.2          tc.2
##  3.597612e-02  1.667551e-03  3.245273e-02  1.984815e-02 -2.111745e-03
##         ldl.2         hdl.2         tch.2         ltg.2         glu.2
## -2.133831e-03 -1.225808e-02  8.662132e-03 -1.461765e-03  1.165417e-02
##       age.sex       age.bmi       age.map        age.tc       age.ldl
##  1.031629e-02 -2.426400e-03  1.176853e-02 -1.288331e-02 -1.831369e-02
##       age.hdl       age.tch       age.ltg       age.glu       sex.bmi
## -6.049891e-03 -4.945737e-06  9.362095e-03  7.396538e-03  6.478675e-03
##       sex.map        sex.tc       sex.ldl       sex.hdl       sex.tch
##  6.769611e-03  3.407140e-03 -7.310675e-04  7.345546e-03  2.401961e-03
##       sex.ltg       sex.glu       bmi.map        bmi.tc       bmi.ldl
##  2.524373e-03  7.238120e-03  1.811912e-02 -1.054121e-02 -1.329167e-02
##       bmi.hdl       bmi.tch       bmi.ltg       bmi.glu        map.tc
## -3.728058e-03  2.026214e-03  6.710189e-03  1.733466e-02 -3.759798e-04
##       map.ldl       map.hdl       map.tch       map.ltg       map.glu
## -5.857773e-03  2.154820e-03  1.753094e-04  9.485493e-03  5.905682e-03
##        tc.ldl        tc.hdl        tc.tch        tc.ltg        tc.glu
## -2.300712e-03 -8.556689e-04  1.191614e-05 -6.234375e-03  2.370279e-03
##       ldl.hdl       ldl.tch       ldl.ltg       ldl.glu       hdl.tch
##  6.485318e-03 -7.460630e-04 -1.239994e-02 -2.177316e-03  1.072869e-03
##       hdl.ltg       hdl.glu       tch.ltg       tch.glu       ltg.glu
##  5.323054e-03 -1.750388e-03 -2.699879e-03  6.169974e-03  8.002961e-03
```

- We can use the predict function for a new value of $\lambda = 50$.

```
predict(mod.ridge, s=50, type="coefficients")[1:11,]
```

```
##   (Intercept)           age           sex           bmi           map
## -0.0038320831  0.0032137826  0.0006119594  0.0114221083  0.0081087271
##            tc           ldl           hdl           tch           ltg
##  0.0029448778  0.0026514615 -0.0074990472  0.0074696992  0.0098810461
##           glu
##  0.0071446382
```

- Lets predict on the testing data, for $\lambda = 5$.

```
pred.ridge <- predict(mod.ridge, s=5, newx=X.test)

mse <- mean( (pred.ridge-y.test)^2)
mse
```

```
## [1] 0.6943536
```

- Let's compare to just the mean of Y from the training data! So we are doing better with Ridge Regression.

```
mse <- mean( (mean(y)-y.test)^2)
mse
```

```
## [1] 0.9657373
```

- Let's compare to standard regression. Standard regression does better. Hmmmm . . .

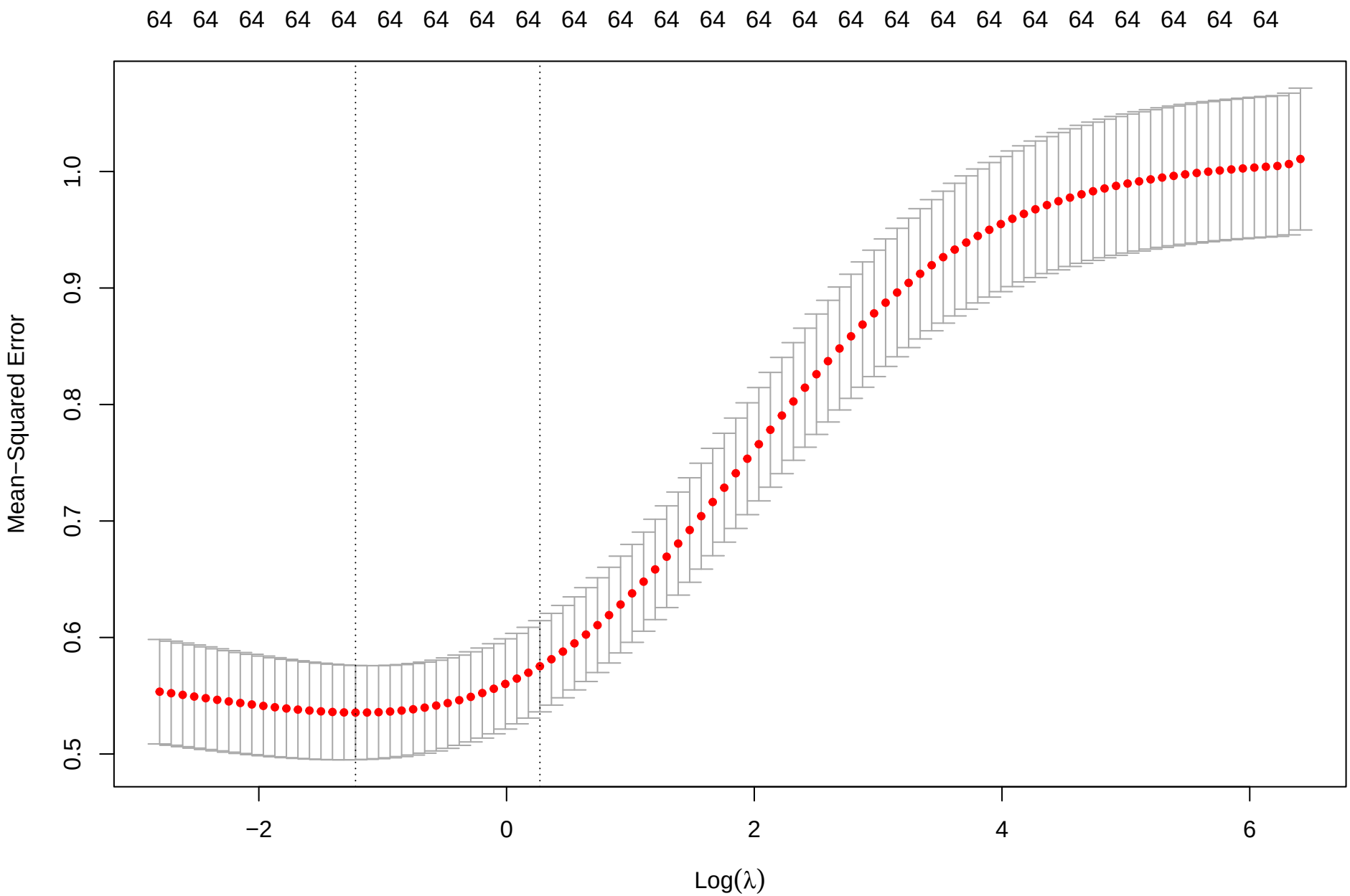
```
train.df <- data.frame(y, X)
test.df <- data.frame(y.test, X.test)
lm.fit <- lm(y ~ ., data=train.df)



pred.lm <- predict(lm.fit,  newdata=test.df[,-1])
mse <- mean( (pred.lm-y.test)^2)
mse
```

```
## [1] 0.672564
```

- Let's use 10-fold cross-validation to determine λ.

```
set.seed(1)
cv.out <- cv.glmnet(X, y, alpha=0, standardize=FALSE)
```

```
plot(cv.out)
```


64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64
Mean−Squared Error
0.5
0.6
0.7
0.8
0.9
1.0
−2
0
2
4
6
Log(λ)

- Let's get the "best" λ.

```
best.lam <- cv.out$lambda.min
best.lam
```

```
## [1] 0.295438
```

```
log(best.lam)
```

```
## [1] -1.219296
```

- Let's get the MSE.

```
pred.ridge <- predict(mod.ridge, s=best.lam, newx=X.test)

mse <- mean( (pred.ridge-y.test)^2)
mse
```

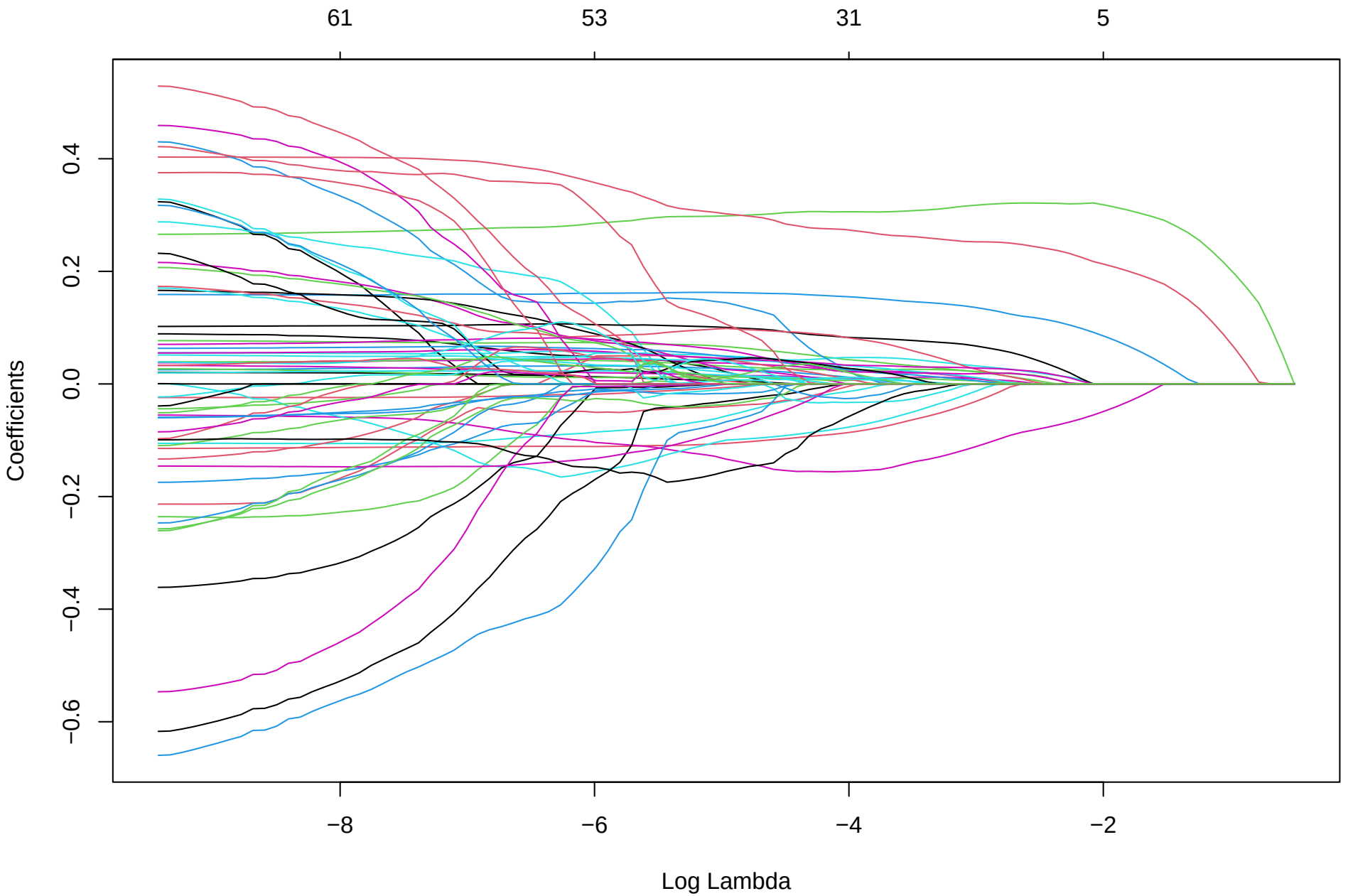
```
## [1] 0.5343424
```

Lasso

- For lasso set $\alpha = 1$.
- All the previous code also works here.

```
mod.lasso <- glmnet(X, y, alpha=1, standardize=FALSE)
```

```
plot(mod.lasso, xvar="lambda")
```



```
set.seed(1)
cv.out <- cv.glmnet(X, y, alpha=1, standardize=FALSE)
best.lam <- cv.out$lambda.min
best.lam
```

```
## [1] 0.04091481
```

```
log(best.lam)
```

```
## [1] -3.196263
```

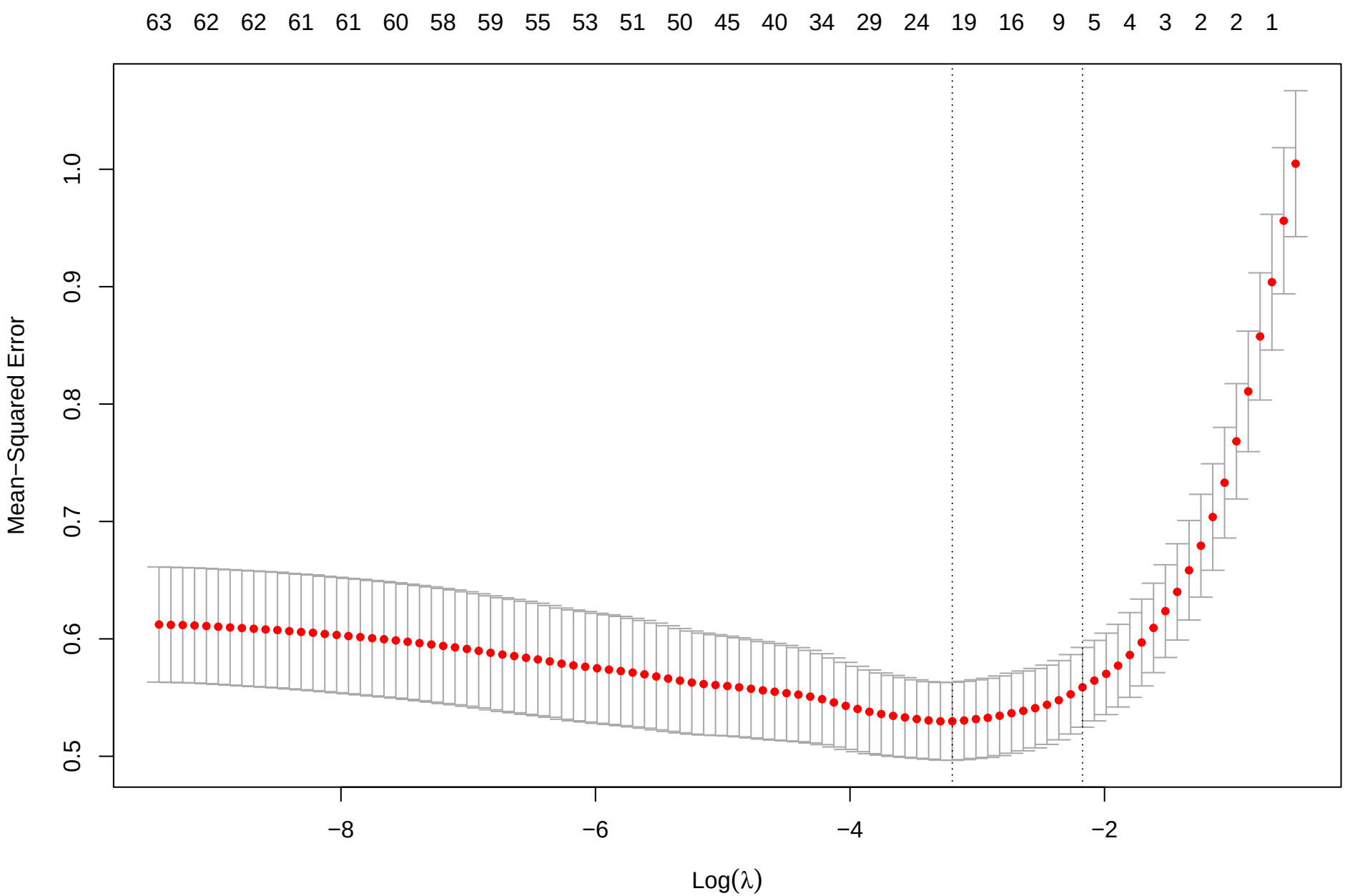
```
lam.1se <- cv.out$lambda.1se
lam.1se
```

```
## [1] 0.1138479
```

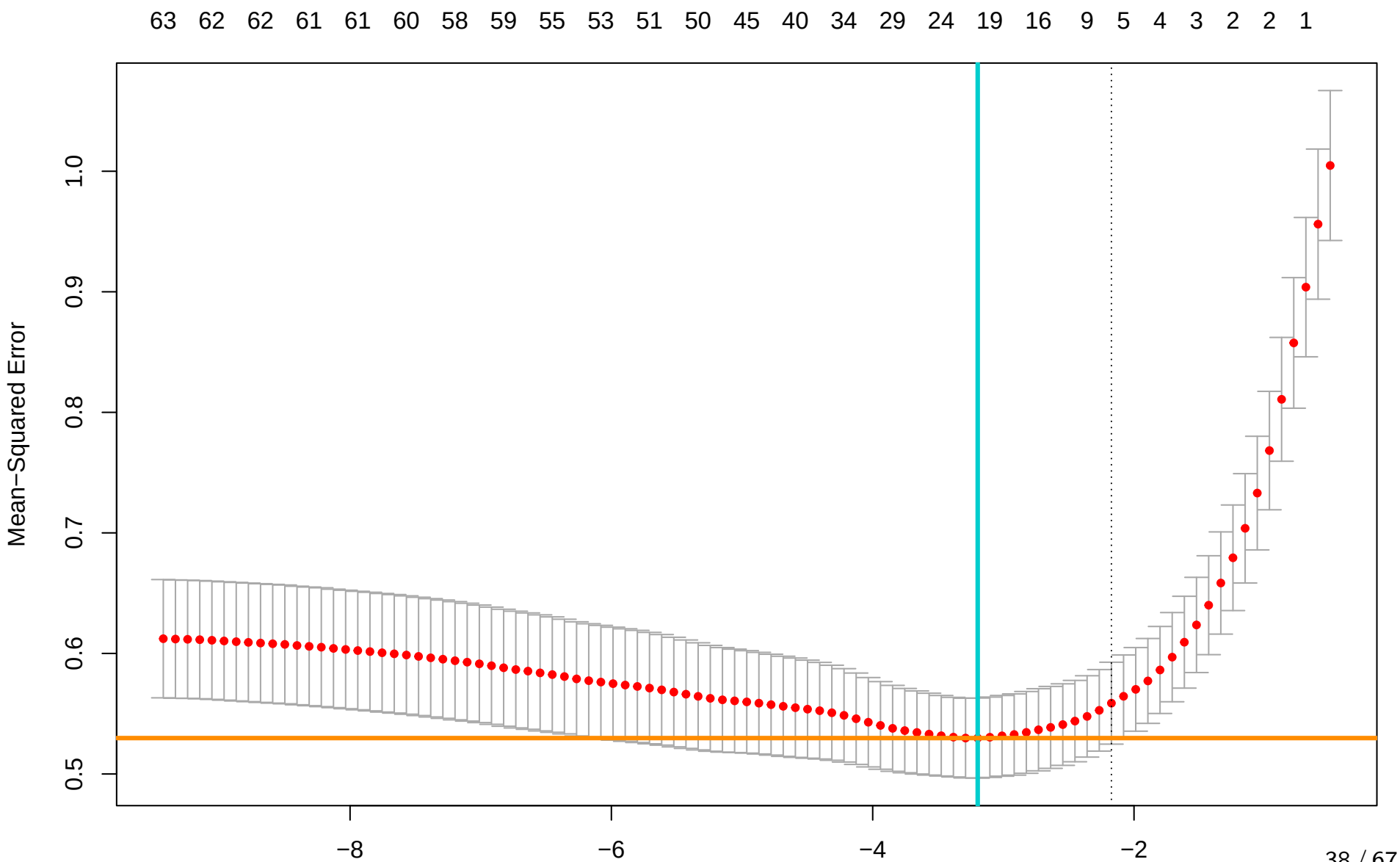
```
log(lam.1se)
```

```
## [1] -2.172892
```

```
plot(cv.out)
```


63 62 62 61 61 60 58 59 55 53 51 50 45 40 34 29 24 19 16 9 5 4 3 2 2 1
Mean−Squared Error
0.5
0.6
0.7
0.8
0.9
1.0
−8
−6
−4
−2
Log(λ)

```
plot(cv.out)
abline(v=log(best.lam), col="cyan3", lwd=3)
abline(h=cv.out$cvm[which.min(cv.out$cvm)], lwd=3,
       col="dark orange")
```



- Let's get the MSE just based on the "best" λ.

```
pred.lasso <- predict(mod.lasso, s=best.lam, newx=X.test)

mse <- mean( (pred.lasso-y.test)^2)
mse
```

```
## [1] 0.4610205
```

- Let's get the MSE just based on the smaller model.

```
pred.lasso <- predict(mod.lasso, s=lam.1se, newx=X.test)

mse <- mean( (pred.lasso-y.test)^2)
mse
```

```
## [1] 0.4955045
```

```
predict(mod.lasso, s=best.lam, type="coefficients")[1:65,]
```

```
##   (Intercept)           age           sex           bmi           map
## -0.0059203628  0.0000000000 -0.0457082363  0.3130753876  0.1410130481
##            tc           ldl           hdl           tch           ltg
##  0.0000000000  0.0000000000 -0.1240497738  0.0000000000  0.2553399747
##           glu         age.2         bmi.2         map.2          tc.2
##  0.0259844433  0.0000000000  0.0366444699  0.0113224068  0.0000000000
##         ldl.2         hdl.2         tch.2         ltg.2         glu.2
##  0.0000000000  0.0000000000  0.0000000000 -0.0101201129  0.0153856568
##       age.sex       age.bmi       age.map        age.tc       age.ldl
##  0.0725845673  0.0000000000  0.0246315363  0.0000000000 -0.0284901895
##       age.hdl       age.tch       age.ltg       age.glu       sex.bmi
##  0.0000000000  0.0000000000  0.0395873131  0.0072966578  0.0101006888
##       sex.map        sex.tc       sex.ldl       sex.hdl       sex.tch
##  0.0165249246  0.0000000000  0.0000000000  0.0000000000  0.0000000000
##       sex.ltg       sex.glu       bmi.map        bmi.tc       bmi.ldl
##  0.0000000000  0.0006070041  0.0295475709  0.0000000000  0.0000000000
##       bmi.hdl       bmi.tch       bmi.ltg       bmi.glu        map.tc
##  0.0000000000  0.0000000000  0.0000000000  0.0184488457  0.0000000000
##       map.ldl       map.hdl       map.tch       map.ltg       map.glu
##  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0000000000
##        tc.ldl        tc.hdl        tc.tch        tc.ltg        tc.glu
##  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0000000000
##       ldl.hdl       ldl.tch       ldl.ltg       ldl.glu       hdl.tch
##  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0000000000
##       hdl.ltg       hdl.glu       tch.ltg       tch.glu       ltg.glu
##  0.0000000000  0.0000000000 -0.0026456329  0.0000000000  0.0000000000
```

```r
beta.hat.lasso <- predict(mod.lasso, s=best.lam,
                          type="coefficients")[1:65,]
beta.hat.lasso[beta.hat.lasso!=0]
```

```
##   (Intercept)           sex           bmi           map           hdl
## -0.0059203628 -0.0457082363  0.3130753876  0.1410130481 -0.1240497738
##           ltg           glu         bmi.2         map.2         ltg.2
##  0.2553399747  0.0259844433  0.0366444699  0.0113224068 -0.0101201129
##         glu.2       age.sex       age.map       age.ldl       age.ltg
##  0.0153856568  0.0725845673  0.0246315363 -0.0284901895  0.0395873131
##       age.glu       sex.bmi       sex.map       sex.glu       bmi.map
##  0.0072966578  0.0101006888  0.0165249246  0.0006070041  0.0295475709
##       bmi.glu       tch.ltg
##  0.0184488457 -0.0026456329
```

- How might we get standard errors or confidence intervals . . . . Bootstrap is a possibility.
- Of course with different bootstrap samples difference coefficients may or may not be zero exactly.

Ridge Regression & Lasso Connections to Bayesian Inference

- What is Bayesian inference?
- Bayesian Inference is simply the application of Bayes' Rule to infer about parameters (which are considered random):

$$\begin{aligned} \pi(\theta|\boldsymbol{y}) &= \frac{f(\boldsymbol{y}|\theta)\pi(\theta)}{\int_{\Theta} f(\boldsymbol{y}|\theta)\pi(\theta)d\theta} \\ &= \frac{f(\boldsymbol{y}|\theta)\pi(\theta)}{m(\boldsymbol{y})} \\ &\propto f(\boldsymbol{y}|\theta)\pi(\theta) \end{aligned}$$

- It is important to remember though that in a Bayesian framework θ is random. In the frequentest framework it is fixed!

Bayesian Inference

- $\pi(\theta|\boldsymbol{y})$ is the posterior distribution for θ.
- $f(\boldsymbol{y}|\theta)$ is the joint sampling distribution for $\boldsymbol{y}$ or the likelihood for θ.
- $\pi(\theta)$ is the prior distribution for θ.
- $m(\boldsymbol{y})$ is the marginal distribution for $\boldsymbol{y}$.

- Bayesian approach:
  - We start with a prior belief about a situation (represented through a probability distribution parameterized by θ).
  - We observe data $\boldsymbol{y}$.
  - Given the data $\boldsymbol{y}$, we update our beliefs about θ via Bayes' rule and obtain the posterior distribution.

Bayesian Inference

- Many times you will just see this notation:

$$
\begin{aligned}
p(\theta|y_1,\ldots,y_n) &= \frac{p(y_1,\ldots,y_n|\theta)p(\theta)}{\int_\Theta p(y_1,\ldots,y_n|\theta)p(\theta)d\theta} \\
&= \frac{p(y_1,\ldots,y_n|\theta)p(\theta)}{p(y_1,\ldots,y_n)} \\
&= \frac{p(\boldsymbol{y}|\theta)p(\theta)}{p(\boldsymbol{y})} \\
&\propto p(\boldsymbol{y}|\theta)p(\theta)
\end{aligned}
$$

Ridge Regression

- Let's consider the following regression model:

$$\begin{aligned} y_i &= \beta_0 + \beta_1 x_{i,1} + \cdots + \beta_p x_{i,p} + \epsilon_i \\ \epsilon_i &\overset{\text{iid}}{\sim} \text{normal}(0, \sigma^2) \end{aligned}$$

- Now consider the following prior for $\boldsymbol{\beta}$:

$$p(\boldsymbol{\beta}) = p(\beta_1, \ldots, \beta_p) = p(\beta_1) \times \cdots \times p(\beta_p)$$

$$\beta_i \overset{\text{iid}}{\sim} \text{normal}(0, \tau^2)$$

Ridge Regression

- We are implicitly assuming an improper prior (not a proper probability distribution) for β_0.

$$p(\beta_0) = k$$

- Assume σ^2, τ^2, and k are known constants. In a proper Bayesian framework σ^2 would also have a prior distribution.

Ridge Regression

- Let's examine the posterior:

$$
\begin{aligned}
p(\beta_0, \beta_1, \ldots \beta_p | \boldsymbol{y}) \quad &\propto \quad p(\boldsymbol{y}|\beta)p(\beta) \\
&= \prod_{i=1}^{n} \left[(2\pi)^{-1/2} (\sigma^2)^{-1/2} exp\left(-\frac{1}{2\sigma^2} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_p x_{i,j} \right)^2 \right) \right] \\
&\times \prod_{j=1}^{p} \left[(2\pi)^{-1/2} (\tau^2)^{-1/2} exp\left(-\frac{1}{2\tau^2} \beta_j^2 \right) \right] \\
&\times k \\
&\propto \quad exp\left(-\frac{1}{2\sigma^2} \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_p x_{i,j} \right)^2 \right) exp\left(-\frac{1}{2\tau^2} \sum_{j=1}^{p} \beta_j^2 \right)
\end{aligned}
$$

Ridge Regression

$$
\begin{aligned}
p(\beta|\boldsymbol{y}) &\propto exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{i,j}\right)^2\right) exp\left(-\frac{1}{2\tau^2}\sum_{j=1}^{p}\beta_j^2\right) \\
&\propto exp\left(-\frac{1}{2\sigma^2}\left[\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{i,j}\right)^2+\frac{\sigma^2}{\tau^2}\sum_{j=1}^{p}\beta_j^2\right]\right) \\
&\propto exp\left(-\frac{1}{2\sigma^2}\left[\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{i,j}\right)^2+\lambda\sum_{j=1}^{p}\beta_j^2\right]\right)
\end{aligned}
$$

- Where $\lambda = \sigma^2/\tau^2$.

Ridge Regression

- We have:

$$p(\beta|\boldsymbol{y}) \quad = \quad C \times exp\left(-\frac{1}{2\sigma^2}\left[\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{i,j}\right)^2 + \lambda\sum_{j=1}^{p}\beta_j^2\right]\right)$$

- Where C is a constant so that the distribution integrates to one.
- Maximizing $p(\beta|\boldsymbol{y})$ wrt β (finding the mode of the posterior) is the same as minimizing the following wrt β:

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

- Further, we can show that $p(\beta|\boldsymbol{y})$ is a normal distribution, so the mean and the mode are the same.

Lasso

- We follow the same procedure, except we change the prior for $\beta_1, \ldots, \beta_p$. Now we assume a double exponential distribution:

$$p(\beta_1, \ldots, \beta_p) = p(\beta_1) \times \ldots \times p(\beta_p) = \prod_{j=1}^{p} \frac{1}{2b} exp\left(-\frac{|\beta_j|}{b}\right)$$

Lasso

- Let's get the posterior:

$$
\begin{aligned}
p(\beta|\boldsymbol{y}) &\propto p(\boldsymbol{y}|\beta)p(\beta) \\
&= \prod_{i=1}^{n}\left[(2\pi)^{-1/2}(\sigma^2)^{-1/2}exp\left(-\frac{1}{2\sigma^2}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{i,j}\right)^2\right)\right] \\
&\quad \times \prod_{j=1}^{p}\frac{1}{2b}exp\left(-\frac{|\beta_j|}{b}\right) \\
&\quad \times k \\
&\propto exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{i,j}\right)^2\right) exp\left(-\frac{1}{b}\sum_{j=1}^{p}|\beta_j|\right) \\
&\propto exp\left(-\frac{1}{2\sigma^2}\left[\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{i,j}\right)^2+\frac{2\sigma^2}{b}\sum_{j=1}^{p}|\beta_j|\right]\right)
\end{aligned}
$$

Lasso

$$
\begin{aligned}
p(\beta|\boldsymbol{y}) &\propto exp\left(-\frac{1}{2\sigma^2}\left[\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{i,j}\right)^2+\frac{2\sigma^2}{b}\sum_{j=1}^{p}\beta_j^2\right]\right) \\
&= C\times exp\left(-\frac{1}{2\sigma^2}\left[\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{i,j}\right)^2+\lambda\sum_{j=1}^{p}|\beta_j|\right]\right)
\end{aligned}
$$

- Where C is a constant so that the distribution integrates to one.
- Maximizing $p(\beta|\boldsymbol{y})$ wrt β (finding the mode of the posterior) is the same as minimizing the following wrt β:

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

Other Bayesian Ideas for Covariate Selection

- We saw that we could change the priors we place on the β to get different formulations (i.e. models) for covariate selection. Another approach is called a **spike-and-slab** prior for β_j:

$$\begin{aligned} \beta_j &\overset{\text{iid}}{\sim} \begin{cases} 0 & \text{if } z_j = 0 \\ \text{normal}(0, 100) & \text{if } z_j = 1 \end{cases} \\ z_j &\overset{\text{iid}}{\sim} \text{Bernoulli}(\pi = 1/2) \end{aligned}$$

- We will be able to examine the posterior distributions to estimate the $p(\beta_j \neq 0 | \boldsymbol{y})$.

- To examine the posterior (Bayesian inference), many times an iterative approach called Markov chain Monte Carlo (MCMC) is used.
- This model, with R-code, is covered in Chapter 9 of *A First Course in Bayesian Statistical Methods* by Peter Hoff.
- If you are taking the Bayesian course with Professor Loong, you may cover this in detail.
- Here, I will simply run the code for a full Bayesian model (i.e. run the MCMC to get samples from the posterior distribution).

```r
## clear all of the objects
rm(list = ls())

diabTrain <- read.table("diabTrain.dat", header=T)
diabTest <- read.table("diabTest.dat", header=T)
```

```r
y <- diabTrain[,1]
X <- as.matrix(diabTrain[,-1])
X <- cbind(rep(1, length(y)), X)

y.test <- diabTest[,1]
X.test <- as.matrix(diabTest[,-1])
X.test <- cbind(rep(1, length(y.test)), X.test)

##
n <- dim(X)[1]
p <- dim(X)[2]
```

```
Beta.ols <- solve(t(X)%*%X)%*%t(X)%*%y
c(Beta.ols)
```

```
##  [1] -0.009659686  0.034510192 -0.119843030  0.281272003  0.158404023
##  [6] -7.709359660  6.727818316  2.761911598  0.028932318  2.909802108
## [11]  0.073729411  0.024773627  0.048165853  0.030244618  5.555899259
## [16]  2.333582766  1.145512276  0.556219176  1.244427518  0.034223667
## [21]  0.085211252 -0.028246293  0.042587370 -0.209505066 -0.024965752
## [26]  0.191285504  0.160688008  0.176622141  0.026875756  0.054896180
## [31]  0.055449950  0.542381922 -0.418896210 -0.172196073 -0.073718105
## [36] -0.174683606  0.013659800  0.055915202 -0.468269051  0.398116522
## [41]  0.171149593 -0.064612652  0.163538010  0.069540412  0.365978724
## [46] -0.205956113 -0.168052231 -0.077806217 -0.064715735 -0.141659547
## [51] -7.338368934 -2.882222273 -2.049815787 -2.204253575  0.330447880
## [56]  1.878185343  1.664927506  1.261846070 -0.242166584  1.017243679
## [61]  0.829593049 -0.101326692  0.489019741  0.003657258 -0.047164160
```

- Let's check with previous standard regression result.

```
y.hat.test <- X.test%*%Beta.ols
mean((y.test-y.hat.test)^2)
```

```
## [1] 0.672564
```

Some Functions for the Bayesian Approach

```
#########################################################################
## Full Bayesian Model Selection Based on the g-prior
## load the MASS library
library(MASS)

### sample from a multivariate normal distribution
rmvnorm<-
function(n,mu,Sigma) {
    p<-length(mu)
    res<-matrix(0,nrow=n,ncol=p)
    if( n>0 & p>0 ) {
        E<-matrix(rnorm(n*p),n,p)
        res<-t(  t(E%*%chol(Sigma)) +c(mu))
    }
    res
}
###


### sample from an inverse-wishart distribution
rinvwish<-function(n,nu0,iS0)
{
    sL0 <- chol(iS0)
    S<-array( dim=c( dim(L0),n ) )
    for(i in 1:n)
    {
        Z <- matrix(rnorm(nu0 * dim(L0)[1]), nu0, dim(iS0)[1]) %*% sL0
        S[,,i]<- solve(t(Z)%*%Z)
    }
    S[,,1:n]
}
```

Some Functions for the Bayesian Approach

```
##### a function to compute the marginal probability
lpy.X <- function(y, X, g, nu0, s20)
{
    n <- dim(X)[1]; p <- dim(X)[2]
    if(p==0){Hg <- 0; s20 <- mean(y^2)}
    if(p>0) { Hg <- (g/(g+1))*X%*%solve(t(X)%*%X)%*%t(X)}
    SSRg <- t(y)%*%(diag(1, nrow=n) - Hg)%*%y
    out <- -0.5*(n*log(pi)+p*log(1+g)+(nu0+n)*log(nu0*s20+SSRg) -
                  nu0*log(nu0*s20)) + lgamma ((nu0+n)/2) - lgamma (nu0/2)
    return(out)
}
```

Priors for the Bayesian Approach

```
## priors
g <- length(y)
nu0 <- 1
s20 <- sum((y - X%*%Beta.ols)^2)/(n-p)

## starting values
z<-rep(1,dim(X)[2])
lpy.c<-lpy.X(y,X[,z==1,drop=FALSE], g=g, nu0=nu0, s20=s20)
```

MCMC for the Bayesian Approach I

- MCMC code not fully shown here . . . see the R-code file for these slides. You would likely want to run the MCMC for more than 2,000 scans.

MCMC for the Bayesian Approach II

```
set.seed(1)
## Set up Gibbs sampler
S <- 2000

## Storage matrices
Z <- BETA <- SIGMA2 <- NULL

## Gibbs Sampler
for(s in 1:S){
print(s)

## update z
    for(j in sample(1:p))
    {
        zp<-z
        zp[j]<-1-zp[j]
        lpy.p<- lpy.X(y,X[,zp==1,drop=FALSE], g=g, nu0=nu0, s20=s20)
        if(zp[j]==1){
        prob.z.1 <- exp(lpy.p)/(exp(lpy.p) + exp(lpy.c))
        }
        if(zp[j]==0){
        prob.z.1 <- exp(lpy.c)/(exp(lpy.p) + exp(lpy.c))
        }
        z[j]<-rbinom(1, 1, prob.z.1)
        lpy.c<- lpy.X(y,X[,z==1,drop=FALSE], g=g, nu0=nu0, s20=s20)
    }

## Update sigma.sq
Hg <- (g/(g+1))*X[,z==1,drop=FALSE]%*%solve(t(X[,z==1,drop=FALSE])%*%X[,z==1,drop=FALSE])%*%t(X[,z==1,drop=FALS
SSRg <- t(y)%*%(diag(1, nrow=n) - Hg)%*%y
sigma.sq <- 1/rgamma(1, (nu0 + n)/2 , (nu0*s20 + SSRg)/2)

## Update beta
```

Results

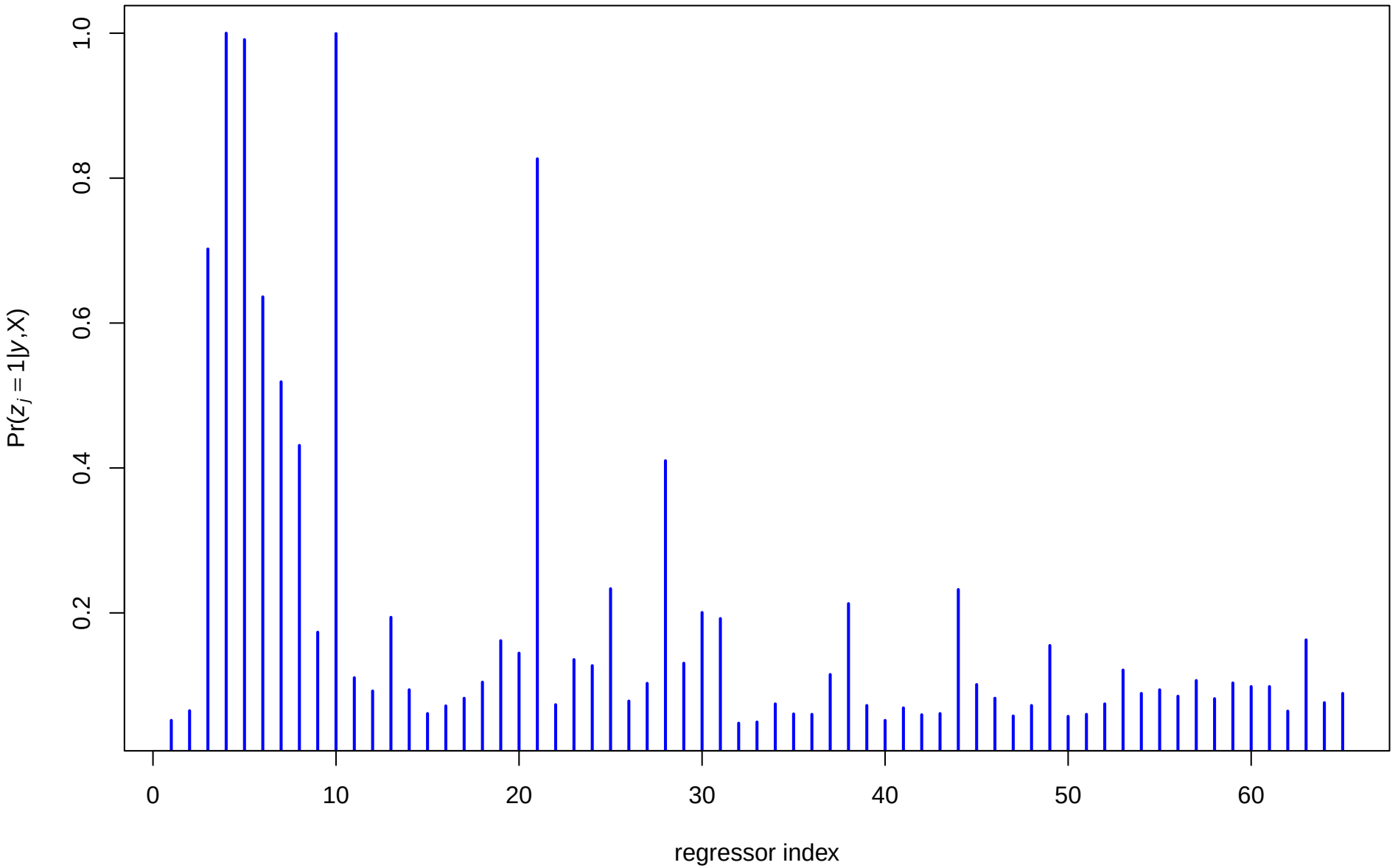
- I will remove the first 200 scans for burn-in.

```
Beta <- read.table("Beta.txt", header=TRUE)[-c(1:200),]
Z <- read.table("Z.txt", header=TRUE)[-c(1:200),]

Beta.bma.pm <- apply(Beta, 2, mean)
y.hat.test <- X.test%*%Beta.bma.pm
mean( (y.test-y.hat.test)^2)
```

```
## [1] 0.451798
```

- Slightly better than the best lasso MSE: 0.46441

```r
Z.prob <- apply(Z, 2, mean)
plot(Z.prob, xlab="regressor index",
     ylab=expression(paste( "Pr(",italic(z[j] == 1),"|",italic(y),",X)",sep="")),
     type="h", lwd=2, col="blue")
```



- Important covariates (probability of inclusion estimated to be greater than 0.8)

```
colnames(X)[Z.prob>0.8]
```

```
## [1] "bmi"     "map"     "ltg"     "age.sex"
```

- We can also get credible intervals (Bayesian confidence intervals)

```
out <- apply(Beta[,Z.prob>0.8], 2, quantile, c(0.025, 0.975))
colnames(out) <- colnames(X)[Z.prob>0.8]
out
```

```
##              bmi        map       ltg   age.sex
## 2.5%  0.2291782 0.08180636 0.2074548 0.0000000
## 97.5% 0.4428489 0.27135862 1.6790159 0.2013113
```

- See "The Bayesian Lasso" by Park and Casella (2008) for further general discussions.