Multiple linear regression

Hedibert F. Lopes & Paulo Marques
INSPER Institute of Education and Research
São Paulo, Brazil

Outline

Outline

Simplest linear regression model

We already studied the homoskedastic linear regression model

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i,$$

where $\{\varepsilon_i\}_{i=1}^n$ are i.i.d. and, for all i,

$$\begin{aligned} E(\varepsilon_i) &= 0 \\ V(\varepsilon_i) &= \sigma^2 \\ COV(x_i, \varepsilon_i) &= 0 \end{aligned}$$

The estimates of β_0 and β_1 are obtained via ordinary least square (OLS):

$$\hat{\beta}_1 = \frac{\sum_{i=1}^n (y_i - \bar{y})(x_i - \bar{x})}{\sum_{i=1}^n (x_i - \bar{x})^2} \quad \text{and} \quad \hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x},$$

where $n\bar{x} = \sum_{i=1}^n x_i$, $n\bar{y} = \sum_{i=1}^n y_i$, $\hat{y}_i = \hat{\beta}_0 - \hat{\beta}_1 x_i$ and $n\hat{\sigma}^2 = \sum_{i=1}^n (y_i - \hat{y}_i)^2$.

Centering and standardizing y's and x's

When y_i and x_i are replaced, respectively, by

$$\tilde{y}_i = \frac{y_i - \bar{y}}{s_y} \quad \text{and} \quad \tilde{x}_i = \frac{x_i - \bar{x}}{s_x},$$

where

$$s_y^2 = \frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y})^2 \quad \text{and} \quad s_x^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2,$$

it is easy to see that the intercept vanishes, i.e.

$$\hat{\beta}_0 = 0,$$

and that

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n} \tilde{y}_i \tilde{x}_i}{\sum_{i=1}^{n} \tilde{x}_i^2} = \frac{\tilde{x}'\tilde{y}}{\tilde{x}'\tilde{x}} = (\tilde{x}'\tilde{x})^{-1}\tilde{x}'\tilde{y}$$

where $\tilde{y} = (\tilde{y}_1, \ldots, \tilde{y}_n)'$ and $\tilde{x} = (\tilde{x}_1, \ldots, \tilde{x}_n)'$.

Gaussian errors and other issues

- OLS estimates and maximum likelihood estimates (MLE) are the same.
- ML is usually more suitable for formal inference.
- The paid price is more modeling assumptions.

We also discussed other departures from the above model, such as

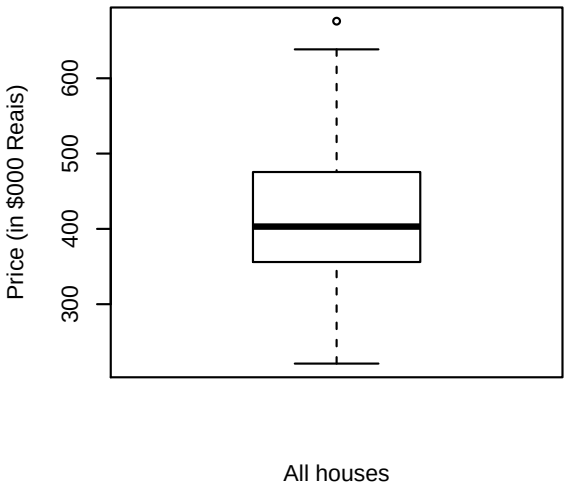
- nonlinearities
- heteroskedasticity
- spurious regression
- endogeneity
- Simpson's paradox

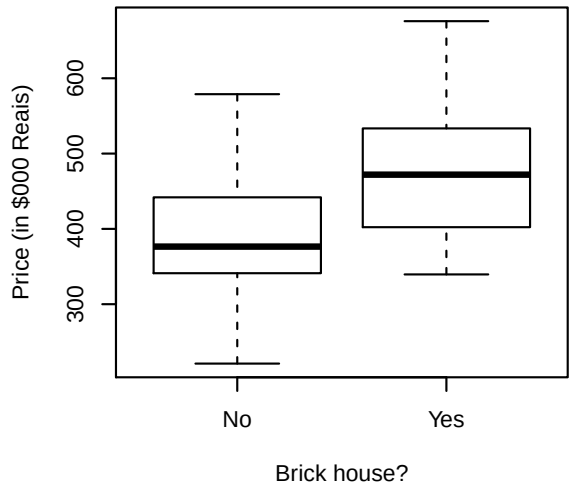
houseprice dataset

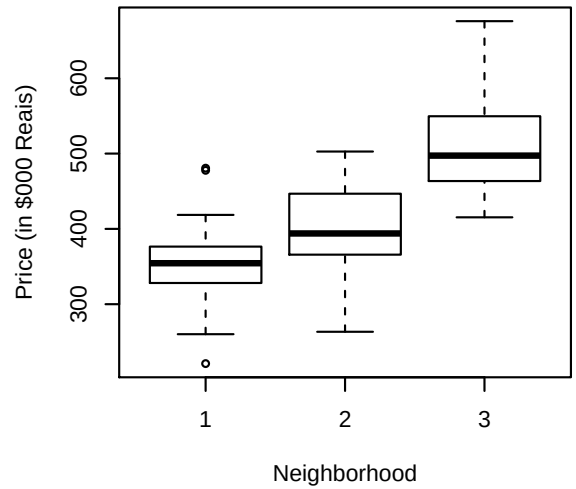
128 observations and 7 variables
5 quantitative variables (2 continuous and 3 counts)
2 qualitative variables

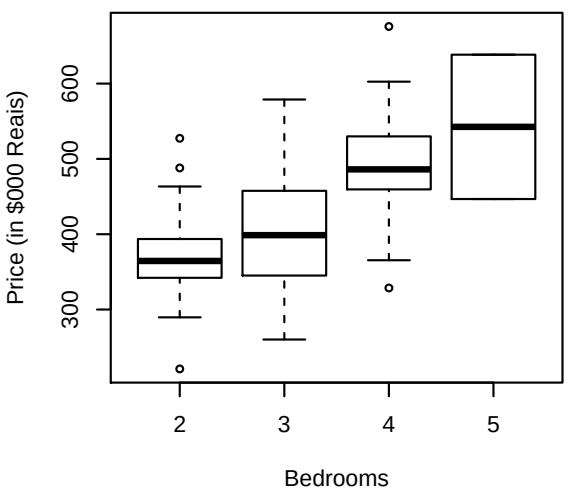
Nbhd	Offers	SqFt	Brick	Bedrooms	Bathrooms	Price
2	2	1790	No	2	2	114300
2	3	2030	No	4	2	114200
2	1	1740	No	3	2	114800
2	3	1980	No	3	2	94700
2	3	2130	No	3	3	119800
1	2	1780	No	3	2	114600
3	3	1830	Yes	3	3	151600
3	2	2160	No	4	2	150700
2	3	2110	No	4	2	119200
2	3	1730	No	3	3	104000

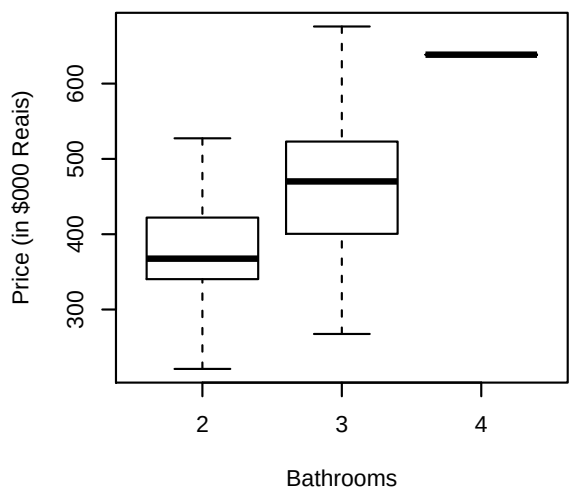
price by covariates

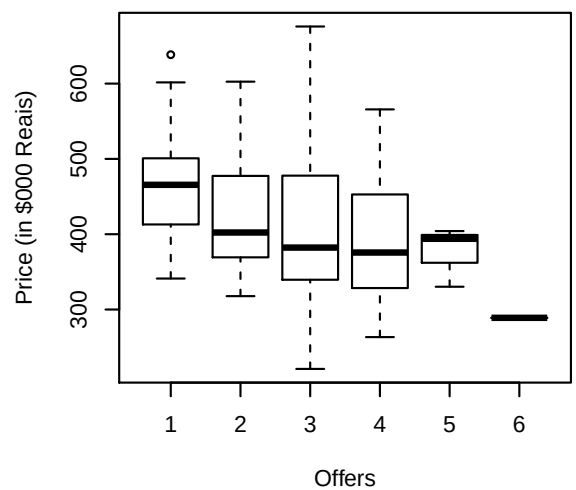




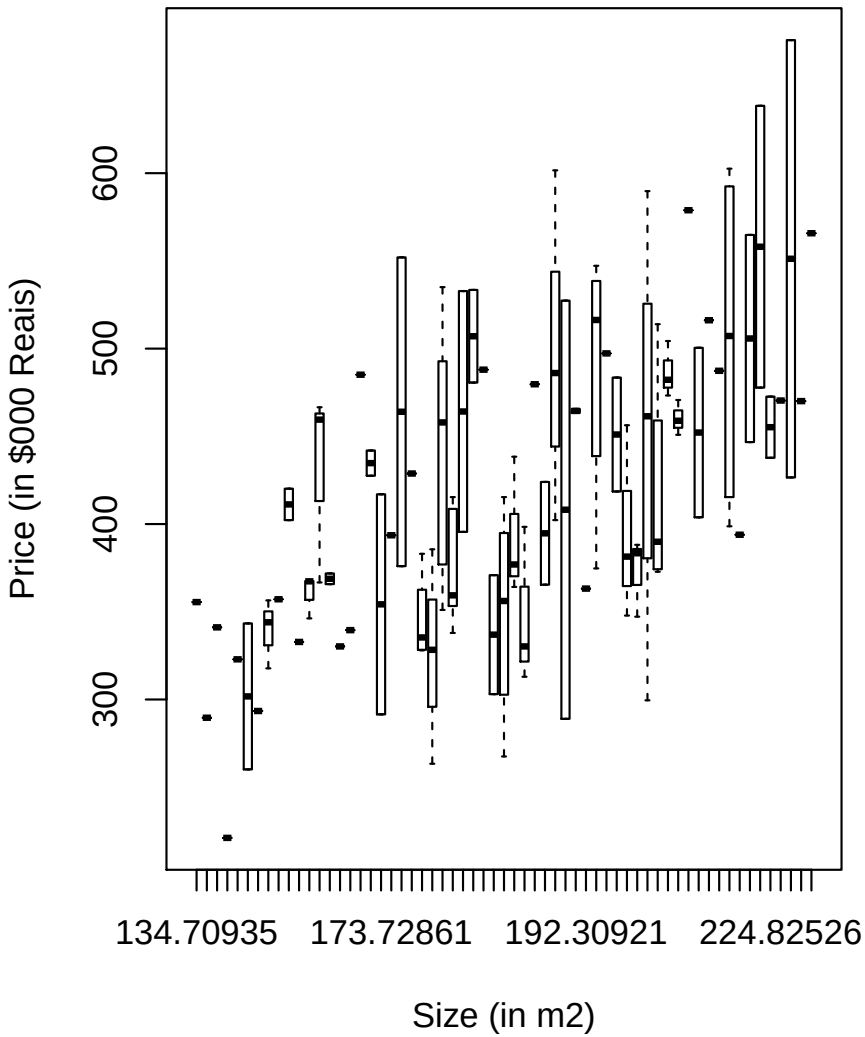


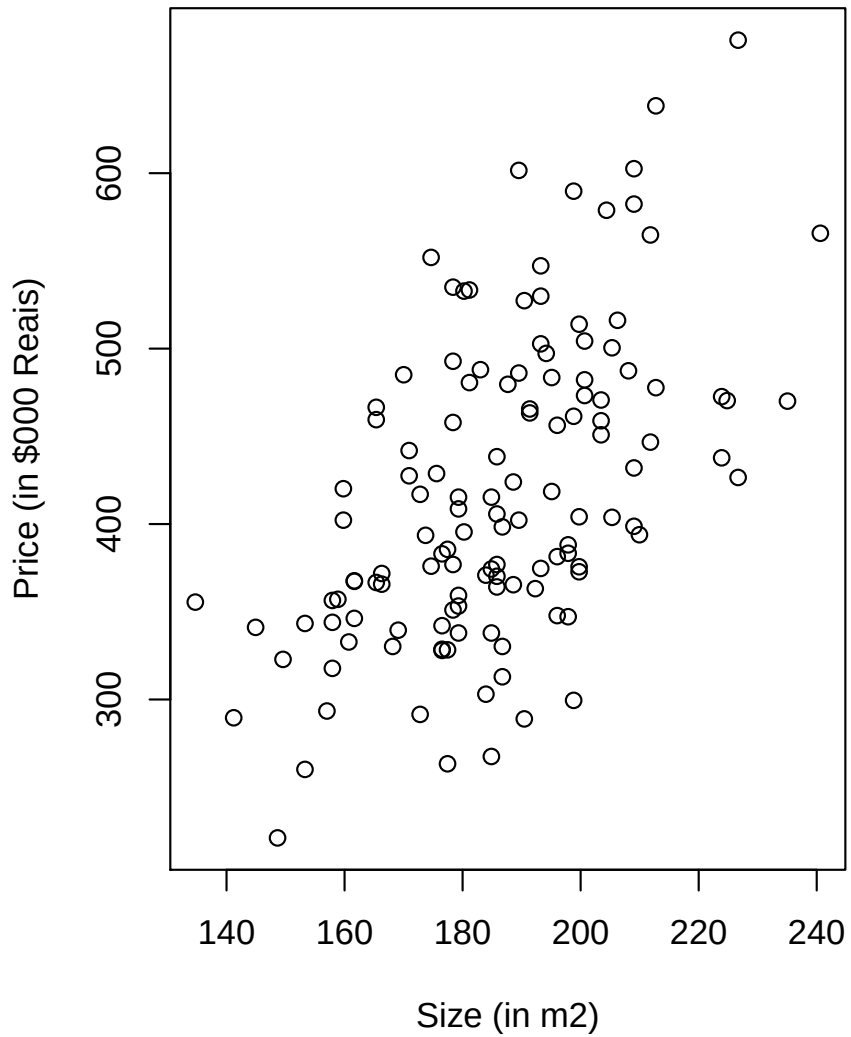




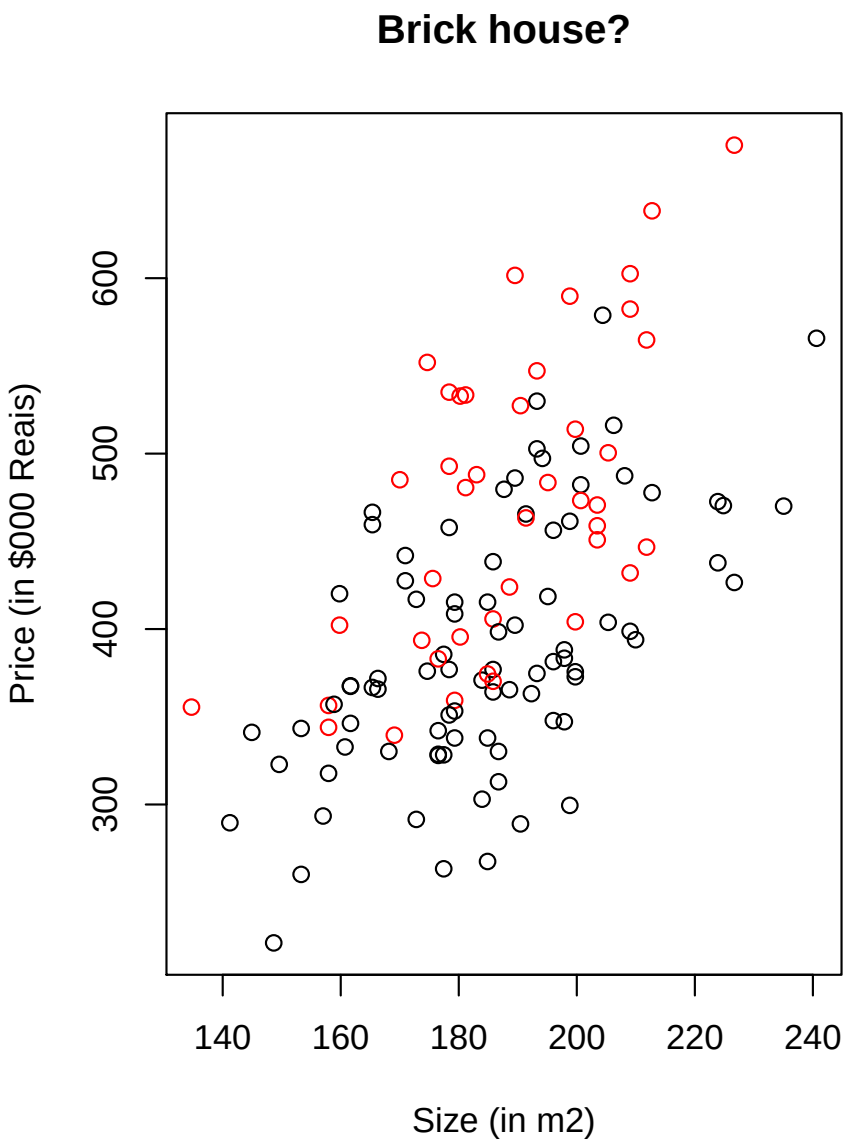


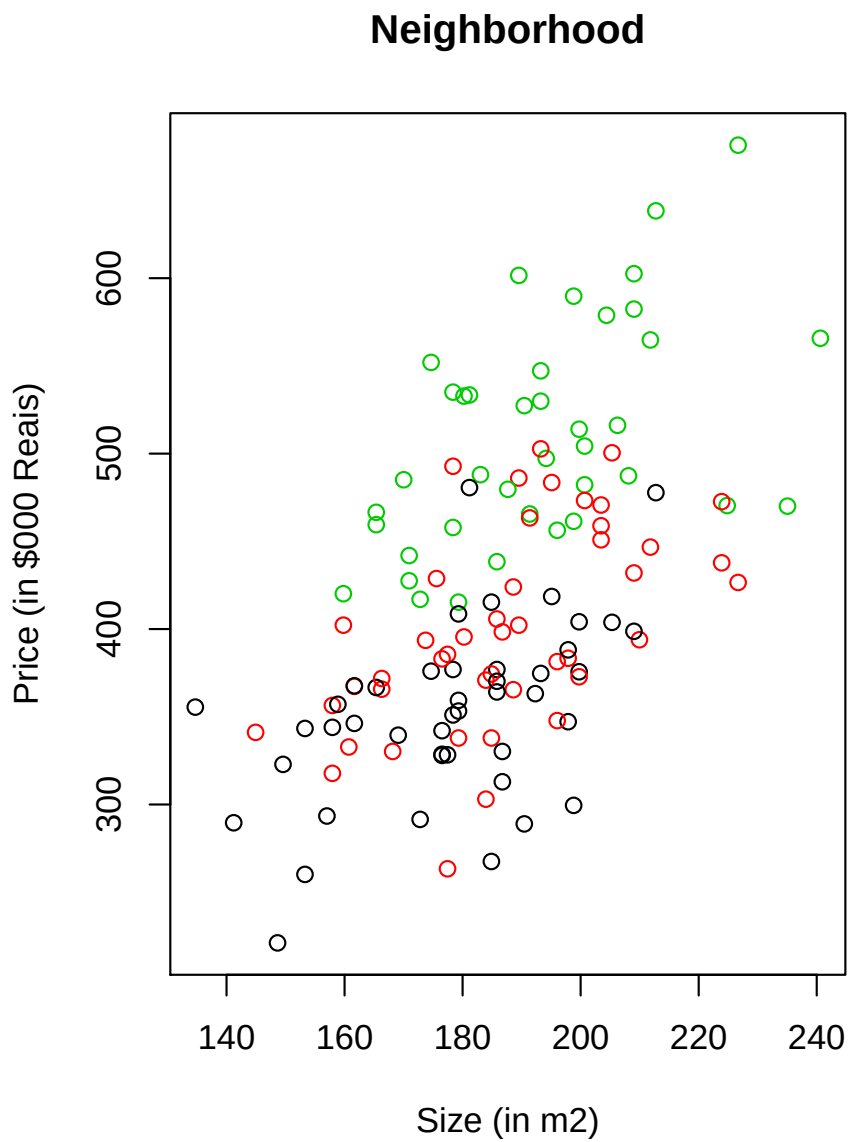
price by size



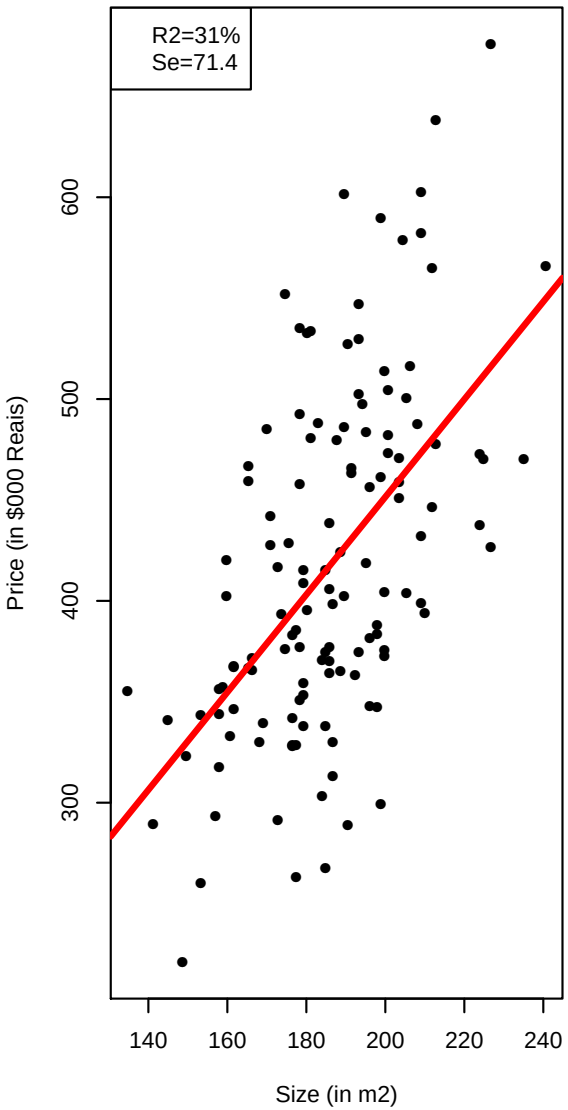


price by size (and type or neighborhood)

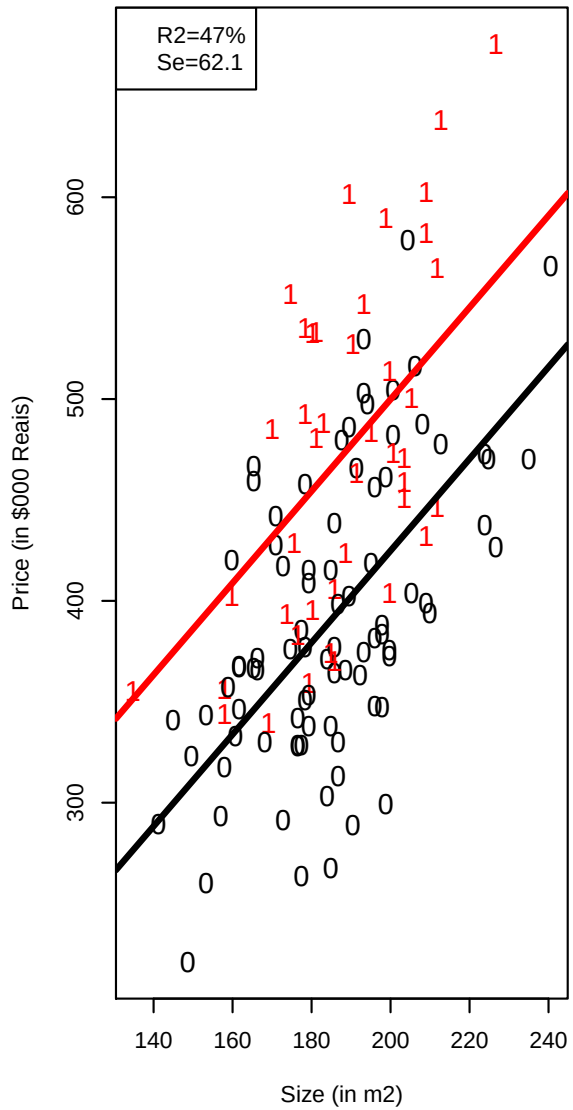




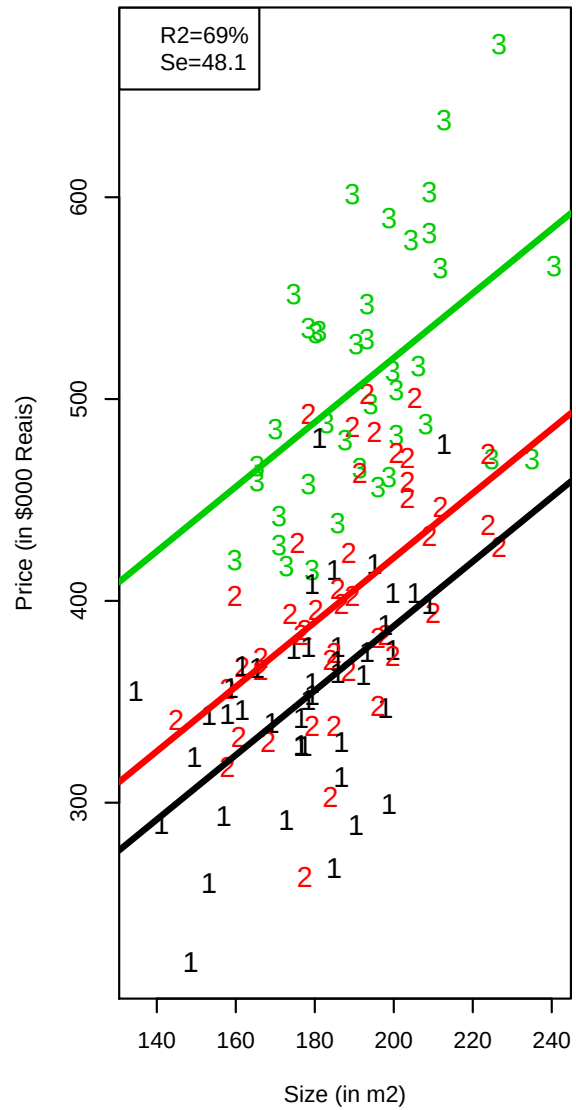
Linear regressions



Brick house?



Neighborhood



Multiple linear regression

Instead of "explaining" y via a single covariate (explanatory or predictor) x, we gain a lot of modeling flexibility by considering several predictors simultaneously:

$$y_i = \beta_0 + \beta_1 x_{i1} + \cdots \beta_p x_{ip} + \varepsilon_i$$

for $i = 1, \ldots, n$ and i.i.d. error terms with mean zero and variance σ^2.

`houseprice:` Here $n = 128$ and a potencial model is

$$\texttt{Price} = \beta_0 + \beta_1 \texttt{Size} + \beta_2 \texttt{Bathrooms} + \beta_3 \texttt{Offers} + \varepsilon$$

OLS: One estimates $\beta_0, \beta_1, \ldots, \beta_p$ by minimizing the sum of squared residuals

$$\text{RSS} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_{i1} - \cdots - \hat{\beta}_p x_{ip})^2$$

Matrix notation

By stacking the y_i's into the vector $\boldsymbol{y}$ and the x_{ij}s into the matrix $\boldsymbol{X}$:

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon},$$

where

- $q = p + 1$
- $\boldsymbol{y}$: n-dimensional vector of continuous responses,
- $\boldsymbol{X}$ is a $(n \times q)$ design matrix with each column representing a covariate,
- $\boldsymbol{\beta}$: q-dimensional vector of regression coefficients,
- $\boldsymbol{\varepsilon} \sim (0, \sigma^2 \boldsymbol{I})$.

Useful constraints:

- $\mathbb{E}(\boldsymbol{X}_j) = 0$ and $\text{Var}(\boldsymbol{X}_j) = 1$, for each column $\boldsymbol{X}_j$ of $\boldsymbol{X}$.
- Replaced y_i by $y_i - \bar{y}$, where $\bar{y} = \sum_{i=1}^{n} y_i / n$, and ignore the intercept β_0.

OLS estimates

Therefore, finding β that minimizes the RSS translates into

$$\begin{aligned}\hat{\beta} &= \arg\min_{\beta \in \Re^p} (\boldsymbol{y} - \boldsymbol{X}\beta)'(\boldsymbol{y} - \boldsymbol{X}\beta) \\ &= \arg\min_{\beta \in \Re^p} \beta \boldsymbol{X}'\boldsymbol{X}\beta - 2\beta'\boldsymbol{X}'\boldsymbol{y}.\end{aligned}$$

Simple matrix algebra leads to

$$\hat{\beta}_{ols} = (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{y}$$

so

$$\hat{\boldsymbol{y}} = \boldsymbol{X}\hat{\beta}_{ols} = \boldsymbol{X}(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{y},$$

which corresponds to an orthogonal projection of $\boldsymbol{y}$ onto the column space of $\boldsymbol{X}$.

Hat matrix: $\boldsymbol{P} = \boldsymbol{X}(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'$
residuals: $\varepsilon = \boldsymbol{y} - \boldsymbol{X}(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{y} = (\boldsymbol{I} - \boldsymbol{P})\boldsymbol{y}$
RSS: $\varepsilon'\varepsilon = \boldsymbol{y}'(\boldsymbol{I} - \boldsymbol{P})'(\boldsymbol{I} - \boldsymbol{P})\boldsymbol{y} = \boldsymbol{y}'(\boldsymbol{I} - \boldsymbol{P})\boldsymbol{y}$
Variance of the errors: $\hat{\sigma}^2 = \varepsilon'\varepsilon/n - p$

R^2 and R^2_{adj}

Since the total sume of squares, $TSS = \boldsymbol{y}'\boldsymbol{y}$, is the same regardless of which multiple linear regression model is being fit, it follows that

$$R^2 = 1 - \frac{RSS}{TSS} = 1 - \frac{\boldsymbol{y}'(\boldsymbol{I} - \boldsymbol{P})\boldsymbol{y}}{\boldsymbol{y}'\boldsymbol{y}}$$

R^2 is the proportion of the variance of y explained by a set of predictors.

Since R^2 always increase with model complexity, an adjusted R^2 is commonly used to avoid (or, at least, diminish) such distortions:

$$R^2_{adj} = 1 - \frac{\boldsymbol{y}'(\boldsymbol{I} - \boldsymbol{P})\boldsymbol{y}}{\boldsymbol{y}'\boldsymbol{y}} \left(\frac{n}{n-p} \right)$$

`houseprice:` With $n = 128$ and $p = 6$, it follows that $n/(n-6) = 1.049$.

C_p, AIC and BIC

Other techniques commonly used to *adjust* for model complexity are:

- Mallow's C_p

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

- Akaike information criterion (AIC)

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

- Bayesian information criterion (BIC)

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

houseprice data

$2^6 = 64$ models

```
install.packages("leaps")
library("leaps")
file = "http://hedibert.org/wp-content/uploads/2013/11/houseprice.txt"
house = read.table(file,header=TRUE)
house[,4]  = house[,4]*0.092903
house[,8]  = house[,8]*3.2/1000
house = house[,2:8]
```

	Nbhd	Offers	SqFt	Brick	Bedrooms	Bathrooms	Price
1	2	2	166.2964	No	2	2	365.76
2	2	3	188.5931	No	4	2	365.44
3	2	1	161.6512	No	3	2	367.36
4	2	3	183.9479	No	3	2	303.04
5	2	3	197.8834	No	3	3	383.36

R package regsubsets

regsubsets {leaps} R Documentation

functions for model selection

Description

Model selection by exhaustive search, forward or backward stepwise, or sequential replacement

Usage

```
regsubsets(x=, ...)

## S3 method for class 'formula'
regsubsets(x=, data=, weights=NULL, nbest=1, nvmax=8,
 force.in=NULL, force.out=NULL, intercept=TRUE,
 method=c("exhaustive", "backward", "forward", "seqrep"),
 really.big=FALSE,
 nested=(nbest==1),...)

## Default S3 method:
regsubsets(x=, y=, weights=rep(1, length(y)), nbest=1, nvmax=8,
force.in=NULL, force.out=NULL, intercept=TRUE,
 method=c("exhaustive","backward", "forward", "seqrep"),
really.big=FALSE,nested=(nbest==1),...)

## S3 method for class 'biglm'
regsubsets(x,nbest=1,nvmax=8,force.in=NULL,
method=c("exhaustive","backward", "forward", "seqrep"),
really.big=FALSE,nested=(nbest==1),...)

## S3 method for class 'regsubsets'
summary(object,all.best=TRUE,matrix=TRUE,matrix.logical=FALSE,df=NULL,...)

## S3 method for class 'regsubsets'
coef(object,id,vcov=FALSE,...)
## S3 method for class 'regsubsets'
vcov(object,id,...)
```

Modeling price

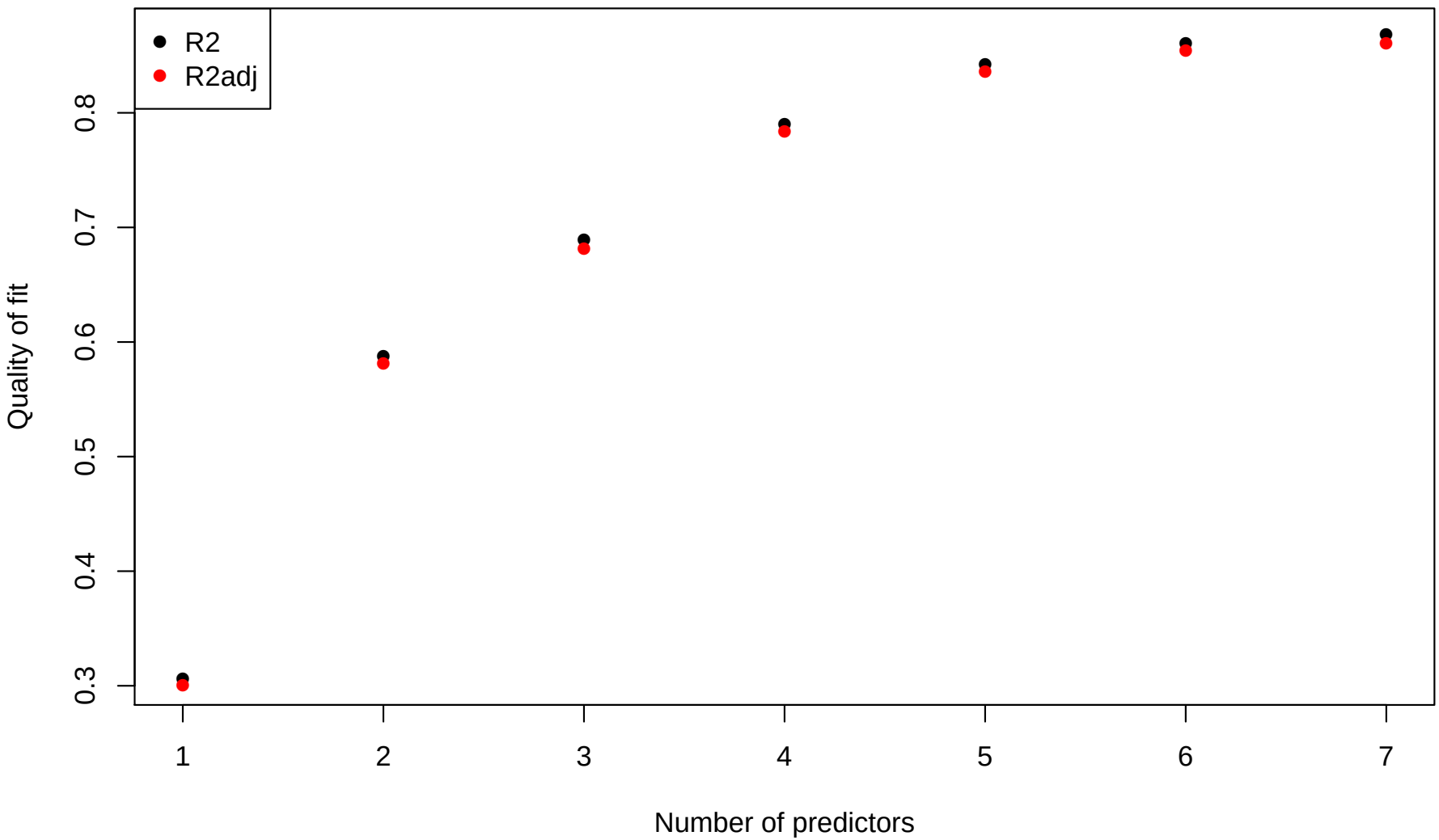
```
install.packages("leaps")

library("leaps")

house = read.table("houseprice.txt",header=TRUE)

house[,4]  = house[,4]*0.092903

house[,8]  = house[,8]*3.2/1000

attach(house)

N1 = rep(0,n)
N2 = rep(0,n)
N1[nbhd==1]=1
N2[nbhd==2]=1

data = cbind(house[,3:8],N1,N2)

regs = regsubsets(Price~.,data=data)

regs.summary = summary(regs)
```

Modeling price

```
regs.summary

Subset selection object
Call: regsubsets.formula(Price ~ ., data = data)
7 Variables  (and intercept)
           Forced in Forced out
Offers         FALSE      FALSE
SqFt           FALSE      FALSE
BrickYes       FALSE      FALSE
Bedrooms       FALSE      FALSE
Bathrooms      FALSE      FALSE
N1             FALSE      FALSE
N2             FALSE      FALSE
1 subsets of each size up to 7
Selection Algorithm: exhaustive
         Offers SqFt BrickYes Bedrooms Bathrooms N1  N2
1  ( 1 ) " "    "*"  " "      " "      " "       " " " "
2  ( 1 ) "*"    "*"  " "      " "      " "       " " " "
3  ( 1 ) "*"    "*"  "*"      " "      " "       " " " "
4  ( 1 ) " "    "*"  "*"      " "      " "       "*" "*"
5  ( 1 ) "*"    "*"  "*"      " "      " "       "*" "*"
6  ( 1 ) "*"    "*"  "*"      " "      "*"       "*" "*"
7  ( 1 ) "*"    "*"  "*"      "*"      "*"       "*" "*"
```

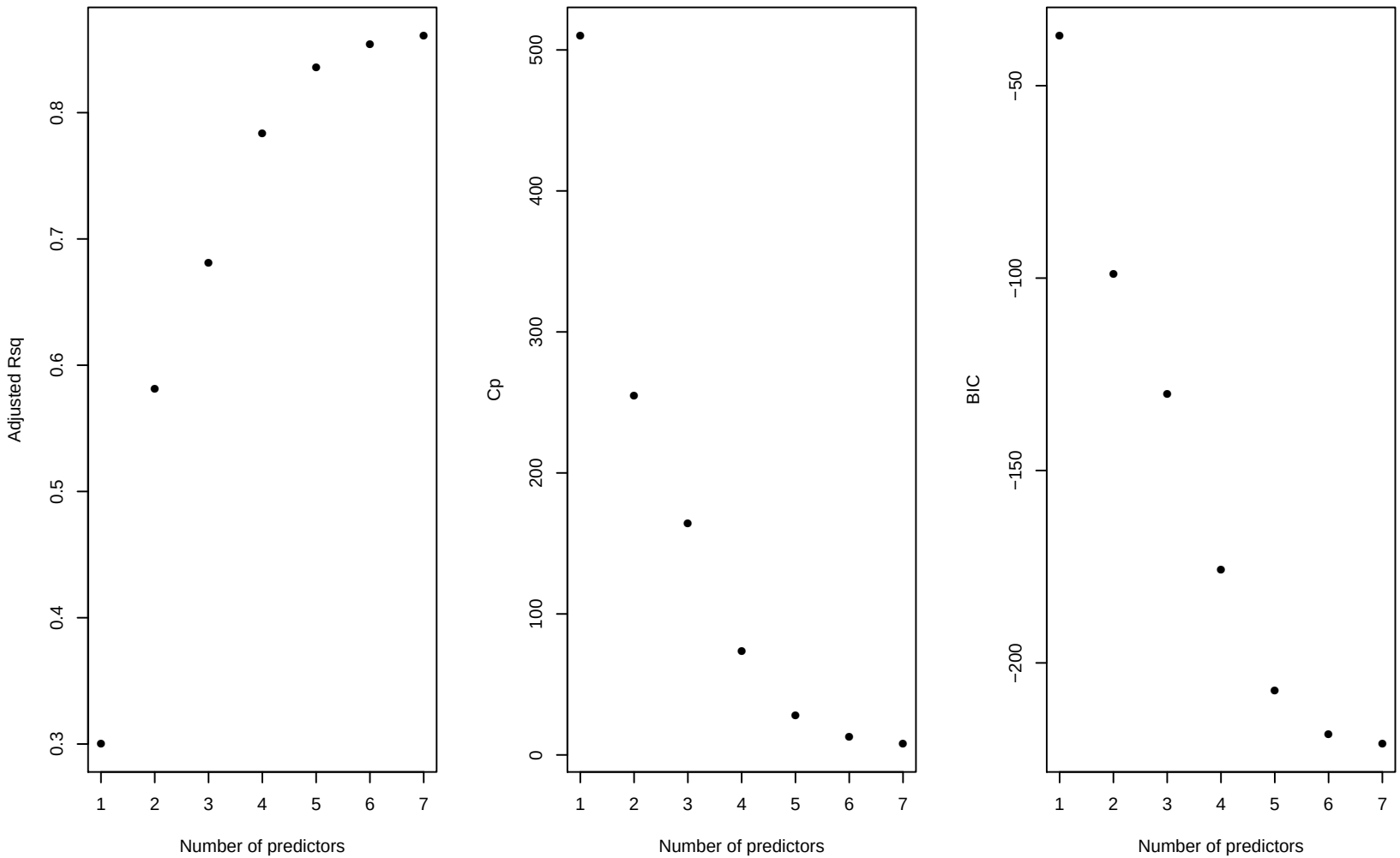
R^2 and R^2_{adj}

```
plot(regs.summary$rsq,pch=16,xlab="Number of predictors",ylab="Quality of fit")
points(regs.summary$adjr2,col=2,pch=16)
legend("topleft",legend=c("R2","R2adj"),col=1:2,pch=16)
```



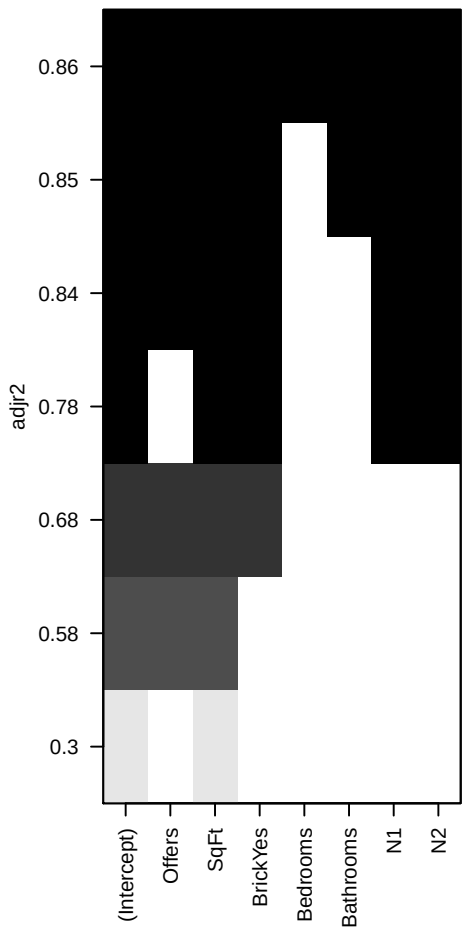
R^2_{adj}, C_p and BIC

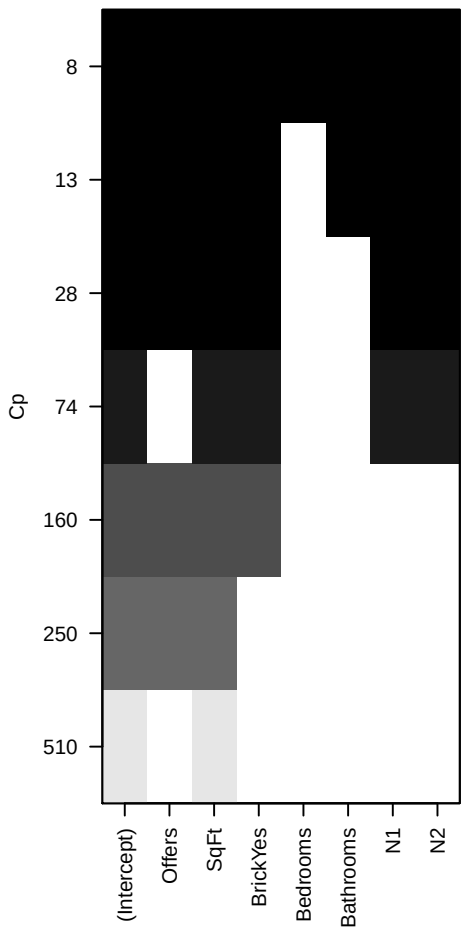
```
par(mfrow=c(1,3))
plot(regs.summary$adjr2,pch=16,xlab="Number of predictors",ylab="Adjusted Rsq")
plot(regs.summary$cp,pch=16,xlab="Number of predictors",ylab="Cp")
plot(regs.summary$bic,pch=16,xlab="Number of predictors",ylab="BIC")
```

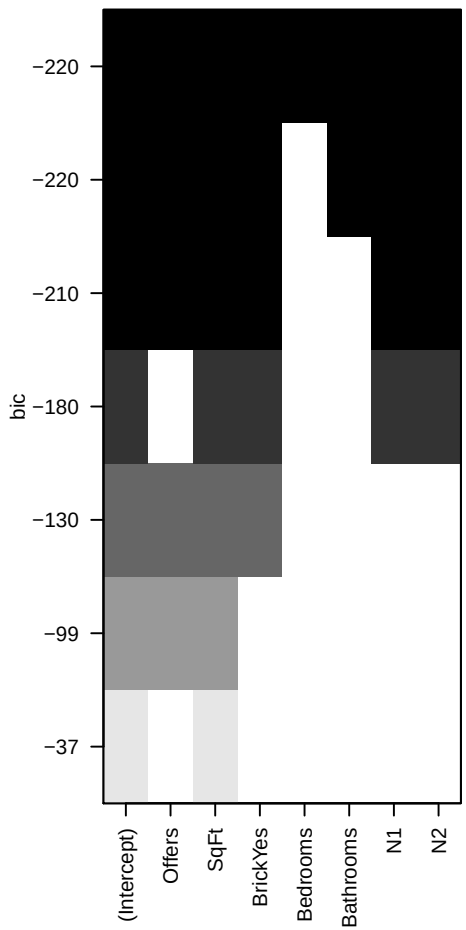


Best models

```
par(mfrow=c(1,3))
plot(regs,scale="adjr2")
plot(regs,scale="Cp")
plot(regs,scale="bic")
```







Best model

```
coef(regs,6)
```

$$\widehat{\texttt{Price}} = 90.54 - 24.68\texttt{Offers} + 1.93\texttt{SqFt} + 54.27\texttt{BrickYes} + 27.78\texttt{Bathrooms} - 77.52\texttt{N}_1 - 79.05\texttt{N}_2$$

Regressing `Price` on 16 covariates

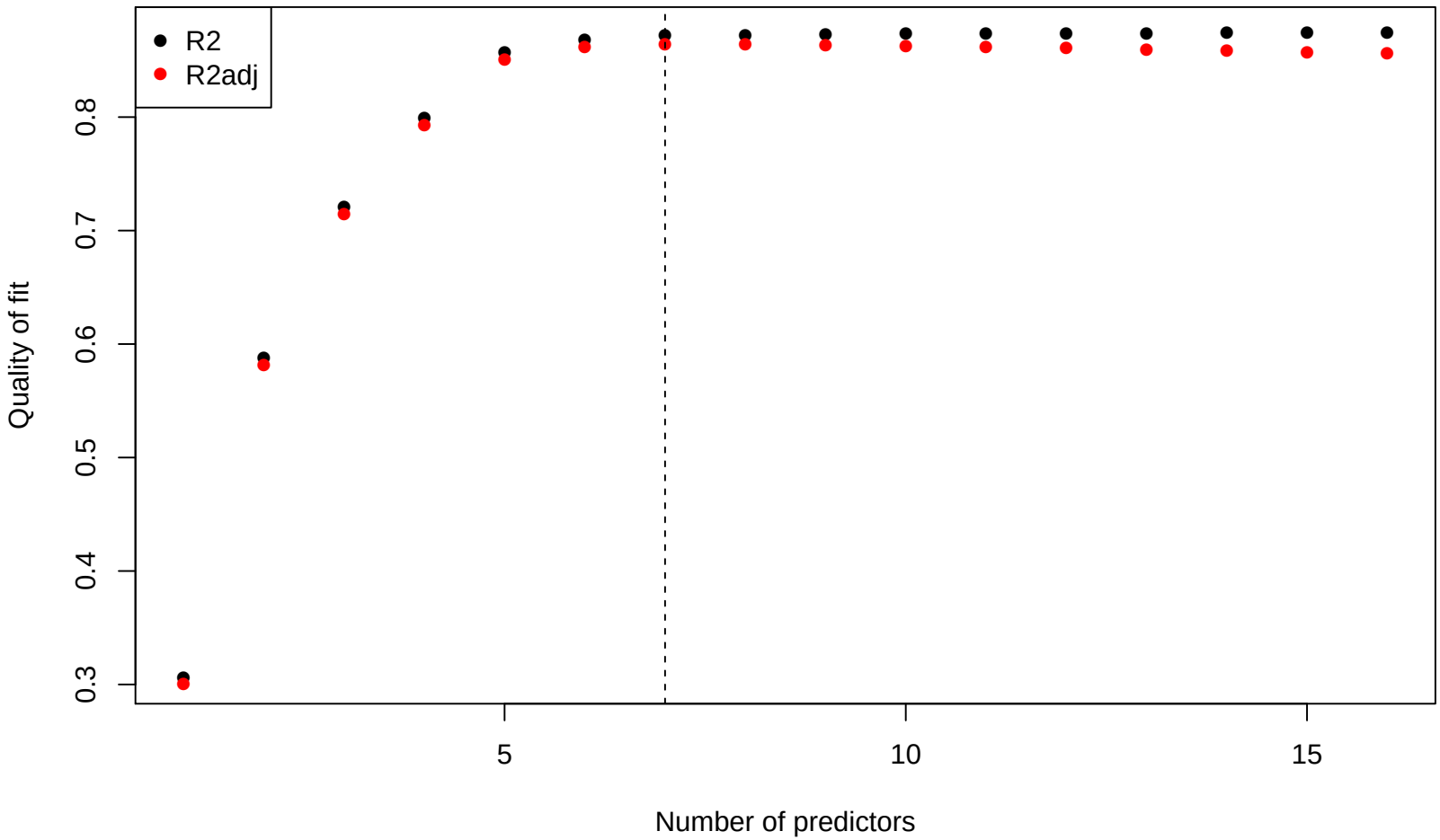
- offers, size, bed bath, brick, N1, N2
- sizeN1, sizeN2, sizebrick
- bedN1, bedN2, bedbrick
- bathN1, bathN2, bathbrick

A total of $2^{16} = 65536$ models.

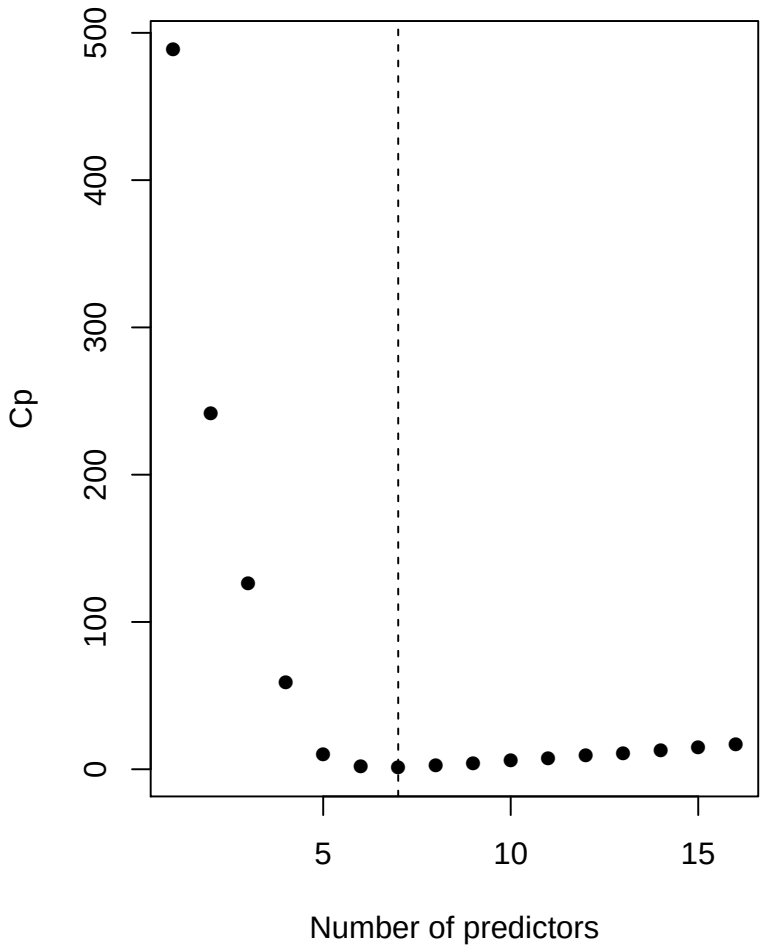
R code

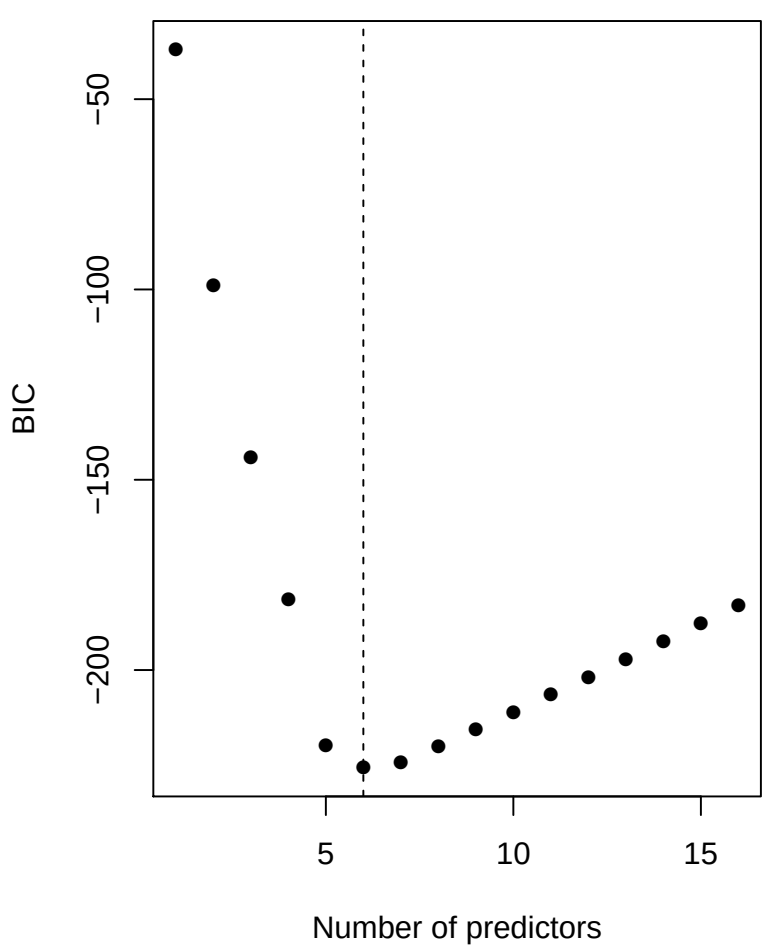
```
file = "http://hedibert.org/wp-content/uploads/2013/11/houseprice.txt"
house = read.table(file,header=TRUE)
house[,4]  = house[,4]*0.092903
house[,8]  = house[,8]*3.2/1000
house = house[,2:8]
attach(house)
n = nrow(house)
brickdum = rep(0,n)
brickdum[house[,4]=="Yes"]=1
N1 = rep(0,n)
N2 = rep(0,n)
N1[house[,1]==1]=1
N2[house[,1]==2]=1

data = cbind(house[,c(2,3,5,6,7)],brickdum,N1,N2,SqFt*N1,SqFt*N2,
SqFt*brickdum,Bedrooms*N1,Bedrooms*N2,Bedrooms*brickdum,
Bathrooms*N1,Bathrooms*N2,Bathrooms*brickdum)

colnames(data) = c("offers","size","bed","bath","Price","brick","N1","N2",
"sizeN1","sizeN2","sizebrick","bedN1","bedN2","bedbrick","bathN1",
"bathN2","bathbrick")

regs = regsubsets(Price~.,data=data,nvmax=16)
```

R^2 and R^2_{adj}


R2
R2adj
Quality of fit
0.3
0.4
0.5
0.6
0.7
0.8
5
10
15
Number of predictors

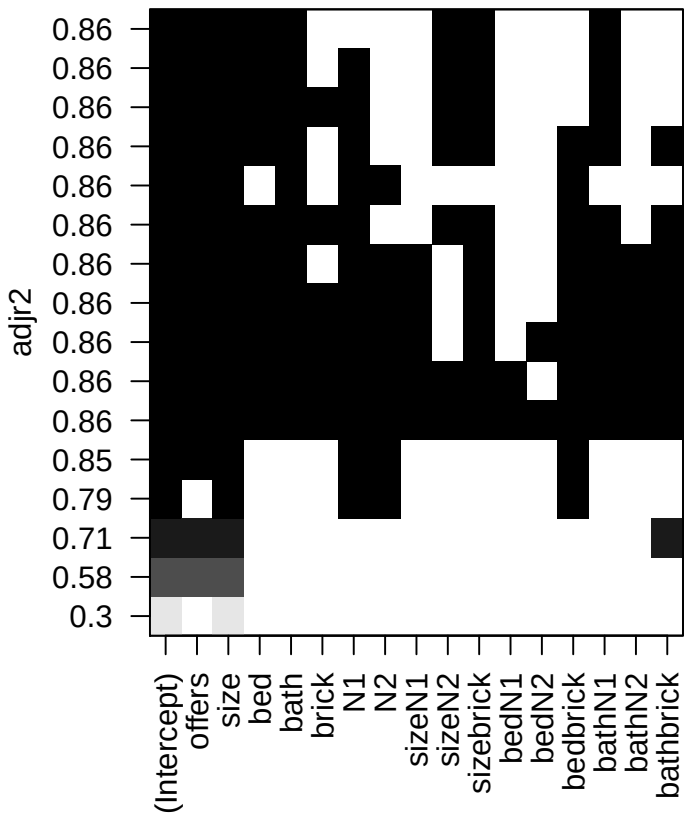
C_p and BIC

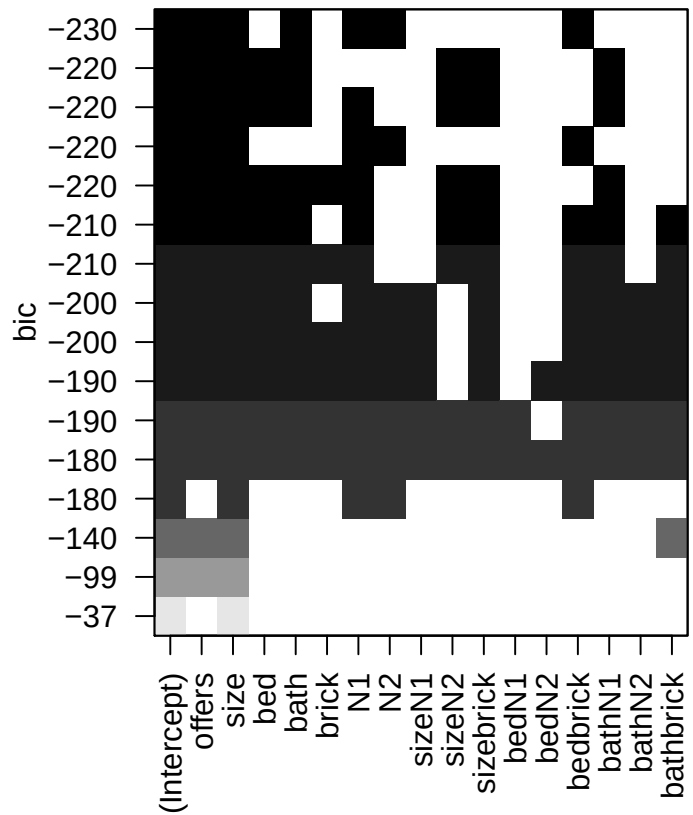

Cp
0
100
200
300
400
500
5
10
15
Number of predictors


BIC
−200
−150
−100
−50
5
10
15
Number of predictors

Top models

```
> coef(regs,6)
(Intercept)      offers        size        bath          N1          N2    bedbrick
 113.130607  -25.539469    1.886008   22.117475  -75.686575  -76.119398   18.037888

> coef(regs,7)
(Intercept)      offers        size         bed        bath      sizeN2   sizebrick      bathN1
 36.9663548 -27.7386190   1.9374668  12.4249439  32.7888754  -0.3614412   0.2907379 -26.0702239
```





Subset Selection[1]

If we just "throw in a ton of x's" our model may be too complex, we may overfit.

Often, we try to start with a "ton of x's" and then see how many we can throw out and still have good fit.

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip} + \varepsilon_i$$

Throwing out an x is equivalent to setting its coefficient to 0.

[1] This and the following 5 slides are taken from Rob McCulloch's personal notes.

The bias variance trade-off

Which coefficients do we set to 0?

The key idea is *the bias variance trade-off !!!*

If we set too many coefficients to 0, we may be throwing out some variables that do important work in explaining y $\Rightarrow$ *bias*.

If we keep too many variables, it may be difficult to get good estimates of all the corresponding coefficients $\Rightarrow$ *variability*.

Which subset to pick?

Our basic problem is that there are a lot of possible way to pick a subset of variables to keep!!

Let k denote the number of variables kept.

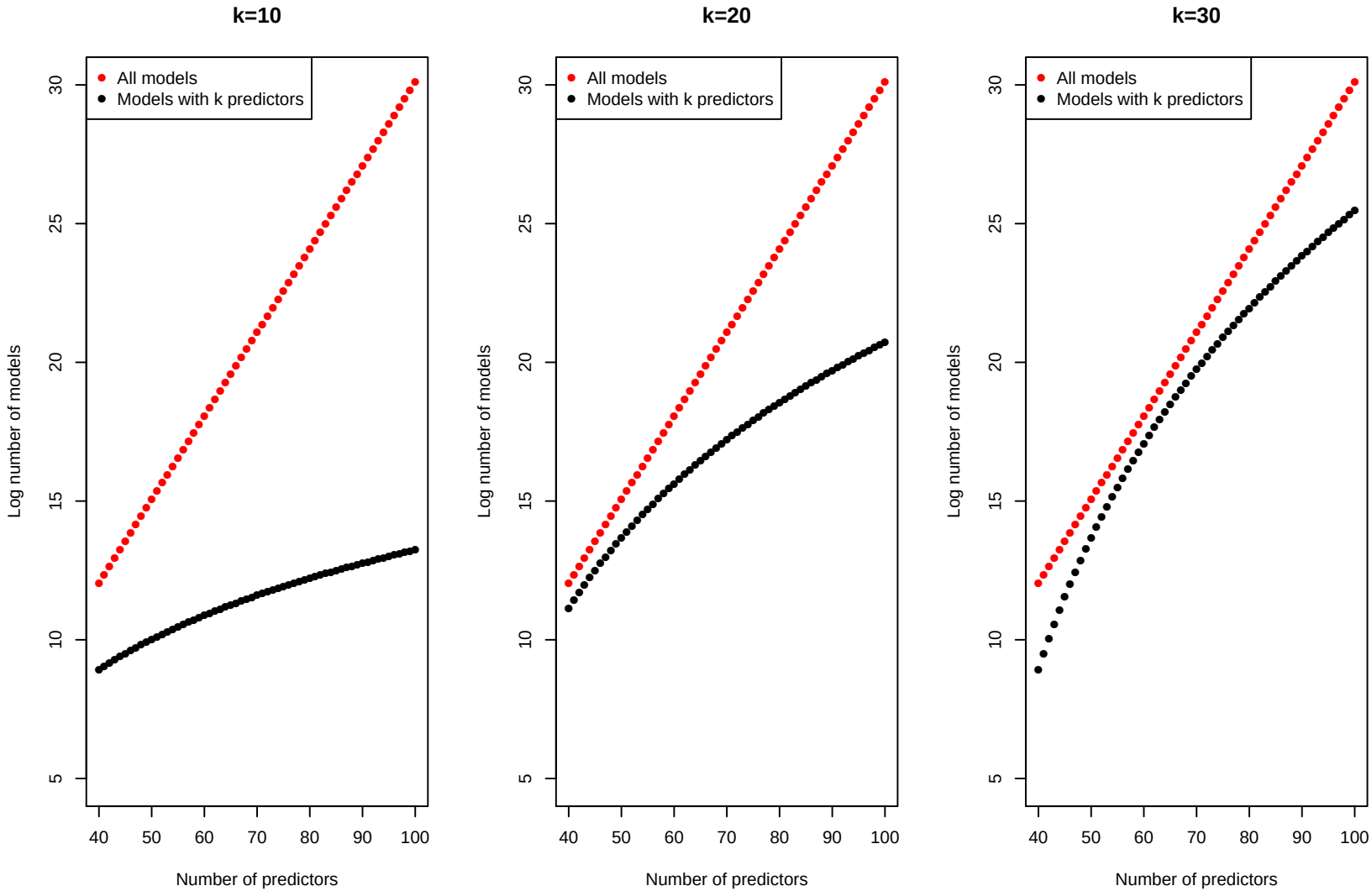
How many ways can you choose k from p: $\left(\begin{array}{c} p \\ k \end{array} \right) = \frac{p!}{k!\,(p-k)!}$.

And, summing over possible $k = 0, 1, 2, \ldots, p$, there are 2^p possible regression models.

Example, $p = 40$ and $k = 10$

$$\begin{aligned} 2^{40} &= 1,099,511,627,776 \\ \left(\begin{array}{c} 40 \\ 10 \end{array} \right) &= 847,660,528 \end{aligned}$$

Way too many models!



What we need is a simple way to move from simpler models to more complex models.

In subset selection we will let k denote the number of variables used, so that k goes from 0 to p.

For each k we will choose a single regression model from the $\begin{pmatrix} p \\ k \end{pmatrix}$ possible models.

All subsets versus stepwise selection

Two possible ways of choosing a subset (a model) given k are:

small p: *All subsets*

For p less than about 40, it is possible to run all the possible regressions. Given the number of variables k, we will pick the subset of variables of size k with the highest R^2.

big p: *Forward Stepwise Selection*

- Start with $k = 0$, no variables selected.
- Given a current k and corresponding subset, add in the new variable which gives you the biggest increase in R^2.
- Stop at $k = p$.

This is a greedy forward search

A simple validation set approach simply splits the data into train and validate, and sees which value of k gives the best prediction.
Or, we could use cross validation.

Training versus testing

For $\alpha \in (0, 1)$

- Randomly split $\boldsymbol{y}$ into
  - $\boldsymbol{y}_1$ - $100\alpha\%$ for training
  - $\boldsymbol{y}_2$ - $100(1-\alpha)\%$ for testing
- Split similarly $\boldsymbol{X}$ into $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$

For $k = 1, \ldots, p$

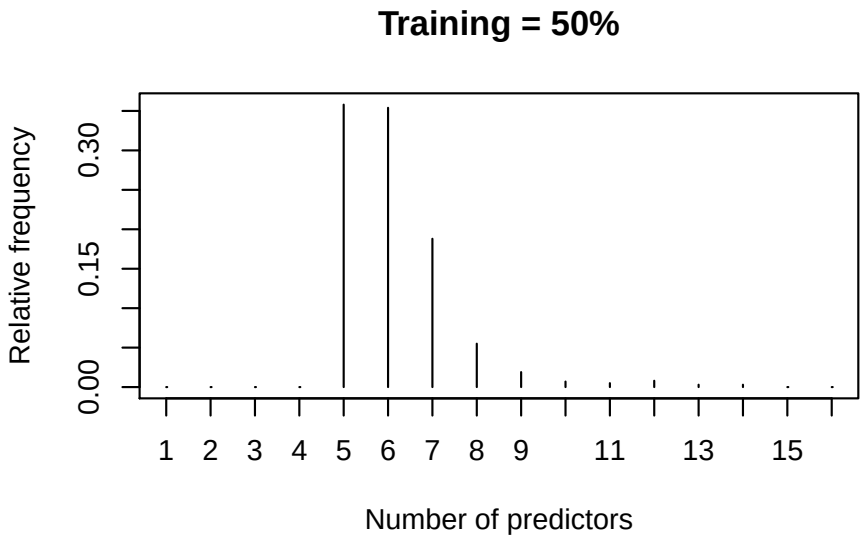
- Use $(\boldsymbol{y}_1, \boldsymbol{X}_1)$ to find *best* model with k predictors by minimizing RSS
- Let $\hat{\boldsymbol{\beta}}_k$ be the estimated coefficients of the *best* model
- Compute the RSS_k based on the testing set

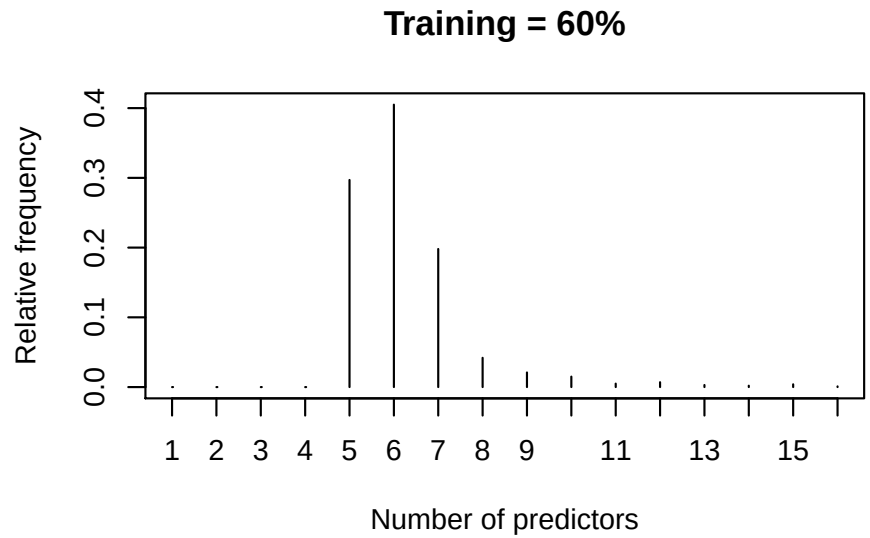
$$RSS_k = (\boldsymbol{y}_2 - \boldsymbol{X}_{2k}\hat{\boldsymbol{\beta}}_k)'(\boldsymbol{y}_2 - \boldsymbol{X}_{2k}\hat{\boldsymbol{\beta}}_k)$$

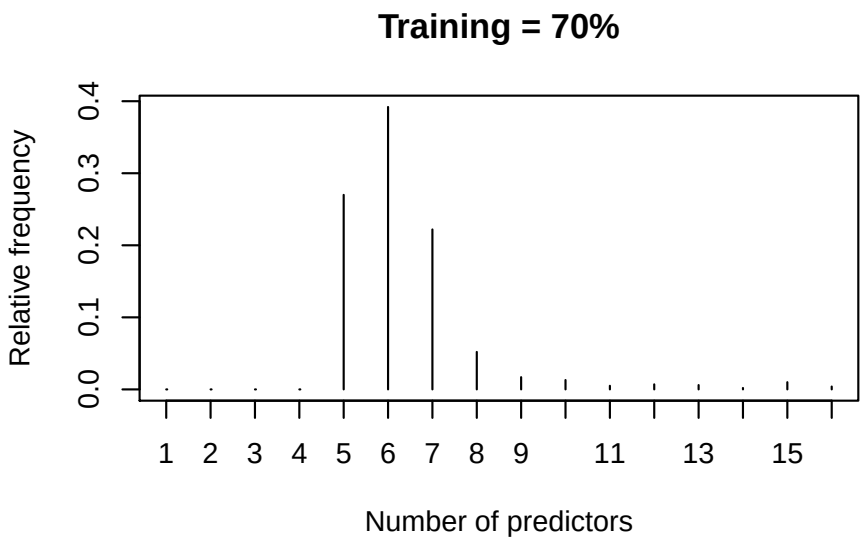
- Select k^* that minimizes RSS_k

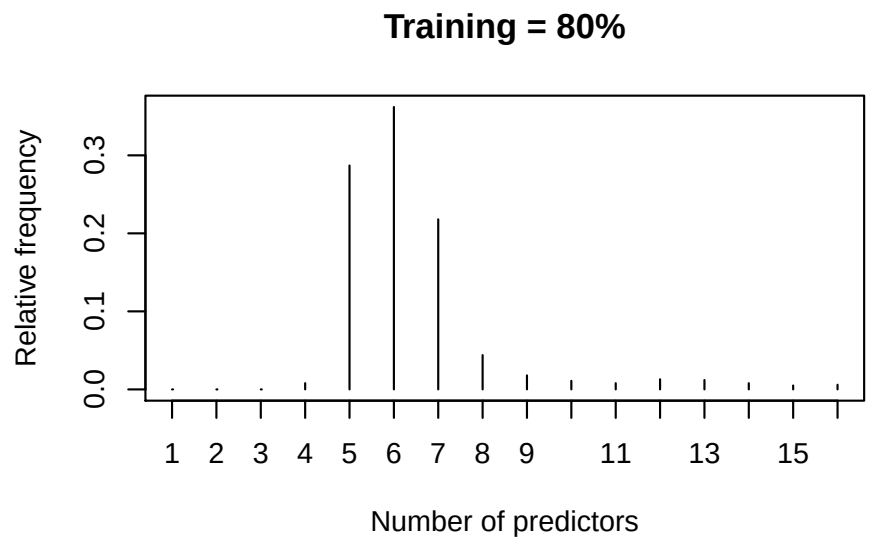
Repeat the above two-step scheme N times

Training versus testing









R code

```
set.seed(31415)
alpha = 0.5
N = 1000
pbests = rep(0,N)

for (j in 1:N){
train = sample(c(TRUE,FALSE),size=n,rep=TRUE,prob=c(alpha,1-alpha))
test = !train
reg.test = regsubsets(Price~.,data=data[train,],nvmax=16)
test.mat = model.matrix(Price~.,data=data[test,])
val.errors = rep(0,p)
for (i in 1:p){
  coefi = coef(reg.test,id=i)
  pred = test.mat[,names(coefi)]%*%coefi
  val.errors[i] = mean((data$Price[test]-pred)^2)
}
pbests[j] = which.min(val.errors)
}

nmodel = rep(0,p)
for (i in 1:N)
  nmodel[pbests[i]] = nmodel[pbests[i]] +1

plot(1:p,nmodel/N,type="h",xlab="Number of predictors",ylab="Relative frequency",axes=FALSE)
axis(2);box()
axis(1,at=1:p)
title(paste("Training = ",100*alpha,"%",sep=""))
```

Credit data

See Figure 3.5, page 83, of *An Introduction to Statistical Learning*

`http://www-bcf.usc.edu/~gareth/ISL/Credit.csv`

Sample size: $n = 400$ individuals

Covariates:

- `balance`: average credit card debt
- `cards`: number of credit cards
- `education`: years of education
- `income`: income in thousands of dollars
- `limit`: credit limit
- `rating`: credit rating
- `age`: Age in years
- `gender`: Male, Female
- `student`: Yes, No
- `married`: Yes, No
- `ethnicity`: Caucasian, African American, Asian

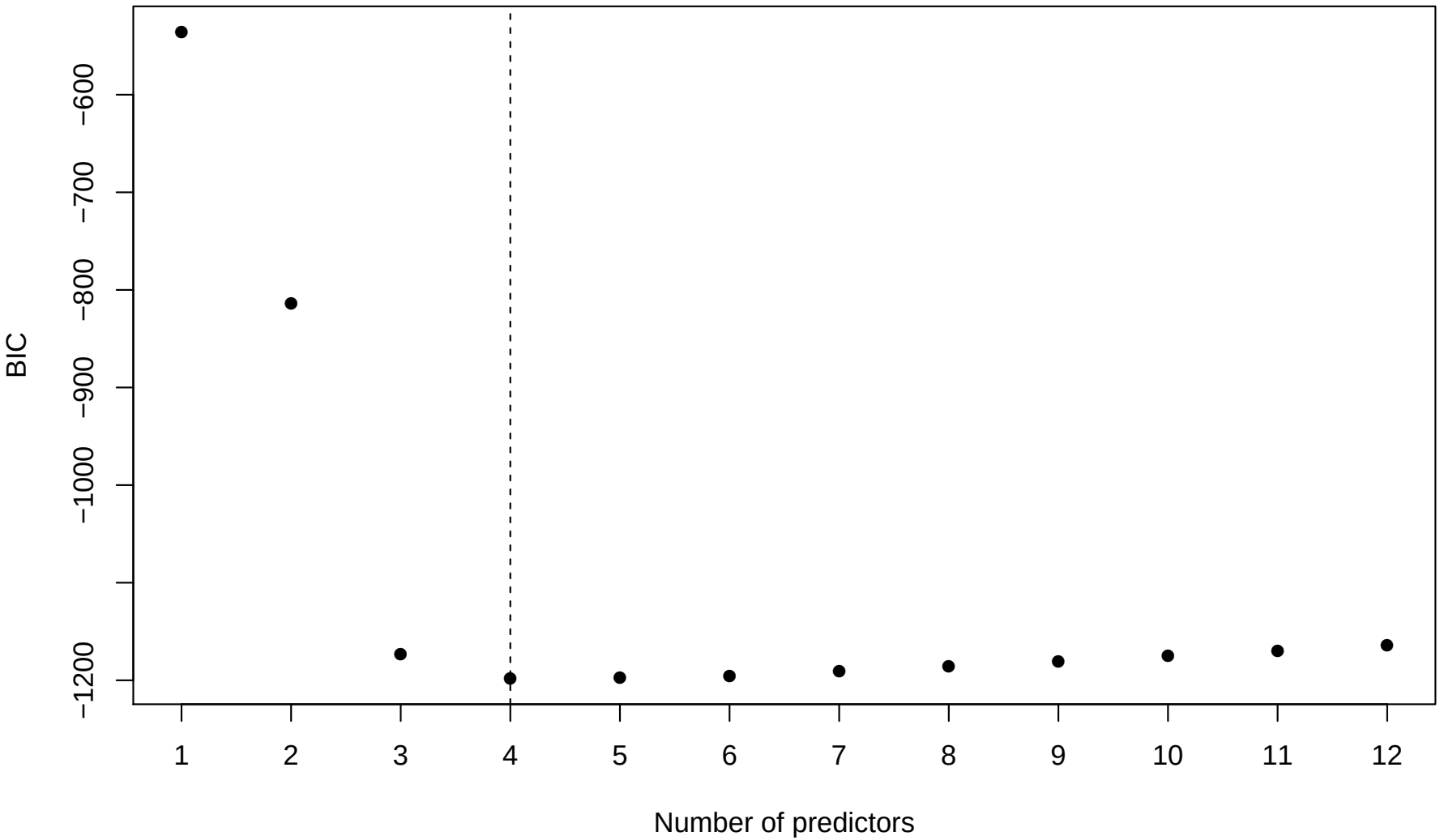
```
> credit[1:10,2:12]
    Income Limit Rating Cards Age Education Gender Student Married        Ethnicity Balance
1   14.891  3606    283     2  34        11   Male      No     Yes        Caucasian     333
2  106.025  6645    483     3  82        15 Female     Yes     Yes            Asian     903
3  104.593  7075    514     4  71        11   Male      No      No            Asian     580
4  148.924  9504    681     3  36        11 Female      No      No            Asian     964
5   55.882  4897    357     2  68        16   Male      No     Yes        Caucasian     331
6   80.180  8047    569     4  77        10   Male      No      No        Caucasian    1151
7   20.996  3388    259     2  37        12 Female      No      No African American     203
8   71.408  7114    512     2  87         9   Male      No      No            Asian     872
9   15.125  3300    266     5  66        13 Female      No      No        Caucasian     279
10  71.061  6819    491     3  41        19 Female     Yes     Yes African American    1350
```

Summary statistics

```
> summary(credit[,2:6])
     Income           Limit           Rating          Cards            Age
 Min.   : 10.35   Min.   :  855   Min.   : 93.0   Min.   :1.000   Min.   :23.00
 1st Qu.: 21.01   1st Qu.: 3088   1st Qu.:247.2   1st Qu.:2.000   1st Qu.:41.75
 Median : 33.12   Median : 4622   Median :344.0   Median :3.000   Median :56.00
 Mean   : 45.22   Mean   : 4736   Mean   :354.9   Mean   :2.958   Mean   :55.67
 3rd Qu.: 57.47   3rd Qu.: 5873   3rd Qu.:437.2   3rd Qu.:4.000   3rd Qu.:70.00
 Max.   :186.63   Max.   :13913   Max.   :982.0   Max.   :9.000   Max.   :98.00
```

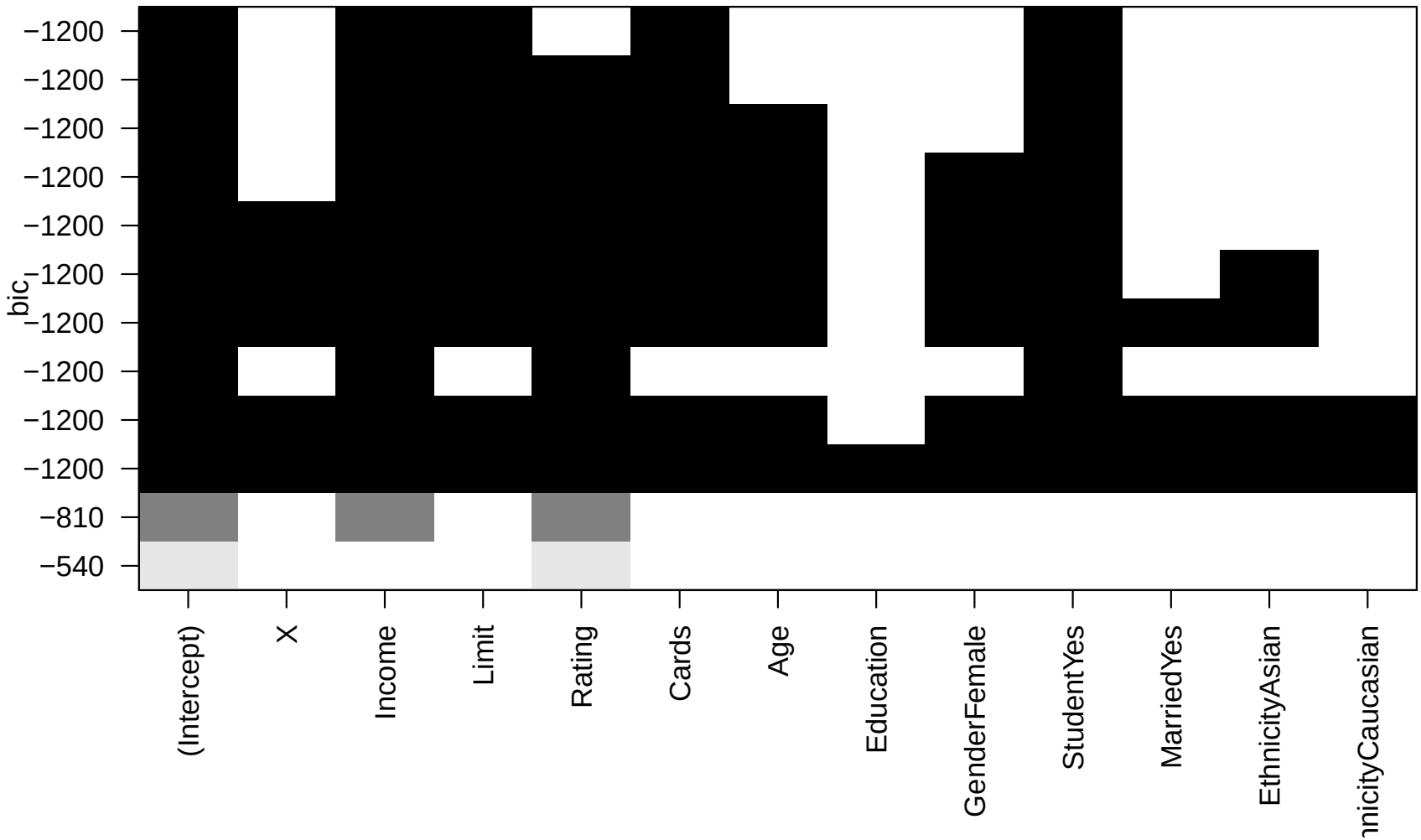
```
> summary(credit[,7:12])
   Education        Gender    Student   Married                Ethnicity      Balance
 Min.   : 5.00    Male :193   No :360   No :155   African American: 99   Min.   :   0.00
 1st Qu.:11.00   Female:207   Yes: 40   Yes:245   Asian            :102   1st Qu.:  68.75
 Median :14.00                                   Caucasian        :199   Median : 459.50
 Mean   :13.45                                                           Mean   : 520.01
 3rd Qu.:16.00                                                           3rd Qu.: 863.00
 Max.   :20.00                                                           Max.   :1999.00
```

Selecting via BIC

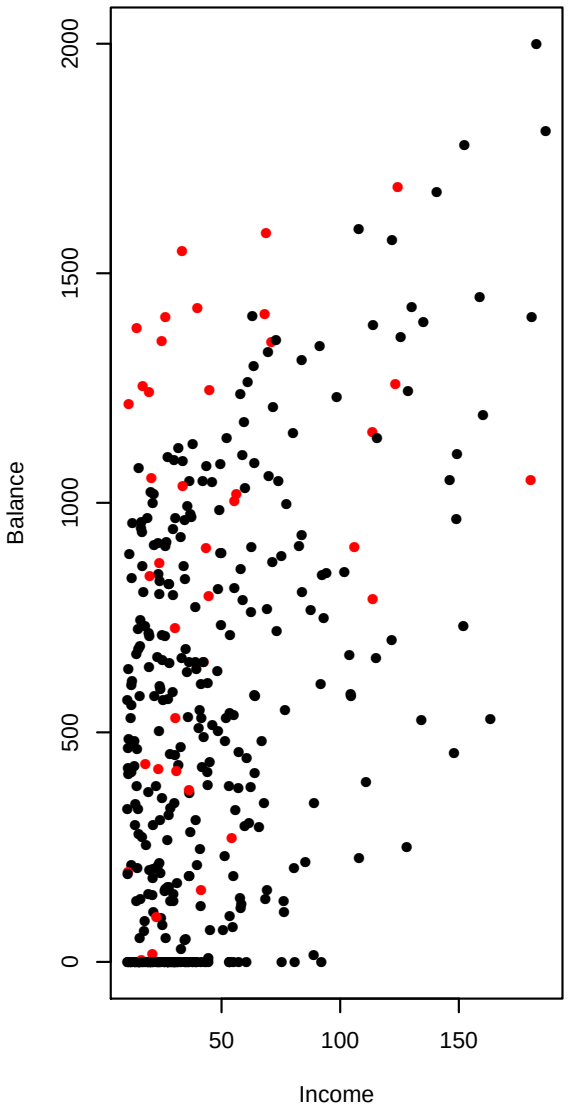


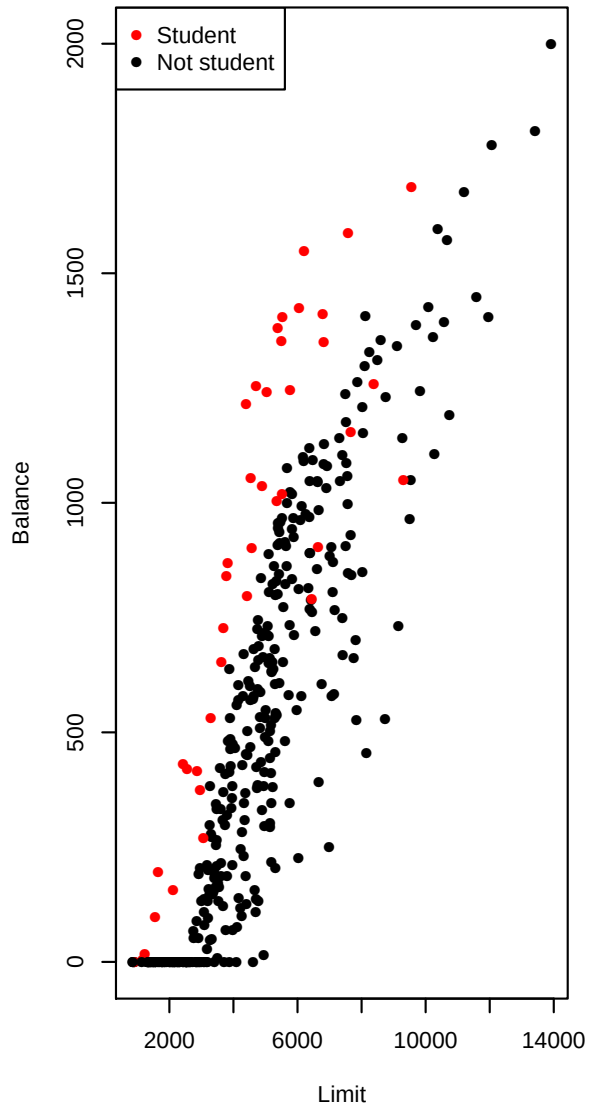
Question: Why 12 predictors?

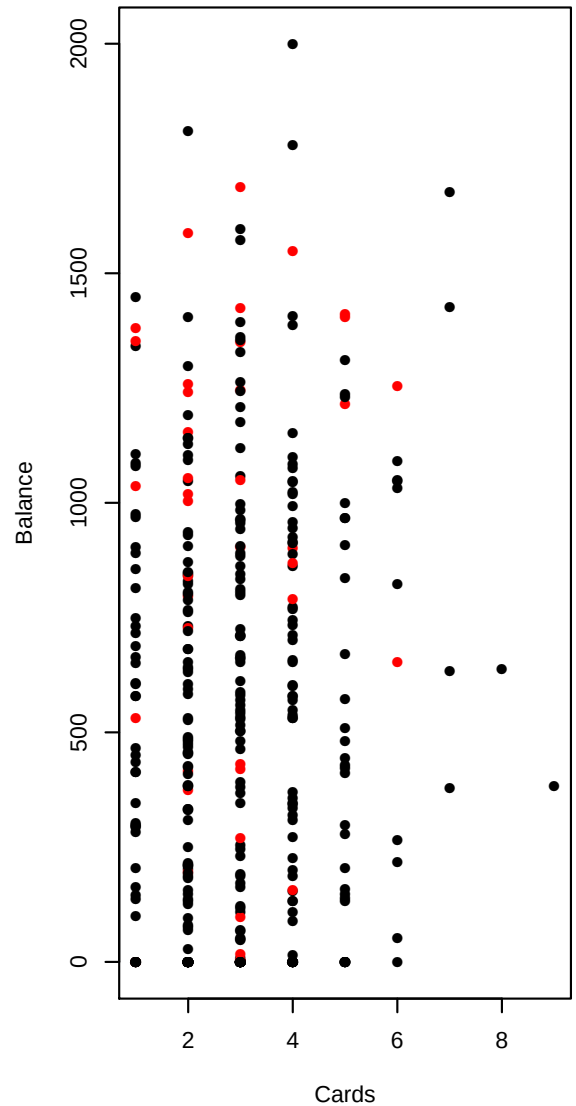
Selecting via BIC



Selected covariates





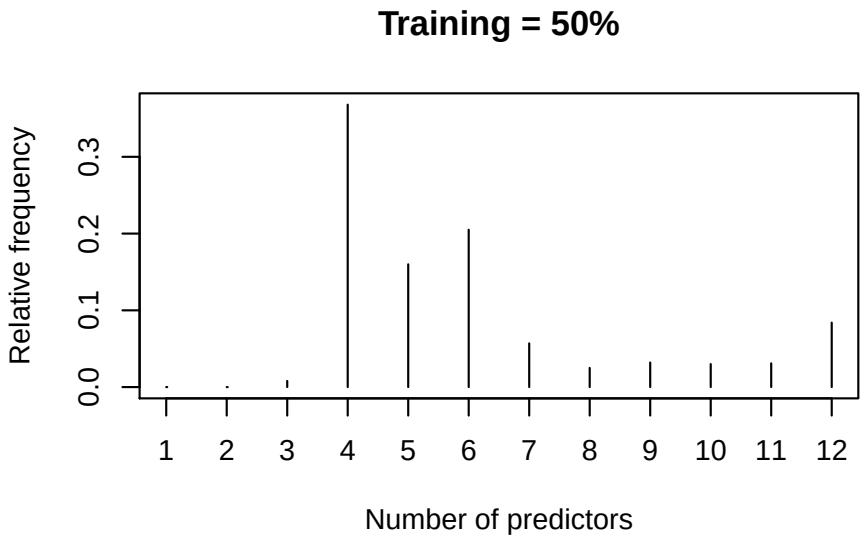


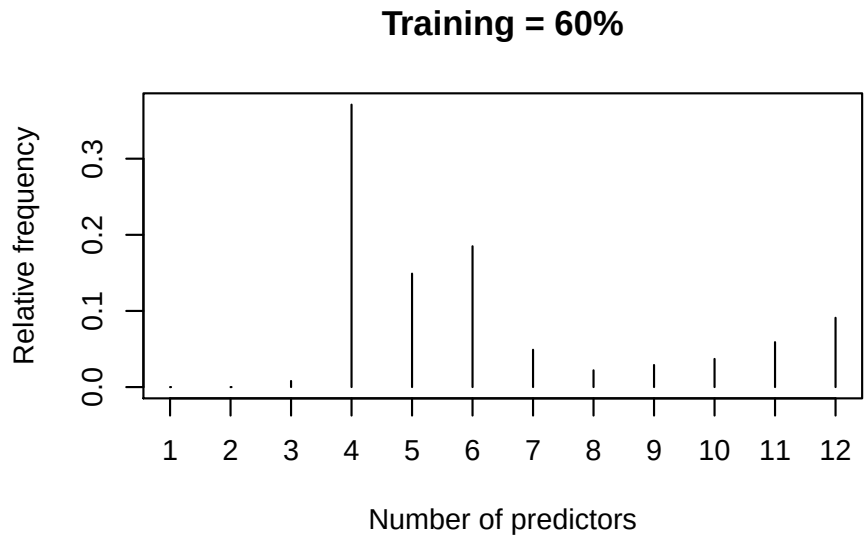
Best model

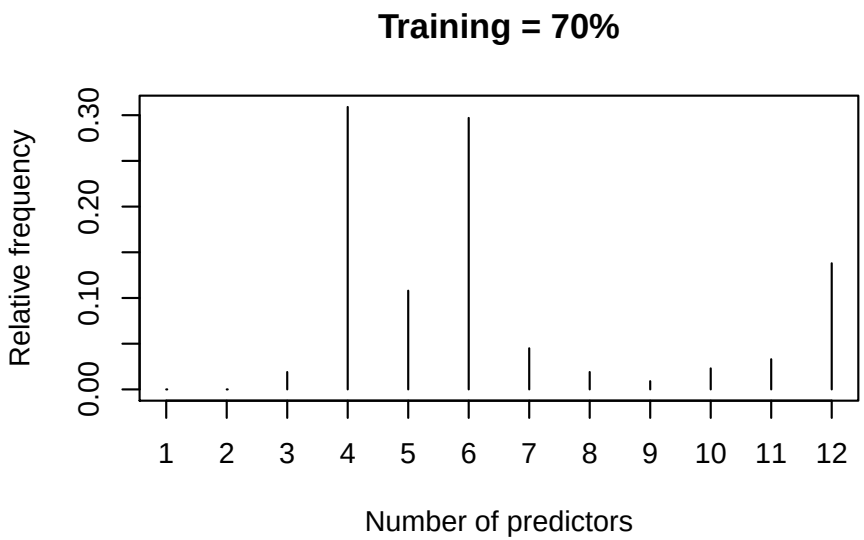
$$\texttt{Balance} = -499.7 - 7.84\texttt{Income} + 0.267\texttt{Limit} + 23.18\texttt{Cards} + 429.6\texttt{Student}$$

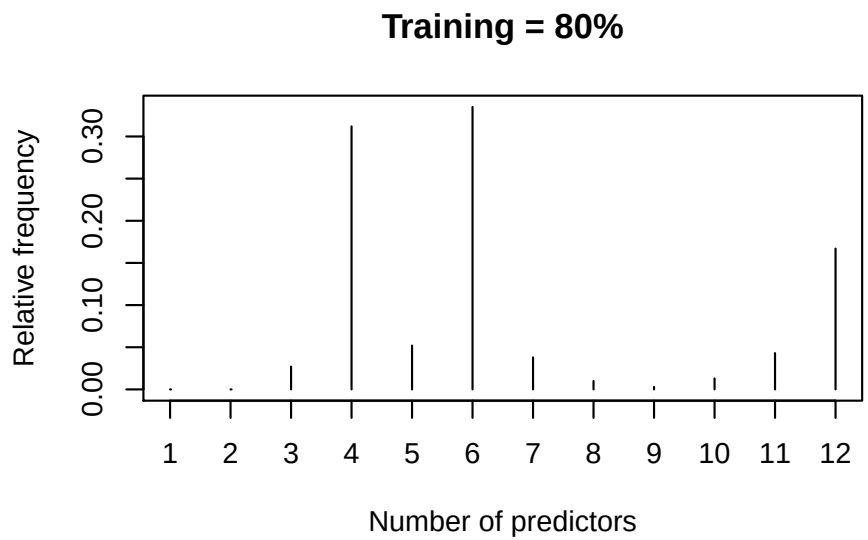
```
> bestmodel = lm(Balance~Income+Limit+Cards+Student,data=credit)
>
> summary(bestmodel)

Call:
lm(formula = Balance ~ Income + Limit + Cards + Student, data = credit)

Residuals:
    Min      1Q  Median      3Q     Max
-202.04  -80.41  -10.51   53.98  334.10

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -4.997e+02  1.589e+01 -31.449  < 2e-16 ***
Income      -7.839e+00  2.321e-01 -33.780  < 2e-16 ***
Limit        2.666e-01  3.542e-03  75.271  < 2e-16 ***
Cards        2.318e+01  3.639e+00   6.368 5.32e-10 ***
StudentYes   4.296e+02  1.661e+01  25.862  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 99.56 on 395 degrees of freedom
Multiple R-squared:  0.9536,Adjusted R-squared:  0.9531
F-statistic:  2029 on 4 and 395 DF,  p-value: < 2.2e-16
```

Training and testing









Outline

Shrinkage-L2, Ridge Regression

Our variable selection approach set some of the coefficients in a multiple regression to 0.

This helped keep our model simple so that we do not overfit.

Another way to keep our model "simple" is to *push* or *shrink* the coefficient towards 0.

This way a coefficient will only be large if the data demands it!

Ridge Regression:

Recall that least squares works by picking the coefficients to minimize

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

Ridge regression works by mimimizing:

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

For large λ you pay a price to make a coefficient large !!

Minimize:

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

λ will be our "walk the bias-variance trade-off" parameter.

small λ: can have big coefficient $\Rightarrow$ *complex model.*

big λ: can't have many big coefficients $\Rightarrow$ *simple model.*

So, for every λ, you will get a different optimizing β:

$$\lambda \Rightarrow \hat{\beta}_{\lambda}^{R}.$$

For example $\hat{\beta}_{0}^{R}$ is just the least squares estimator.

How do you choose λ ?

cross-validation, or another out-of-sample criterion!!.

Note:

We are minimizing

fit: $\sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij})^2$

$+$

penalty: $\lambda \sum_{j=1}^{p} \beta_j^2$.

Since the penalty treats all the β_j the same you have to be thinking about all the x's the same. What are the units of β_j?

Usually people *standardize* the x's before the do this kind of shrinkage.

Hitters dataset

Let's look at the "Hitters" example used in the Lab in the ISLR book.

Major League Baseball Data from the 1986 and 1987 seasons.

A data frame with 322 observations of major league players on 20 variables.

Each observation corresponds to a baseball player.

Variables

Salary: 1987 annual salary on opening day in thousands of dollars

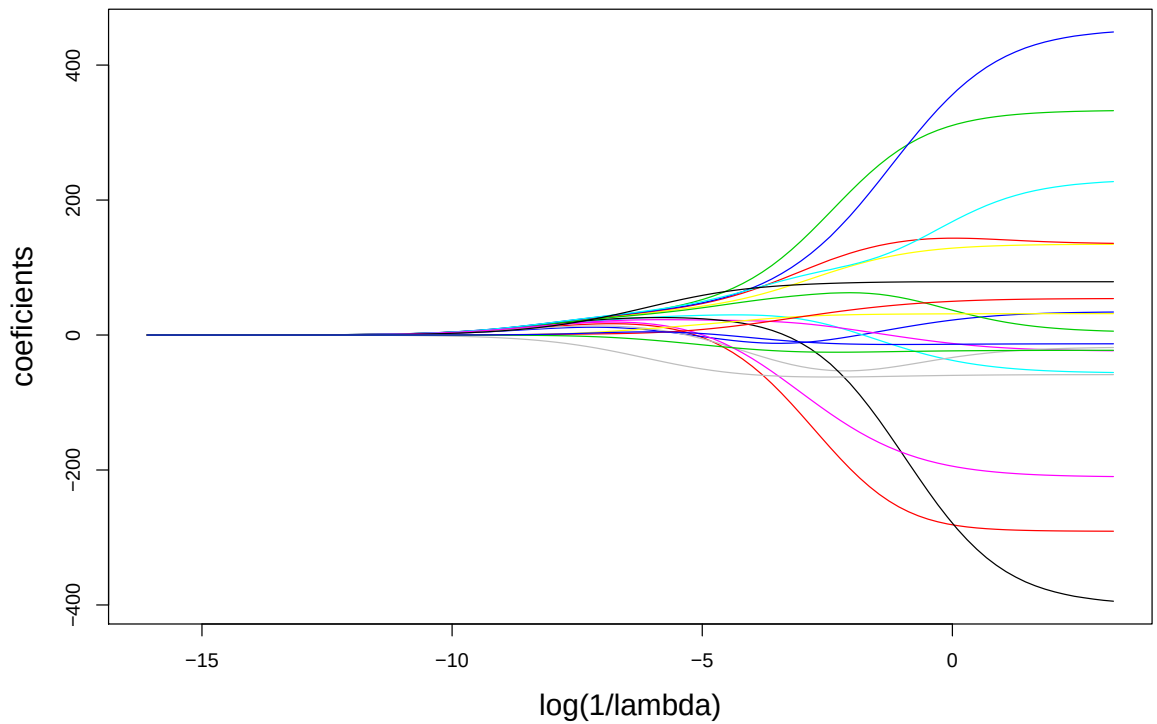
AtBat: Number of times at bat in 1986
Hits: Number of hits in 1986
HmRun: Number of home runs in 1986
Runs: Number of runs in 1986
RBI: Number of runs batted in 1986
Walks: Number of walks in 1986

CAtBat: Number of times at bat during his career
CHits: Number of hits during his career
CHmRun: Number of home runs during his career
CRuns: Number of runs during his career
CRBI: Number of runs batted in during his career
CWalks: Number of walks during his career
Years: Number of years in the major leagues

League: A factor with levels A and N indicating player's league at the end of 1986
Division: A factor with levels E and W indicating player's division at the end of 1986
PutOuts: Number of put outs in 1986
Assists: Number of assists in 1986
Errors: Number of errors in 1986
NewLeague: A factor with levels A and N indicating player's league at the beginning of 1987

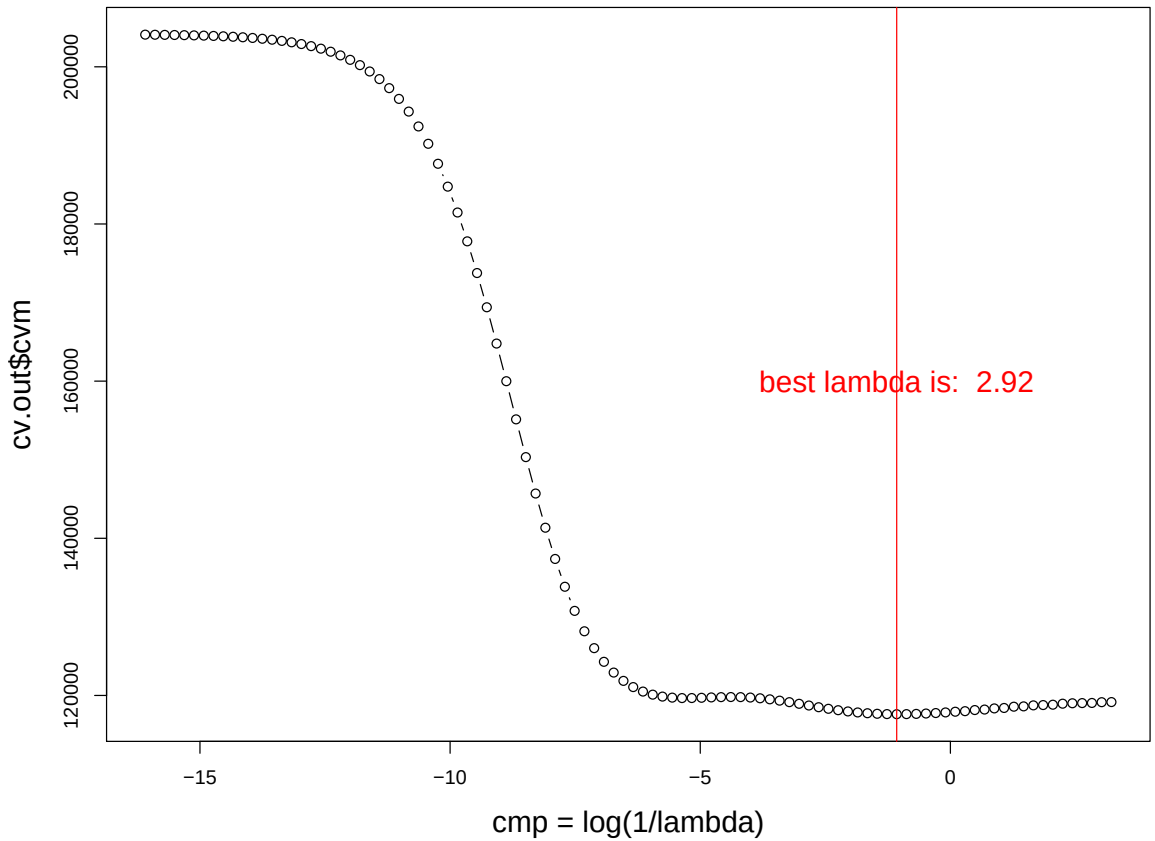
Let's try Ridge regression with the Hitters data.
I standardized all the x's.

Here we plot $\log(1/\lambda)$ vs. $\hat{\beta}^R_\lambda$.



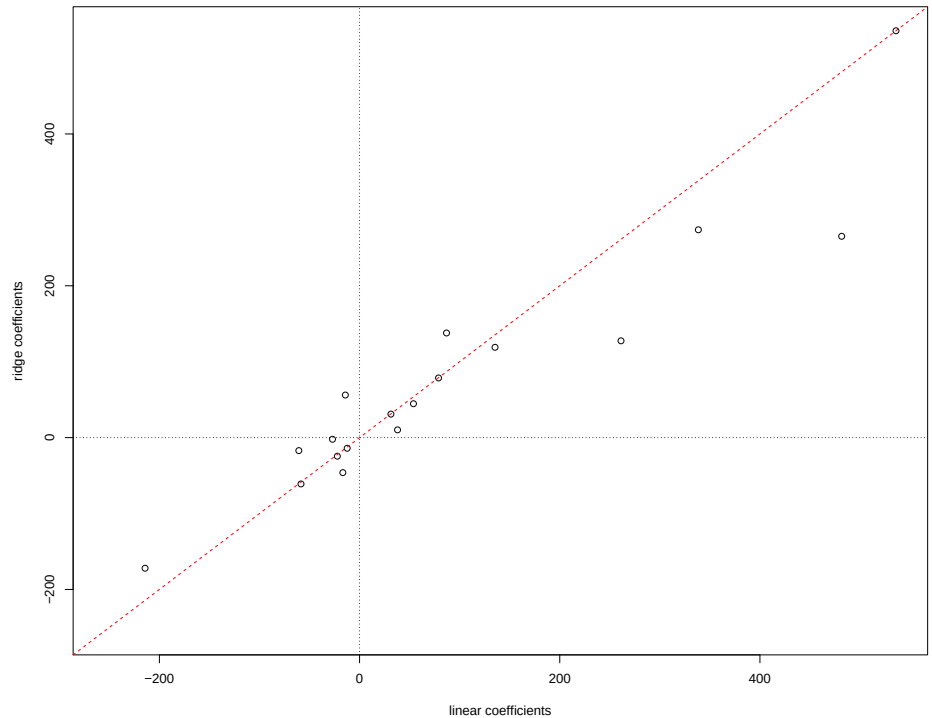
A *complex* model is one where the coefficients are allowed to be big.

Here is the cross-validation estimate of the out of sample loss.



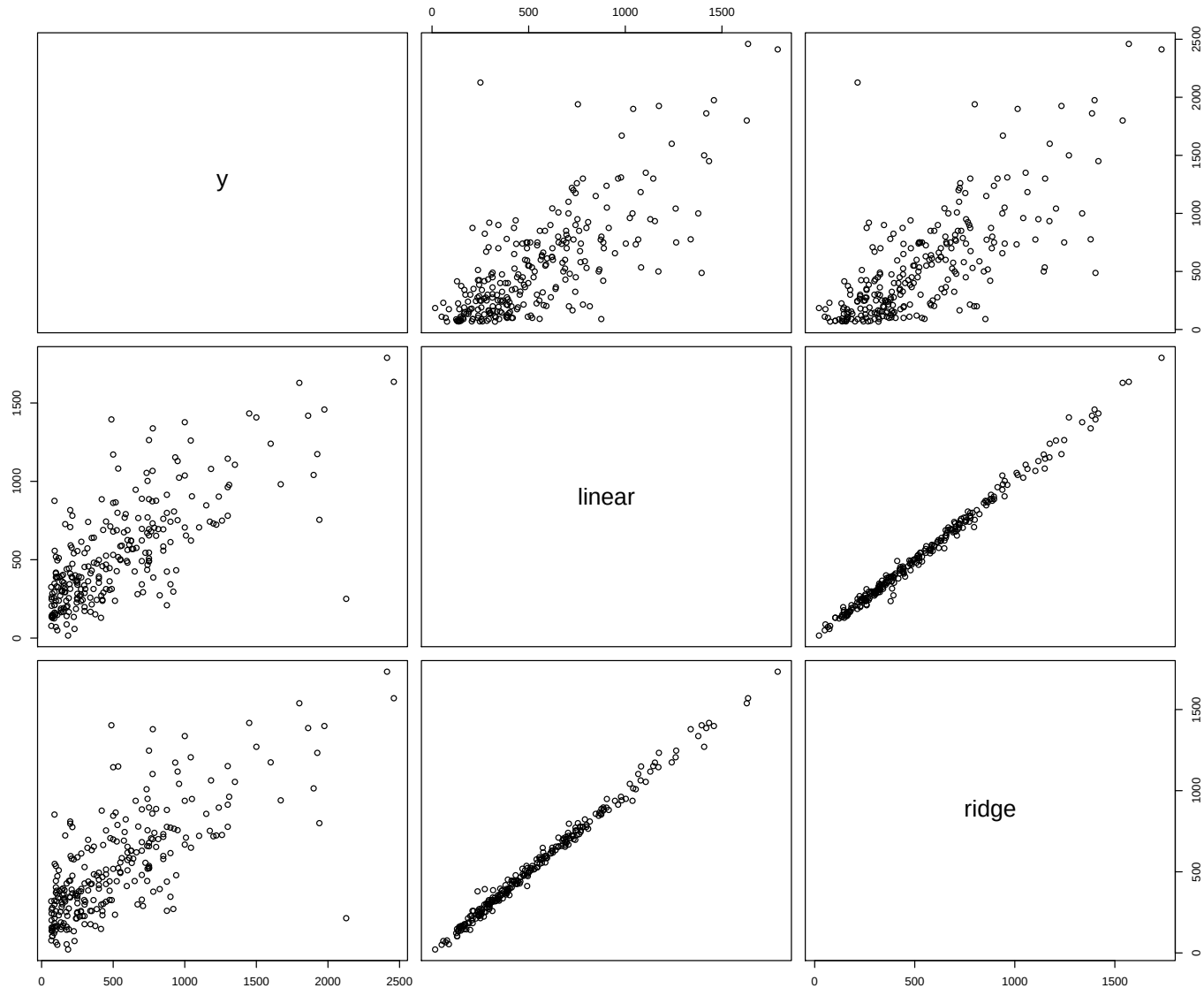
$\log(1/2.92) = -1.071584$.

Here we plot the coefficients from linear regression against those we get using ridge regression with the optimal λ.



They are not too different in this case.
You can see some of the bigger coefficients are shrunk a bit.
A lot of the coeficients are close to 0, (we standardized the x's).
The x's with absolute values bigger than 100 are "AtBat" "Hits" "Walks" "CAtBat" "CHits" "CRuns" "CRBI" "CWalks"

Here we compare the in-sample fits from regression and ridge.



What is the ridge regression $\hat{\beta}_\lambda^R$?

Using a basic Lagrange multiplier argument, $\hat{\beta}_\lambda^R$ is the solution of

$$\min_\beta L_\lambda(\beta)$$

where

$$L_\lambda(\beta) = \sum_{i=1}^{n}(y_i - x_i'\beta)^2 + \lambda \sum_{j=1}^{p} \beta_j^2.$$

Taking derivatives and equating to zero (1st order conditions) leads to

$$\lambda\beta = X'(y - X\beta)$$

or

$$\hat{\beta}_\lambda^R = (X'X + \lambda I_p)X'y.$$

Constrained minimization

It is also useful to view the problem as a constrained fit:

$$\min_{\beta} \sum_{i=1}^{n} (y_i - x_i'\beta)^2 \quad \text{such that} \quad \sum_{j=1}^{p} \beta_j^2 \leq \kappa.$$

If OLS leads to $\sum_{j=1}^{p} \hat{\beta}_j^2 \leq \kappa$, then there is no problem. Otherwise, the constraint is "active".

If $f(\beta) = \sum_{i=1}^{n} (y_i - x_i'\beta)^2$ and $g(\beta) = \sum_{j=1}^{p} \hat{\beta}_j^2$, then the problem of

$$\min_{\beta} f(\beta) \quad \text{such that} \quad g(\beta) = \kappa,$$

is simpler.

At the minimum, $\hat{\beta}_\lambda^R$,

$$\nabla f + \lambda \nabla g = 0, \quad \text{for} \quad \lambda > 0.$$

We can easily solve the first order conditions:

$$\begin{aligned} -\nabla f' &= 2X'(y - X\beta) \\ \nabla g' &= 2\beta \end{aligned}$$

so

$$2\lambda\beta = 2X'(y - X\beta)$$

and

$$\hat{\beta}_\lambda^R = (X'X + \lambda I_p)^{-1} X'y$$

We would then solve (the easy problem) of finding the λ such that $||\hat{\beta}_\lambda^R||^2 = \kappa$.

Karush Kuhn Tucker (KKT) conditions[2]

Note that this is an example of the Karush-Kuhn-Tucker approach.

To minimize $f(\beta)$ subject to $g(\beta) \leq 0$, form

$$L(\beta, \lambda) = f(\beta) + \lambda g(\beta)$$

and then solve

$$\min_{\beta} max_{\lambda \geq 0} \; L(\beta, \lambda).$$

With $\lambda \geq 0$ we must have $g(\beta) \leq 0$, since otherwise we could get a max of infinity.

[2] Allowing inequality constraints, the KKT approach to nonlinear programming generalizes the method of Lagrange multipliers, which allows only equality constraints.

Also note that at the solution:

$$\lambda^* \, g(\beta^*) = 0.$$

This captures the fact that there are two possibilities:

- If the constraint is *binding* then $g(\beta^*) = 0$.
- If the constraint is not binding so that $g(\beta) < 0$ then the max over non-negative λ is clearly obtained at $\lambda = 0$.

The general form of the KKT theorem

Minimize $f(\beta)$ such that

$$\{h_i(\beta) = 0\} \quad \text{and} \quad \{g_j(\beta) \leq 0\}$$

or

$$\min_{\beta} \max_{\gamma} \max_{\lambda \geq 0} L(\beta, \gamma, \lambda)$$

where

$$L(\beta, \gamma, \lambda) = f(\beta) + \sum \gamma_i h_i(\beta) + \sum \lambda_j g_j(\beta)$$

Just notice that with equality constraints you don't know the sign of the constraint coefficient.

Outline

Shrinkage: The LASSO

The LASSO (least absolute shrinkage and selection operator) changes the form of the penalty.

Now, we minimize:

$$\sum_{i=1}^{n}(y_i - \beta_0 - \beta_1 x_{i1} - \cdots - \beta_p x_{ip})^2 + \lambda \sum_{j=1}^{p} |\beta_j|.$$

This may not seem like a big deal, but it turns out the solution to this problem can set a β_j exactly to 0, so that you get variable selection.

In the LASSO there is shrinkage as well as selection and the shrinkage takes on a different form than in L2 regularization.

Also, with the LASSO, variables can to out as λ decreases, whereas with forward, once you are in, your are always in.

Why do people like the LASSO?

- Simple way to walk the bias variance trade-off.
- Zero coefficients give variable selection, can get more interpretable models.
- Computationally fast.

Stepwise compared to LASSO

LASSO is a quadratic (and hence convex and differentiable) loss function optimized under a convex constraint.

Hence, the LASSO problem has a guaranteed global optimum and we have very efficient algorithms for finding that optimum.

The `stepwise` algorithms are greedy searches so there is no guarantee the global optimum has been found.

But, since they do not shrink, the step wise methods can find more parsimonious solutions (use fewer x's) faster!!

In our Hitters example, the `all subsets` *method ended up using just 6 x's but the LASSO only set two coefficients to 0!!*

Understanding the LASSO Solution

Why does the LASSO give solutions with coeficients at 0?

How is Ridge different from LASSO?

To get a good simple intuition, it is helpful to consider the constained optimization view of LASSO and Ridge.

RIDGE

$$\underset{\beta_0,\beta}{\text{minimize}} \quad \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_{i1} - \cdots - \beta_p x_{ip})^2$$

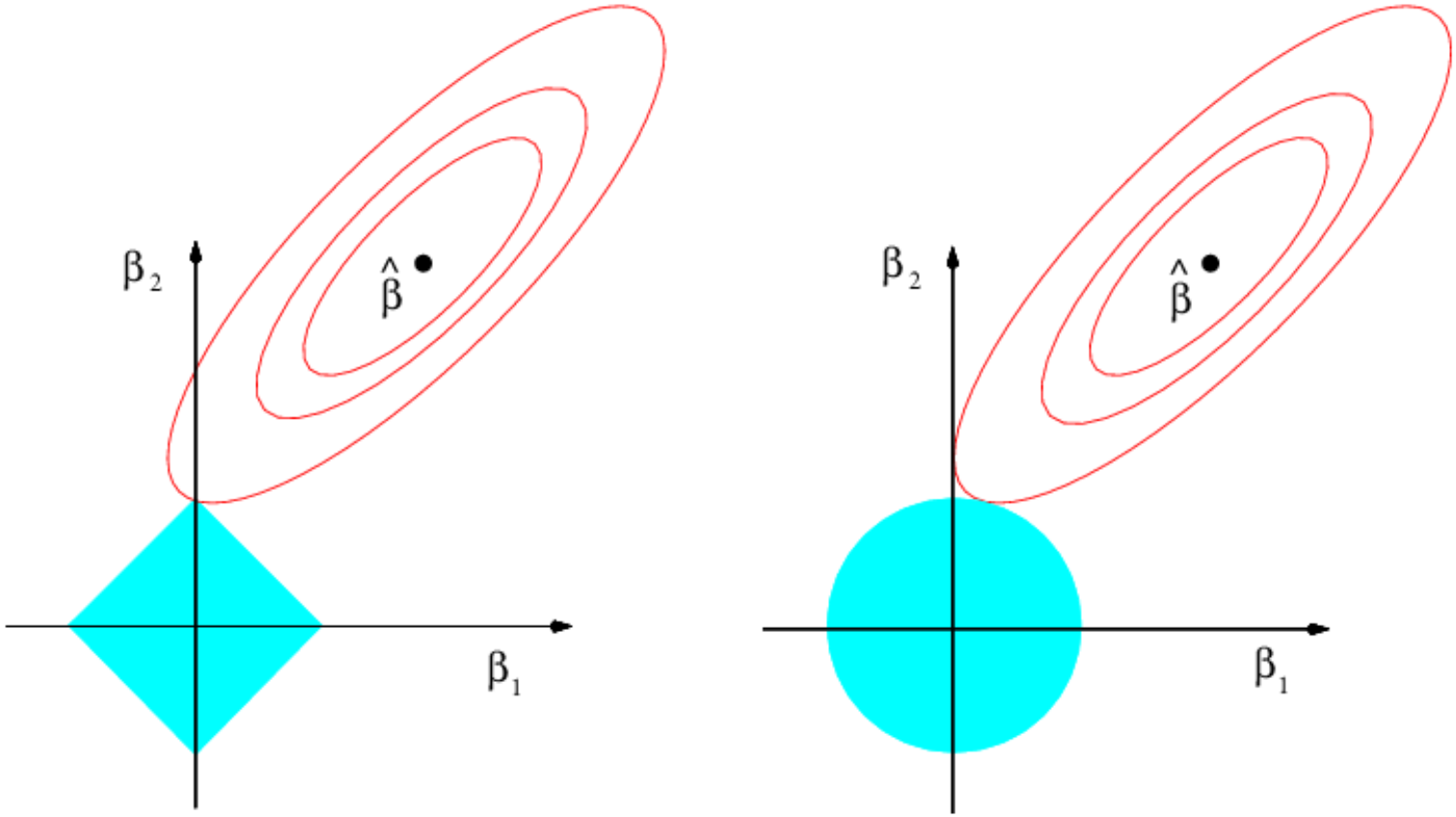
$$\text{subject to} \quad \sum_{j=1}^{p} \beta_j^2 \leq t^2$$

LASSO

$$\underset{\beta_0,\beta}{\text{minimize}} \quad \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_{i1} - \cdots - \beta_p x_{ip})^2$$

$$\text{subject to} \quad \sum_{j=1}^{p} |\beta_j| \leq t$$

This is a very famous picture



Left: LASSO problem, where the constraint set looks like a diamond.

Right: Ridge problem, where the constraint set looks like a circle.

The diamond constraint can give solutions at an axis.

The simplest version

Let's consider the simplest possible version of our problems back in the "Lagrangian" formulation:

Ridge:

$$\underset{\beta}{\text{minimize}} \;\; (y - \beta)^2 + \lambda \, \beta^2$$

LASSO:

$$\underset{\beta}{\text{minimize}} \;\; \tfrac{1}{2} \, (y - \beta)^2 + \lambda \, |\beta|$$

Adding the $1/2$ for the LASSO changes nothing and makes the expressions look nicer.

For the Ridge version we are minimizing a quadatric so we can easily find the global miniumum by setting the derivative equal to 0:

$$2(y - \beta)(-1) + 2\lambda\beta = 0$$

such that

$$\hat{\beta}^R = \frac{y}{1 + \lambda}.$$

Of course the unconstrained solution is

$$\hat{\beta} = y$$

so we can very nicely see how a choice of λ shrinks the estimate towards 0.

For the LASSO problem, we suddenly have a basic technical problem:

$$f(\beta) = |\beta| \quad \text{is not differentiable at 0.}$$

Our function is convex, so there is a global minimum, but can we find it in a simple way?

We can, and the solution will shed light on the LASSO and on how to solve the general regression problem.

Before we derive the LASSO solution, let's see how it works out in practice.

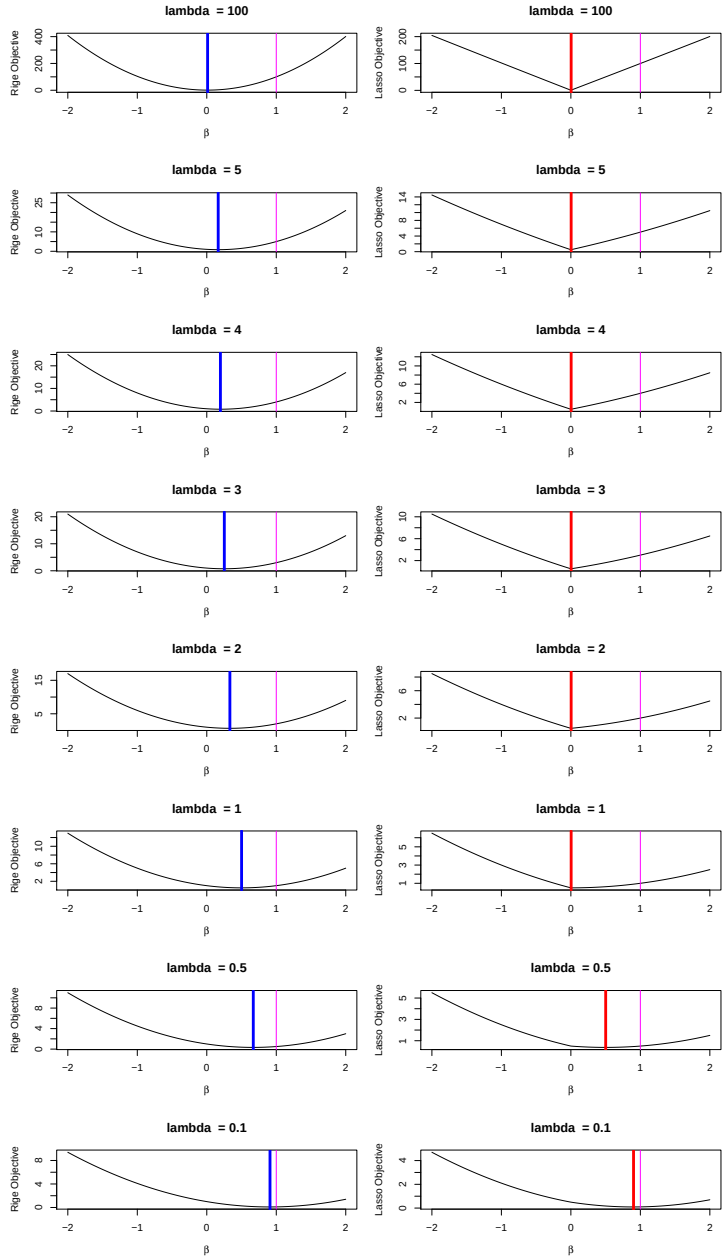
Let's say $y = 1$.
We plot y with the solid magenta line.

λ decreases as we go down the plots.

At left we have the Ridge criterion plotted with the minimizing β indicated by the solid blue line.

At right we have the LASSO criterion plotted with the minimizing β indicated by the solid red line.

Each estimate moves from 0 to 1, but the LASSO estimates sticks at 0 for a while and then moves faster to 1.



To derive the LASSO solution, suppose the optimal β is greater than 0.

Then locally our differential first order conditions apply *and* our criterion is differentiable since we know $|\beta| = \beta$.

$$(y - \beta)(-1) + \lambda = 0 \Rightarrow \hat{\beta}^L = y - \lambda.$$

Similarly, if the optimal is less than 0, then $|\beta| = -\beta$ so,

$$(y - \beta)(-1) - \lambda = 0 \Rightarrow \hat{\beta}^L = y + \lambda.$$

Shrink towards 0 by an amount λ !!

Now we only have three posibilities for the optimal β and you can just check that the minimum is obtained with

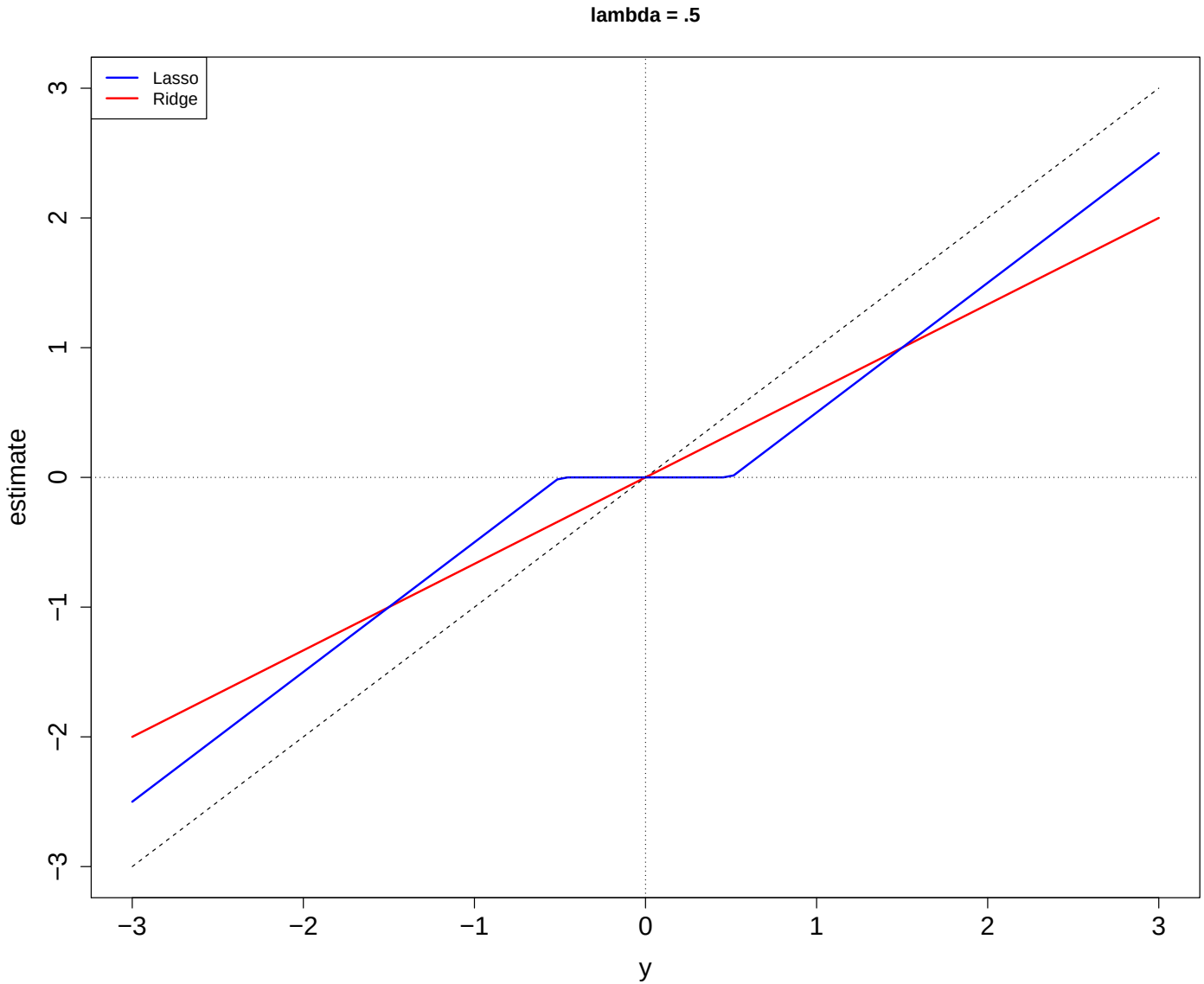
$$\hat{\beta}^L = \begin{cases} y - \lambda & y > \lambda \\ 0 & |y| \leq \lambda \\ y + \lambda & y < -\lambda \end{cases}$$

For example, suppose $0 < y < \lambda$.
Which is better, $\beta = 0$ or $\beta = y - \lambda$?.
At $\beta = y - \lambda$ we have

$$\begin{aligned} (y - (y - \lambda))^2 + \lambda|y - \lambda| &= \lambda^2 + \lambda|y - \lambda| \\ &\geq (y - 0)^2 + \lambda|0|. \end{aligned}$$

Intuitively, if $0 < y < \lambda$, there is no way I want negative estimate $y - \lambda$.

Here is a plot of the LASSO and Ridge shinkage.



Soft threshholding function

We can express these solutions succintly using the *soft threshholding function* S_λ.

$$\hat{\beta}^R = \frac{y}{1+\lambda}.$$

$$\hat{\beta}^L = S_\lambda(y)$$

where

$$S_\lambda(y) = \text{sign}(y)(|y| - \lambda)_+$$

with $x_+ = x$ if x is positive and 0 otherwise.

Standardization

Ok, now let's try LASSO with some x's !!

But first, we emphasize again that for this to make sense you have to put the x's on the same scale by standarizing them.

The LASSO literature strongly favors standardization using the sample mean and variance.

Since we are not trying to regularize (shrink) the intercept, it is usual to start by demeaning y and x:

$$y_i \to y_i - \bar{y}; \quad x_{ij} \to x_{ij} - \bar{x}_j.$$

Recall that if you run a regression using the demeaned variables, you get the same slope estimates.

We then scale the x's:

$$x_{ij} \to \frac{x_{ij}}{s_j}$$

where

$$s_j^2 = \frac{\sum x_{ij}^2}{n}$$

Note that after you do this standardization $\sum_i x_{ij}^2 = n$ for each $j = 1, 2, \ldots, p$.

LASSO with one x

Let's now see what happens when we just have one x variable.

After standardizing we miminize:

$$\frac{1}{2n}\sum_{i=1}^{n}(y_i - \beta\, x_i)^2 + \lambda\,|\beta|.$$

Dividing by $2n$ does not change the problem, but makes the formulas turn out nicer.

Again if the solution were positive, we must have

$$\frac{1}{n}\sum(y_i - \beta x_i)(-x_i) + \lambda = 0 \rightarrow \hat{\beta}^L = \frac{1}{n}x'y - \lambda.$$

And if negative,

$$\frac{1}{n}\sum(y_i - \beta x_i)(-x_i) - \lambda = 0 \rightarrow \hat{\beta}^L = \frac{1}{n}x'y + \lambda.$$

So that,

$$\hat{\beta}^L = S_\lambda(\frac{1}{n}x'y).$$

The General Problem, p Variables

$$\underset{\beta}{\text{minimize}} \; \frac{1}{2n} \sum_{i=1}^{n} (y_i - \sum_j \beta_j \, x_{ij})^2 + \lambda \sum_j |\beta_j|.$$

or,

$$\underset{\beta}{\text{minimize}} \; \frac{1}{2n} ||y - X\beta||^2 + \lambda \, ||\beta||_1$$

Cyclic Coordinate Descent

Given a choice of λ, suppose we knew all of the coefficients except β_j.

We can write our objective as:

$$\underset{\beta_j}{\text{minimize}} \frac{1}{2n} \sum_{i=1}^{n} (y_i - \sum_{k \neq j} \beta_k x_{ik} - \beta_j x_{ij})^2 + \lambda |\beta_j| + \lambda \sum_{k \neq j} |\beta_k|.$$

Which is the same problem as

$$\underset{\beta_j}{\text{minimize}} \frac{1}{2n} \sum_{i=1}^{n} (r_i^{(j)} - \beta_j x_{ij})^2 + \lambda |\beta_j|$$

with

$$r_i^{(j)} = y_i - \sum_{k \neq j} \beta_k x_{ik}$$

The $r_i^{(j)}$ are the *partial residuals*.

We already know that

$$\underset{\beta_j}{\text{minimize}} \frac{1}{2n} \sum_{i=1}^{n} (r_i^{(j)} - \beta_j x_{ij})^2 + \lambda |\beta_j|$$

has solution

$$\hat{\beta}_j = S_\lambda(\frac{1}{n} x_j' r^{(j)}).$$

This gives us a very simple *cyclic coordinate descent algorithm*

- Pick a fixed order for the coefficients (variables), e.g $1, 2, \ldots, p$.
- Cycle through the coefficient updating each with the soft thresholding formula: $\hat{\beta}_j = S_\lambda(\frac{1}{n} x_j' r^{(j)})$.
- Repeat until covergence.

Simple !!!

Note

We often want to do this for a set of λ values.

If we start with all the β_j at 0, then our initial $r^{(j)} = y$.

Thus we know that if we set

$$\lambda_{max} = \max_j \ |\frac{1}{n} x_j' y|$$

then for that λ, and all larger, no matter what coefficient we attempted to update, we would get 0. So, there is no need to consider $\lambda > \lambda_{max}$.

So, we can,

- Start at $\lambda = \lambda_{max}$.
- Slowly decrease, λ.
- At each λ, find a solution using cyclic coordinate descent.
- *warm start*, each cyclic descent by starting at the solution from the previous λ.

Orthogonal x's

Suppose our x's are orthogonal:

$$x_j' x_i = 0, \quad i \neq j.$$

Since we have demeaned, this is equivalent to the x's being uncorrelated.

Then,

$$x_j' r^{(j)} = x_j' y$$

So our cyclic alorgithm converges immediately to

$$\hat{\beta}_j = S_\lambda(\frac{1}{n} x_j' y).$$

Just as in least squares regression, we can fit the model one x at a time if the x's are uncorrelated.

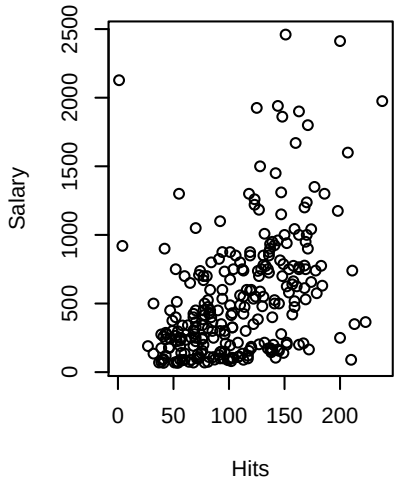
R package glmnet - ridge regression

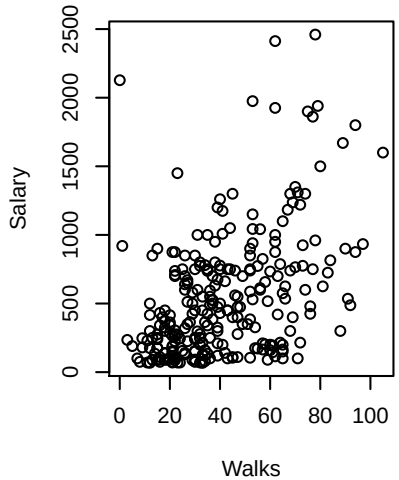
```
'

install.packages("ISLR")
library(ISLR)
library(glmnet)

fix(Hitters)
names(Hitters)
dim(Hitters)

sum(is.na(Hitters$Salary))
Hitters = na.omit(Hitters)

x = model.matrix(Salary~.,Hitters)[,-1]
y = Hitters$Salary


grid=10^seq(10,-2,length=100)

ridge.mod = glmnet(x,y,alpha=0,lambda=grid)

plot(log(1/grid),coef(ridge.mod)[2,],type="l",ylim=range(coef(ridge.mod)[2:20,]),xlab="log(1/lambda)",ylab="Coefficients")
for (i in 3:20)
  lines(log(1/grid),coef(ridge.mod)[i,])


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

cv.out = cv.glmnet(x[train,],y[train],alpha=0)
plot(cv.out)

bestlam = cv.out$lambda.min
ridge.pred = predict(ridge.mod,s=bestlam,newx=x[test,])
mean((ridge.pred-y.test)^2)

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

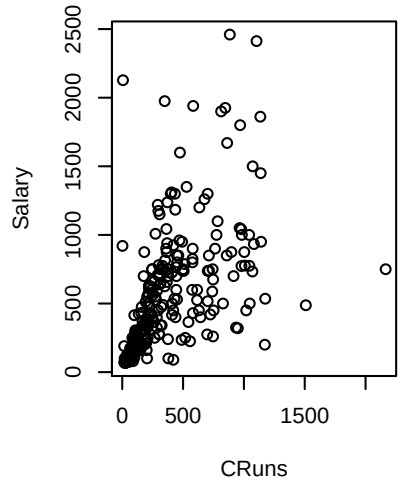
R package glmnet - LASSO regression

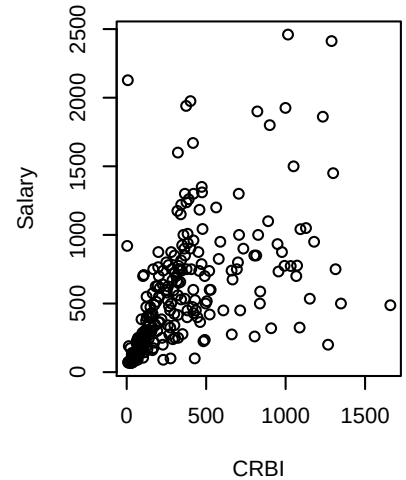
```
LASSO.mod = glmnet(x,y,alpha=1,lambda=grid)

par(mfrow=c(1,1))
plot(log(1/grid),coef(ridge.mod)[2,],type="l",ylim=range(coef(ridge.mod)[2:20,]),
       xlab="log(1/lambda)",ylab="Coefficients")
for (i in 2:20){
  lines(log(1/grid),coef(ridge.mod)[i,])
  lines(log(1/grid),coef(LASSO.mod)[i,],col=2)
}
legend("topleft",legend=c("Ridge","LASSO"),col=1:2,lwd=2)

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

bestlam = cv.out$lambda.min
LASSO.pred = predict(LASSO.mod,s=bestlam,newx=x[test,])
mean((LASSO.pred-y.test)^2)

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

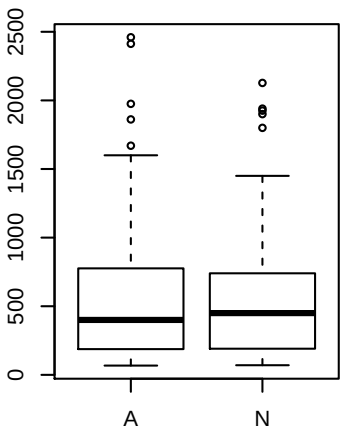
Hitters data

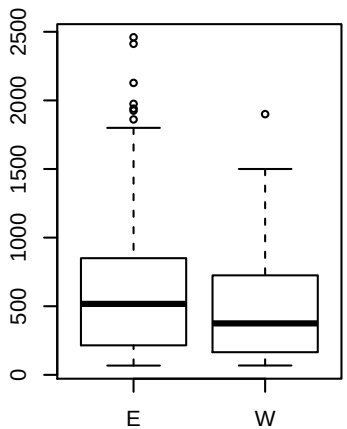


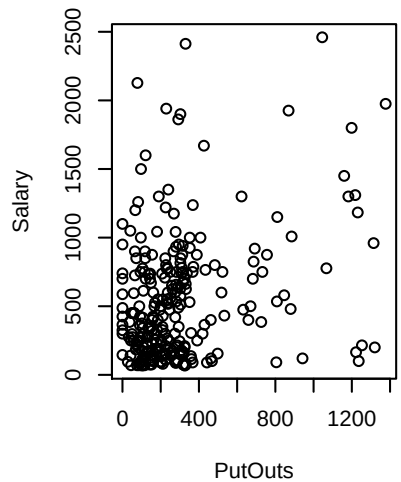




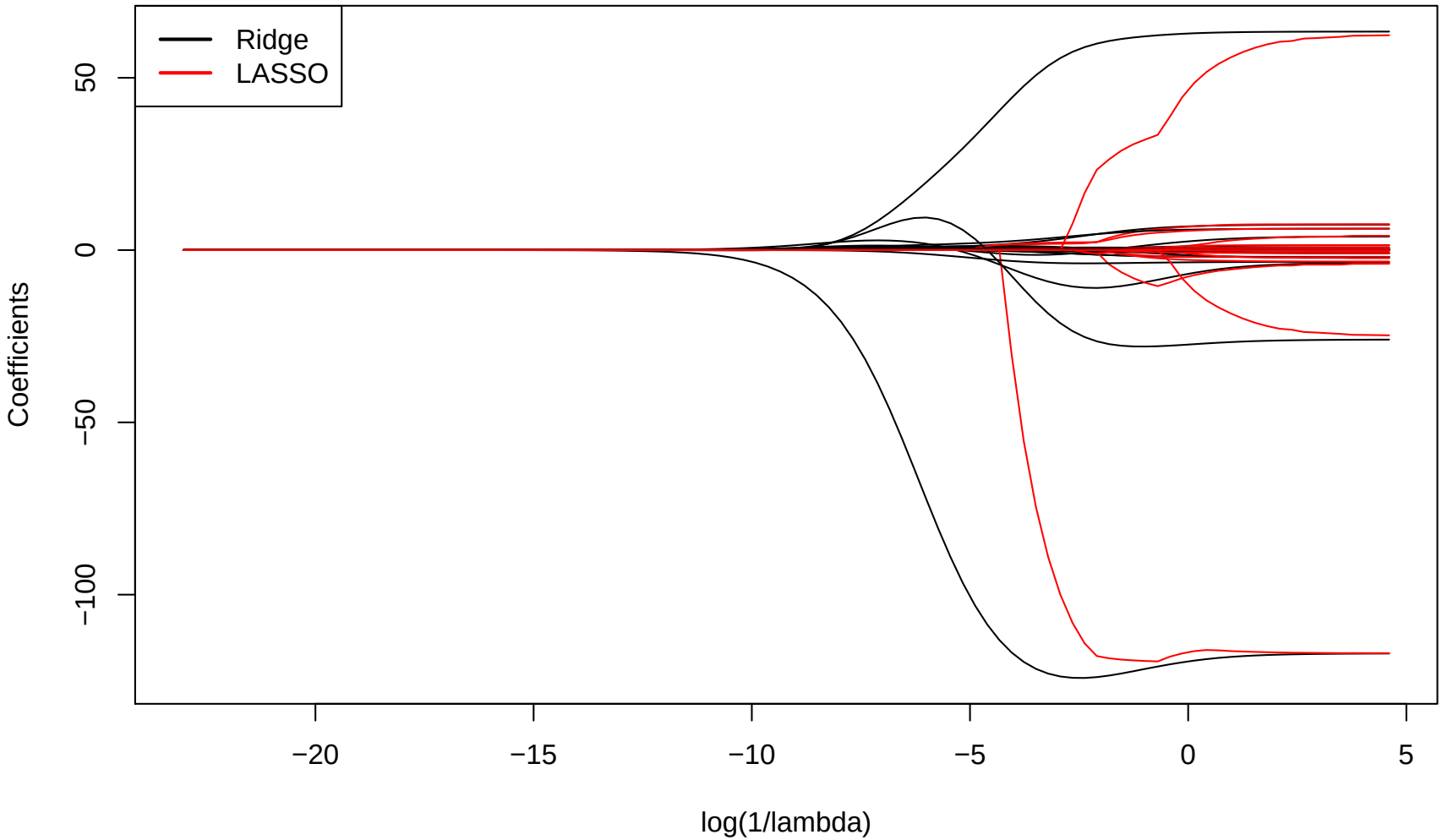




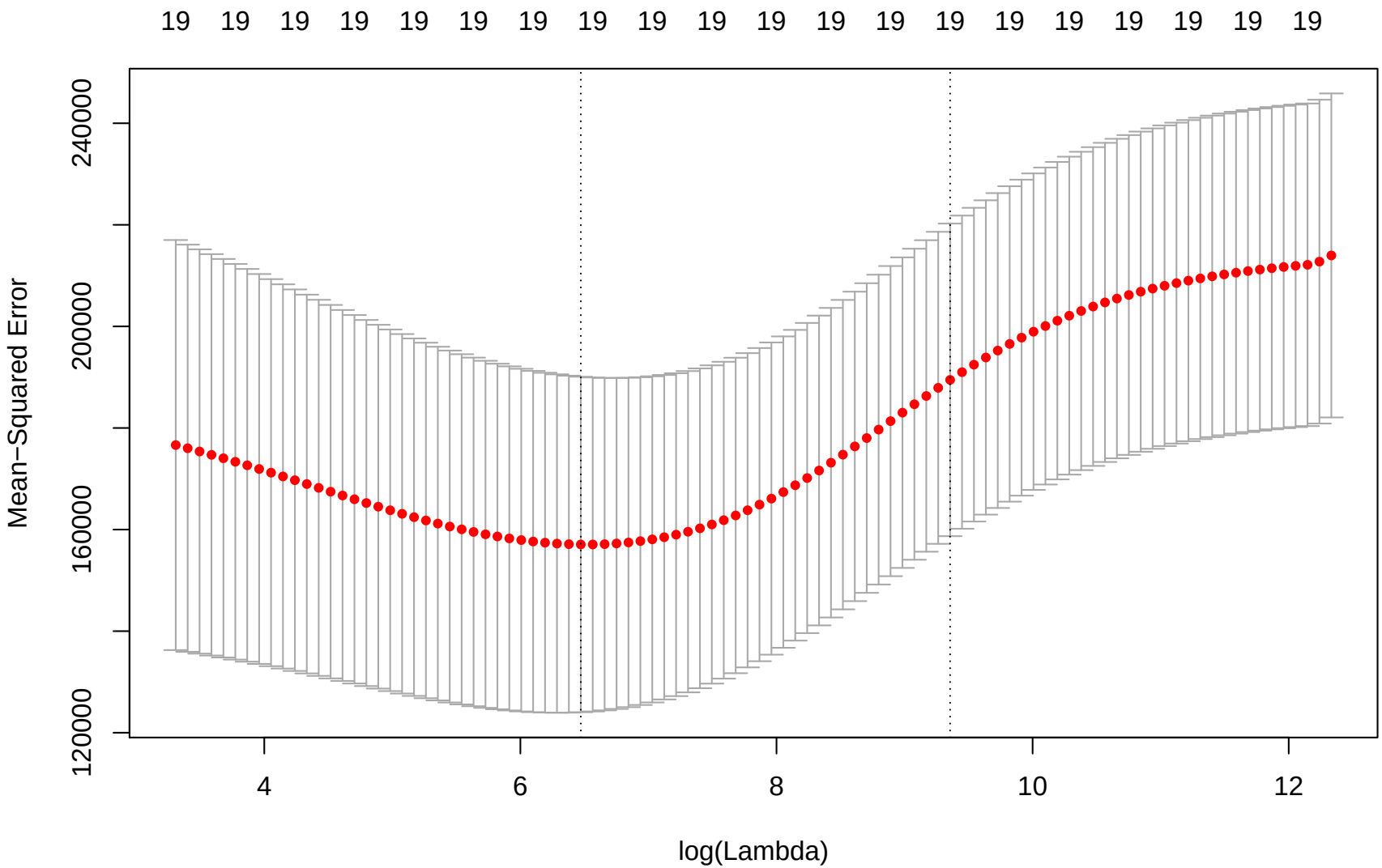




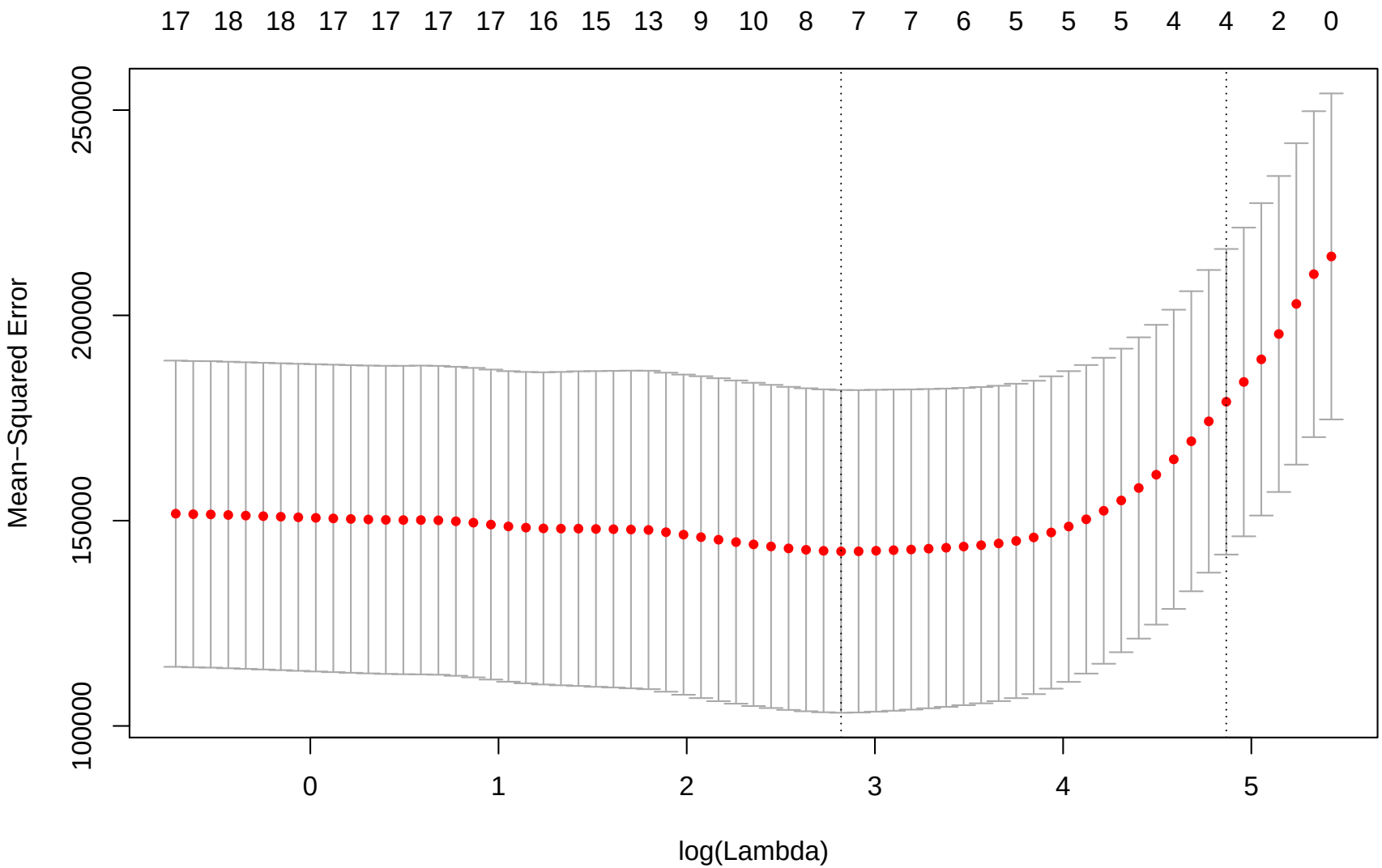
Hitters data



Ridge regression: 10-fold cross validation



LASSO regression: 10-fold cross validation



Hitters data

OLS's MSE: 125,154

RIDGE's MSE: 87,150

LASSO's MSE: 77,426

Comparing Ridge and LASSO regressions

	ridge.coef	LASSO.coef
(Intercept)	47.92183564	18.5394844
AtBat	0.11044921	0.0000000
Hits	0.67662342	1.8735390
HmRun	1.13312348	0.0000000
Runs	0.95450953	0.0000000
RBI	0.85450060	0.0000000
Walks	1.35472462	2.2178444
Years	2.50554472	0.0000000
CAtBat	0.01094280	0.0000000
CHits	0.04790211	0.0000000
CHmRun	0.34549715	0.0000000
CRuns	0.09585853	0.2071252
CRBI	0.10027471	0.4130132
CWalks	0.07096837	0.0000000
LeagueN	14.58789771	3.2666677
DivisionW	-57.39269817	-103.4845458
PutOuts	0.12384683	0.2204284
Assists	0.01709043	0.0000000
Errors	-0.76600546	0.0000000
NewLeagueN	8.86825574	0.0000000

Outline

More on regularization

Consider again the Gaussian linear model

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}, \qquad \boldsymbol{\varepsilon} \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I}_n),$$

where $\boldsymbol{\beta}$ is p-dimensional.

Ridge Regression: ℓ_2 penalty on $\boldsymbol{\beta}$:

$$\hat{\boldsymbol{\beta}}_R = \arg\min_{\beta} \{ \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|^2 + \lambda \|\boldsymbol{\beta}\|_2^2 \}, \qquad \lambda \geq 0,$$

leading to $\hat{\boldsymbol{\beta}}_{ridge} = (\boldsymbol{X}'\boldsymbol{X} + \lambda \boldsymbol{I})^{-1}\boldsymbol{X}'\boldsymbol{y}$.

LASSO Regression: ℓ_1 penalty on $\boldsymbol{\beta}$:

$$\hat{\beta}_L = \arg\min_{\beta} \{ \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|^2 + \lambda \|\boldsymbol{\beta}\|_1 \}, \qquad \lambda \geq 0,$$

which can be solved by using quadratic programming techniques such as a *coordinate gradient descent* algorithm.

Elastic net

The Elastic net combines the Ridge and the LASSO approaches:

$$\hat{\beta}_{EN} = \arg\min_{\beta}\{\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|^2 + \lambda_1 \|\boldsymbol{\beta}\|_1 + \lambda_2 \|\boldsymbol{\beta}\|_2^2\},\ \lambda_1 \geq 0, \lambda_2 \geq 0,$$

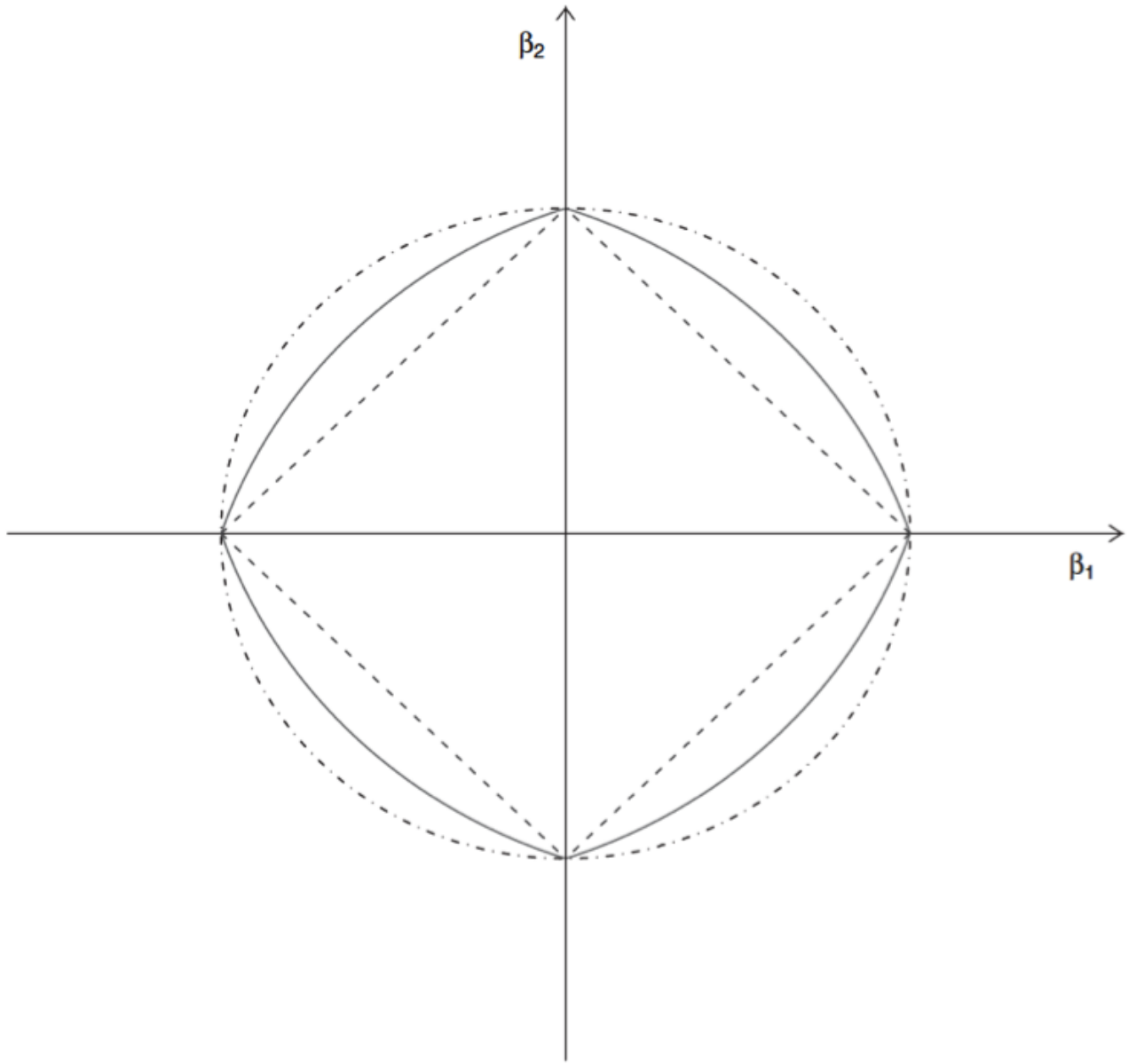
The ℓ_1 part of the penalty generates a sparse model.

The ℓ_2 part of the penalty

- Removes the limitation on the number of selected variables;
- Encourages grouping effect;
- Stabilizes the ℓ_1 regularization path.

R package `elasticnet`

Two dimension contour plots of the three penalty functions



Ridge (dot-dashed), LASSO (dashed) and Elastic net (solid)

Bayesian regularization

- Regularization and variable selection are done by assuming independent prior distributions from a scale mixture of normals (SMN) class:

$$\boldsymbol{\beta}|\boldsymbol{\psi} \sim \mathcal{N}(0, \boldsymbol{\psi}) \qquad \text{and} \qquad \boldsymbol{\psi} \sim p(\boldsymbol{\psi}),$$

- The posterior mode or the maximum a posteriori (MAP) is

$$\arg\max_{\boldsymbol{\beta}} \; \{\log p(\boldsymbol{y}|\boldsymbol{\beta}) + \log p(\boldsymbol{\beta}|\boldsymbol{\psi})\}$$

which is equivalent to penalizing the log-likelihood

$$\log p(\boldsymbol{y}|\boldsymbol{\beta})$$

with penalty equal to the log prior

$$\log p(\boldsymbol{\beta}|\boldsymbol{\psi})$$

when $\boldsymbol{\psi}$ is held fixed.

Bayesian regularization in linear regression problems

The marginal prior distribution of β

$$p(\beta) = \int p(\beta|\psi)p(\psi)d\psi$$

can assume many forms depending on the mixing distribution $p(\psi)$:

	Distribution of ψ	Distribution of β
Bayesian LASSO	$\mathcal{E}(\lambda^2/2)$	Laplace
Ridge	$\mathcal{IG}(\alpha, \delta)$	Scaled Student's t
NG prior	$\mathcal{G}(\lambda, 1/(2\gamma^2))$	below

The Normal-Gamma prior

$$p(\beta) = \frac{1}{\sqrt{\pi}2^{\lambda-1/2}\gamma^{\lambda+1/2}\Gamma(\lambda)}|\beta|^{\lambda-1/2}K_{\lambda-1/2}(|\beta|/\gamma),$$

where K is the modified Bessel function of the 3rd kind,

$$Var(\beta|\lambda, \gamma^2) = 2\lambda\gamma^2 \quad \text{and} \quad \text{excess kurtosis} = 3/\lambda.$$

Horseshoe prior

The horseshoe prior assumes that

$$\beta|\lambda \sim N(0, \lambda^2)$$

where

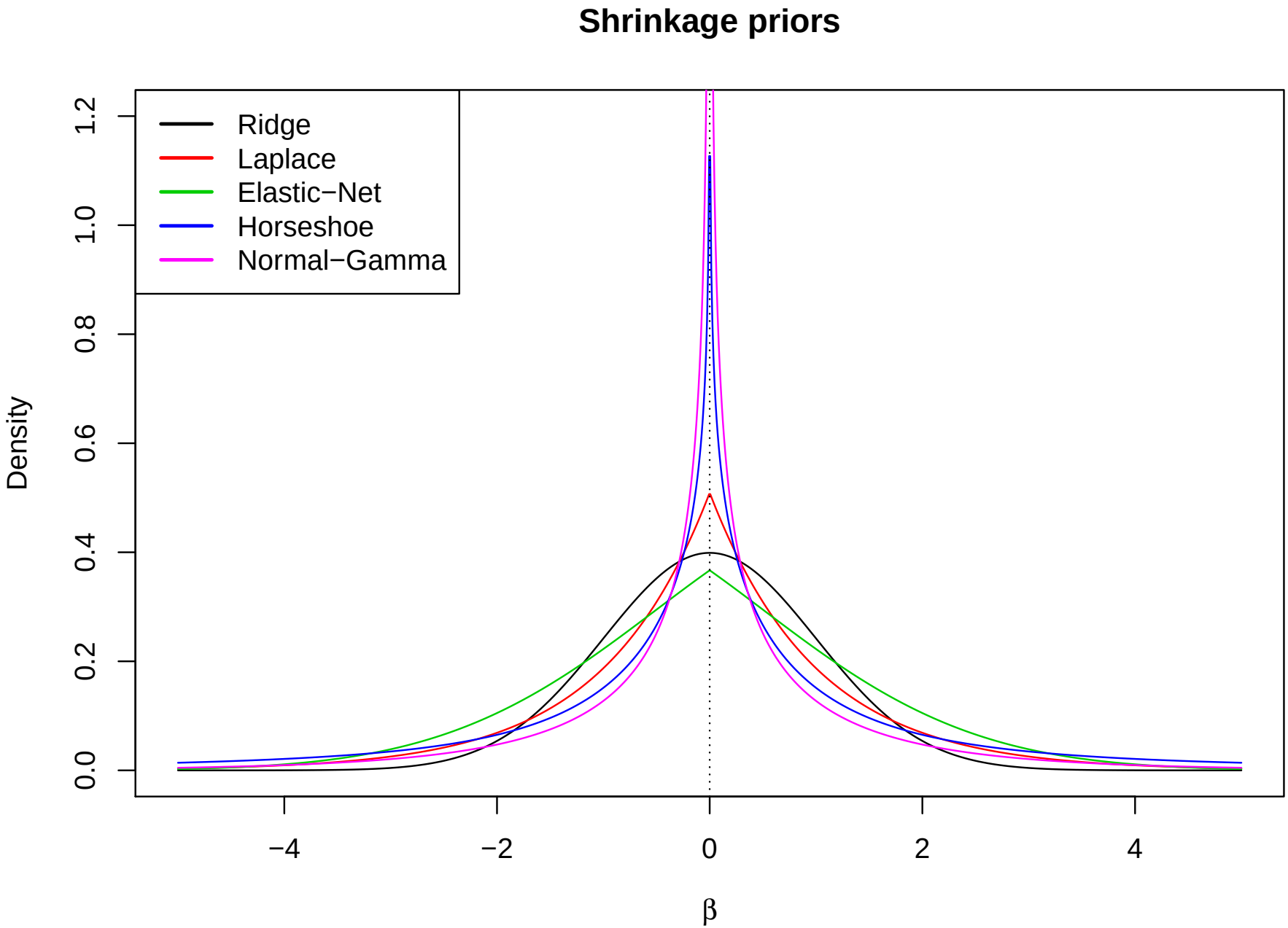
$$\lambda \sim C^+(0, 1),$$

a truncated Cauchy distribution.

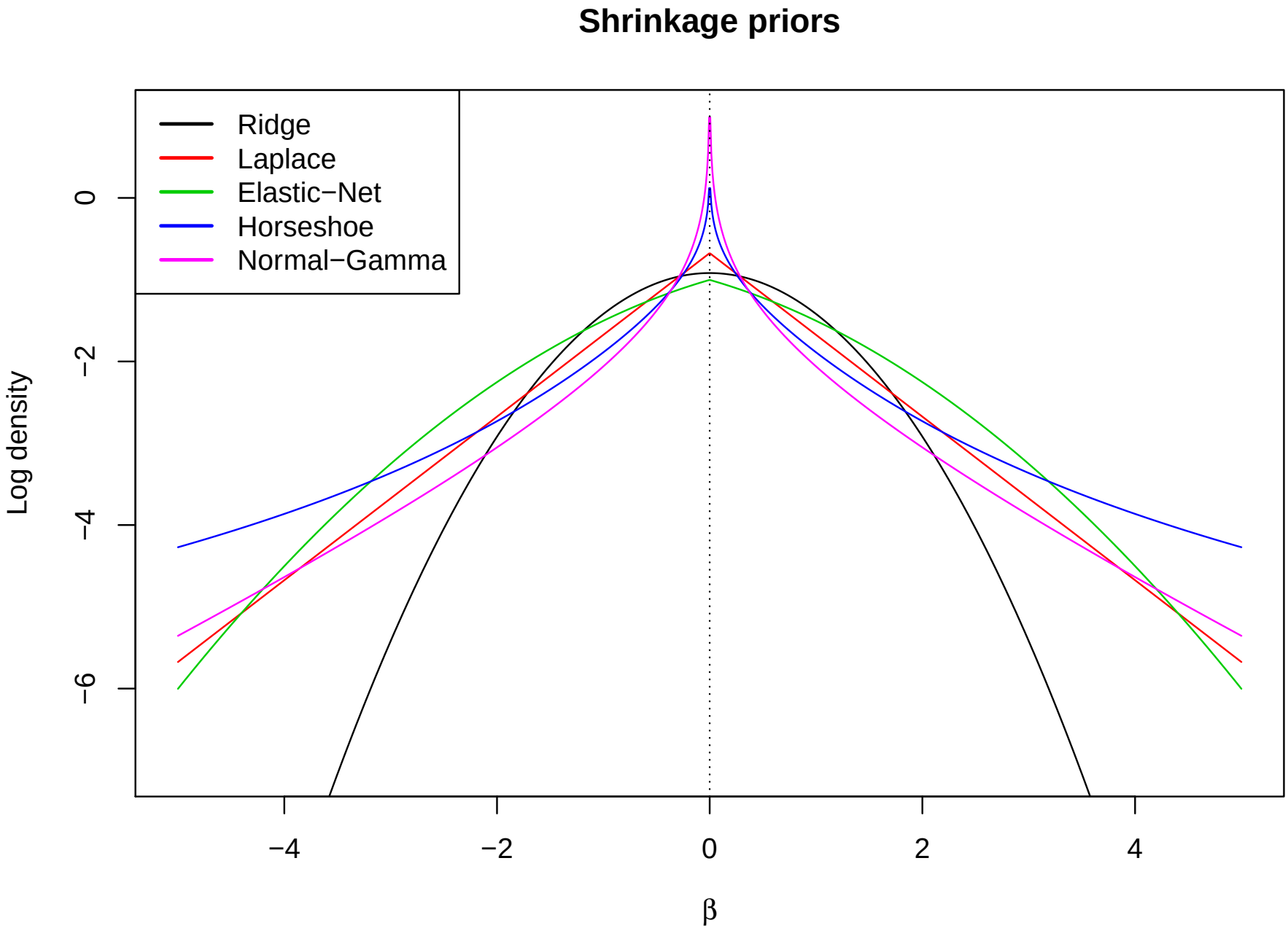
The log-density is approximately

$$\log\left(1 + \frac{4}{\beta^2}\right)$$

Comparing shrinkage priors



Comparing shrinkage priors



R package bayeslm

This package implements an efficient sampler for Gaussian Bayesian linear regression.

The package uses elliptical slice sampler instead of regular Gibbs sampler.

The function has several built-in priors and users can also provide their own priors.

Source: Hahn, He and Lopes (2017) Efficient sampling for Gaussian linear regression with arbitrary priors. *Journal of Computational and Graphical Statistics* (to appear).

R package bayeslm

```
## Default S3 method:
bayeslm(Y, X = FALSE, prior = "horseshoe", penalize = NULL, block_vec = NULL, sigma = NULL,
s2 = 1, kap2 = 1, N = 20000L, burnin = 0L, thinning = 1L, vglobal = 1, verb = FALSE,
icept = TRUE, standardize = TRUE, singular = FALSE, prior_mean = NULL, prob_vec=NULL,cc,...)

Arguments

Y - data.frame, matrix, or vector of inputs Y. Response variable.

X - data.frame, matrix, or vector of inputs X. Regressors.

prior - Indicating shrinkage prior to use. "horseshoe" for approximate horseshoe
prior (default), "laplace" for laplace prior,  "ridge" for ridge prior,...

penalize - A vector indicating shrink regressors or not. It's length should be the same as
number of regressors. 1 indicates shrink corresponding coefficient, 0 indicates no shrinkage.
The default value is rep(1, p), shrink all coefficients

N - Number of posterior samples (after burn-in).

burnin - Number of burn-in samples. If burnin > 0, the function will draw N + burnin samples
and return the last N samples only.

thinning - Number of thinnings. thinning = 1 means no thinning.

icept - Bool, if TRUE, add an intercept. Default value is TRUE.

standardize - Bool, if TRUE, standardize X and Y before sampling.
```

Simulation exercise

Sample size: $n = 100$

Predictors: $p = 20$

Coefficients: $\boldsymbol{\beta} = (2, 3, 4, 0_{p-3})'$

Variance of error: $\sigma^2 = \kappa\sqrt{\beta'\beta} = 1.25$

Design matrix $\boldsymbol{X}$: x_{ij} are i.i.d. $N(0, 1)$

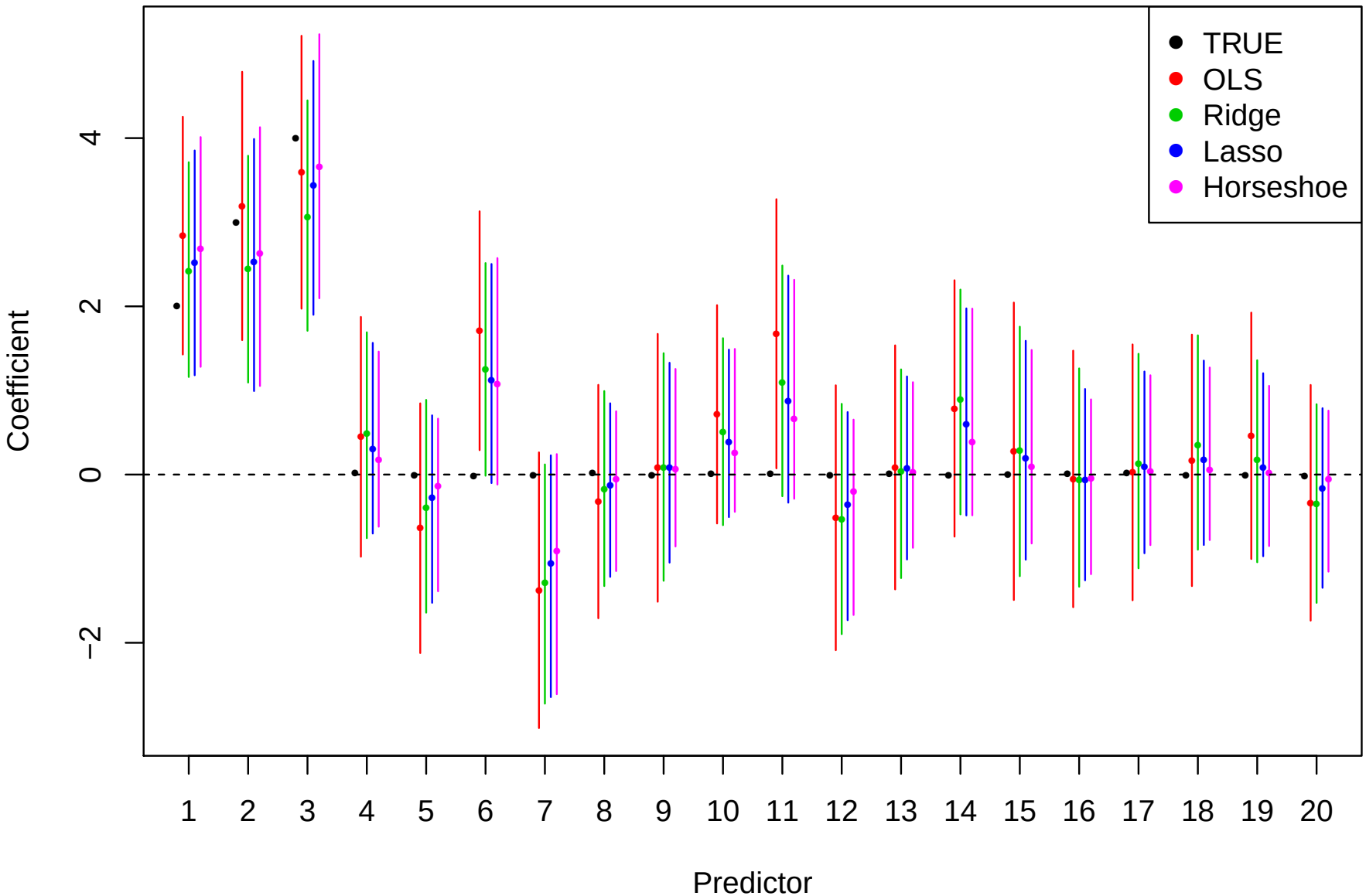
Error term: $\boldsymbol{\varepsilon} \sim N(0, \sigma^2 \boldsymbol{I}_n)$

Dependent variable: $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$

Posterior means

	true	ols	ridge	laplace	hshoe
x1	2.000	2.841	2.673	2.515	2.433
x2	3.000	3.194	2.620	2.518	2.442
x3	4.000	3.594	3.654	3.426	3.057
x4	0.016	0.449	0.275	0.346	0.482
x5	-0.011	-0.638	-0.240	-0.322	-0.391
x6	-0.015	1.710	1.093	1.145	1.255
x7	-0.007	-1.375	-0.970	-1.091	-1.292
x8	0.015	-0.321	-0.123	-0.156	-0.170
x9	-0.011	0.080	0.122	0.120	0.083
x10	0.008	0.716	0.356	0.422	0.508
x11	0.009	1.674	0.767	0.902	1.100
x12	-0.011	-0.513	-0.327	-0.407	-0.533
x13	0.012	0.085	0.078	0.084	0.028
x14	-0.007	0.786	0.504	0.639	0.892
x15	-0.001	0.277	0.194	0.241	0.283
x16	0.013	-0.052	-0.082	-0.084	-0.059
x17	0.022	0.026	0.098	0.117	0.136
x18	-0.006	0.169	0.136	0.215	0.361
x19	-0.013	0.461	0.060	0.101	0.176
x20	-0.018	-0.335	-0.123	-0.210	-0.345
RMSE		3.325	2.820	2.307	2.044

Comparing posteriors



R code

```
install.packages("bayeslm")
library(bayeslm)

set.seed(31415)
p = 20
n = 100
kappa = 1.25
beta_true = c(c(2,3,4),rnorm(p-3,0,0.01))
sig_true = kappa*sqrt(sum(beta_true^2))
x = matrix(rnorm(p*n),n,p)
y = x %*% beta_true + sig_true * rnorm(n)
x = as.matrix(x)
y = as.matrix(y)
data = data.frame(x = x, y = y)


# OLS fit
fitOLS = lm(y~x-1)
se = sqrt(diag(solve(t(x)%*%x)))*summary(fitOLS)$sigma
qols  = cbind(fitOLS$coef-2*se,fitOLS$coef,fitOLS$coef+2*se)


# Bayesian regularization
fit1 = bayeslm(y,x,prior = 'horseshoe', icept = FALSE, N = 10000, burnin=2000)
fit2 = bayeslm(y,x,prior = 'laplace', icept = FALSE, N = 10000, burnin=2000)
fit3 = bayeslm(y,x,prior = 'ridge', icept = FALSE, N = 10000, burnin=2000)

round(cbind(beta_true,fitOLS$coef,apply(fit1$beta,2,mean),apply(fit2$beta,2,mean),apply(fit3$beta,2,mean)),3)

rmseOLS = sqrt(sum((fitOLS$coef-beta_true)^2))
rmse1 = sqrt(sum((beta_est1-beta_true)^2))
rmse2 = sqrt(sum((beta_est2-beta_true)^2))
rmse3 = sqrt(sum((beta_est3-beta_true)^2))

print(cbind(ols = rmseOLS, ridge = rmse3, laplace = rmse2, horseshoe = rmse1))
```

Let us revisit what we covered

Multiple linear regression

- Simplest linear regression model
- `houseprice` dataset
- R^2, R^2_{adj}, C_p, AIC and BIC
- R package `regsubsets`
- `Credit` dataset

Shrinkage-L2, Ridge Regression

- `Hitters` dataset
- Constrained minimization
- Karush Kuhn Tucker (KKT) conditions

Shrinkage-L1: The LASSO

- Soft threshholding function
- Cyclic Coordinate Descent
- `R` package `glmnet`

More on regularization

- Elastic net
- Normal-gamma prior
- Horseshoe prior
- `R` package `bayeslm`
- Simulation exercise

A few references

Chipman (1964) On Least Squares with Insufficient Observations. *JASA*, 59(308), 1078-1111.

Hoerl and Kennard (1970) Ridge regression: biased estimation for non-orthogonal problems. *Technometrics*, 12, 55-67.

Goldstein and Smith (1974) Ridge-Type Estimators for Regression Analysis. *JRSS-B*, 36(2), 284-291.

Tibshirani, R. (1996). Regression shrinkage and selection via the LASSO. *JRSS-B*, 58(1), 267-288.

Efron, Johnstone, Hastie and Tibshirani (2004) Least angle regression. *The Annals of Statistics*, 32(2), 407-499.

Zou and Hastie (2005) Regularization and variable selection via the elastic net. *JRSS-B*, 67, 301?320.

Zou (2006) The adaptive LASSO and its oracle properties. *JASA*, 101, 1418-1429.

Park and Casella (2008) The Bayesian LASSO. *JASA*, 103(482), 681-686.

Hans (2009) Bayesian LASSO regression. *Biometrika*, 96(4), 835-845.

Carvalho, Polson and Scott (2010) The horseshoe estimator for sparse signals. *Biometrika*, 97(2), 465-480.

Griffin and Brown (2010) Inference with normal-gamma prior distributions in regression problems. *Bayesian Analysis*, 5(1), 171-188.

Tibshirani (2011) Regression shrinkage and selection via the LASSO: a retrospective. *JRSS-B*, 73(3) 273-282

Hahn, He and Lopes (2017) Efficient sampling for Gaussian linear regression with arbitrary priors. *JCGS* (to appear).