Economic Forecasting

Shrinkage methods

Sebastian Fossati

University of Alberta | E493 | 2023

Outline

Model selection

Suppose we observe a dependent variable y and p potential explanatory variables (or predictors) $x_1, x_2, \ldots, x_p$ such that

$$y = f(x_1, x_2, \ldots, x_p) + \varepsilon$$

Two main reasons to perform variable selection:

- *model interpretability*: including irrelevant variables leads to unnecessary complexity
- *prediction accuracy*: including irrelevant variables leads to less accurate predictions, specially when N is not much larger than p

Model selection

The methods described before work well with small p...

- but the number of models grows as p increases, with $p = 10$ there are $2^{10} = 1024$ possible models
- if $p < N$ but close to N, OLS will not be very accurate
- if $p > N$, OLS cannot be used!

Shrinkage methods

Shrinkage methods involve fitting a model with all p predictors (even if $p > N$).

- estimated coefficients are shrunk towards 0, some may be estimated to be exactly 0
- shrinkage reduces the variance of the estimators

Outline

Ridge regression

The **ridge regression estimates** ($\hat{\beta}^R$) are given by

$$\hat{\beta}^R = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

Remarks:

- that is, minimize SSR + $\lambda \sum_{j=1}^{p} \beta_j^2$
- $\lambda \geq 0$ is a *tuning parameter*
- $\lambda \sum_{j=1}^{p} \beta_j^2$ is the *shrinkage penalty*

Ridge regression

Remarks:

- if $\lambda = 0$, $\hat{\beta}^R$ is the OLS estimate
- if $\lambda \to \infty$, $\hat{\beta}^R = \mathbf{0}$ (the null model)
- λ balances fit vs. shrinking of the estimates towards 0
- selecting a good value of λ is critical, more later

Ridge regression

The ridge regression estimates can change substantially when a predictor is multiplied by a constant. Why?

As a result, it is better to first *standardize the predictors* such that they all have the same variance (= 1):

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

Ridge regression vs. OLS

In general, if the relationship between y and the predictors is close to linear, OLS will have low bias but *may* have high variance.

Remarks:

- if p is close to N, OLS estimates will have very large variance
- if $p > N$, OLS cannot be used (variance is ∞)!
- ridge regression trades off an increase in bias for a decrease in variance (the effect of shrinkage)
- ridge regression works best when OLS estimates have high variance

Simulated example 1

True model: $y_i = \beta_1 x_{i1} + \beta_2 x_{i2} + \varepsilon_i, \quad i = 1, \ldots, N$

And consider a small data set with 3 predictor variables such that:

- $\beta_1 = .75$, $\beta_2 = .50$, $\beta_j = 0$ for $j = 0, 3$
- $\sigma = 2$
- $N = 200$

Note that y is a function of x_1 and x_2, all other variables are irrelevant.

- OLS is perfectly capable of handling this one
- but we may still get some insights on how ridge regression works

Simulated example 1

First, simulate data according to data generating process (DGP). Next, use `scale(x)` to standardize the variables.

```
# simulate data
n.obs <- 200
n.var <- 3
#
e <- rnorm(n.obs, mean = 0, sd = 2) # errors
x <- matrix(0, n.obs, n.var)
for (i in 1:n.var){ x[,i] <- rnorm(n.obs) }
#
y <- .75*x[,1] + .50*x[,2] + e
x <- scale(x)
data.all <- data.frame(x,y)
```

Simulated example 1

```
model1 <- lm(y ~ X1 + X2 + X3, data = data.all)
coeftest(model1)
```

```
##
## t test of coefficients:
##
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -0.100      0.143   -0.70   0.4853
## X1             0.983      0.144    6.83  1.1e-10 ***
## X2             0.387      0.144    2.69   0.0077 **
## X3             0.233      0.144    1.62   0.1069
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Simulated example 1

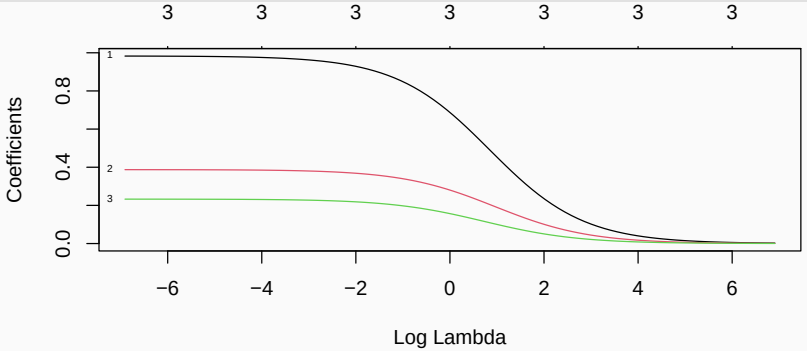
We will use the R package `glmnet`.

- `alpha=0` selects the ridge regression penalty, $\lambda \sum_{j=1}^{p} \beta_j^2$
- we are estimating the ridge regression for a range of values of λ, $\lambda \in (.001, 1000)$

```
# ridge reg for different values of lambda
grid <- 10^seq(3, -3, length = 100)
ridge.mod <- glmnet(x, y, alpha = 0, lambda = grid)
```

Simulated example 1

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



- each line corresponds to a coefficient estimate as a function of λ (if $\lambda \to \infty$?)

Simulated example 1

We can compare the OLS and ridge estimates for a given λ.

- a large λ produces substantial shrinkage (relative to OLS)

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

```
## [1] 1000
```

```
coef(ridge.mod)[,1]
```

```
## (Intercept)          V1          V2          V3
##  -0.1002473   0.0022821   0.0010010   0.0004738
```

```
# least squares
model1$coef
```

```
## (Intercept)          X1          X2          X3
##     -0.1002      0.9832      0.3875      0.2329
```

Simulated example 1

We can compare the OLS and ridge estimates for a given λ.

- a small λ produces less shrinkage

```
# ridge
ridge.mod$lambda[80]
```

```
## [1] 0.0163
```

```
coef(ridge.mod)[,80]
```

```
## (Intercept)          V1          V2          V3
##      -0.1002      0.9764      0.3851      0.2311
```

```
# least squares
model1$coef
```

```
## (Intercept)          X1          X2          X3
##      -0.1002      0.9832      0.3875      0.2329
```

Outline

The lasso

The **lasso estimates** ($\hat{\beta}^L$) are given by

$$\hat{\beta}^L = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

Remarks:

- that is, minimize SSR + $\lambda \sum_{j=1}^{p} |\beta_j|$
- $\lambda \geq 0$ is a *tuning parameter*
- $\lambda \sum_{j=1}^{p} |\beta_j|$ is the *shrinkage penalty*

The lasso

Remarks:

- if $\lambda = 0$, $\hat{\beta}^L$ is the OLS estimate
- if $\lambda \to \infty$, $\hat{\beta}^L = \mathbf{0}$
- λ balances fit vs. shrinking of the estimates towards 0
- selecting a good value of λ is critical, more later

The lasso estimates can also change substantially when a predictor is multiplied by a constant. As a result, it is better to first standardize the predictors such that they all have the same variance (= 1).

The lasso

In general, the lasso works very much like ridge regression.

But the lasso performs **variable selection**, ridge regression does not.

- ridge regression will include all p predictors in the final model
- the lasso may force some estimates to be exactly equal to 0
- if λ is sufficiently large, the ridge regression and lasso models may differ
- lasso models are generally much easier to interpret

Simulated example 1

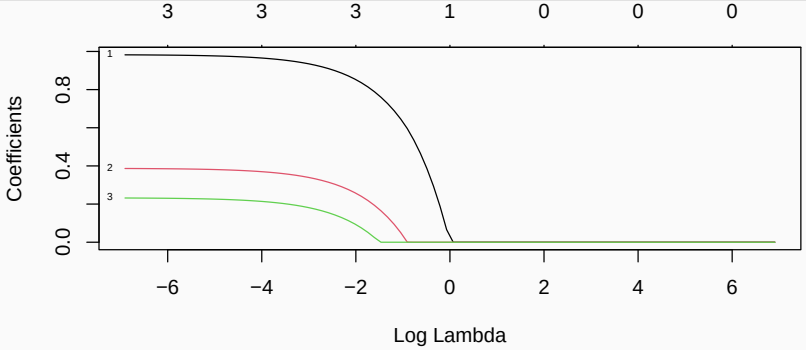
We will use the R package `glmnet`.

- `alpha=1` selects the lasso regression penalty, $\lambda \sum_{j=1}^{p} |\beta_j|$
- we are estimating the lasso regression for a range of values of λ, $\lambda \in (.001, 1000)$

```
# lasso reg for different values of lambda
grid <- 10^seq(3, -3, length = 100)
lasso.mod <- glmnet(x, y, alpha = 1, lambda = grid)
```

Simulated example 1

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



- each line corresponds to a coefficient estimate as a function of λ (if $\lambda \to \infty$?)

Simulated example 1

We can compare the OLS and lasso estimates for a given λ.

- a large λ produces substantial shrinkage (relative to OLS)

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

```
## [1] 1000
```

```
coef(lasso.mod)[,1]
```

```
## (Intercept)          V1          V2          V3
##     -0.1002      0.0000      0.0000      0.0000
```

```
# least squares
model1$coef
```

```
## (Intercept)          X1          X2          X3
##     -0.1002      0.9832      0.3875      0.2329
```

Simulated example 1

We can compare the OLS and lasso estimates for a given λ.

- a small λ produces less shrinkage

```
# ridge
lasso.mod$lambda[80]
```

```
## [1] 0.0163
```

```
coef(lasso.mod)[,80]
```

```
## (Intercept)          V1          V2          V3
##      -0.1002      0.9674      0.3717      0.2160
```

```
# least squares
model1$coef
```

```
## (Intercept)          X1          X2          X3
##      -0.1002      0.9832      0.3875      0.2329
```

Outline

Ridge regression vs. lasso

The ridge regression and lasso coefficients solve the problems:

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

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

Ridge regression vs. lasso

Remarks:

- OLS is the unconstrained solution
- lasso and ridge estimates are constrained solutions
- this reduces the variance as it keeps the estimates close to 0
- the “shape” of the restriction matters

Ridge regression vs. lasso

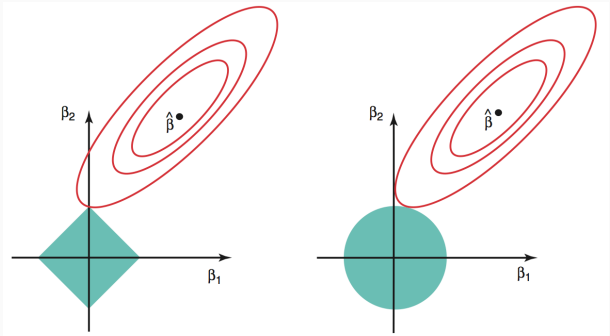


FIGURE 6.7. *Contours of the error and constraint functions for the lasso* (left) *and ridge regression* (right). *The solid blue areas are the constraint regions,* $|\beta_1| + |\beta_2| \leq s$ *and* $\beta_1^2 + \beta_2^2 \leq s$, *while the red ellipses are the contours of the RSS.*

- $\hat{\beta}$ is the (unrestricted) OLS solution, the lasso and ridge regression solutions may differ

Ridge regression vs. lasso

Which is better?

- the lasso should perform better when a relatively small number of predictors are important and the remaining predictors have coefficients that are very small or equal to 0
- ridge regression should perform better when many predictors are (roughly) equally important
- cross-validation can be used to determine which approach is better in a particular data set

Simulated example 2

True model: $y_i = \beta_1 x_{i1} + \beta_2 x_{i2} + \varepsilon_i, \quad i = 1, \ldots, N$

And consider a large data set of 50 mostly irrelevant predictor variables such that:

- $\beta_1 = .75$, $\beta_2 = .50$, $\beta_j = 0$ for $j = 0, 3, \ldots, 50$
- $\sigma = 2$
- $N = 200$

Note that y is a function of x_1 and x_2, all other variables are irrelevant.

- how can we find and estimate the model?

Simulated example 2

First, simulate data according to data generating process (DGP). Next, use `scale(x)` to standardize the variables.

```
# simulate data
n.obs <- 200
n.var <- 50
#
e <- rnorm(n.obs, mean = 0, sd = 2) # errors
x <- matrix(0, n.obs, n.var)
for (i in 1:n.var){ x[,i] <- rnorm(n.obs) }
#
y <- .75*x[,1] + .50*x[,2] + e
x <- scale(x)
data.all <- data.frame(x,y)
```

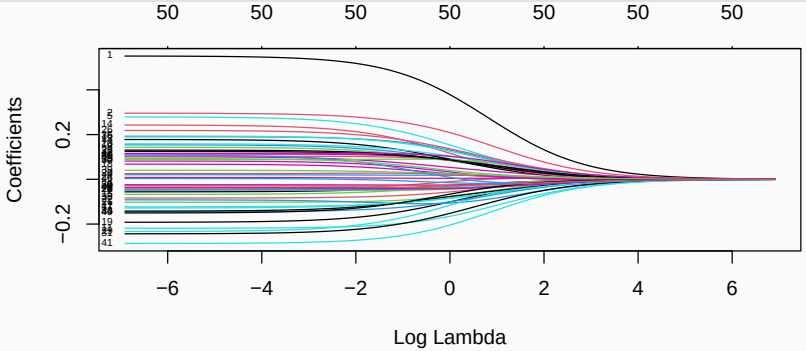
Simulated example 2

We are estimating the ridge and lasso regressions for a range of values of λ, $\lambda \in (.001, 1000)$.

```
#
grid <- 10^seq(3, -3, length = 100)

# ridge reg for different values of lambda
ridge.mod <- glmnet(x, y, alpha = 0, lambda = grid)

# lasso reg for different values of lambda
lasso.mod <- glmnet(x, y, alpha = 1, lambda = grid)
```

Simulated example 2

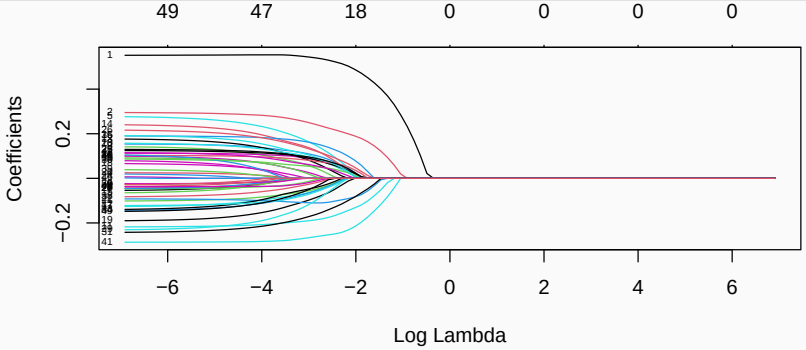
```
# plot ridge results
plot(ridge.mod, xvar = "lambda", label = TRUE)
```



- each line corresponds to a coefficient estimate
- ridge does not perform variable selection

Simulated example 2

```
# plot lasso results
plot(lasso.mod, xvar = "lambda", label = TRUE)
```



- each line corresponds to a coefficient estimate
- lasso performs variable selection

Selecting the tuning parameter

We can use cross-validation to select the tuning parameter λ.

Steps:

- split the sample into a *training set* and a *test set*
- use *k*-fold CV in the training set to find the value of λ that minimizes the cross-validation error
- compute the MSE for observations in the test set and evaluate the performance of the different models
- re-estimate using all available observations and the selected value of λ

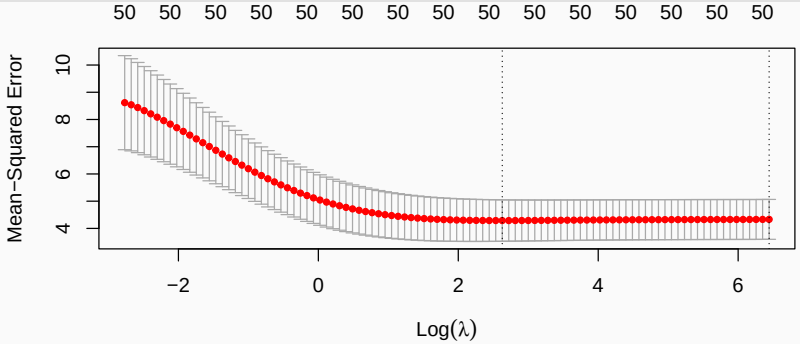
Use the function `cv.glmnet` to find the λ value that minimizes the CV error.

Simulated example 2

```
# split the sample into train/test sets
train <- sample(nrow(data.all), round(nrow(data.all)/2))
cv.error <- rep(NA,3)

# least squares
ols.cv <- lm(y ~ x, data = data.all, subset = train)
cv.error[1] <- mean((y - predict(ols.cv, data.all))[-train]^2)

# ridge
ridge.cv <- cv.glmnet(x[train,], y[train], alpha = 0, nfolds = 10)
ridge.lam <- ridge.cv$lambda.min
cv.error[2] <- mean((y - predict(ridge.cv, s = ridge.lam, newx = x))[-train]^2)

# lasso
lasso.cv <- cv.glmnet(x[train,], y[train], alpha = 1, nfolds = 10)
lasso.lam <- lasso.cv$lambda.min
cv.error[3] <- mean((y - predict(lasso.cv, s = lasso.lam, newx = x))[-train]^2)
```

Simulated example 2

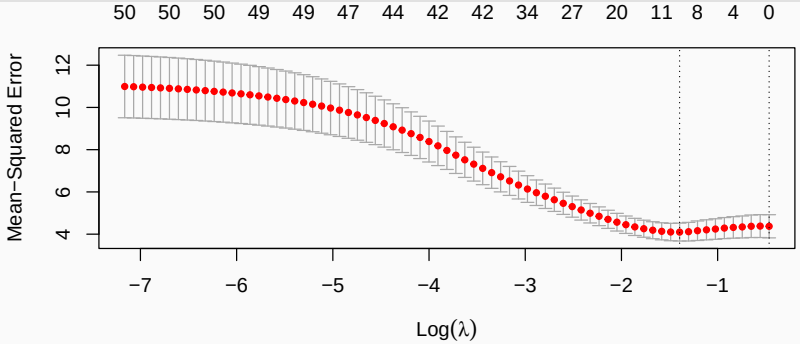
```
# plot ridge cv
plot(ridge.cv)
```



- select the value of λ that minimizes the cross-validation error
- wide confidence intervals (estimate not very accurate)

Simulated example 2

```
# plot lasso cv
plot(lasso.cv)
```



- select the value of λ that minimizes the cross-validation error
- narrow confidence intervals (estimate more accurate)

Simulated example 2

The MSE for observations in the test set is reported below.

```
cv.error <- data.frame(cv.error)
rownames(cv.error) <- c("ols","ridge","lasso")
cv.error
```

```
##       cv.error
## ols      8.015
## ridge    4.497
## lasso    4.758
```

Remarks:

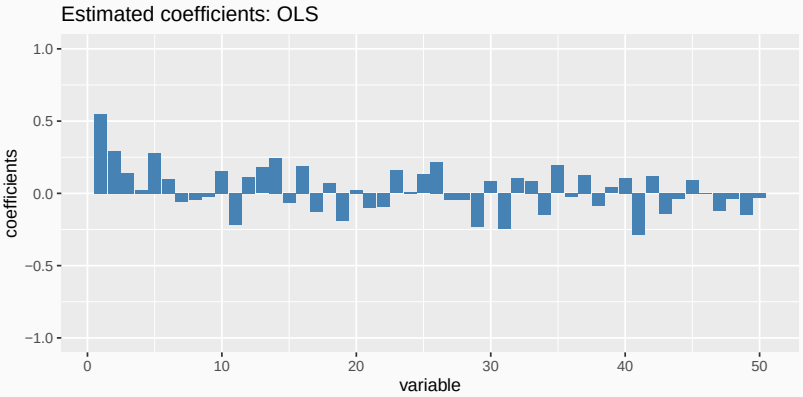
- OLS performs poorly (relative to the other methods)
- ridge and lasso work best in this data set (why?)

Simulated example 2

Re-estimate regression using all the available observations and the selected value of λ.

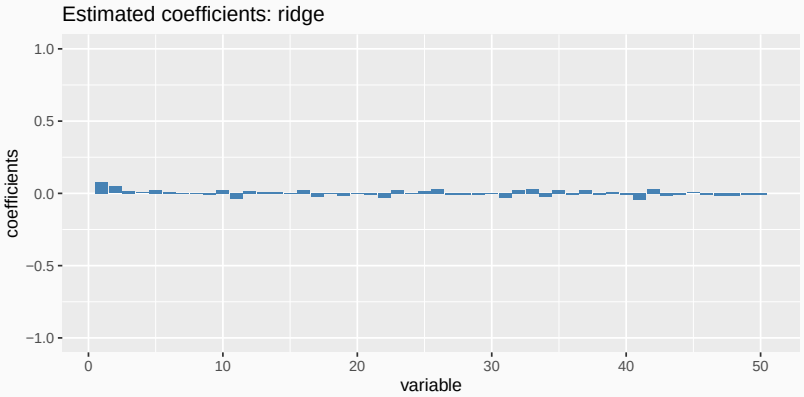
```
# least squares coefficients
model1 <- lm(y ~ x, data = data.all)
coef.ls <- data.frame("x" = seq(1:n.var), "beta" = coef(model1)[-1])

# ridge coefficients
model2 <- glmnet(x, y, alpha = 0, lambda = ridge.lam)
coef.ridge <- data.frame("x" = seq(1:n.var), "beta" = coef(model2)[-1])

# lasso coefficients
model3 <- glmnet(x, y, alpha = 1, lambda = lasso.lam)
coef.lasso <- data.frame("x" = seq(1:n.var), "beta" = coef(model3)[-1])
```

Simulated example 2



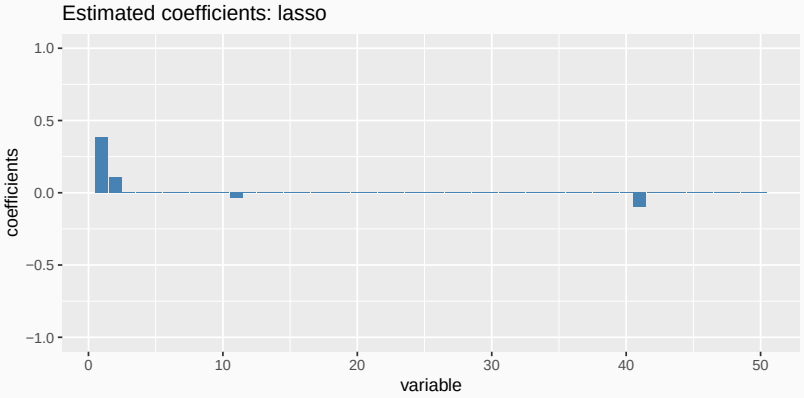
- no shrinkage, all (most) coefficients different from 0

Simulated example 2



- noticeable shrinkage, all (most) coefficients different from 0

Simulated example 2



- most coefficients shrinked to 0

Simulated example 2

Since lasso performs variable selection, we can re-estimate the model using all available observations and only the selected variables.

Least squares estimates are less biased than lasso estimates.

Simulated example 2

```
# estimate the lasso selected model
coeftest(lm(y ~ X1 + X2 + X11 + X41, data = data.all))
```

```
##
## t test of coefficients:
##
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -0.0292     0.1368   -0.21    0.831
## X1            0.6134     0.1382    4.44  1.5e-05 ***
## X2            0.3142     0.1381    2.27    0.024 *
## X11          -0.2587     0.1382   -1.87    0.063 .
## X41          -0.3447     0.1382   -2.49    0.013 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Outline

Example: Used cars

Suppose you want to sell your car of a certain make, type, year, miles, condition and other features.

Prediction analysis can help you uncover the average advertised price of cars with similar characteristics.

- that helps decide what price you may want to put on your ad

Example: Used cars

Consider a sample of offers for used Toyota Camry cars in 2018 in Chicago (Békés and Kézdi, 2021).

Data:

- source: scraped from a website
- characteristics: year of make (age), odometer (miles), etc.
- data cleaning: drop erroneous observations, hybrid cars, trucks…

Example: Used cars

Load data set and filter offers from the Chicago area.

```
# read data
data <-
  read.csv(
    "data/used_cars_work.csv",
    header = TRUE,
    stringsAsFactors = TRUE
  )
# focus only on Chicago
data <- data %>%
  filter(area == "chicago")
```

Example: Used cars

We will use the following predictors:

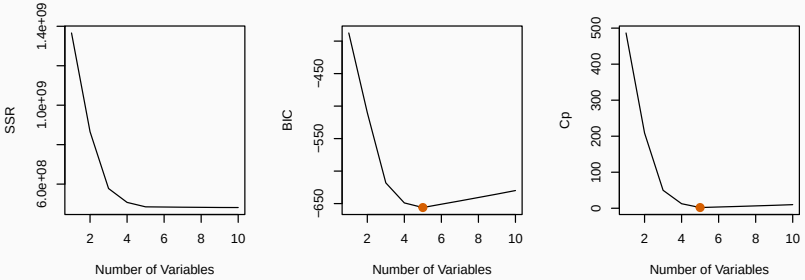
- age: measuring how old is the car (continuous, linear)
- odometer: measuring miles the car traveled (continuous, linear)
- car type: LE, XLE, SE (missing in about 30% of the observations, factor–set of dummies, incl. N/A)
- condition: good condition, excellent condition, or it is like new (missing for about one third of the ads, factor–set of dummies, incl. N/A)
- car's engine has 6 cylinders (20% of ads say this; 43% says 4 cylinders, and the rest has no information, binary for 6 cylinders)

Example: Used cars

Perform **best subset selection** using all the variables in Model 4 allowing for up to 10 variables.

```
# select variables
x <- data %>%
  dplyr::select(
    age, agesq, odometer, odometersq, SE, LE, XLE,
    cond_likenew, cond_excellent, cond_good, cylind6
  ) %>%
  data.matrix()
#
price <- data$price
data2 <- data.frame(x,price)
#
regfit.all <- regsubsets(price~. , data = data2, nvmax = 10)
reg.summary <- summary(regfit.all)
```

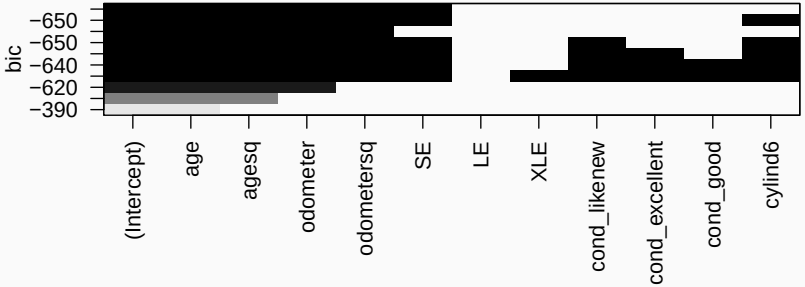
Example: Used cars



- BIC and AIC (Cp) select $k = 5$

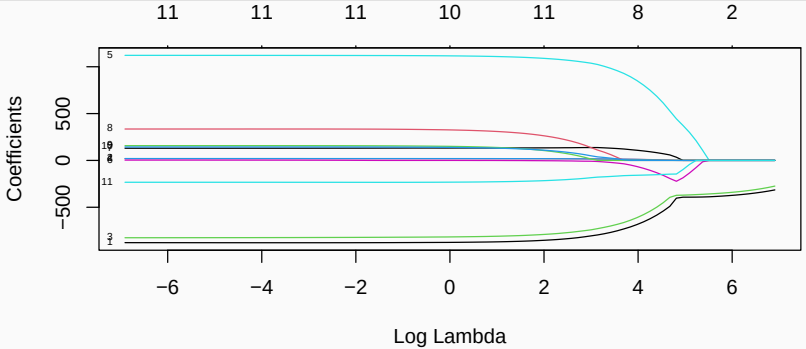
Example: Used cars

```
# plot variable
plot(regfit.all, scale = "bic")
```



Example: Used cars

```
# lasso reg for different values of lambda
lasso.mod <- glmnet(x, price, alpha = 1, lambda = 10^seq(3, -3, length = 100))
plot(lasso.mod, xvar = "lambda", label = TRUE)
```



Example: Used cars

A more detailed look (prices in levels).

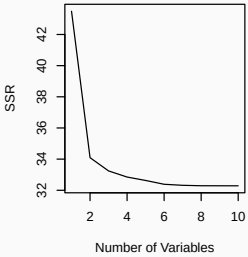
```
##                       ols      bss     lasso
## (Intercept)    19948.916 20035.02 19764.700
## age             -876.959  -873.74  -855.876
## agesq             18.653    18.42    17.837
## odometer        -826.489  -832.01  -805.685
## odometersq        19.832    20.12    18.929
## SE              1122.004  1173.09  1101.982
## LE                 3.256       NA    -2.465
## XLE              130.122       NA   134.158
## cond_likenew     334.838       NA   286.443
## cond_excellent   157.129       NA   121.354
## cond_good        143.040       NA   120.532
## cylind6         -233.506       NA  -224.179
```

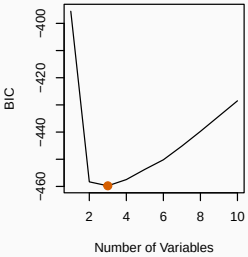
Example: Used cars

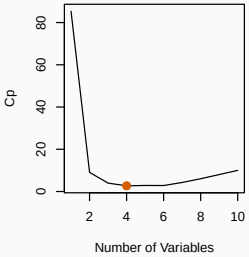
Should we work in levels or **logs**?

```
# select variables
x <- data %>%
  dplyr::select(
    age, agesq, odometer, odometersq, SE, LE, XLE,
    cond_likenew, cond_excellent, cond_good, cylind6
  ) %>%
  data.matrix()
#
lnprice <- data$lnprice
data2 <- data.frame(x,lnprice)
#
regfit.all <- regsubsets(lnprice~. , data = data2, nvmax = 10)
reg.summary <- summary(regfit.all)
```

Example: Used cars



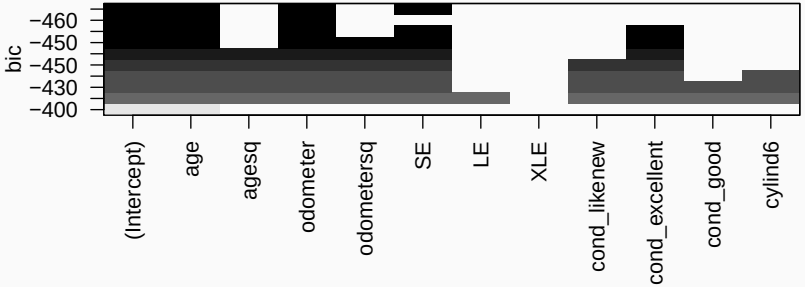




- BIC selects $k = 3$, AIC (Cp) selects $k = 4$

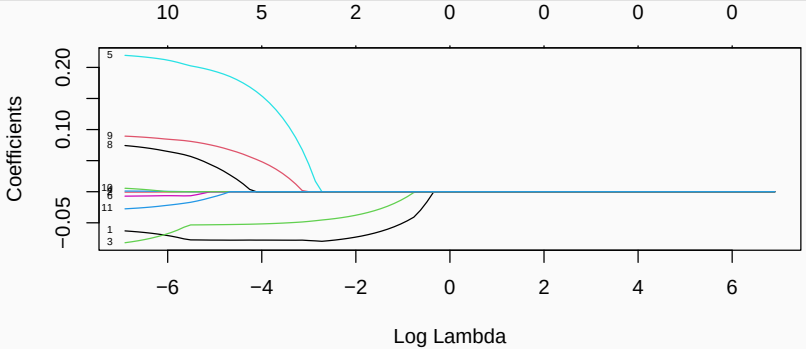
Example: Used cars

```
# plot variable
plot(regfit.all, scale = "bic")
```



Example: Used cars

```
# lasso reg for different values of lambda
lasso.mod <- glmnet(x, lnprice, alpha = 1, lambda = 10^seq(3, -3, length = 100))
plot(lasso.mod, xvar = "lambda", label = TRUE)
```



Example: Used cars

A more detailed look (prices in logs).

```
##                       ols      bss      lasso
## (Intercept)    10.1355346 10.10577 10.0318223
## age            -0.0563904 -0.08429 -0.0778367
## agesq          -0.0009484       NA -0.0001926
## odometer       -0.0935311 -0.05372 -0.0521044
## odometersq      0.0015117       NA  0.0000000
## SE              0.2247556  0.22733  0.1634097
## LE             -0.0080826       NA  0.0000000
## XLE            -0.0057798       NA  0.0000000
## cond_likenew    0.0808348       NA  0.0000000
## cond_excellent  0.0927394       NA  0.0532273
## cond_good       0.0090029       NA  0.0000000
## cylind6        -0.0311883       NA  0.0000000
```

Outline

Simulated example 3

True model: $y_i = \beta_1 x_{i1} + \beta_2 x_{i2} + \varepsilon_i, \quad i = 1, \ldots, N$

And consider a large data set of 500 mostly irrelevant predictor variables such that:

- $\beta_1 = .75$, $\beta_2 = .50$, $\beta_j = 0$ for $j = 0, 3, \ldots, 500$
- $\sigma = 2$
- $N = 200$

Note that y is a function of x_1 and x_2, all other variables are irrelevant.

- how can we find and estimate the model?

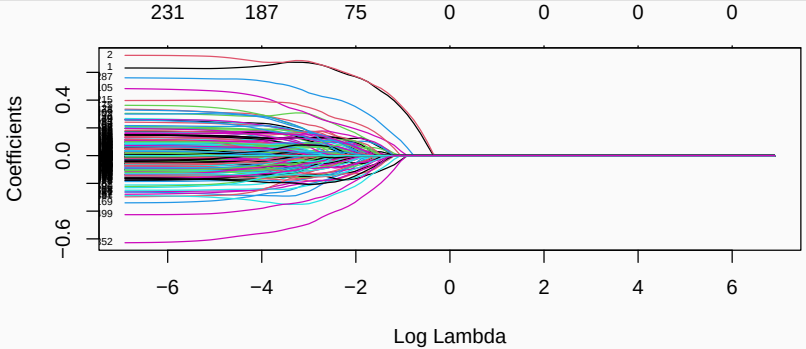
Simulated example 3

First, simulate data according to data generating process (DGP). Next, use `scale(x)` to standardize the variables.

```
# simulate data
n.obs <- 200
n.var <- 500
#
e <- rnorm(n.obs, mean = 0, sd = 2) # errors
x <- matrix(0, n.obs, n.var)
for (i in 1:n.var){ x[,i] <- rnorm(n.obs) }
#
y <- .75*x[,1] + .50*x[,2] + e
x <- scale(x)
data.all <- data.frame(x,y)
```

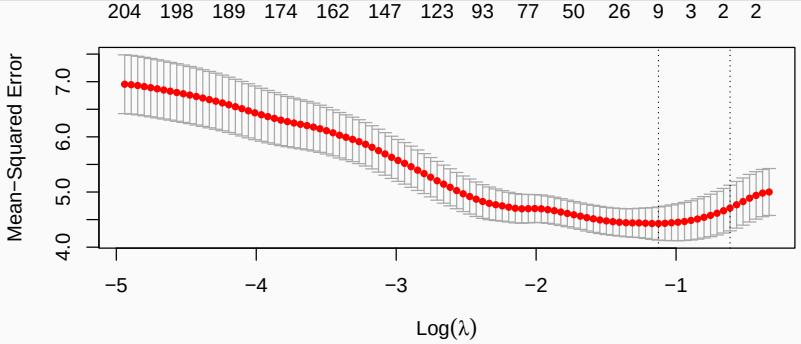
Simulated example 3

```
# lasso reg for different values of lambda
lasso.mod <- glmnet(x, y, alpha = 1, lambda = 10^seq(3, -3, length = 100))
plot(lasso.mod, xvar = "lambda", label = TRUE)
```

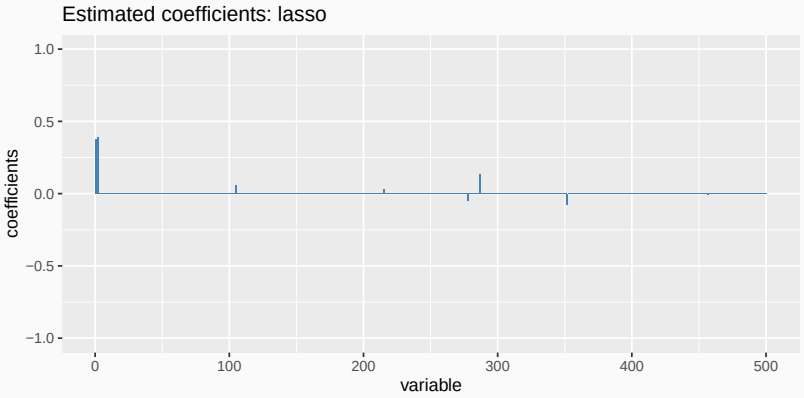


Simulated example 3

```
lasso.cv <- cv.glmnet(x, y, alpha = 1, nfolds = 10)
plot(lasso.cv)
```



Simulated example 3



Simulated example 3

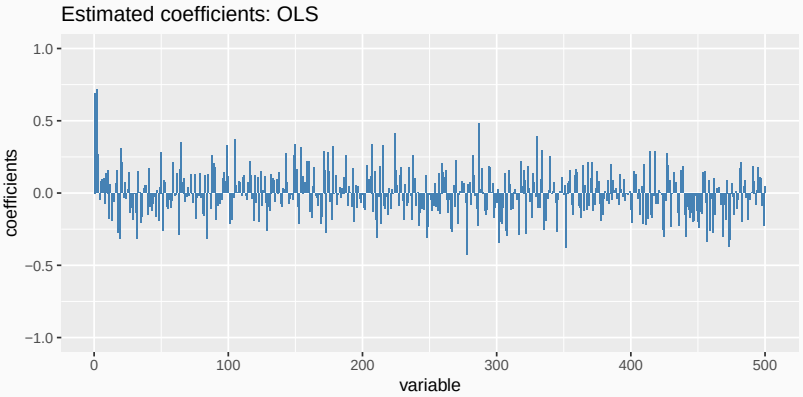
What would we find if we run 500 single OLS regressions?

Simulated example 3

What would we find if we run 500 single OLS regressions?

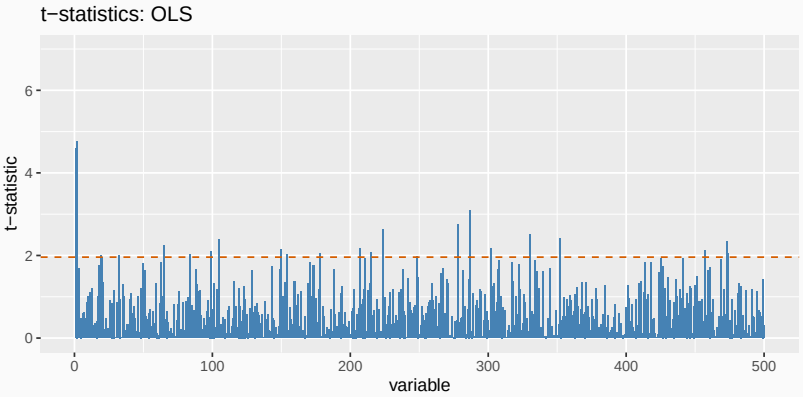
```
#
coef.ols <- rep(NA, n.var)
test.ols <- rep(NA, n.var)
for (i in 1:n.var){
  tmp0 <- lm(y ~ x[,i])
  coef.ols[i] <- coeftest(tmp0)[2,1] # estimated slope
  test.ols[i] <- abs(coeftest(tmp0)[2,3]) # t-stat
  }
# single regression coefficients
coef.ols <-
  data.frame(
    "x" = seq(1:n.var),
    "beta" = coef.ols,
    "test" = test.ols
  )
```

Simulated example 3



- x_1 and x_2 show stronger effect but many coefficients $\neq 0$

Simulated example 3



- many t-statistics > 2!