Chapter 6 in ISL: Regularization

DJM, Revised: NAK

5 March 2019

Model Selection

Suppose we are trying to predict Y using p predictors (including an intercept): $\underline{X}^\top = (1, X_1, X_2, \ldots, X_p)$.

$$Y = \underline{X}^\top \underline{\beta} + \epsilon = \beta_0 + \sum_{j=1}^{n} \beta_j X_j + \epsilon$$

Some of the predictors may be related to Y, and others may not. There are three main methods for model selection.

1. Subset selection (we briefly talked about this before, we'll go into more detail today)
2. Regularization
3. Shrinkage

These are all based off of modifying the minimization problem at the core of **least squares** regression.

$$\widehat{\underline{\beta}} = \underset{\underline{\beta}}{\operatorname{argmin}} \sum_{i=1}^{n} \left(y_i - \underline{x}_i \underline{\beta}\right)^2 = \sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij})^2 = SSE$$

Some Model Selection Criterion

In past applications, especially from academic/research settings, various methods are used for selecting models. They are based on examining a "Criterion" calculated using the estimated model.

- R^2 or its adjusted form: a "measure of variability accounted for by our predictors" and its not good.
- Other more complex ones:
  - $C_p = \frac{1}{n}(SSE + 2p\widehat{\sigma}^2)$
  - $AIC = \frac{1}{n\widehat{\sigma}^2}(SSE + 2p\widehat{\sigma}^2)$
  - $BIC = \frac{1}{n\widehat{\sigma}^2}(SSE + log(n)p\widehat{\sigma}^2)$
- CV MSE
  - LOO CV $= \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \widehat{y}_{-i}\right)^2$
  - k-Fold CV
  - Actually, there are a lot of CV methods.

The concern with any of these methods is will it choose the "correct" predictors, i.e., the ones that truly have a relationship with Y.

Subset Selection

A more classic method of model selection involves algorithms for looking at different subsets of the predictors.

- Take a potentially large set of predictors.
- Whittle down these predictors to some smaller set
- We choose which predictors we want to use by putting some in the model and pulling some out
- We may try transformations of predictors and the response
- We keep messing around until the model fits best according to some criterion.

So we have transformations, multiple predictors, multiple transformations, multiple interactions, etc., that we use to select a model.

We are selecting a model M from a typically huge set of hypothetical models $\mathcal{M}$

$$M \in \mathcal{M}$$

We throw around tons of models:

$$M_1, M_2, \ldots, M_k$$

Potentially each with different sets predictors, transformations, etc.

Sometimes this is done without much regard for reality. Crappy models will get selected and some criterion gets cited as for why the model is good.

"Best" Subset Selection

1. Let M_0 denote the *null model*, which contains no predictors. (What is this model trying to estimate?)
2. For $k = 1, 2, \ldots, p$:

- Fit all $\binom{p}{k}$ models that contain exactly k predictors
- Pick the "best" among the models, call it M_p. "*Best*" means for the chosen criterion.

3. Select the "best" model among $M_0, \ldots, M_p$.

Forward stepwise selection

1. Let M_0 denote the *null model*, which contains no predictors.
2. For $k = 0, 1, ..., p-1$

- Consider all $p - k$ models that add in only one predictor variable.
- Choose the model with the best criterion, among the $p - k$ models.

3. Select the "best" model among $M_0, \ldots, M_p$.

Backward Stepwise Regression

1. Let M_p denote the *full model*, which contains all predictors.
2. For $k = p, p-1, \ldots, 1$

- Consider all k models that remove only one predictor variable.
- Choose the model with the best criterion, among the k models.

3. Select the "best" model among $M_0, \ldots, M_p$.

Packages for Subset Selection

`leaps` package: `regsubsets` function in this package can do all three selection methods

- `regsubsets(formula, data, nvmax, method)`
  - `formula` is the formula for the full model

- `data` is the dataframe (nothing special here)
- `nvmax` is the maximum number of predictors to consider, default it 8
- `method=c("exhaustive","backward", "forward")`

Example: Mobility Data

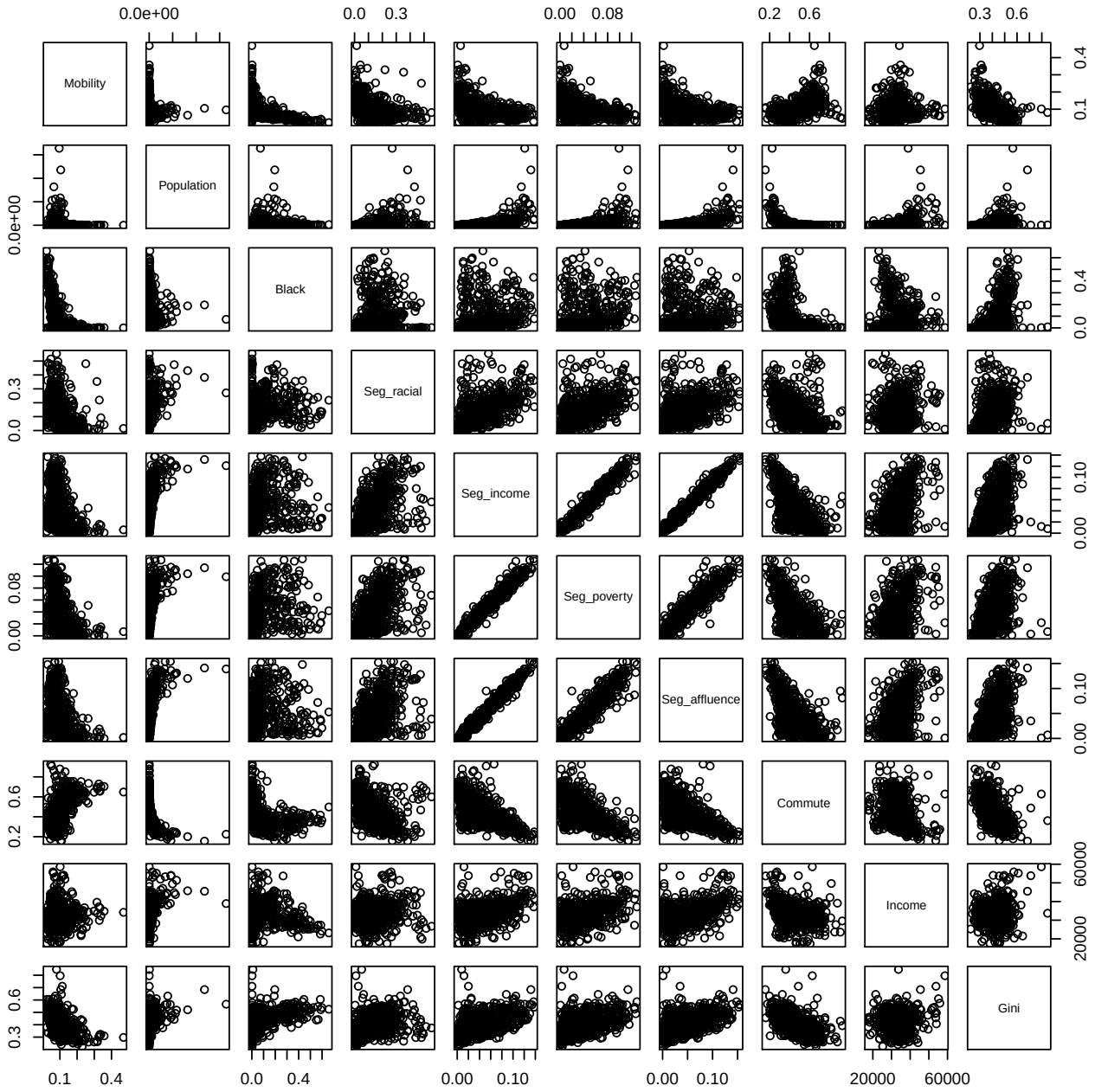
We've seen this data before, but only examined a few potential predictors.

Let's see how the methods work with stepwise regression on an expanded version of the data.

```
mob.full <- read.csv("http://www.stat.cmu.edu/~cshalizi/uADA/15/hw/01/mobility.csv")
mob.full <- mob.full[,c(3,5,7:14)]
mob.full <- na.omit(mob.full)

head(mob.full)
```

```
##     Mobility Population Black Seg_racial Seg_income Seg_poverty
## 1 0.06219881     576081 0.021      0.090      0.035       0.030
## 2 0.05365194     227816 0.020      0.093      0.026       0.028
## 3 0.07263514      66708 0.015      0.064      0.024       0.015
## 4 0.05628121     727600 0.056      0.210      0.092       0.084
## 5 0.04480079     493180 0.174      0.262      0.072       0.061
## 6 0.05183585      92753 0.224      0.137      0.024       0.015
##   Seg_affluence Commute Income  Gini
## 1         0.038   0.325  31560 0.468
## 2         0.025   0.276  29959 0.435
## 3         0.026   0.359  22328 0.441
## 4         0.102   0.269  35884 0.508
## 5         0.081   0.292  38892 0.466
## 6         0.028   0.313  31265 0.444
```

```
pairs(mob.full)
```



Mobility Data: Best Subset

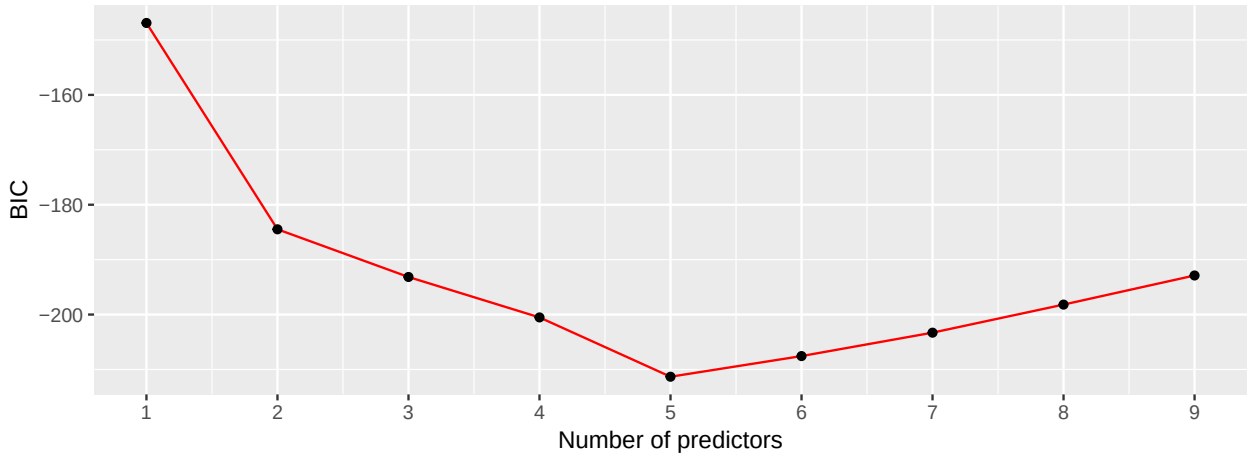
```
library(leaps)
```

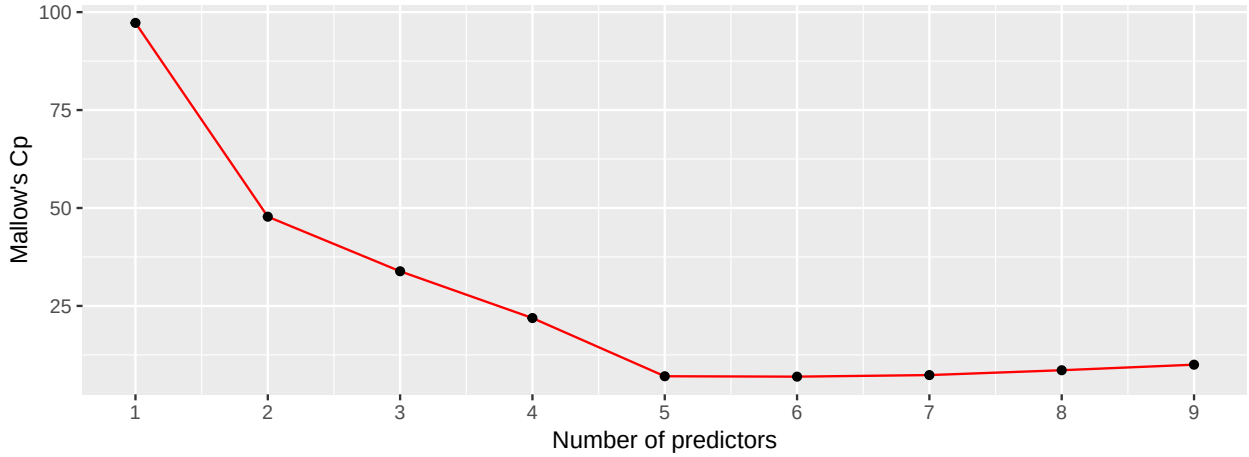
```
# Lets do some data splitting.

set.seed(333)
train <- sample(1:nrow(mob.full), size = nrow(mob.full)/2)
mob <- mob.full[train,]
mob.test <- mob.full[-train,]

mob.best.in.quotes <- regsubsets(Mobility ~ ., data = mob, nvmax = 9)
```

```r
# summary() will report the best set of variables for each model size

summary(mob.best.in.quotes)
```

```
## Subset selection object
## Call: regsubsets.formula(Mobility ~ ., data = mob, nvmax = 9)
## 9 Variables  (and intercept)
##               Forced in Forced out
## Population        FALSE      FALSE
## Black             FALSE      FALSE
## Seg_racial        FALSE      FALSE
## Seg_income        FALSE      FALSE
## Seg_poverty       FALSE      FALSE
## Seg_affluence     FALSE      FALSE
## Commute           FALSE      FALSE
## Income            FALSE      FALSE
## Gini              FALSE      FALSE
## 1 subsets of each size up to 9
## Selection Algorithm: exhaustive
##          Population Black Seg_racial Seg_income Seg_poverty Seg_affluence
## 1  ( 1 ) " "        " "   " "        " "         " "          " "
## 2  ( 1 ) " "        "*"   " "        " "         " "          " "
## 3  ( 1 ) " "        "*"   " "        " "         " "          " "
## 4  ( 1 ) "*"        "*"   "*"        " "         " "          " "
## 5  ( 1 ) "*"        "*"   "*"        " "         " "          " "
## 6  ( 1 ) "*"        "*"   "*"        " "         " "          " "
## 7  ( 1 ) "*"        "*"   "*"        " "         "*"          " "
## 8  ( 1 ) "*"        "*"   "*"        "*"         " "          "*"
## 9  ( 1 ) "*"        "*"   "*"        "*"         "*"          "*"
##          Commute Income Gini
## 1  ( 1 ) "*"     " "    " "
## 2  ( 1 ) "*"     " "    " "
## 3  ( 1 ) "*"     " "    "*"
## 4  ( 1 ) "*"     " "    " "
## 5  ( 1 ) "*"     " "    "*"
## 6  ( 1 ) "*"     "*"    "*"
## 7  ( 1 ) "*"     "*"    "*"
## 8  ( 1 ) "*"     "*"    "*"
## 9  ( 1 ) "*"     "*"    "*"
```

```r
# The summary() also contains information about the assessment criteria, adj R^2, Cp, BIC
mob.sum <- summary(mob.best.in.quotes)

crit <- data.frame(p = 1:9, BIC = mob.sum$bic, Cp = mob.sum$cp)

p1 <- ggplot(crit, aes(x = p, y = BIC)) + geom_line(color = "red") + geom_point() +
  xlab("Number of predictors") + ylab("BIC") +
  scale_x_continuous(breaks = 1:9)

p2 <- ggplot(crit, aes(x = p, y = Cp)) + geom_line(color = "red") + geom_point() +
  xlab("Number of predictors") + ylab("Mallow's Cp") +
  scale_x_continuous(breaks = 1:9)
grid.arrange(p1, p2)
```






```
data.frame(
  Adj.R2 = which.max(mob.sum$adjr2),
  CP = which.min(mob.sum$cp),
  BIC = which.min(mob.sum$bic)
)
```

```
##   Adj.R2 CP BIC
## 1      7  6   5
```

Helper functions for doing CV on the models.

Courtesy of Statistics Tools for High-throughput Data Analysis (STHDA)

```
library(caret)

# id: model id
# object: regsubsets object
# data: data used to fit regsubsets
# outcome: outcome variable
get_model_formula <- function(id, object, outcome){
  # get models data
  models <- summary(object)$which[id,-1]
  # Get outcome variable
```

```
  #form <- as.formula(object$call[[2]])
  #outcome <- all.vars(form)[1]
  # Get model predictors
  predictors <- names(which(models == TRUE))
  predictors <- paste(predictors, collapse = "+")
  # Build model formula
  as.formula(paste0(outcome, "~", predictors))
}

# number is number of folds
get_cv_error <- function(model.formula, data, number = 5){
  set.seed(nrow(data)) # All models have same folds
  train.control <- trainControl(method = "cv", number = number)
  cv <- train(model.formula, data = data, method = "lm",
              trControl = train.control)
  cv$results$RMSE
}
```

Applying to model subsets.

```
ids <- 1:9

cv.errors <-  map(ids, get_model_formula, mob.best.in.quotes, "Mobility") %>%
  map(get_cv_error, data = mob, number = ) %>%
  unlist()
cv.errors
```

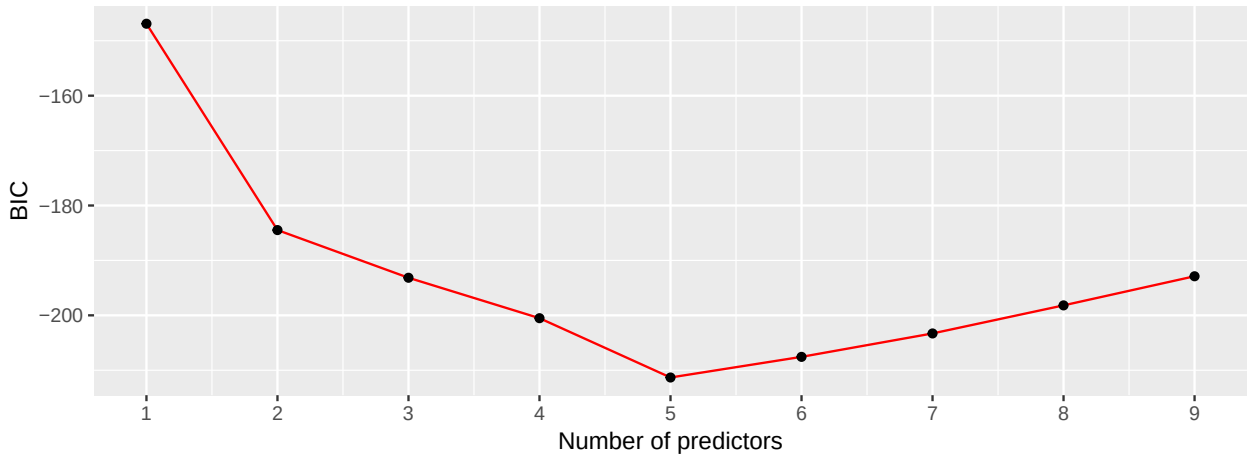
```
## [1] 0.04140775 0.03907286 0.03832089 0.03759709 0.03675913 0.03689629
## [7] 0.03687222 0.03735766 0.03754905
```

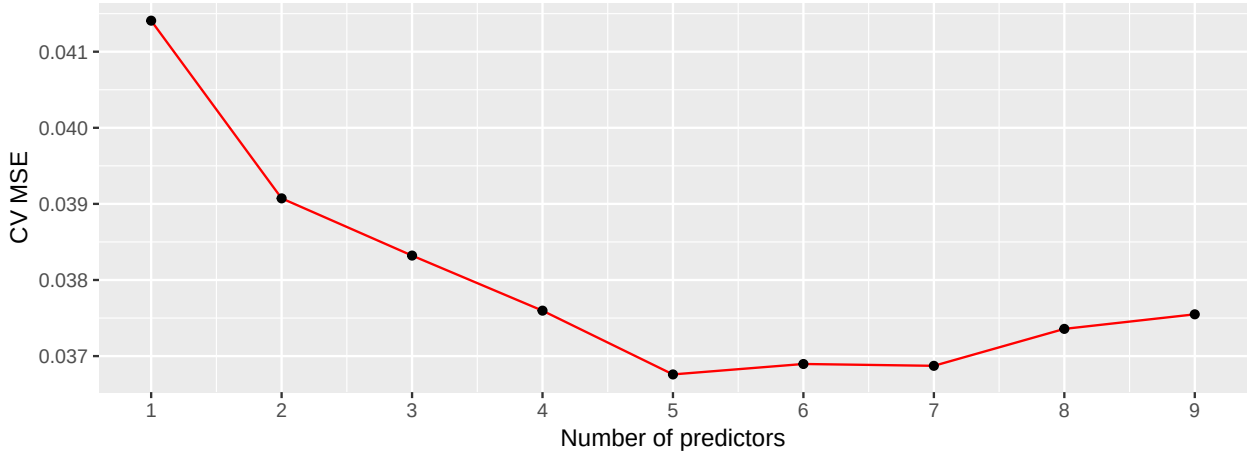
```
which.min(cv.errors)
```

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

```
crit$CV <- cv.errors
```

```
p1 <- ggplot(crit, aes(x = p, y = BIC)) + geom_line(color = "red") + geom_point() +
  xlab("Number of predictors") + ylab("BIC") +
  scale_x_continuous(breaks = 1:9)

p2 <- ggplot(crit, aes(x = p, y = CV)) + geom_line(color = "red") + geom_point() +
  xlab("Number of predictors") + ylab("CV MSE") +
  scale_x_continuous(breaks = 1:9)
grid.arrange(p1, p2)
```





Examining the "Best" Model

```
summary(mob.best.in.quotes)
```

```
## Subset selection object
## Call: regsubsets.formula(Mobility ~ ., data = mob, nvmax = 9)
## 9 Variables  (and intercept)
##               Forced in Forced out
## Population        FALSE      FALSE
## Black             FALSE      FALSE
## Seg_racial        FALSE      FALSE
## Seg_income        FALSE      FALSE
## Seg_poverty       FALSE      FALSE
## Seg_affluence     FALSE      FALSE
## Commute           FALSE      FALSE
## Income            FALSE      FALSE
## Gini              FALSE      FALSE
## 1 subsets of each size up to 9
## Selection Algorithm: exhaustive
##          Population Black Seg_racial Seg_income Seg_poverty Seg_affluence
## 1  ( 1 ) " "        " "   " "        " "         " "          " "
## 2  ( 1 ) " "        "*"   " "        " "         " "          " "
## 3  ( 1 ) " "        "*"   " "        " "         " "          " "
```

```
## 4  ( 1 ) "*"        "*"   "*"        " "        " "         " "
## 5  ( 1 ) "*"        "*"   "*"        " "        " "         " "
## 6  ( 1 ) "*"        "*"   "*"        " "        " "         " "
## 7  ( 1 ) "*"        "*"   "*"        " "        "*"         " "
## 8  ( 1 ) "*"        "*"   "*"        "*"        " "         "*"
## 9  ( 1 ) "*"        "*"   "*"        "*"        "*"         "*"
##          Commute Income Gini
## 1  ( 1 ) "*"     " "    " "
## 2  ( 1 ) "*"     " "    " "
## 3  ( 1 ) "*"     " "    "*"
## 4  ( 1 ) "*"     " "    " "
## 5  ( 1 ) "*"     " "    "*"
## 6  ( 1 ) "*"     "*"    "*"
## 7  ( 1 ) "*"     "*"    "*"
## 8  ( 1 ) "*"     "*"    "*"
## 9  ( 1 ) "*"     "*"    "*"
```

Both CV and BIC give the same suggested model.

```
best.formula <- get_model_formula(5, mob.best.in.quotes, "Mobility")

mob.lm <- lm(best.formula, mob)

summary(mob.lm)
```

```
##
## Call:
## lm(formula = best.formula, data = mob)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.131793 -0.021975 -0.004818  0.016409  0.181510
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  9.340e-02  1.714e-02   5.449 9.46e-08 ***
## Population   1.276e-08  2.859e-09   4.464 1.08e-05 ***
## Black       -8.149e-02  1.961e-02  -4.155 4.07e-05 ***
## Seg_racial  -1.003e-01  2.223e-02  -4.510 8.79e-06 ***
## Commute      1.600e-01  1.788e-02   8.948  < 2e-16 ***
## Gini        -1.244e-01  3.034e-02  -4.099 5.13e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.03685 on 358 degrees of freedom
## Multiple R-squared:  0.4922, Adjusted R-squared:  0.4852
## F-statistic: 69.41 on 5 and 358 DF,  p-value: < 2.2e-16
```

```
mob.test.lm <- lm(best.formula, mob.test)

summary(mob.test.lm)
```

```
##
## Call:
## lm(formula = best.formula, data = mob.test)
##
```

```
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.121730 -0.020387 -0.005131  0.014217  0.313919
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  9.948e-02  2.113e-02   4.707 3.60e-06 ***
## Population   7.355e-09  1.924e-09   3.823 0.000156 ***
## Black       -9.159e-02  2.075e-02  -4.413 1.35e-05 ***
## Seg_racial  -5.237e-02  2.408e-02  -2.175 0.030251 *
## Commute      1.521e-01  2.082e-02   7.304 1.82e-12 ***
## Gini        -1.396e-01  3.738e-02  -3.736 0.000218 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0398 on 359 degrees of freedom
## Multiple R-squared:  0.4639, Adjusted R-squared:  0.4564
## F-statistic: 62.12 on 5 and 359 DF,  p-value: < 2.2e-16
```

Shrinkage Methods

Another way to control bias and variance is through **regularization** or **shrinkage** of the slope coefficients.

Previously, we looked through a methodology that selects some subset of the predictors based off of a criterion outside of the OLS solutions.

$$\underline{\widehat{\beta}} = \underset{\underline{\beta}}{\operatorname{argmin}} \sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij})^2$$

Shrinkage methods instead modify the sum of squares formula and chooses $\underline{\widehat{\beta}}$ based off of constraints on the $\underline{\beta}$ vector.

This will take three forms:

1. Ridge Regression
2. Lasso Regression
3. Elastic Net Regression (A combination of the first two.)

Constraints in Optimization

An optimization problem has 2 components:

1. The "Objective function": e.g., $\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij})^2$.
2. The "constraint": e.g., "fewer than 5 non-zero entries in β".

A constrained minimization problem is written

$$\min_{\beta} f(\beta) \text{ subject to } C(\beta)$$

- $f(\beta)$ is the objective function. (Sum of squares for us!)
- $C(\beta)$ is the constraint

Ridge Regression

One way to do this for regression is to solve (say):

$$\min_{\beta} \sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij})^2$$
$$\text{s.t.} \sum_{j=1}^{p} \beta_j^2 < t$$

for some $t > 0$.

- This is called "ridge regression".
- The **minimizer** of this problem is called $\widehat{\beta}_{r,t}$

Compare this to least squares:

$$\min_{\beta} \sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij})^2$$
$$\text{s.t.} \beta \in \mathbb{R}^p$$

Try to think about how the constraint in ridge regression changes things.

- If t is small, what happens?
- What about if t is large, or $t \to \infty$

Rewriting Ridge Regression

An equivalent way to write ridge regression

$$\widehat{\beta}_{r,t} = \underset{\sum_{j=1}^{p} \beta_j^2 \leq t}{\operatorname{argmin}} \sum_i (y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij})^2$$

is in the **Lagrangian** (ahhhh... calculus) form

$$\widehat{\beta}_{r,\lambda} = \underset{\beta}{\operatorname{argmin}} \sum_i (y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij})^2 + \lambda \sum_{j=1}^{p} \beta_j^2.$$

For every λ there is a unique t (and vice versa) that makes

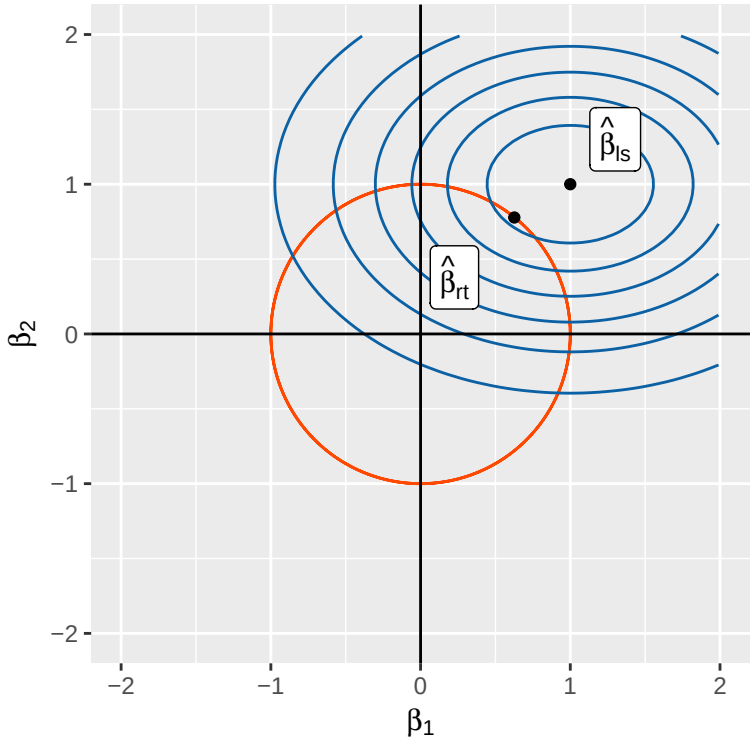
$$\widehat{\beta}_{r,t} = \widehat{\beta}_{r,\lambda}$$

Observe:

- $\lambda = 0$ (or $t = \infty$) makes $\widehat{\beta}_{r,\lambda}$ equivalent to the OLS solution $\widehat{\beta}_{ls}$
- Any $\lambda > 0$ (or $t < \infty$) penalizes larger values of β, effectively shrinking them.

Note: λ and t are known as **tuning parameters**. Most of the time, Ridge regression (And LASSO) is characterized in the Lagrangian form with the tuning parameter λ

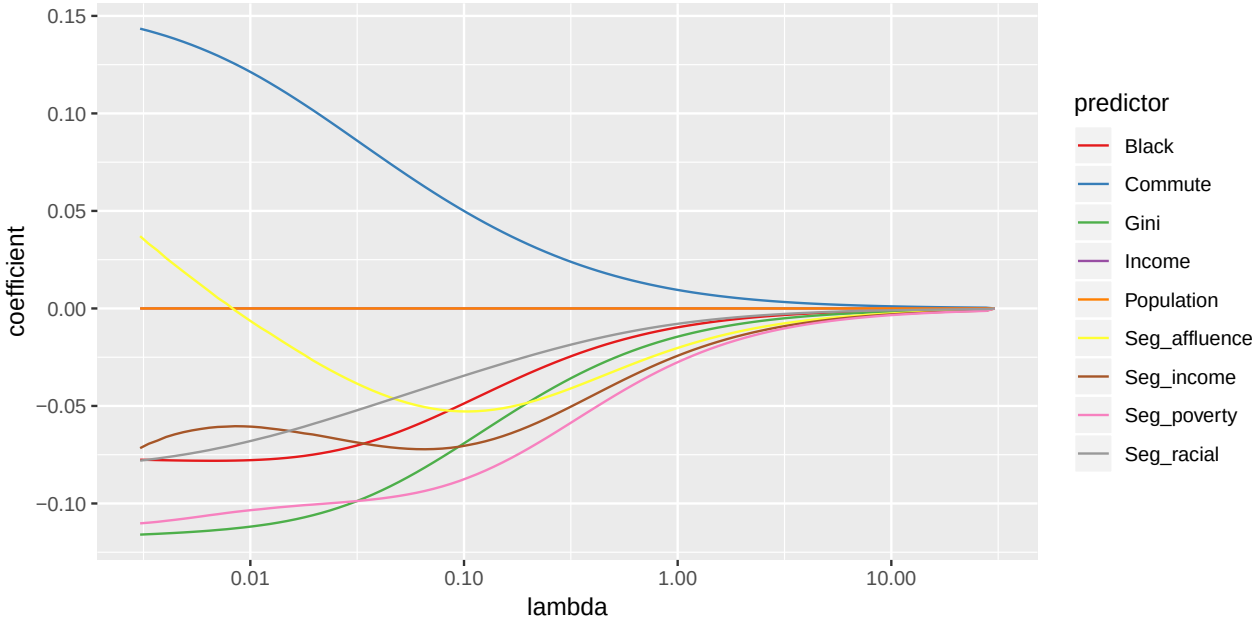
Geometry of ridge regression (2 dimensions)



Ridge regression path

We will go over the mechanics of computing ridge regression models. For now, here is an example using the `mob` data from earlier.

Each colored curve indicates the coefficient values for each predictor variable as λ is increased. This curve can be referred to as the **path** of the coefficient.



Some things to think about:

- All predictors are included
- Coefficients shrink to zero as $\lambda \to \infty$
- Why do the coefficients shrink?

Least Squares Scaling Invariance

Let's multiply our design matrix by a factor of 10 to get $\widetilde{\underline{X}} = 10\underline{X}$.

Then:

$$\widetilde{\beta}_{\text{ls}} = (\widetilde{\underline{X}}^\top \widetilde{\underline{X}})^{-1}\widetilde{\underline{X}}^\top Y = \frac{1}{10}(\widetilde{\underline{X}}^\top \widetilde{\underline{X}})^{-1}\widetilde{\underline{X}}^\top Y = \frac{\widehat{\beta}_{\text{ls}}}{10}$$

So, multiplying our data by ten just results in our estimates being reduced by one tenth.
Hence, any prediction is left unchanged:

$$\widetilde{\underline{X}}\widetilde{\beta}_{\text{ls}} = \underline{X}\widehat{\beta}_{ls}$$

This means, for instance, if we have a predictor measured in miles, then we will get the same predictios even if we change it to kilometers.

Least Squares Scaling Invariance Example

Sometimes we will change the scale of our predictor variables. Ex: Convert dollars to thousands of dollars, convert Fahrenheit to Celsius, pounds to kilograms, etc.

In Ordinary Least Squares regression, the coefficients respond to this change such that there is no impact on our predictions.

```
n = 20
set.seed(777)
X = matrix(runif(2*n,0,1), ncol=2)
Y = X %*% c(.5,1.5) + rnorm(n,0,.25)
Xtilde   = 2*X
Ytilde   = Y - mean(Y)
summary(lm(Y~X))$coefficients
```

```
##               Estimate Std. Error   t value     Pr(>|t|)
## (Intercept) -0.1381674  0.2325170 -0.594225 5.601872e-01
## X1           0.7130707  0.2498274  2.854253 1.097518e-02
## X2           1.5353739  0.2790432  5.502280 3.885673e-05
```

```
summary(lm(Y~Xtilde))$coefficients
```

```
##               Estimate Std. Error   t value     Pr(>|t|)
## (Intercept) -0.1381674  0.2325170 -0.594225 5.601872e-01
## Xtilde1      0.3565353  0.1249137  2.854253 1.097518e-02
## Xtilde2      0.7676870  0.1395216  5.502280 3.885673e-05
```

```
summary(lm(Ytilde~Xtilde))$coefficients
```

```
##               Estimate Std. Error   t value     Pr(>|t|)
## (Intercept) -1.2153624  0.2325170 -5.226982 6.834199e-05
## Xtilde1      0.3565353  0.1249137  2.854253 1.097518e-02
## Xtilde2      0.7676870  0.1395216  5.502280 3.885673e-05
```

Ridge regression (and other regularized methods) is not

In Ridge regression, the coefficients respond weirdly to scaling. To bypass this issue, all predictor variables are standardized before a model is estimated.

$$\widetilde{x}_{ij} = \frac{x_{ij}}{\sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(x_{ij} - \overline{x}_j\right)^2}}$$

This makes it so changes in scale have no effect on the coefficients.

```
library(MASS)
lm.ridge(Y~X, lambda=1)$coef
```

```
##        X1        X2
## 0.1832046 0.3588517
```

```
lm.ridge(Y~Xtilde, lambda=1)$coef
```

```
##   Xtilde1   Xtilde2
## 0.1832046 0.3588517
```

```
lm.ridge(Ytilde~Xtilde, lambda=1)$coef
```

```
##   Xtilde1   Xtilde2
## 0.1832046 0.3588517
```

`lm.ridge` automatically scales every column of $\underline{X}$ to have mean zero and Euclidean norm 1.

It also centers Y.

Together, this means there is no intercept. (We don't penalize the intercept)

In `R`: `scale(X)` defaults to mean 0, SD 1. But you can change either.

Another version is in the package `glmnet`. More on this in a bit.

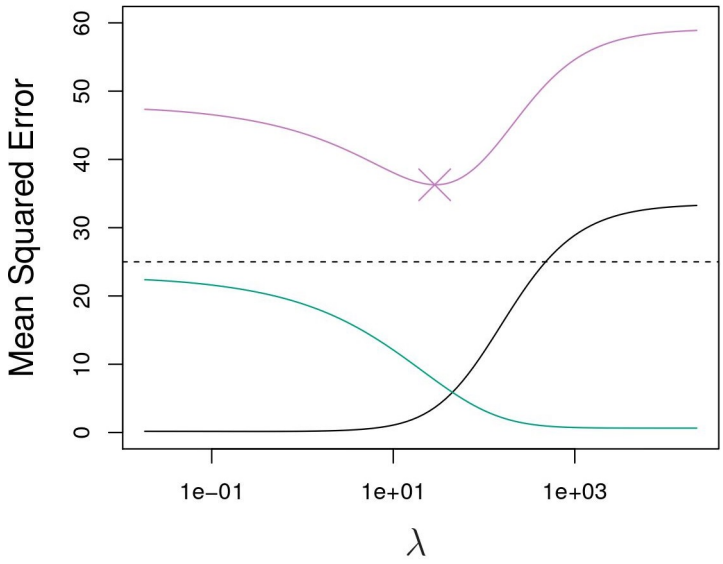


Figure 1: Pink = MSE, Green = Variance, Black = Bias

Why Ridge Regression?

Ridge regression is a method meant to balance the trade-offs of bias versus variances by shrinking coefficients.

Ridge Regression and Multicollinearity

One nice thing about ridge regression is that it has a closed-form solution (like OLS)

$$\widehat{\beta}_{r,\lambda} = (\underline{X}'\underline{X} + \lambda I)^{-1}\underline{X}'Y$$

This is easy to calculate in `R` for any λ.

- Ridge regression makes the coefficients smaller relative to OLS.
- If there is multicollinearity, ridge regression will prevent numerical issues. Specifically with the inversion of $\underline{X}^\top\underline{X}$.

Multicollinearity is a phenomenon in which a combination of predictor variables is extremely similar to another predictor variable. Some comments:

- A better phrase that is sometimes used is "$\underline{X}$ is ill-conditioned"

Conclusion: $(\underline{X}^\top\underline{X})^{-1}$ can be really unstable, while $(\underline{X}^\top\underline{X} + \lambda I)^{-1}$ is not.

Can we get the best of both worlds? Variable Selection and Multicollinearity Stability

To recap:

- Deciding which predictors to include, adding quadratic terms, or interactions is **model selection**.

- Ridge regression provides regularization, which trades off bias and variance and also stabilizes multicollinearity.

Ridge regression: $\min \sum_i (y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij})^2 + \lambda \sum_{j=1}^p \beta_j^2$.

- Let $\underline{\beta}$ be the vector of coefficients. Ridge regression is based off of a constraint of the ℓ_2 norm: $\sum_{j=1}^p \beta_j^2 = ||\underline{\beta}||_2^2 \leq t$

Best linear regression model: $\min \sum_{i=1}^n (y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij})^2$ subject to the number of non-zero β_j is less than or equal to p

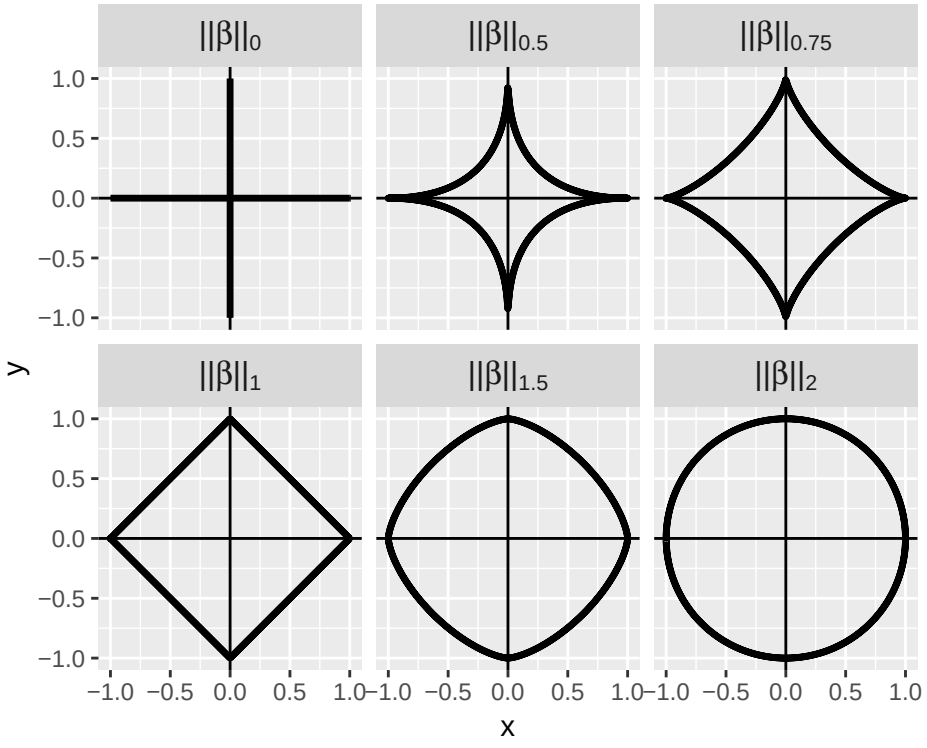
- Best subset selection is based on constraining the l_0 norm. $||\underline{\beta}||_0 \leq s$. The ℓ_0 norm is simply the number of non-zero elements of a vector.

Finding the best linear model is a nonconvex optimization problem (In fact, it is NP-hard)

Ridge regression is convex (easy to solve), but doesn't do model selection. (We always have the maximum number of predictors.)

It would be great if we could do model selection within a convex optimization problem.

Geometry of convexity



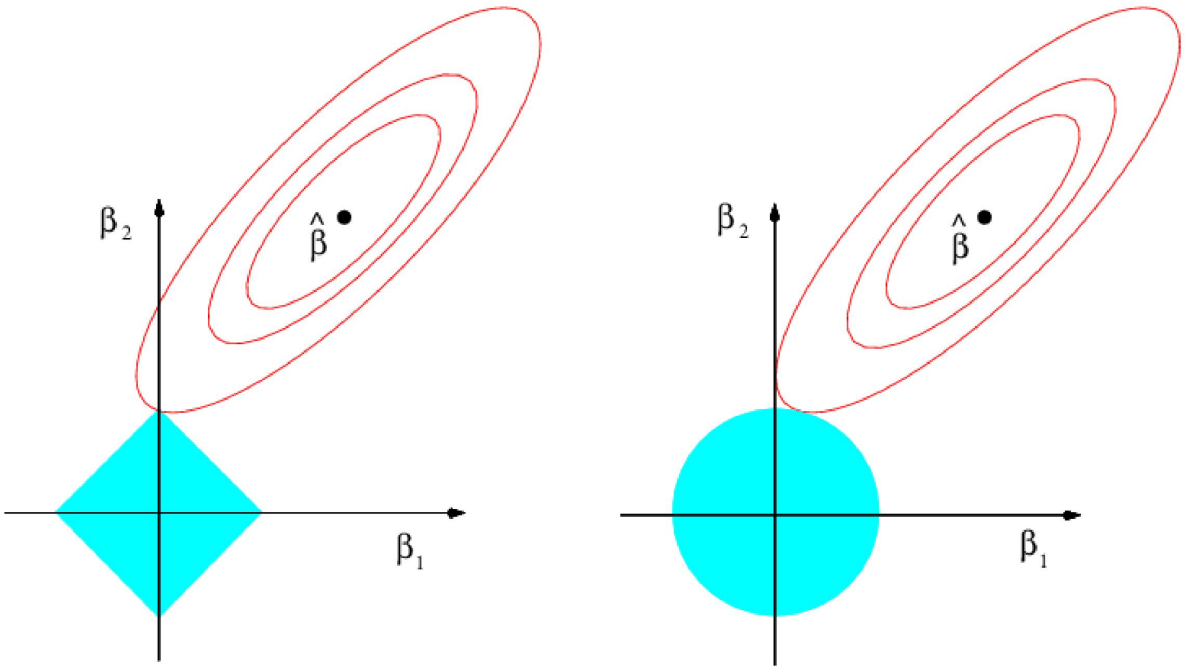


Figure 2: Contours of the SSE in red, and contraint regions in blue

The best of both worlds

The main important point here is that the set of potential values for $\widehat{\beta}_{l,t}$ has corners. This means that depending on the value for λ, certain variables will have a coefficient of zero.

The LASSO: ℓ_1-regularized regression

Known as

- "LASSO" (least absolute shrinkage and selection operator)
- "basis pursuit"

This form of regression makes use of the ℓ_1 norm. $||\underline{\beta}||_1 = \sum_{j=1}^{p} |\beta_j|$

Lasso minimizes the least squares formula:

$$\sum_i (y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij})^2$$

subject to the constraint $||\underline{\beta}||_1 = \sum_{j=1}^{p} |\beta_j| \leq t$

In its corresponding Lagrangian dual form:

$$\widehat{\beta}_{l,\lambda} = \arg\min_{\underline{\beta}} \sum_i (y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij})^2 + \lambda ||\underline{\beta}||_1$$

Lasso

While the ridge solution can be easily computed

$$\widehat{\beta}_{r,\lambda} = \underset{\underline{\beta}}{\operatorname{argmin}} \sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \lambda||\underline{\beta}||_2^2 = (\underline{X}^\top \underline{X} + \lambda I)^{-1}\underline{X}^\top Y$$

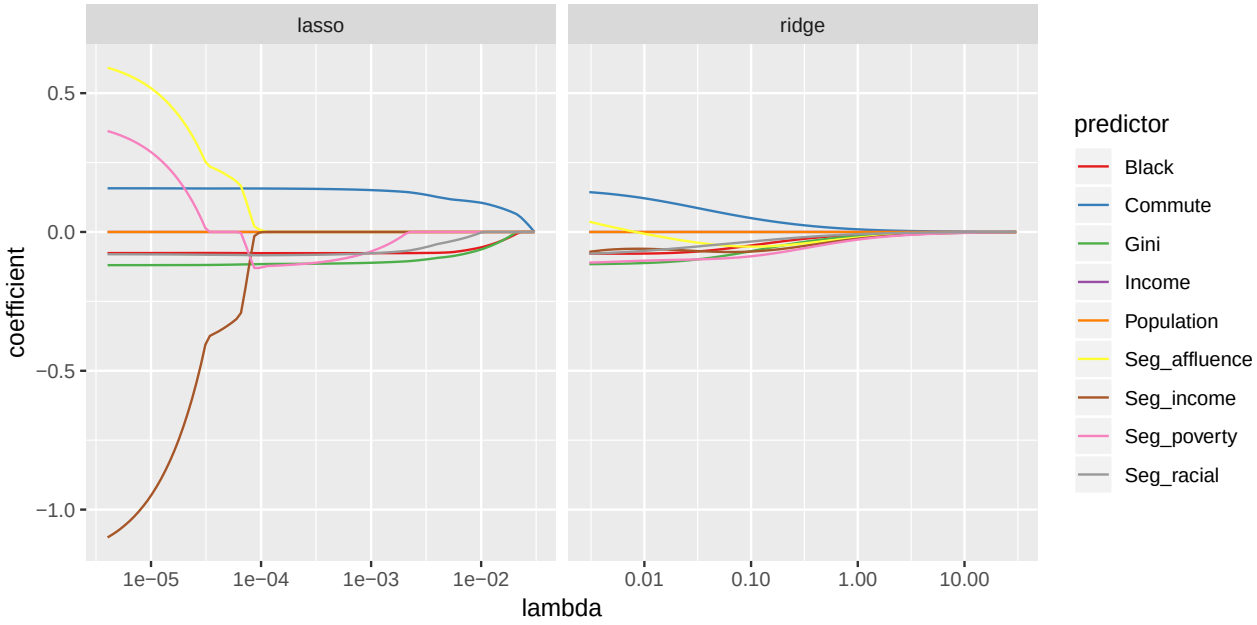
the lasso solution

$$\widehat{\beta}_{l,\lambda} = \underset{\underline{\beta}}{\operatorname{argmin}} \sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \lambda||\underline{\beta}||_1 = ??$$

doesn't have a closed form solution.

However, because the optimization problem is convex, there exist efficient algorithms for computing it

Coefficient path: ridge vs lasso



Packages

There are three main `R` implementations for doing ridge and/or lasso regression

- `glmnet`, the most commonly used.
- `lars`, this does exist, and it has its uses, but we will concentrate on `glmnet`
- `lm.ridge` in the `MASS` package which only does ridge regression.

Earlier, `lm.ridge` from the MASS library was used purely for a visual aid.

Using `glmnet`

```r
library(glmnet)
```

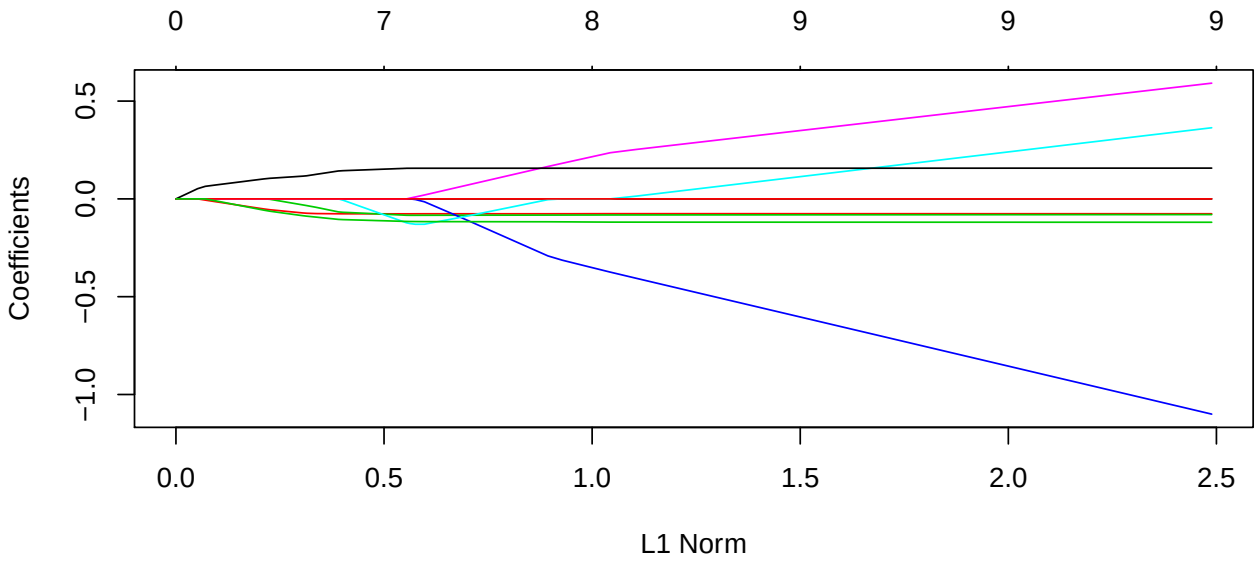
Basic function takes the form `glmnet(x, y, alpha = 1, nlambda = 100, lambda = NULL)`

- `x` is the matrix of predictor variables, rows are observations, columns are the variables
- `y` is the response vector.
- `alpha` determines whether ridge or lasso or a combination of the two (elastic net) is used.
  - `alpha = 0` is ridge regression
  - `alpha = 1` is lasso regression (default)
  - `alpha` in between is elastic net regression
- `nlambda`is the number of λ values to try, default is 100.
- `lambda` if specified is a grid of λ values to try.

If you want to do ridge regression, it is as easy as `glmnet(x,y)`.

Lasso paths

```r
# Using the mob data
# Glmnet needs the the response vector and data matrix

Y = mob$Mobility
X = as.matrix(mob[,names(mob)!=c('Mobility')])

lasso = glmnet(X,Y)
# lars.out = lars(X,Y)
plot(lasso)
```



Choosing the lambda

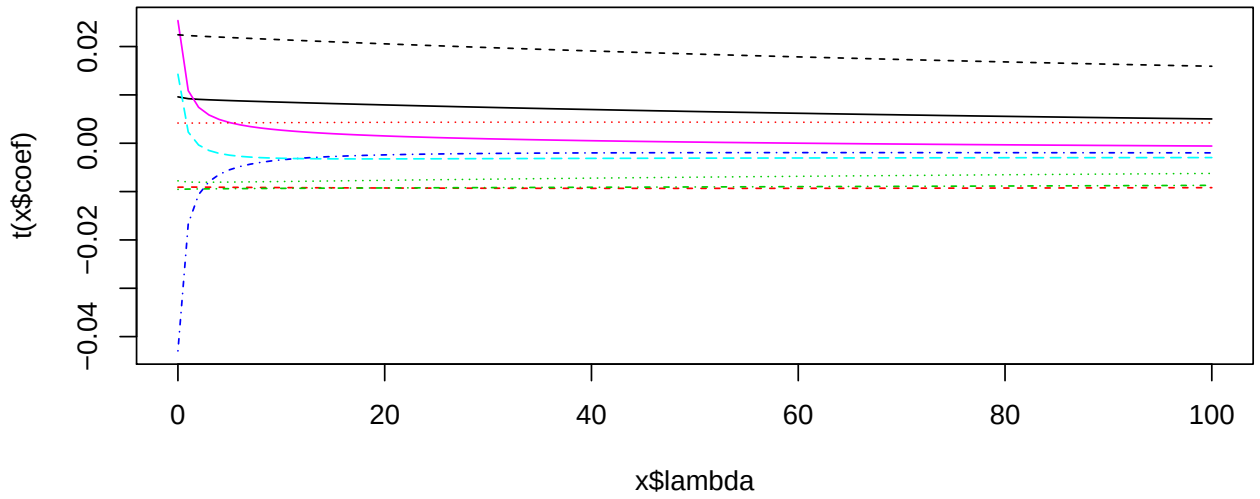
We have so far covered λ as just a set of values. However, there is a choice of lambda that minimizes the prediction error.

This choice (maybe unsurprisingly at this point) is done via cross validation.

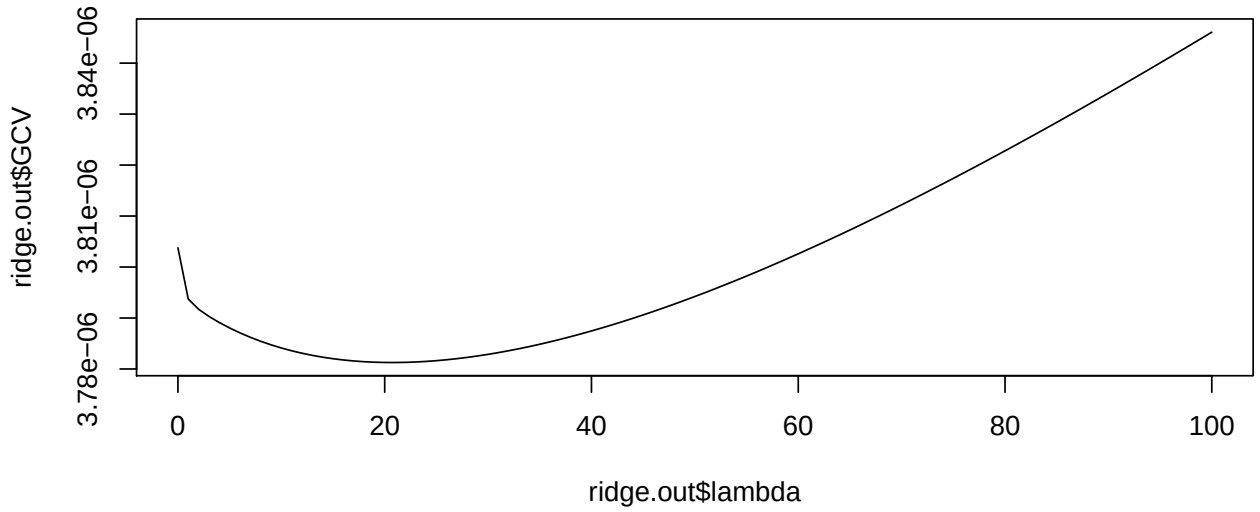
The different functions provide methods for selecting λ based on which one has the smallest cross validation error.

Ridge regression, `lm.ridge` version

```
par(mfrow=c(1,1))
# 1. Estimate the model (note, this uses a formula, and you must supply lambda)
ridge.out = lm.ridge(Mobility ~ ., data=mob, lambda = 0:100)

# 2. Plot it
plot(ridge.out)
```




```
# (2a). If you chose lambda poorly, this will look bad, try again
# 3. Choose lambda using GCV
plot(ridge.out$lambda,ridge.out$GCV,ty='l')
```



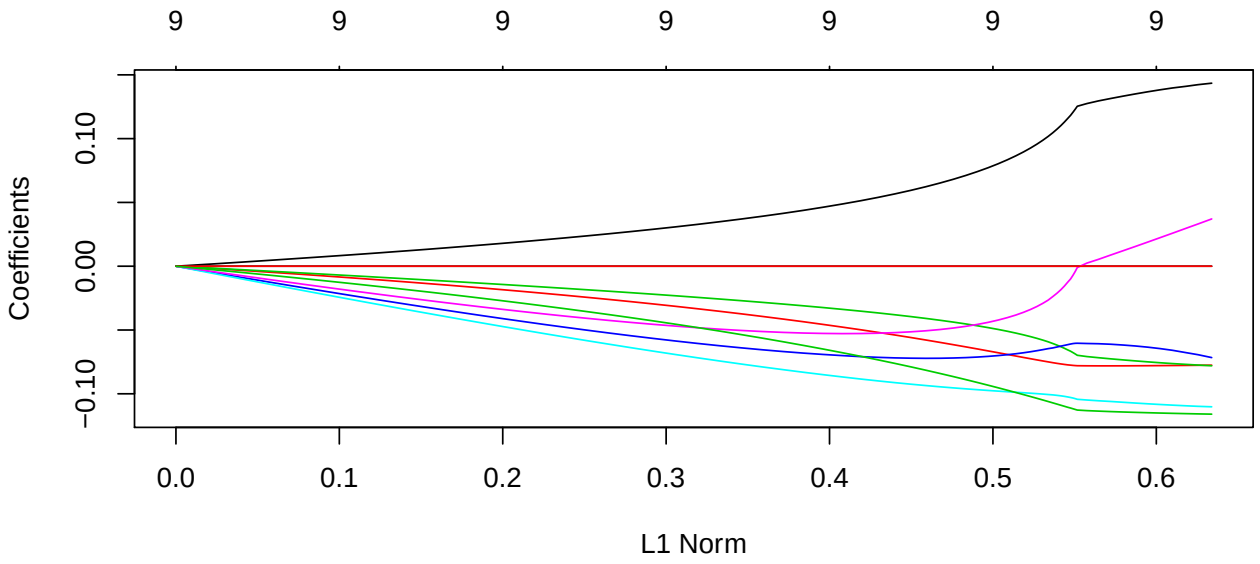

```
# 4. If there's a minimum, FIND IT, else try again
best.lam = which.min(ridge.out$GCV) # the index, not the value
```

```r
# 5. Return the coefs/predictions for the best model

coefs = coefficients(ridge.out)[best.lam,]
preds = as.matrix(dplyr::select(mob,-Mobility)) %*% coefs[-1] + coefs[1]

coefs
```

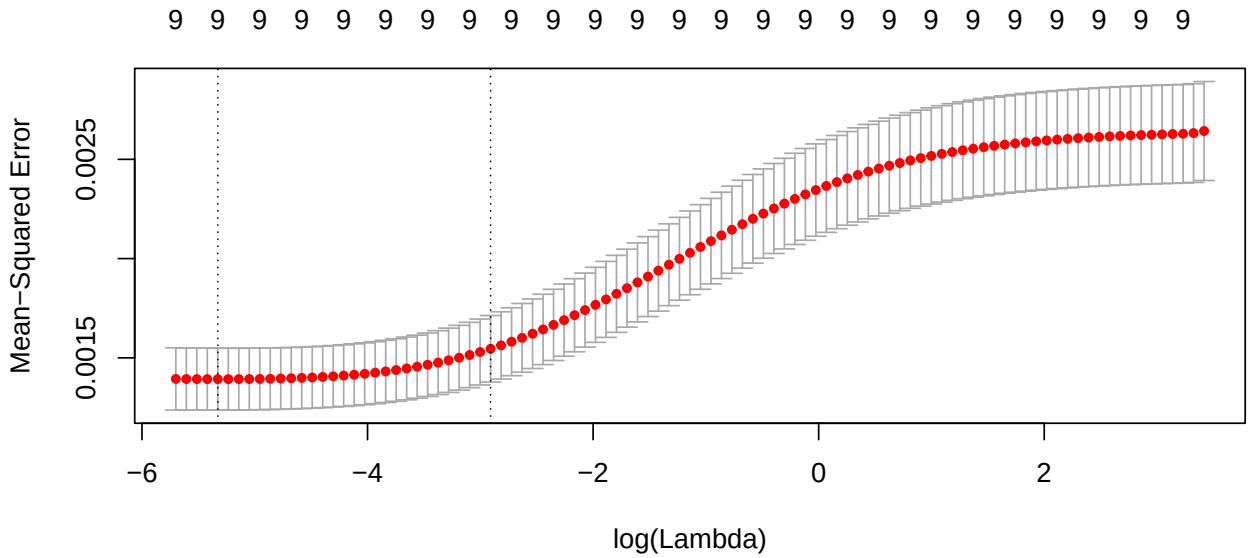
```
##                  Population         Black    Seg_racial    Seg_income
##  7.699870e-02  9.718259e-09 -7.751067e-02 -7.810203e-02 -7.475534e-02
##   Seg_poverty Seg_affluence       Commute        Income          Gini
## -1.113293e-01  4.070960e-02  1.437519e-01  7.228606e-07 -1.160585e-01
```

glmnet version (ridge)

```r
# 1. Estimate cv and model at once, no formula version
lasso.glmnet = cv.glmnet(X,Y, alpha = 0)

# 2. Plot the coefficient path
plot(lasso.glmnet$glmnet.fit) # the glmnet.fit == glmnet(X,Y)
```




```r
# 3. Choose lambda using CV
plot(lasso.glmnet) #a different plot method for the cv fit
```



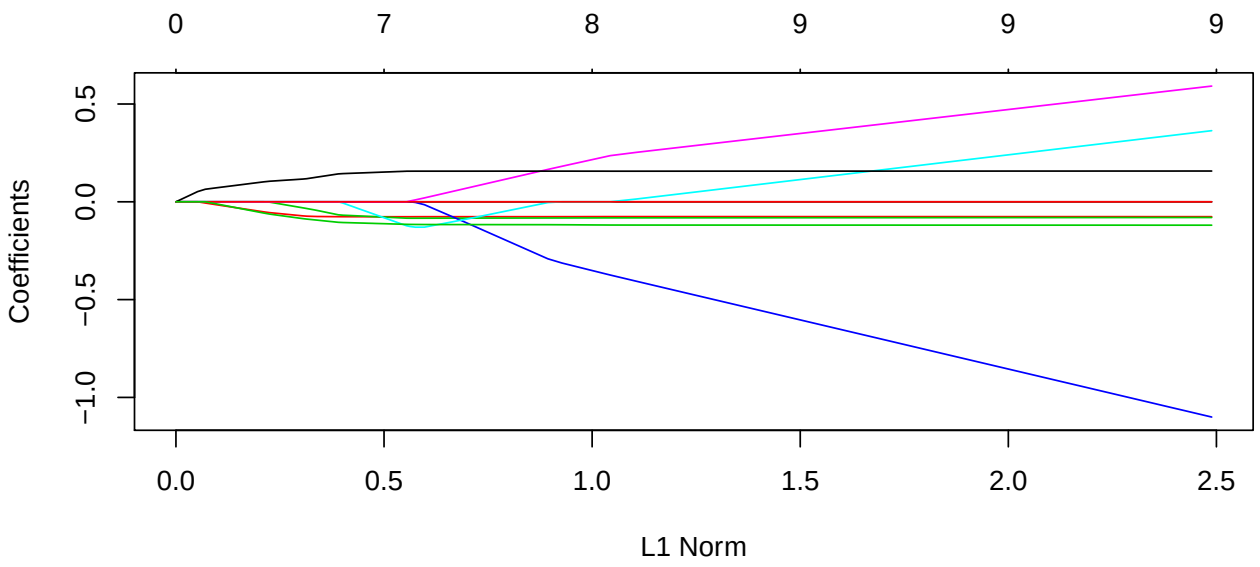

```
# 4. If the dashed lines are at the boundaries, redo it with a better set of lambda
best.lam = lasso.glmnet$lambda.min # the value, not the location (or lasso$lambda.1se)

# 5. Return the coefs/predictions for the best model
coefs.glmnet = coefficients(lasso.glmnet, s = best.lam)
coefs.glmnet
```

```
## 10 x 1 sparse Matrix of class "dgCMatrix"
##                              1
## (Intercept)    7.977602e-02
## Population     8.906872e-09
## Black         -7.798690e-02
## Seg_racial    -7.498099e-02
## Seg_income    -6.345949e-02
## Seg_poverty   -1.078454e-01
## Seg_affluence  1.894033e-02
## Commute        1.367299e-01
## Income         7.282192e-07
## Gini          -1.148499e-01
```

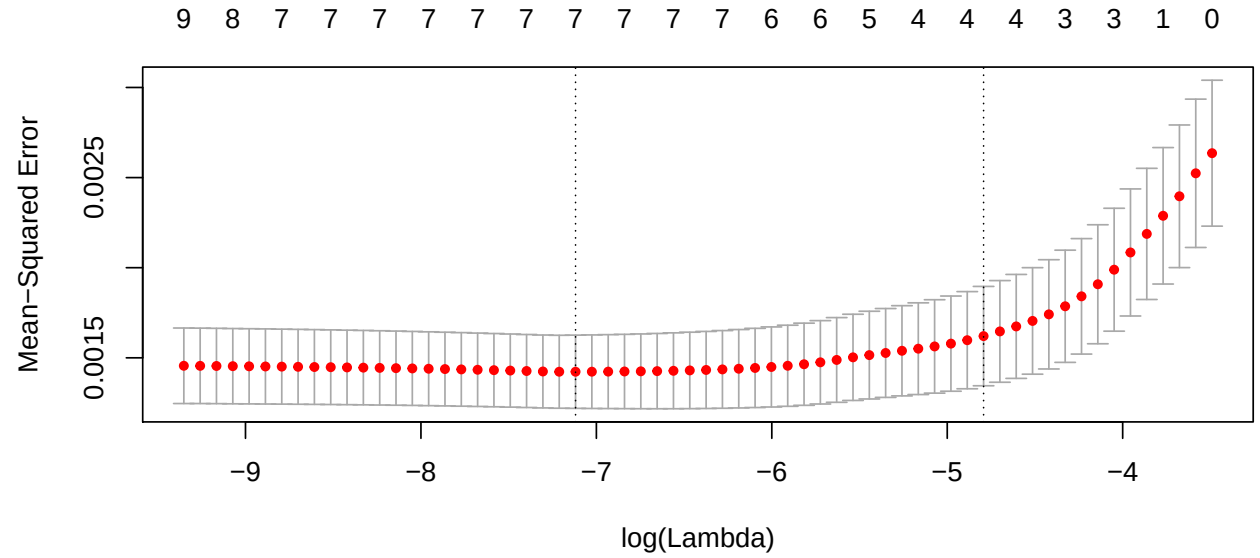
```
preds.glmnet = predict(lasso.glmnet, newx = X, s = best.lam) # must supply `newx`
```

glmnet version (lasso)

```
# 1. Estimate cv and model at once, no formula version
lasso.glmnet = cv.glmnet(X,Y, alpha = 1)

# 2. Plot the coefficient path
plot(lasso.glmnet$glmnet.fit) # the glmnet.fit == glmnet(X,Y)
```




```
# 3. Choose lambda using CV
plot(lasso.glmnet) #a different plot method for the cv fit
```




```
# 4. If the dashed lines are at the boundaries, redo it with a better set of lambda
best.lam = lasso.glmnet$lambda.min # the value, not the location (or lasso$lambda.1se)

# 5. Return the coefs/predictions for the best model
coefs.glmnet = coefficients(lasso.glmnet, s = best.lam)
coefs.glmnet
```

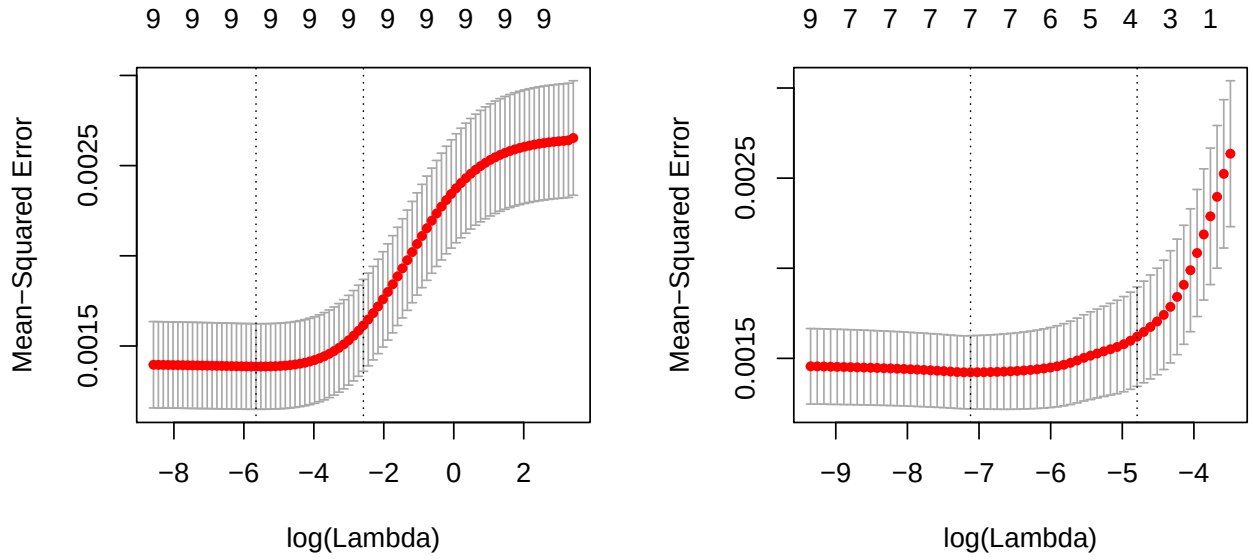
```
## 10 x 1 sparse Matrix of class "dgCMatrix"
##                              1
## (Intercept)    7.311971e-02
## Population     8.906398e-09
## Black         -7.633101e-02
## Seg_racial    -7.823532e-02
## Seg_income     .
## Seg_poverty   -8.128598e-02
```

```
## Seg_affluence  .
## Commute         1.520063e-01
## Income          6.087810e-07
## Gini           -1.118854e-01
```

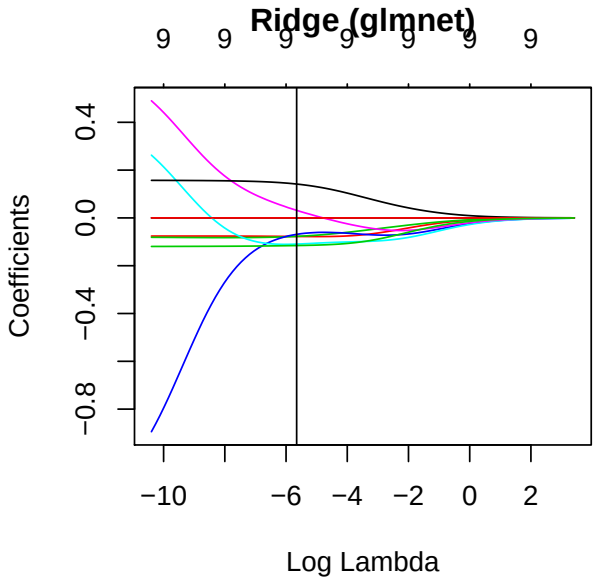
```
preds.glmnet = predict(lasso.glmnet, newx = X, s = best.lam) # must supply `newx`
```

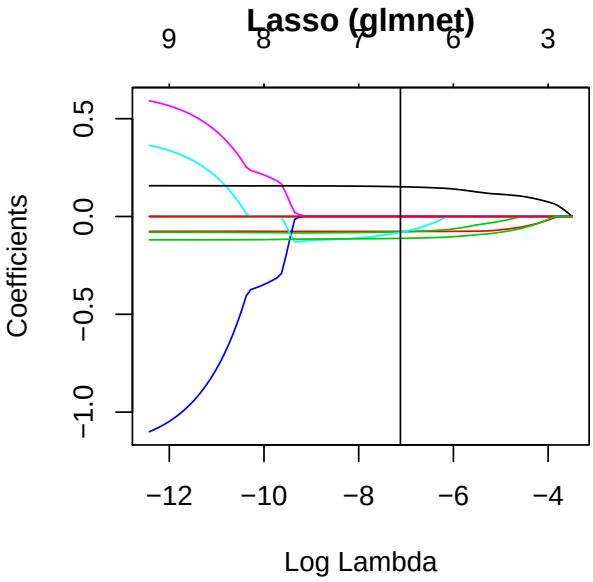
Paths with chosen lambda (lasso and ridge)

```
ridge.glmnet = cv.glmnet(X,Y,alpha=0,lambda.min.ratio=1e-10) # added to get a minimum
par(mfrow=c(1,2))
plot(ridge.glmnet)
plot(lasso.glmnet)
```




```
best.lam.ridge = ridge.glmnet$lambda.min
plot(ridge.glmnet$glmnet.fit,xvar='lambda', main='Ridge (glmnet)')
abline(v=log(best.lam.ridge))
plot(lasso.glmnet$glmnet.fit,xvar='lambda', main='Lasso (glmnet)')
abline(v=log(best.lam))
```


Ridge (glmnet)
9 9 9 9 9 9 9
Coefficients
0.4
0.0
−0.4
−0.8
−10
−6
−4
−2
0
2
Log Lambda


Lasso (glmnet)
9 8 7 6 3
Coefficients
0.5
0.0
−0.5
−1.0
−12
−10
−8
−6
−4
Log Lambda