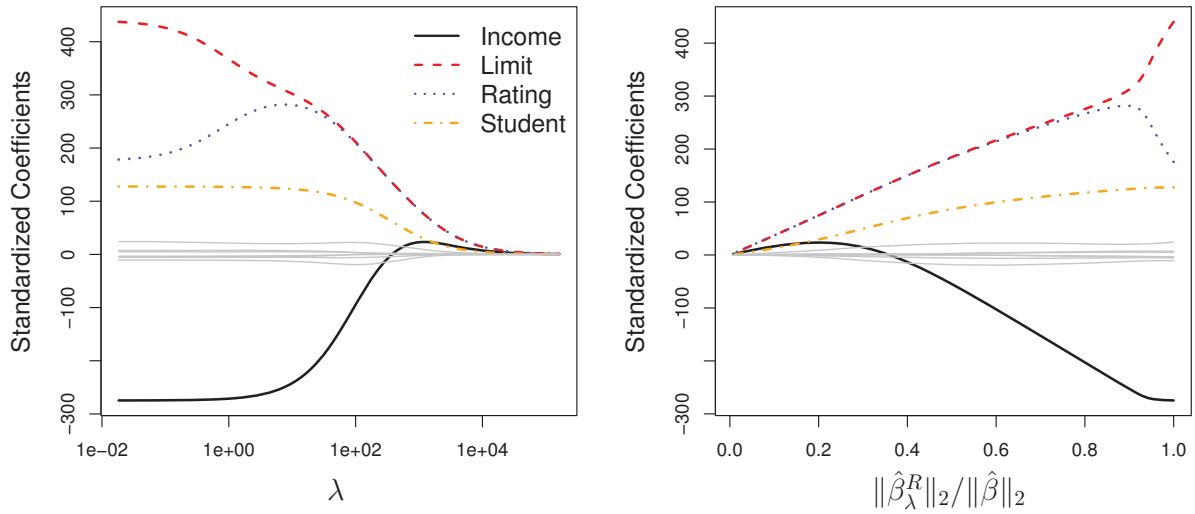


The standardized ridge regression coefficients are displayed for the Credit data set, as a function of λ and $\|\hat{\beta}^R_\lambda\|_2/\|\hat{\beta}\|_2$.

- In the left-hand panel, each curve corresponds to the ridge regression coefficient estimate for one of the ten variables, plotted as a function of λ.
- The right-hand panel displays the same ridge coefficient estimates as the left-hand panel, but instead of displaying λ on the x-axis, we now display $\|\hat{\beta}^R_\lambda\|_2/\|\hat{\beta}\|_2$, where $\hat{\beta}$ denotes the vector of least squares coefficient estimates. The notation $\|\beta\|_2$ denotes the ℓ_2 norm (pronounced "ell 2") of a vector, and is defined as $\|\beta\|_2 = \sqrt{\sum_{j=1}^{p} \beta_j^2}$. It measures the distance of β from zero.

Scaling of predictors in ridge regression:

- The standard least squares coefficient estimates are *scale equivariant*: multiplying X_j by a constant c simply leads to a scaling of the least squares coefficient estimates by a factor of $1/c$. In other words, regardless of how the jth predictor is scaled, $X_j\hat{\beta}_j$ will remain the same.
- In contrast, the ridge regression coefficient estimates can change substantially when multiplying a given predictor by a constant, due to the sum of squared coefficients term in the ridge regression formulation.

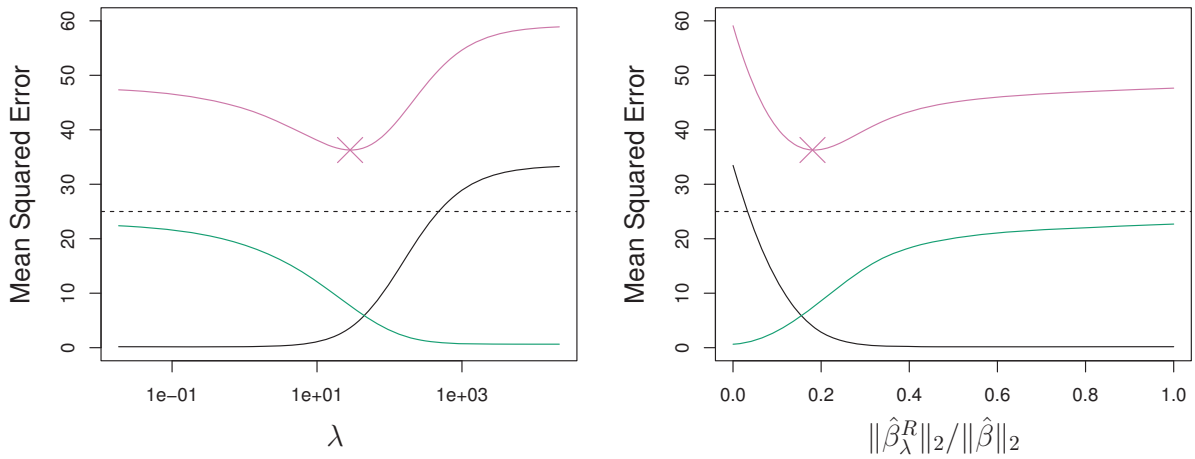
- Therefore, it is best to apply ridge regression after standardizing the predictors, using the formula

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

so that they are all on the same scale.

Why does ridge regression improve over least squares?

Ridge regression's advantage over least squares is rooted in bias-variance trade-off. As λ increases, the flexibility of the ridge regression fit decreases, leading to decreased variance but increased bias.



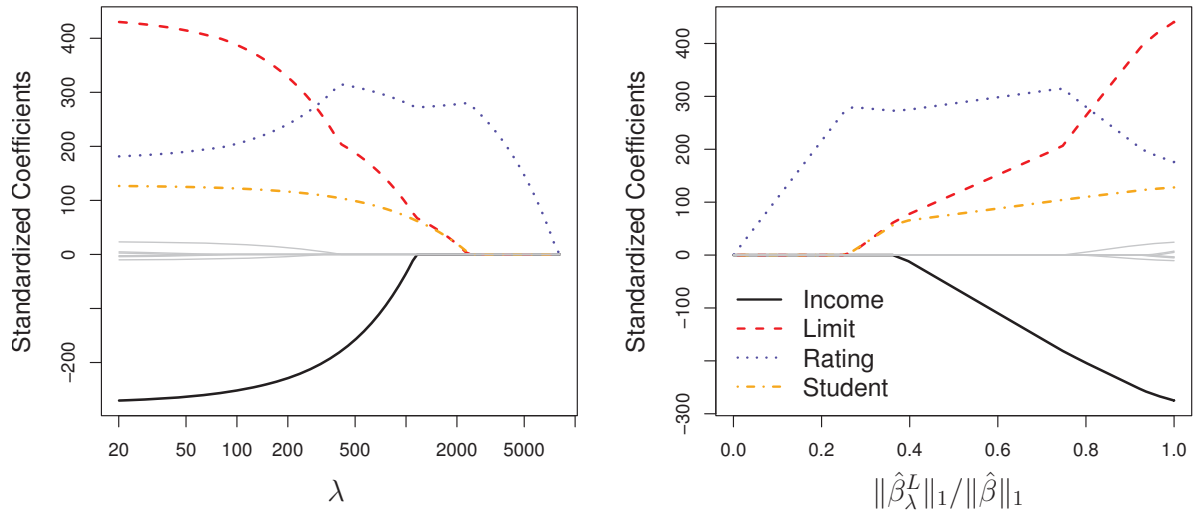
Simulated data with $n = 50$ observations, $p = 45$ predictors, all having nonzero coefficients. Squared bias (black), variance (green), and test mean squared error (purple) for the ridge regression predictions on the simulated data set, as a function of λ and $\|\hat{\beta}_\lambda^R\|_2/\|\hat{\beta}\|_2$. The horizontal dashed lines indicate the minimum possible MSE. The purple crosses indicate the ridge regression models for which the MSE is smallest.

The Lasso:

- Ridge regression does have one obvious disadvantage. Unlike best subset, forward stepwise, and backward stepwise selection, which will generally select models that involve just a subset of the variables, ridge regression will include all p predictors in the final model.
- The lasso is a relatively recent alternative to ridge regression that overcomes this disadvantage. The lasso coefficients, $\hat{\beta}_\lambda^L$, minimize the quantity

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

  In statistical parlance, the lasso uses an ℓ_1 (pronounced "ell 1") penalty instead of an ℓ_2 penalty. The ℓ_1 norm of a coefficient vector β is given by $\|\beta\|_1 = \sum|\beta_j|$.
- As with ridge regression, the lasso shrinks the coefficient estimates towards zero.
- However, in the case of the lasso, the ℓ_1 penalty has the effect of forcing some of the coefficient estimates to be exactly equal to zero when the tuning parameter λ is sufficiently large.
- Hence, much like best subset selection, the lasso performs variable selection.
- We say that the lasso yields sparse models—that is, models that involve only a subset of the variables.
- As in ridge regression, selecting a good value of λ for the lasso is critical; cross-validation is again the method of choice.



The standardized lasso coefficients on the Credit data set are shown as a function of λ and $\|\hat{\beta}_\lambda^L\|_1/\|\hat{\beta}\|_1$.

The Variable Selection Property of the Lasso:

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

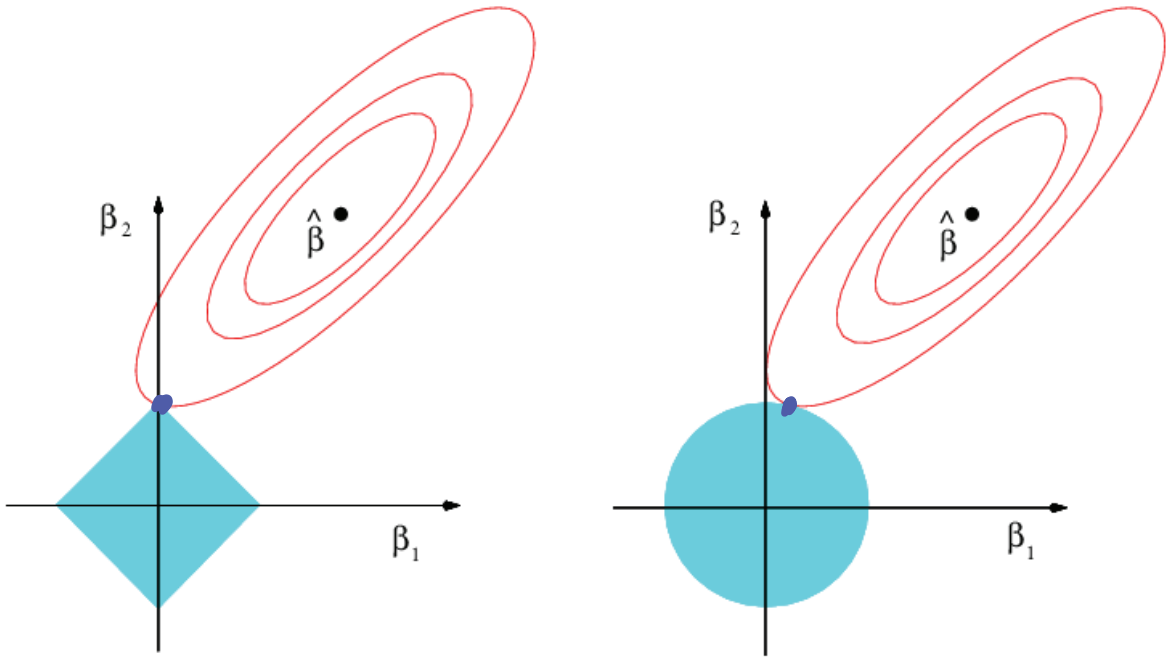
One can show that the lasso and ridge regression coefficient estimates solve the problems

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

and

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

respectively.

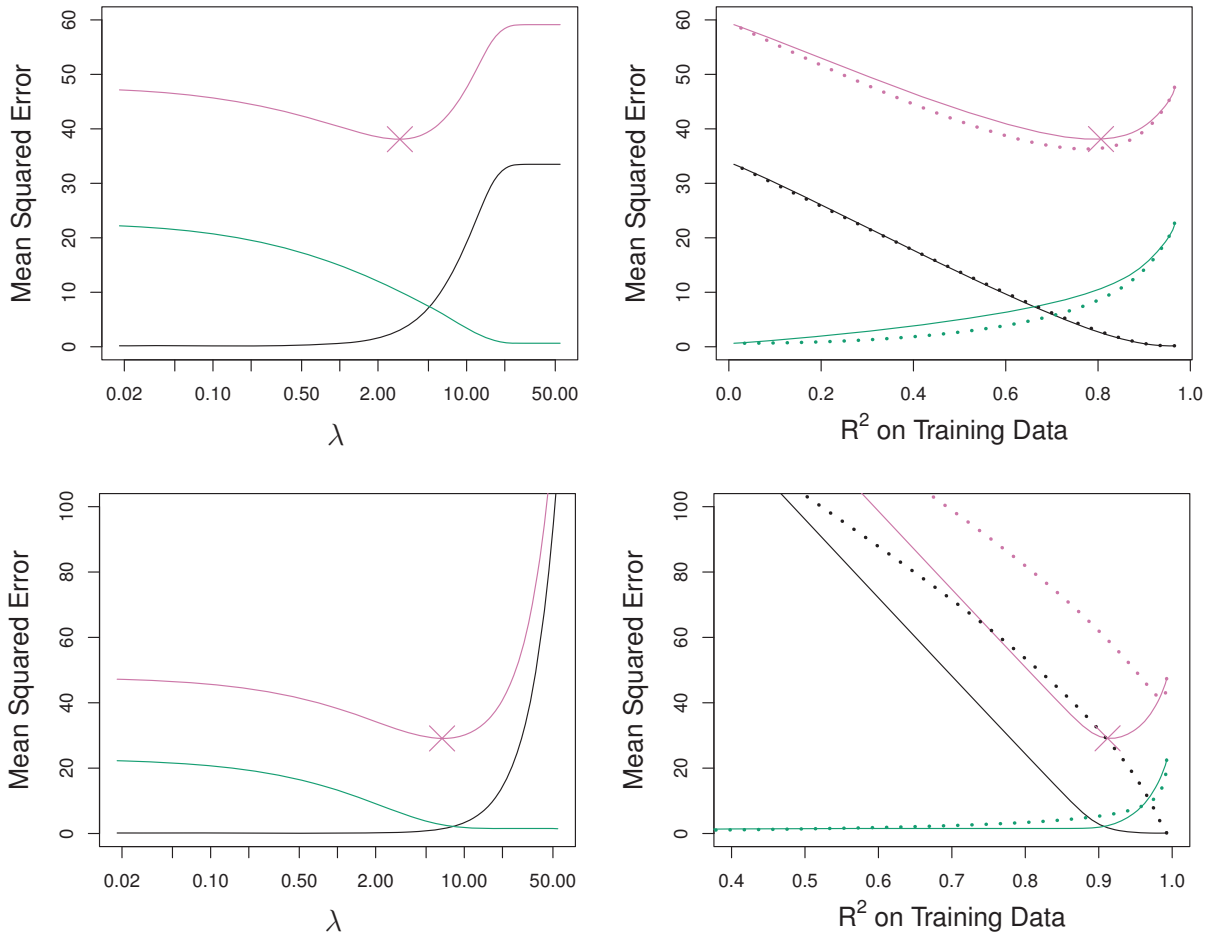


Contours of the error and constraint functions for the lasso (left) and ridge regression (right). The solid blue areas are the constraint regions, $|\beta_1|+|\beta_2| \leq s$ and $\beta_1^2 + \beta_2^2 \leq s$, while the red ellipses are the contours of the RSS.

$\hat{\beta}$: least square solution

Each of the ellipses centered around $\hat{\beta}$ represent a contour. As the ellipses expand away from the least squares coefficient estimates, the RSS increases.

The lasso and ridge regression coefficient estimates are given by the first point at which an ellipse contacts the constraint region.



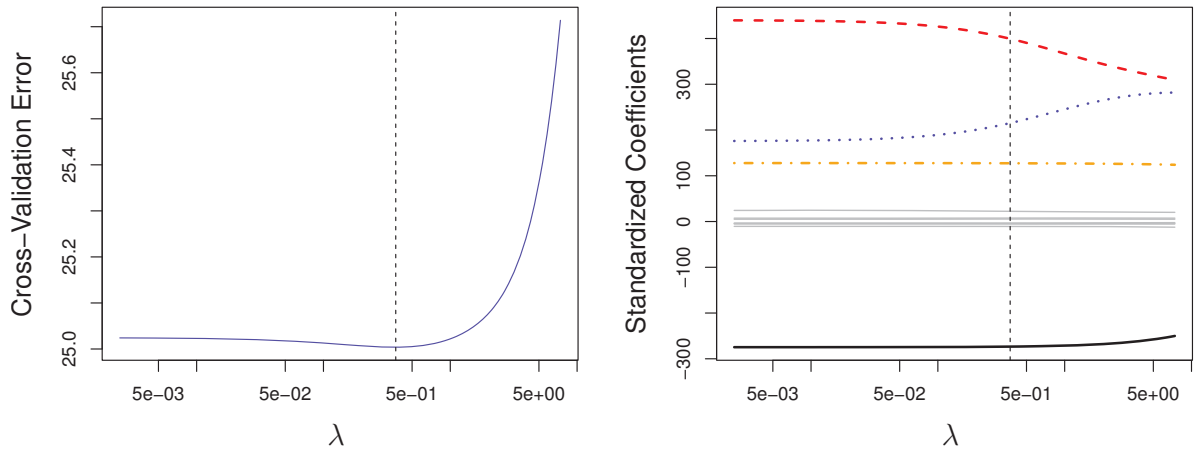
Simulated data with $n = 50$ observations, $p = 45$ predictors. Top: All predictors are related to the response so they have nonzero coefficients. Bottom: Only 2 out of 45 predictors are related to the response. Left: Plots of squared bias (black), variance (green), and test MSE (purple) for the lasso on the simulated data set. Right: Comparison of squared bias, variance, and test MSE between lasso (solid) and ridge (dotted). Both are plotted against their R^2 on the training data, as a common form of indexing. The crosses in plots indicate the lasso model for which the MSE is smallest.

Conclusions:

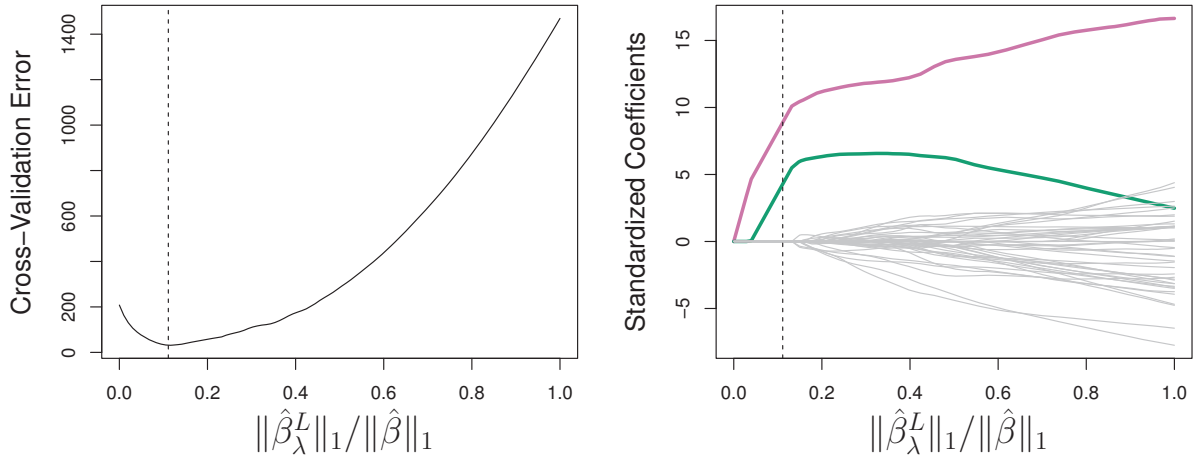
- These two examples illustrate that neither ridge regression nor the lasso will universally dominate the other.
- In general, one might expect the lasso to perform better in a setting where a relatively small number of predictors have substantial coefficients, and the remaining predictors have coefficients that are very small or that equal zero.
- Ridge regression will perform better when the response is a function of many predictors, all with coefficients of roughly equal size.
- However, the number of predictors that is related to the response is never known a priori for real data sets.
- A technique such as cross-validation can be used in order to determine which approach is better on a particular data set.

Selecting the Tuning Parameter for Ridge Regression and Lasso

- As for subset selection, for ridge regression and lasso we require a method to determine which of the models under consideration is best.
- That is, we require a method selecting a value for the tuning parameter λ or equivalently, the value of the constraint s.
- Cross-validation provides a simple way to tackle this problem. We choose a grid of λ values, and compute the cross-validation error for each value of λ.
- We then select the tuning parameter value for which the cross-validation error is smallest.
- Finally, the model is re-fit using all of the available observations and the selected value of the tuning parameter.



Left: Cross-validation errors that result from applying ridge regression to the Credit data set with various values of λ. Right: The coefficient estimates as a function of λ. The vertical dashed lines indicate the value of λ selected by cross-validation.



Left: Ten-fold cross-validation MSE for the lasso, applied to the sparse simulated data with $n = 50$ observations, $p = 45$ predictors; only 2 out of 45 predictors are related to the response. Right: The corresponding lasso coefficient estimates are displayed. The two signal variables are shown in color, and the noise variables are in gray. The vertical dashed lines indicate the lasso fit for which the cross-validation error is smallest.

Performing Ridge Regression and the Lasso in R

We will use the glmnet package in order to perform ridge regression and the lasso. The main function in this package is glmnet(), which can be used to fit ridge regression models, lasso models, and more. We will now perform ridge regression and the lasso in order to predict Balance on the Credit data.

```
> library(ISLR2)
> names(Credit)
 [1] "Income"    "Limit"     "Rating"    "Cards"     "Age"
 [6] "Education" "Own"       "Student"   "Married"   "Region"
[11] "Balance"
> dim(Credit)
[1] 400  11
> x <- model.matrix(Balance ~ ., Credit)[, -1]
> y <- Credit$Balance
> dim(x)
[1] 400  11
```

The model.matrix() function is particularly useful for creating x; not only does it produce a matrix corresponding to the 10 predictors but it also automatically transforms any qualitative variables into dummy variables. The latter property is important because glmnet() can only take numerical, quantitative inputs.

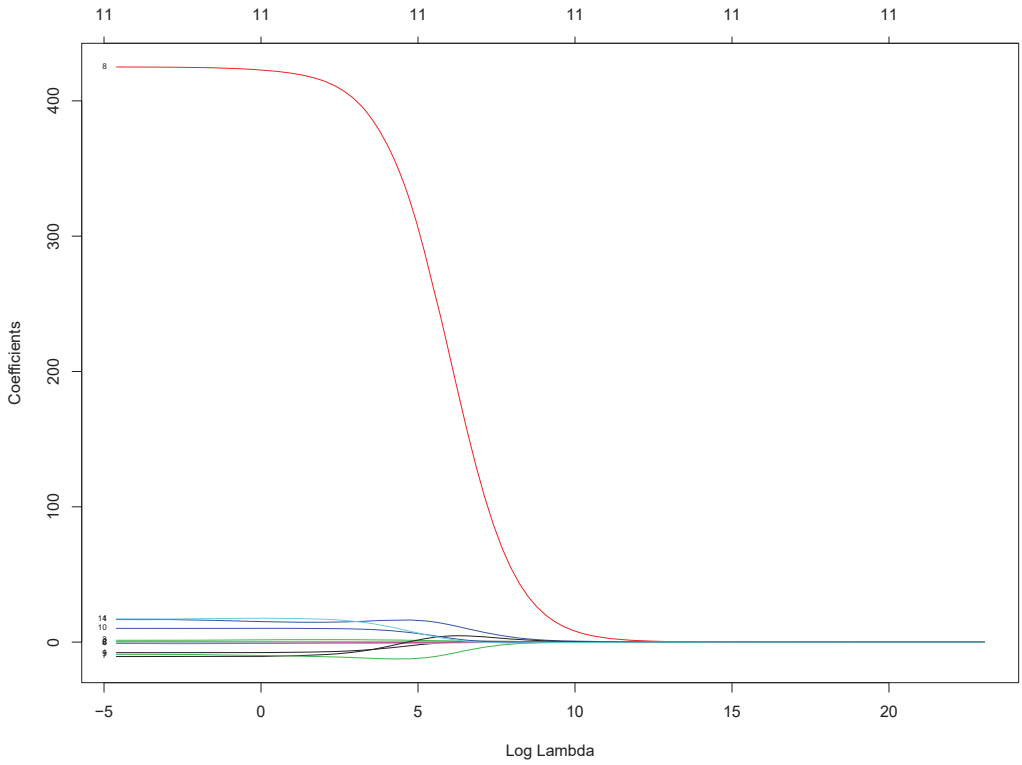
The glmnet() function has an alpha argument that determines what type of model is fit. If alpha=0 then a ridge regression model is fit, and if alpha=1 then a lasso model is fit. We first fit a ridge regression model.

```
> library(glmnet)
> grid <- 10^seq(10, -2, length = 100)
> ridge_mod <- glmnet(x, y, alpha = 0, lambda = grid)
```

By default the glmnet() function performs ridge regression for an automatically selected range of λ values. However, here we have chosen to implement

the function over a grid of values ranging from $\lambda = 10^{10}$ to $\lambda = 10^{-2}$, essentially covering the full range of scenarios from the null model containing only the intercept, to the least squares fit.

```
> dim(coef(ridge_mod))
[1]  12 100
> plot(ridge_mod, xvar="lambda", label=TRUE)
```



Associated with each value of λ is a vector of ridge regression coefficients, stored in a matrix that can be accessed by coef(). In this case, it is a 12×100 matrix, with 12 rows (one for each predictor, plus an intercept) and 100 columns (one for each value of λ).

We expect the coefficient estimates to be much smaller, in terms of l_2 norm, when a large value of λ is used, as compared to when a small value of λ is used.

```
> ridge_mod$lambda[50]
[1] 11497.57
> coef(ridge_mod)[, 50]
  (Intercept)        Income         Limit        Rating         Cards
443.882328225   0.207297549   0.006255511   0.093529429   1.094782827
          Age     Education        OwnYes    StudentYes     MarriedYes
 -0.007423108  -0.036491175   0.727221775  15.224286108  -0.273927934
  RegionSouth    RegionWest
 -0.088984491  -0.323636287
> sqrt(sum(coef(ridge_mod)[-1, 50]^2))
[1] 15.28879
```

```
> ridge_mod$lambda[60]
[1] 705.4802
> coef(ridge_mod)[, 60]
 (Intercept)       Income        Limit       Rating        Cards
 11.21321982   0.47047438   0.04709689   0.70328171  10.00746449
         Age    Education       OwnYes   StudentYes   MarriedYes
 -0.54060993   0.01398454   4.61657007 156.68825480  -6.12238443
 RegionSouth   RegionWest
  1.43860055   0.62047594
> sqrt(sum(coef(ridge_mod)[-1, 60]^2))
[1] 157.2057
```

We can use the predict() function for a number of purposes. For instance, we can obtain the ridge regression coefficients for a new value of λ, say 50:

```
> predict(ridge_mod, s = 50, type ="coefficients")[1:12, ]
 (Intercept)       Income        Limit       Rating        Cards
-381.7323042   -4.7079668    0.1102715    1.6083591   16.0197687
```

```
         Age    Education       OwnYes   StudentYes   MarriedYes
  -0.9992699   -0.4049489   -3.8463658  372.6728938  -12.2655089
 RegionSouth   RegionWest
   8.8118403   12.1975413
```

We now split the samples into a training set and a test set in order to estimate the test error of ridge regression and the lasso.

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

Next we fit a ridge regression model on the training set, and evaluate its MSE on the test set, using $\lambda = 6$.

```
> ridge_mod <- glmnet(x[train, ], y[train], alpha = 0, lambda = grid,
   thresh = 1e-12)
> ridge_pred <- predict(ridge_mod, s = 6, newx = x[test, ])
> mean((ridge_pred - y_test)^2)
[1] 10744.38
```

The test MSE is 10744.38. Note that if we had instead simply fit a model with just an intercept, we would have predicted each test observation using the mean of the training observations. In that case, we could compute the test set MSE like this:

```
> mean((mean(y[train]) - y_test)^2)
[1] 208684
```

We could also get the same result by fitting a ridge regression model with a very large value of λ.

```
> ridge_pred <- predict(ridge_mod, s = 1e10, newx = x[test, ])
> mean((ridge_pred - y_test)^2)
[1] 208684
```

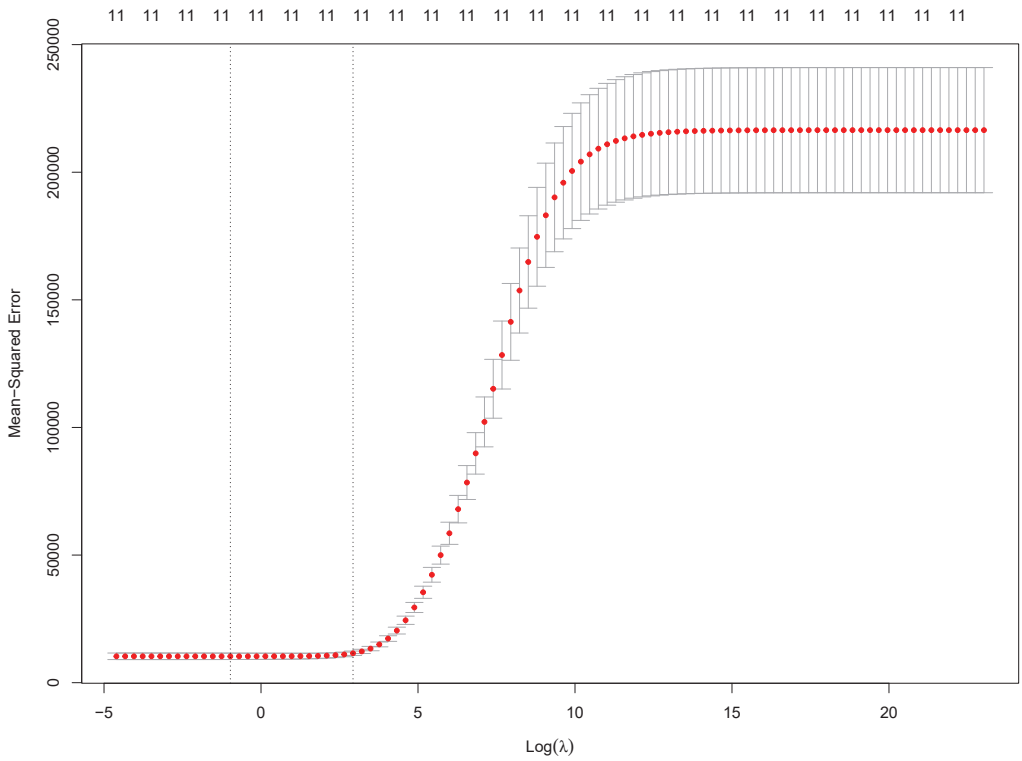
We now check whether there is any benefit to performing ridge regression with $\lambda = 6$ instead of just performing least squares regression. Recall that least squares is simply ridge regression with $\lambda = 0$.

```
> ridge_pred <- predict(ridge_mod, s = 0, newx = x[test, ], exact = T,
   x = x[train, ], y = y[train])
> mean((ridge_pred - y_test)^2)
[1] 10754.22
```

In general, instead of arbitrarily choosing $\lambda = 6$, it would be better to use cross-validation to choose the tuning parameter λ. We can do this using the built-in cross-validation function, cv.glmnet(). By default, the function performs ten-fold cross-validation, though this can be changed using the argument nfolds.

```
> set.seed(2)
> cv_out <- cv.glmnet(x[train, ], y[train], alpha = 0, lambda = grid)
> plot(cv_out)
> bestlam <- cv_out$lambda.min
> bestlam
[1] 0.3764936
```

The value of λ that results in the smallest cross-validation error




```
> ridge_pred <- predict(ridge_mod, s = bestlam, newx = x[test, ])
> mean((ridge_pred - y_test)^2)
[1] 10699.25
```

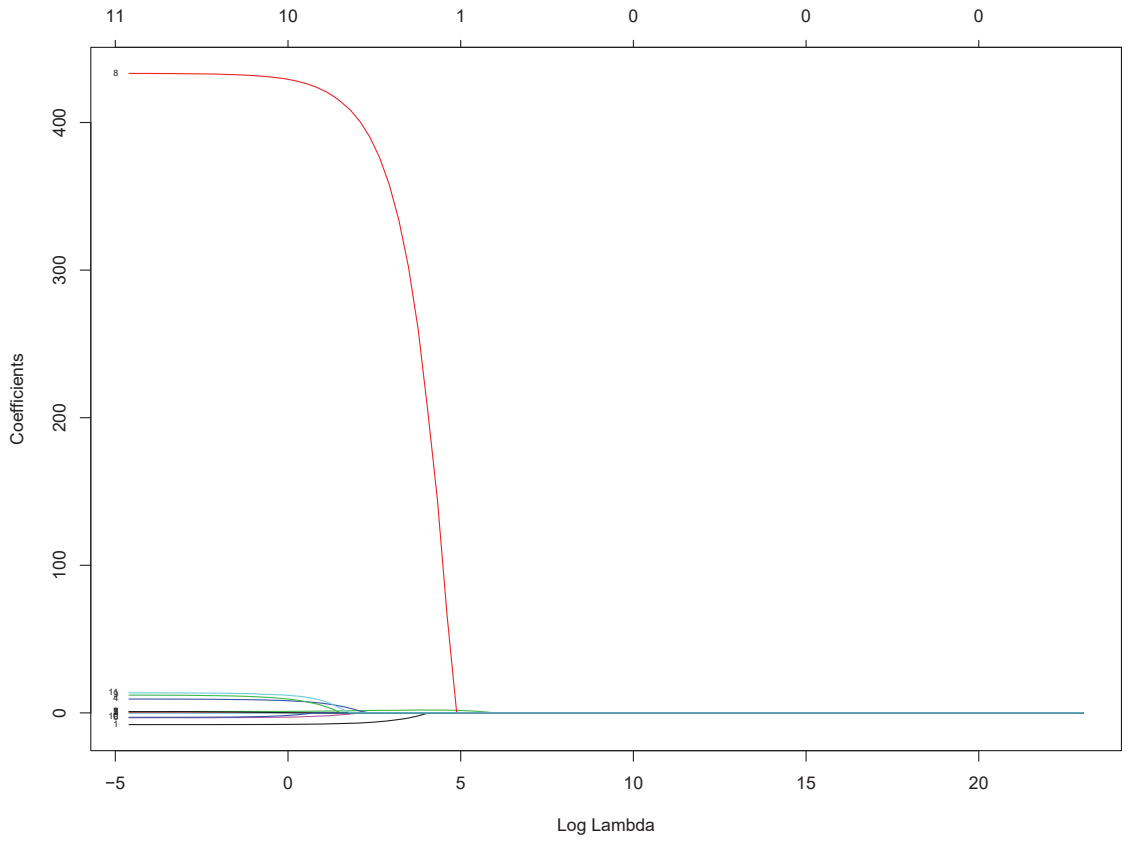
Finally, we refit our ridge regression model on the full data set, using the value of λ chosen by cross-validation, and examine the coefficient estimates.

```
> out <- glmnet(x, y, alpha = 0)
> predict(out, type = "coefficients", s = bestlam)[1:12, ]
 (Intercept)      Income       Limit      Rating       Cards
-400.8600303   -5.1751846    0.1144898    1.6607396   15.8061283
         Age   Education       OwnYes   StudentYes   MarriedYes
  -0.9565677   -0.4740528   -4.8551956  381.6896031  -12.0976032
 RegionSouth   RegionWest
   9.1146438   13.1298156
```

As expected, none of the coefficients are zero. Ridge regression does not perform variable selection.

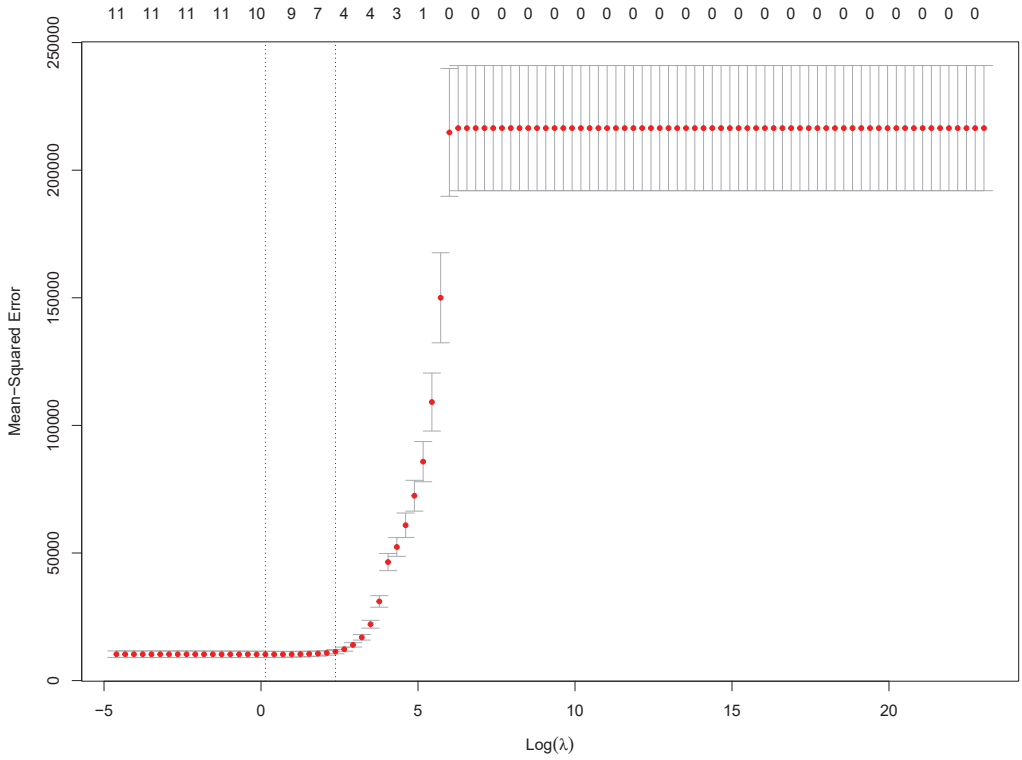
In order to fit a lasso model, we once again use the glmnet() function; however, this time we use the argument alpha=1. Other than that change, we proceed just as we did in fitting a ridge model.

```
> lasso_mod <- glmnet(x[train, ], y[train], alpha = 1, lambda = grid)
> plot(lasso_mod, xvar="lambda", label=TRUE)
```



We now perform cross-validation and compute the associated test error.

```
> set.seed(2)
> cv_out_lasso <- cv.glmnet(x[train, ], y[train], alpha = 1, lambda = grid)
> plot(cv_out_lasso)
> bestlam_lasso <- cv_out_lasso$lambda.min
> bestlam_lasso
[1] 1.149757
> lasso_pred <- predict(lasso_mod, s = bestlam_lasso, newx = x[test, ])
> mean((lasso_pred - y_test)^2)
[1] 10667.61
```




```
> out_lasso <- glmnet(x, y, alpha = 1, lambda = grid)
> lasso_coef <- predict(out_lasso, type = "coefficients",
   s = bestlam_lasso)[1:12, ]
> lasso_coef
```

```
 (Intercept)       Income        Limit       Rating        Cards
-484.1178625   -7.6496972    0.1712963    1.3928286   15.7837738
         Age    Education       OwnYes   StudentYes   MarriedYes
  -0.5766138   -0.6701087   -7.9688576  420.4854558   -6.4780158
 RegionSouth   RegionWest
   4.5656338   10.5291605
```

We see that none of the coefficient estimates are exactly zero. However, if we choose more restricted models, some of the coefficient estimates will be exactly zero.

```
> lasso_coef <- predict(out_lasso, type = "coefficients", s = 10)[1:12, ]
> lasso_coef
 (Intercept)       Income        Limit       Rating        Cards
-469.8791642   -6.4884352    0.1278652    1.7700209    8.1460234
         Age    Education       OwnYes   StudentYes   MarriedYes
  -0.2372460    0.0000000    0.0000000  386.3937236    0.0000000
 RegionSouth   RegionWest
   0.0000000    0.0000000
```