Linear Model Selection and Regularization

Recall the standard linear model

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

As we have seen before, one typically fits this model using least squares.

Here, we will discuss some ways in which the simple linear model can be improved, by replacing plain least squares fitting with some alternative fitting procedures.

Why might we want to use another fitting procedure instead of least squares?

- *Prediction Accuracy*: especially when $p > n$, to control the variance.
- *Model Interpretability*: By removing irrelevant features — that is, by setting the corresponding coefficient estimates to zero — we can obtain a model that is more easily interpreted.

Here, we will discuss three important classes of methods.

- *Subset Selection*. This approach involves identifying a subset of the p predictors that we believe to be related to the response. We then fit a model using least squares on the reduced set of variables.
- *Shrinkage*. This approach involves fitting a model involving all p predictors. However, the estimated coefficients are shrunken towards zero relative to the least squares estimates. This shrinkage (also known as regularization) has the effect of reducing variance. Depending on what type of shrinkage is performed, some of the coefficients may be estimated to be exactly zero. Hence, shrinkage methods can also perform variable selection.
- *Dimension Reduction*. This approach involves projecting the p predictors into an M-dimensional subspace, where $M < p$. This is achieved by

computing M different linear combinations, or projections, of the variables. Then these M projections are used as predictors to fit a linear regression model by least squares.

Subset Selection

Here we consider some methods for selecting subsets of predictors. These include best subset and stepwise model selection procedures.

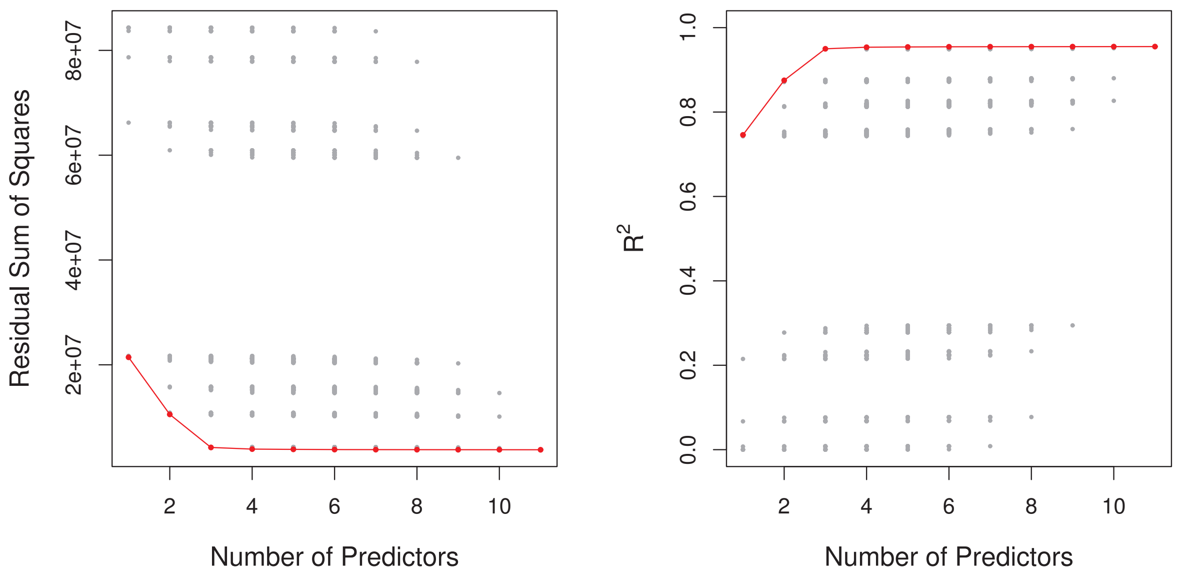
Best Subset Selection

Best Subset Selection Procedure:

1. Let $\mathcal{M}_0$ denote the null model, which contains no predictors. This model simply predicts the sample mean for each observation.
2. For $k = 1, 2, ..., p$:
   (a) Fit all $\binom{p}{k}$ models that contain exactly k predictors.
   (b) Pick the best among these $\binom{p}{k}$ models, and call it $\mathcal{M}_k$. Here best is defined as having the smallest RSS, or equivalently largest R^2.
3. Select a single best model from among $\mathcal{M}_0, \ldots, \mathcal{M}_p$ using cross-validated prediction error, C_p (AIC), BIC, or adjusted R^2.

In total, there are 2^p possible models:

$$\binom{p}{0} + \binom{p}{1} + \binom{p}{2} + \cdots + \binom{p}{p} = 2^p$$



For each possible model containing a subset of the ten predictors in the Credit data set, the RSS and R^2 are displayed. The red frontier tracks the best model for a given number of predictors, according to RSS and R^2. Though the data set contains only ten predictors, the x-axis ranges from 1 to 11, since one of the variables is categorical and takes on three values, leading to the creation of two dummy variables.

Notes:

- Although we have presented best subset selection here for least squares regression, the same ideas apply to other types of models, such as logistic regression.
- In the case of logistic regression, instead of ordering models by RSS, we instead use the deviance, a measure that plays the role of RSS for a broader class of models. The deviance is negative two times the maximized log-likelihood; the smaller the deviance, the better the fit.

Stepwise Selection

- For computational reasons, best subset selection cannot be applied with very large p.

In general, there are 2^p models that involve subsets of p predictors.

If $p=10$, $2^p \approx 1000$ possible models

If $p=20$, $2^p \approx 1,000,000$ possible models

- Best subset selection may also suffer from statistical problems when p is large. The larger the search space, the higher the chance of finding models that look good on the training data, even though they might not have any predictive power on future data.
- Thus an enormous search space can lead to overfitting and high variance of the coefficient estimates.
- For both of these reasons, stepwise methods, which explore a far more restricted set of models, are attractive alternatives to best subset selection.

Forward Stepwise Selection

Forward stepwise selection begins with a model containing no predictors, and then adds predictors to the model, one-at-a-time, until all of the predictors are in the model. In particular, at each step the variable that gives the greatest additional improvement to the fit is added to the model.

Forward Stepwise Selection Procedure:

1. Let $\mathcal{M}_0$ denote the null model, which contains no predictors.
2. For $k = 0, ..., p-1$:
   (a) Consider all $p-k$ models that augment the predictors in $\mathcal{M}_k$ with one additional predictor.
   (b) Choose the best among these $p-k$ models, and call it $\mathcal{M}_{k+1}$. Here best is defined as having smallest RSS or highest R^2.
3. Select a single best model from among $\mathcal{M}_0, \dots, \mathcal{M}_p$ using cross-validated prediction error, C_p (AIC), BIC, or adjusted R^2.

Unlike best subset selection, which involved fitting 2^p models, forward stepwise selection involves fitting one null model, along with $p-k$ models in the kth iteration, for $k = 0, ..., p-1$.

This amounts to a total of

$$1+\sum_{k=0}^{p-1}(p-k) = 1+\frac{p(p+1)}{2}$$

models.

when $p=20$, $2^p \approx 1{,}000{,}000$ but $1+\frac{p(p+1)}{2} = 211$

Note: Although forward stepwise tends to do well in practice, it is not guaranteed to find the best possible model out of all 2^p models containing subsets of the p predictors.

For instance, suppose that in a given data set with $p = 3$ predictors, the best possible one-variable model contains X_1, and the best possible two-variable model instead contains X_2 and X_3. Then forward stepwise selection will fail to select the best possible two-variable model, because $\mathcal{M}_1$ will contain X_1, so $\mathcal{M}_2$ must also contain X_1 together with one additional variable.

# Variables	Best subset	Forward stepwise
One	`rating`	`rating`
Two	`rating`, `income`	`rating`, `income`
Three	`rating`, `income`, `student`	`rating`, `income`, `student`
Four	`cards`, `income`, `student`, `limit`	`rating`, `income`, `student`, `limit`

The first four selected models for best subset selection and forward stepwise selection on the Credit data set. The first three models are identical but the fourth models differ.

Backward Stepwise Selection

Like forward stepwise selection, backward stepwise selection provides an efficient alternative to best subset selection. However, unlike forward stepwise selection, it begins with the full least squares model containing all p predictors, and then iteratively removes the least useful predictor, one-at-a-time.

Backward Stepwise Selection Procedure:

1. Let $\mathcal{M}_p$ denote the full model, which contains all p predictors.
2. For $k = p, p-1, ..., 1$:
   (a) Consider all k models that contain all but one of the predictors in $\mathcal{M}_k$, for a total of $k-1$ predictors.
   (b) Choose the best among these k models, and call it $\mathcal{M}_{k-1}$. Here best is defined as having smallest RSS or highest R^2.
3. Select a single best model from among $\mathcal{M}_0, \ldots, \mathcal{M}_p$ using cross-validated prediction error, C_p (AIC), BIC, or adjusted R^2.

Notes:

- Like forward stepwise selection, the backward selection approach searches through only $1 + p(p+1)/2$ models, and so can be applied in settings where p is too large to apply best subset selection.
- Also like forward stepwise selection, backward stepwise selection is not guaranteed to yield the best model containing a subset of the p predictors.
- Backward selection requires that the number of samples n is larger than the number of variables p (so that the full model can be fit). In contrast, forward stepwise can be used even when $n < p$, and so is the only viable subset method when p is very large.

Choosing the Optimal Model

The model containing all of the predictors will always have the smallest RSS and the largest R^2, since these quantities are related to the training error. Instead, we wish to choose a model with a low test error.

The training error can be a poor estimate of the test error. Therefore, RSS and R^2 are not suitable for selecting the best model among a collection of models with different numbers of predictors.

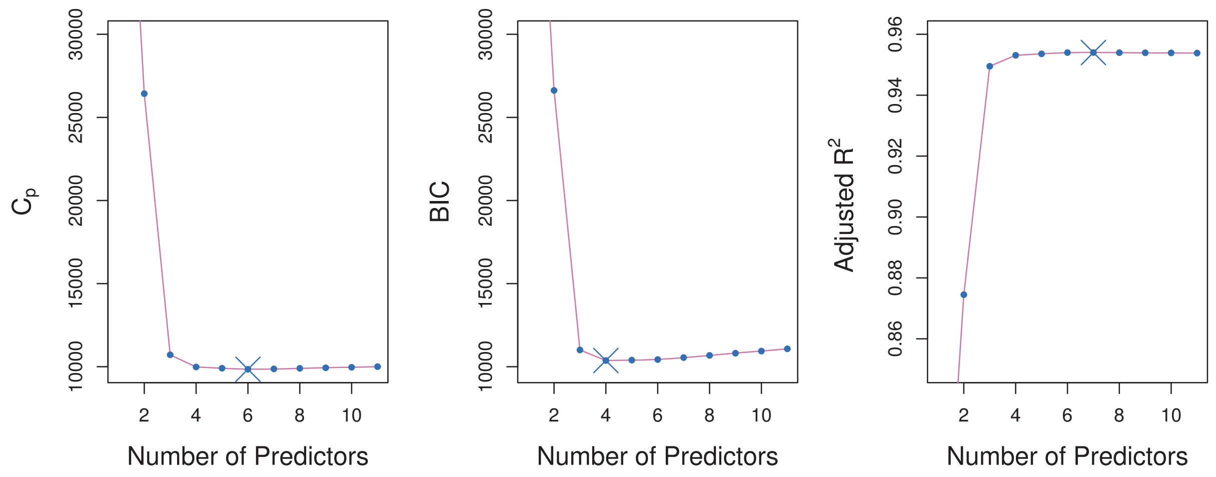
In order to select the best model with respect to test error, we need to estimate this test error. There are two common approaches:

1. We can indirectly estimate test error by making an adjustment to the training error to account for the bias due to overfitting.
2. We can directly estimate the test error, using either a validation set approach or a cross-validation approach.

C_p, AIC, BIC, and Adjusted R^2

There are a number of techniques for adjusting the training error for the model size which can be used to select among a set of models with different numbers of variables.

We consider four such approaches: C_p, Akaike information criterion (AIC), Bayesian information criterion (BIC), and adjusted R^2.



C_p, BIC, and adjusted R^2 are shown for the best models of each size for the Credit data set. C_p and BIC are estimates of test MSE. In the middle plot we see that the BIC estimate of test error shows an increase after four variables are selected. The other two plots are rather flat after four variables are included.

Mallow's C_p:

For a fitted least squares model containing d predictors, the C_p estimate of test MSE is computed using the equation

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

$$\text{MSE} = \frac{\text{RSS}}{n}$$

where $\hat{\sigma}^2$ is an estimate of the variance of the error ϵ associated with each response measurement.

- The C_p statistic adds a penalty of $2d\hat{\sigma}^2$ to the training RSS in order to adjust for the fact that the training error tends to underestimate the test error.
- Clearly, the penalty increases as the number of predictors in the model increases; this is intended to adjust for the corresponding decrease in training RSS.
- The C_p statistic tends to take on a small value for models with a low test error, so when determining which of a set of models is best, we choose the model with the lowest C_p value.

AIC:

The AIC criterion is defined for a large class of models fit by maximum likelihood:

$$\text{AIC} = -2\log L + 2d$$

where L is the maximized value of the likelihood function for the estimated model.

In the case of the linear model with Gaussian errors, maximum likelihood and least squares are the same thing, and C_p and AIC are equivalent.

BIC:

For the least squares model with d predictors, the BIC is, up to irrelevant constants, given by

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

- Like C_p, the BIC will tend to take on a small value for a model with a low test error, and so generally we select the model that has the lowest BIC value.

- Notice that BIC replaces the $2d\hat{\sigma}^2$ used by C_p with a $\log(n)d\hat{\sigma}^2$ term, where n is the number of observations.

- Since $\log n > 2$ for any $n > 7$, the BIC statistic generally places a heavier penalty on models with many variables, and hence results in the selection of smaller models than C_p.

Adjusted R^2:

For a least squares model with d variables, the adjusted R^2 statistic is calculated as

$$\text{Adjusted } R^2 = 1 - \frac{\text{RSS}/(n-d-1)}{\text{TSS}/(n-1)}$$

Recall that

$$R^2 = 1 - \frac{RSS}{TSS} \quad \text{where } TSS = \sum_{i=1}^{n} (y_i - \bar{y})^2$$

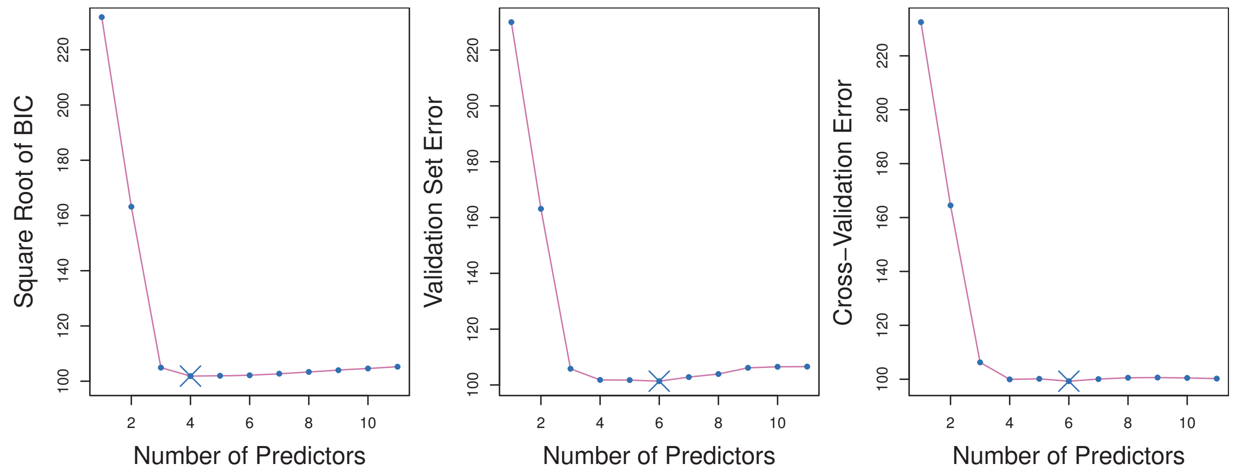
- Unlike C_p, AIC, and BIC, for which a small value indicates a model with a low test error, a large value of adjusted R^2 indicates a model with a small test error.

- Maximizing the adjusted R^2 is equivalent to minimizing $\frac{\text{RSS}}{n-d-1}$. While RSS always decreases as the number of variables in the model increases, $\frac{\text{RSS}}{n-d-1}$ may increase or decrease, due to the presence of d in the denominator.

- Unlike the R^2 statistic, the adjusted R^2 statistic pays a price for the inclusion of unnecessary variables in the model.

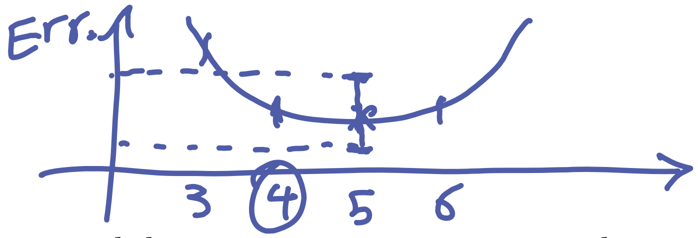
Validation and Cross-Validation

- We can compute the validation set error or the cross-validation error for each model under consideration, and then select the model for which the resulting estimated test error is smallest.
- This procedure has an advantage relative to AIC, BIC, C_p, and adjusted R^2, in that it provides a direct estimate of the test error, and makes fewer assumptions about the true underlying model.



For the Credit data set, three quantities are displayed for the best model containing d predictors, for d ranging from 1 to 11. The overall best model, based on each of these quantities, is shown as a blue cross. Left: Square root of BIC. Center: Validation set errors. Right: Cross-validation errors.

- The figure displays, as a function of d, the BIC, validation set errors, and cross-validation errors on the Credit data, for the best d-variable model.
- The validation errors were calculated by randomly selecting three-quarters of the observations as the training set, and the remainder as the validation set.



- The cross-validation errors were computed using $k = 10$ folds. In this case, the validation and cross-validation methods both result in a six-variable model.
- However, all three approaches suggest that the four-, five-, and six-variable models are roughly equivalent in terms of their test errors.
- In this setting, we can select a model using the one-standard-error rule. We first calculate the standard error of the estimated test MSE for each model size, and then select the smallest model for which the estimated test error is within one standard error of the lowest point on the curve.

Performing Subset Selection Methods in R

Here we apply the best subset selection approach to the Credit data in R. We wish to predict an individual's balance (average credit card debt for each individual) on the basis of various predictors.

```
> library(ISLR2)
> names(Credit)
 [1] "Income"    "Limit"     "Rating"    "Cards"     "Age"
 [6] "Education" "Own"       "Student"   "Married"   "Region"
[11] "Balance"
> dim(Credit)
[1] 400  11
```

The regsubsets() function (part of the leaps library) performs best subset selection by identifying the best model that contains a given number of predictors, where best is quantified using RSS. The syntax is the same as for lm(). The summary() command outputs the best set of variables for each model size.

```
> library(leaps)
> reg_fit_Credit_full <- regsubsets(Balance ~ ., Credit)
> summary(reg_fit_Credit_full)
```

```
Subset selection object
Call: regsubsets.formula(Balance ~ ., Credit)
11 Variables  (and intercept)
            Forced in Forced out
Income          FALSE      FALSE
Limit           FALSE      FALSE
Rating          FALSE      FALSE
Cards           FALSE      FALSE
Age             FALSE      FALSE
Education       FALSE      FALSE
OwnYes          FALSE      FALSE
StudentYes      FALSE      FALSE
MarriedYes      FALSE      FALSE
RegionSouth     FALSE      FALSE
RegionWest      FALSE      FALSE
1 subsets of each size up to 8
Selection Algorithm: exhaustive
         Income Limit Rating Cards Age Education OwnYes StudentYes
1  ( 1 ) " "    " "   "*"    " "   " " " "       " "    " "
2  ( 1 ) "*"    " "   "*"    " "   " " " "       " "    " "
3  ( 1 ) "*"    " "   "*"    " "   " " " "       " "    "*"
4  ( 1 ) "*"    "*"   " "    "*"   " " " "       " "    "*"
5  ( 1 ) "*"    "*"   "*"    "*"   " " " "       " "    "*"
6  ( 1 ) "*"    "*"   "*"    "*"   "*" " "       " "    "*"
7  ( 1 ) "*"    "*"   "*"    "*"   "*" " "       "*"    "*"
8  ( 1 ) "*"    "*"   "*"    "*"   "*" " "       "*"    "*"
         MarriedYes RegionSouth RegionWest
1  ( 1 ) " "        " "         " "
2  ( 1 ) " "        " "         " "
3  ( 1 ) " "        " "         " "
4  ( 1 ) " "        " "         " "
5  ( 1 ) " "        " "         " "
6  ( 1 ) " "        " "         " "
7  ( 1 ) " "        " "         " "
8  ( 1 ) " "        " "         "*"
```

An asterisk indicates that a given variable is included in the corresponding model. For instance, this output indicates that the best two-variable model contains only Income and Rating. By default, regsubsets() only reports results up to the best eight-variable model. But the nvmax option can be used in order to return as many variables as are desired. Here we fit up to an 11-variable model.

```
> reg_fit_Credit_full <- regsubsets(Balance ~ ., Credit, nvmax = 11)
> reg_summary <- summary(reg_fit_Credit_full)
> reg_summary
Subset selection object
Call: regsubsets.formula(Balance ~ ., Credit, nvmax = 11)
11 Variables  (and intercept)
            Forced in Forced out
Income          FALSE      FALSE
Limit           FALSE      FALSE
Rating          FALSE      FALSE
Cards           FALSE      FALSE
Age             FALSE      FALSE
Education       FALSE      FALSE
OwnYes          FALSE      FALSE
StudentYes      FALSE      FALSE
MarriedYes      FALSE      FALSE
RegionSouth     FALSE      FALSE
RegionWest      FALSE      FALSE
1 subsets of each size up to 11
Selection Algorithm: exhaustive
          Income Limit Rating Cards Age Education OwnYes StudentYes
1  ( 1 )  " "    " "   "*"    " "   " " " "       " "    " "
2  ( 1 )  "*"    " "   "*"    " "   " " " "       " "    " "
3  ( 1 )  "*"    " "   "*"    " "   " " " "       " "    "*"
4  ( 1 )  "*"    "*"   " "    "*"   " " " "       " "    "*"
5  ( 1 )  "*"    "*"   "*"    "*"   " " " "       " "    "*"
6  ( 1 )  "*"    "*"   "*"    "*"   "*" " "       " "    "*"
7  ( 1 )  "*"    "*"   "*"    "*"   "*" " "       "*"    "*"
8  ( 1 )  "*"    "*"   "*"    "*"   "*" " "       "*"    "*"
9  ( 1 )  "*"    "*"   "*"    "*"   "*" " "       "*"    "*"
10  ( 1 ) "*"    "*"   "*"    "*"   "*" " "       "*"    "*"
11  ( 1 ) "*"    "*"   "*"    "*"   "*" "*"       "*"    "*"
          MarriedYes RegionSouth RegionWest
1  ( 1 )  " "        " "         " "
2  ( 1 )  " "        " "         " "
3  ( 1 )  " "        " "         " "
4  ( 1 )  " "        " "         " "
5  ( 1 )  " "        " "         " "
```

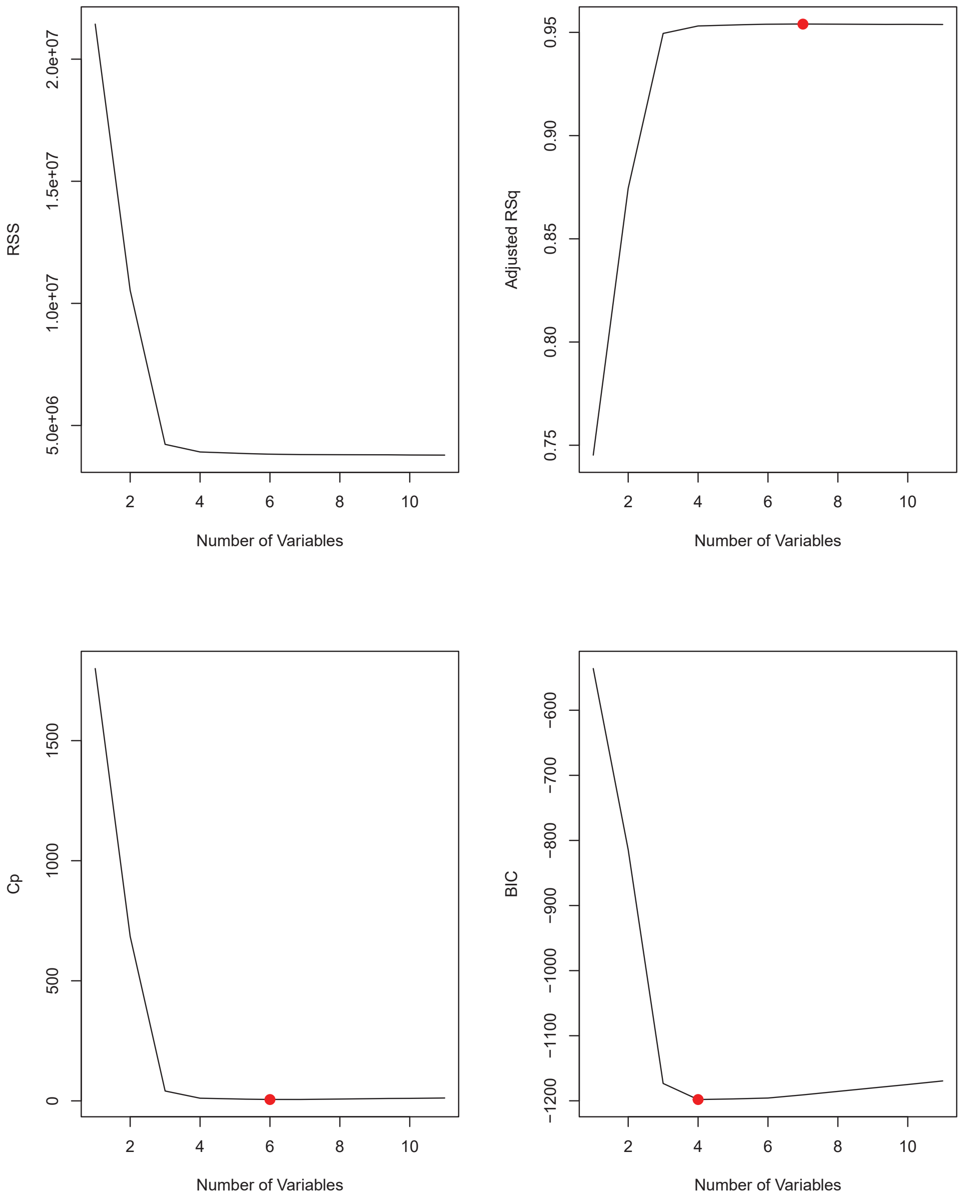
```
6  ( 1 )  " "         " "          " "
7  ( 1 )  " "         " "          " "
8  ( 1 )  " "         " "          "*"
9  ( 1 )  "*"         " "          "*"
10  ( 1 ) "*"         "*"          "*"
11  ( 1 ) "*"         "*"          "*"
```

The summary() function also returns R^2, RSS, adjusted R^2, C_p, and BIC. We can examine these to try to select the best overall model.

```
> names(reg_summary)
[1] "which"  "rsq"    "rss"    "adjr2"  "cp"     "bic"    "outmat"
[8] "obj"
> reg_summary$rsq
 [1] 0.7458484 0.8751179 0.9498788 0.9535800 0.9541606 0.9546879
 [7] 0.9548167 0.9548880 0.9549636 0.9550468 0.9551016
```

Plotting RSS, adjusted R^2, C_p, and BIC for all of the models at once will help us decide which model to select. Note the type = ”l” option tells R to connect the plotted points with lines.

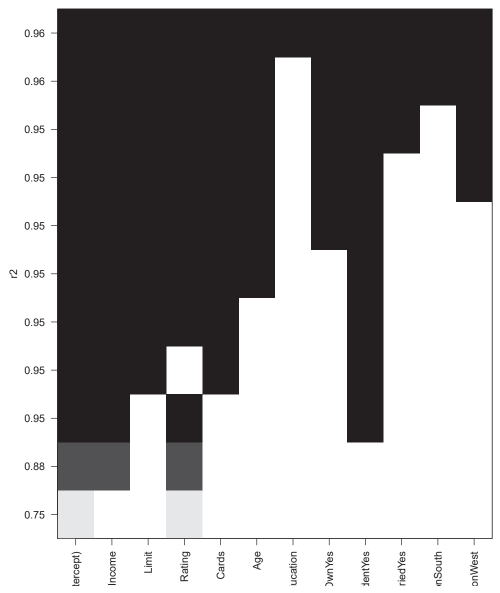
```
> par(mfrow = c(2, 2))
> plot(reg_summary$rss, xlab = "Number of Variables", ylab = "RSS",
   type = "l")
> plot(reg_summary$adjr2, xlab = "Number of Variables", ylab = "Adjusted RSq",
   type = "l")
> which.max(reg_summary$adjr2)
[1] 7
> points(which.max(reg_summary$adjr2),
   reg_summary$adjr2[which.max(reg_summary$adjr2)],
   col = "red", cex = 2, pch = 20)
> plot(reg_summary$cp, xlab = "Number of Variables", ylab = "Cp", type = "l")
> which.min(reg_summary$cp)
[1] 6
> points(which.min(reg_summary$cp), reg_summary$cp[which.min(reg_summary$cp)],
   col = "red", cex = 2, pch = 20)
> plot(reg_summary$bic, xlab = "Number of Variables", ylab = "BIC",
   type = "l")
> which.min(reg_summary$bic)
[1] 4
> points(which.min(reg_summary$bic),
   reg_summary$bic[which.min(reg_summary$bic)], col = "red", cex = 2, pch = 20)
```

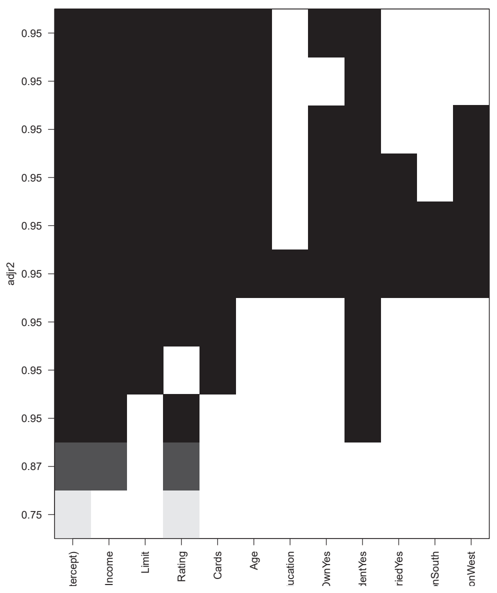


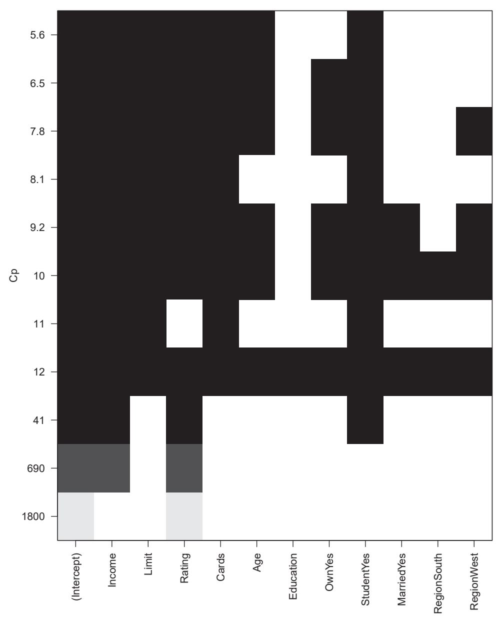
RSS, adjusted R^2, C_p, and BIC are shown for the best models of each size for the Credit data set.

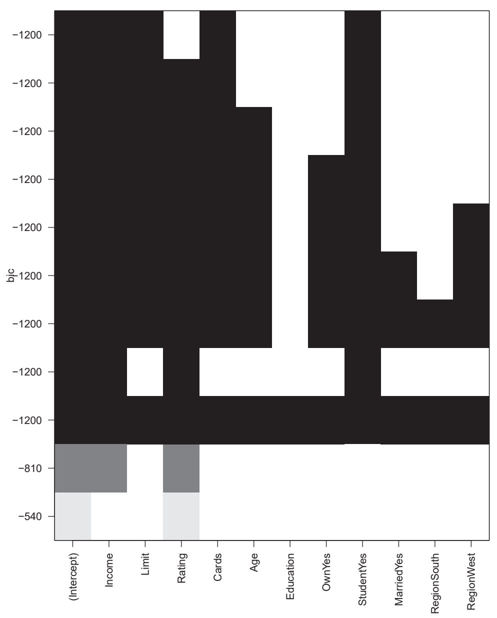
The regsubsets() function has a built-in plot() command which can be used to display the selected variables for the best model with a given number of predictors, ranked according to the BIC, C_p, adjusted R^2, or AIC.

```
> plot(reg_fit_Credit_full, scale = "r2")
> plot(reg_fit_Credit_full, scale = "adjr2")
> plot(reg_fit_Credit_full, scale = "Cp")
> plot(reg_fit_Credit_full, scale = "bic")
```









The top row of each plot contains a black square for each variable selected according to the optimal model associated with that statistic. For instance, we see that several models share a BIC close to -1200. However, the model with the lowest BIC is the four-variable model that contains only Income, Limit, Cards, and StudentYes. We can use the coef() function to see the coefficient estimates associated with this model.

```
> coef(reg_fit_Credit_full, 4)
 (Intercept)       Income        Limit        Cards   StudentYes
-499.7272117   -7.8392288    0.2666445   23.1753794  429.6064203
```

We can also use the regsubsets() function to perform forward stepwise or backward stepwise selection, using the argument method = ”forward” or method = ”backward”.

```
> reg_fit_Credit_fwd <- regsubsets(Balance ~ ., Credit, nvmax = 11,
   method = "forward")
> summary(reg_fit_Credit_fwd)
Subset selection object
Call: regsubsets.formula(Balance ~ ., Credit, nvmax = 11, method = "forward")
11 Variables  (and intercept)
            Forced in Forced out
Income          FALSE      FALSE
Limit           FALSE      FALSE
Rating          FALSE      FALSE
Cards           FALSE      FALSE
Age             FALSE      FALSE
Education       FALSE      FALSE
OwnYes          FALSE      FALSE
StudentYes      FALSE      FALSE
MarriedYes      FALSE      FALSE
RegionSouth     FALSE      FALSE
RegionWest      FALSE      FALSE
1 subsets of each size up to 11
Selection Algorithm: forward
          Income Limit Rating Cards Age Education OwnYes StudentYes
1  ( 1 )  " "    " "   "*"    " "   " " " "       " "    " "
2  ( 1 )  "*"    " "   "*"    " "   " " " "       " "    " "
3  ( 1 )  "*"    " "   "*"    " "   " " " "       " "    "*"
4  ( 1 )  "*"    "*"   "*"    " "   " " " "       " "    "*"
5  ( 1 )  "*"    "*"   "*"    "*"   " " " "       " "    "*"
6  ( 1 )  "*"    "*"   "*"    "*"   "*" " "       " "    "*"
7  ( 1 )  "*"    "*"   "*"    "*"   "*" " "       "*"    "*"
8  ( 1 )  "*"    "*"   "*"    "*"   "*" " "       "*"    "*"
9  ( 1 )  "*"    "*"   "*"    "*"   "*" " "       "*"    "*"
10  ( 1 ) "*"    "*"   "*"    "*"   "*" " "       "*"    "*"
```

```
11  ( 1 ) "*"    "*"   "*"    "*"   "*" "*"       "*"    "*"
          MarriedYes RegionSouth RegionWest
1  ( 1 )  " "        " "         " "
2  ( 1 )  " "        " "         " "
3  ( 1 )  " "        " "         " "
4  ( 1 )  " "        " "         " "
5  ( 1 )  " "        " "         " "
6  ( 1 )  " "        " "         " "
7  ( 1 )  " "        " "         " "
8  ( 1 )  " "        " "         "*"
9  ( 1 )  "*"        " "         "*"
10  ( 1 ) "*"        "*"         "*"
11  ( 1 ) "*"        "*"         "*"
> reg_fit_Credit_bwd <- regsubsets(Balance ~ ., Credit, nvmax = 11,
   method = "backward")
> summary(reg_fit_Credit_bwd)
Subset selection object
Call: regsubsets.formula(Balance ~ ., Credit, nvmax = 11, method = "backward")
11 Variables  (and intercept)
            Forced in Forced out
Income          FALSE      FALSE
Limit           FALSE      FALSE
Rating          FALSE      FALSE
Cards           FALSE      FALSE
Age             FALSE      FALSE
Education       FALSE      FALSE
OwnYes          FALSE      FALSE
StudentYes      FALSE      FALSE
MarriedYes      FALSE      FALSE
RegionSouth     FALSE      FALSE
RegionWest      FALSE      FALSE
1 subsets of each size up to 11
Selection Algorithm: backward
          Income Limit Rating Cards Age Education OwnYes StudentYes
1  ( 1 )  " "    "*"   " "    " "   " " " "       " "    " "
2  ( 1 )  "*"    "*"   " "    " "   " " " "       " "    " "
3  ( 1 )  "*"    "*"   " "    " "   " " " "       " "    "*"
4  ( 1 )  "*"    "*"   " "    "*"   " " " "       " "    "*"
5  ( 1 )  "*"    "*"   "*"    "*"   " " " "       " "    "*"
6  ( 1 )  "*"    "*"   "*"    "*"   "*" " "       " "    "*"
7  ( 1 )  "*"    "*"   "*"    "*"   "*" " "       "*"    "*"
8  ( 1 )  "*"    "*"   "*"    "*"   "*" " "       "*"    "*"
```

```
9  ( 1 )  "*"    "*"   "*"    "*"   "*" " "        "*"    "*"
10  ( 1 ) "*"    "*"   "*"    "*"   "*" " "        "*"    "*"
11  ( 1 ) "*"    "*"   "*"    "*"   "*" "*"        "*"    "*"
          MarriedYes RegionSouth RegionWest
1  ( 1 )  " "        " "         " "
2  ( 1 )  " "        " "         " "
3  ( 1 )  " "        " "         " "
4  ( 1 )  " "        " "         " "
5  ( 1 )  " "        " "         " "
6  ( 1 )  " "        " "         " "
7  ( 1 )  " "        " "         " "
8  ( 1 )  " "        " "         "*"
9  ( 1 )  "*"        " "         "*"
10  ( 1 ) "*"        "*"         "*"
11  ( 1 ) "*"        "*"         "*"
```

```
> coef(reg_fit_Credit_fwd, which.min(summary(reg_fit_Credit_fwd)$bic))
 (Intercept)       Income        Limit       Rating        Cards
-526.1555233   -7.8749239    0.1944093    1.0879014   17.8517307
  StudentYes
 426.8501456
> coef(reg_fit_Credit_bwd, which.min(summary(reg_fit_Credit_bwd)$bic))
 (Intercept)       Income        Limit        Cards   StudentYes
-499.7272117   -7.8392288    0.2666445   23.1753794  429.6064203
```

We will now show how to choose among a set of models of different sizes using the validation set and cross-validation approaches in R.

In order to use the validation set approach, we begin by splitting the observations into a training set and a test set. Then we apply regsubsets() to the training set in order to perform best subset selection.

```
> set.seed(1)
> train <- sample(c(TRUE, FALSE), nrow(Credit), replace = TRUE)
> test <- (!train)
> reg_fit_Credit_full_val <- regsubsets(Balance ~ ., Credit[train, ],
   nvmax = 11)
```

We now compute the validation set error for the best model of each model size. We first make a model matrix from the test data.

```
> test_mat <- model.matrix(Balance ~ ., data = Credit[test, ])
> dim(test_mat)
[1] 183  12
```

The model.matrix() function is used in many regression packages for building an "X" matrix from data. Now we run a loop, and for each size i, we extract the coefficients from reg_fit_Credit_full_val for the best model of that size, multiply them into the appropriate columns of the test model matrix to form the predictions, and compute the test MSE.

```
> val_errors <- rep(NA, 11)
> for (i in 1:11) {
+         coefi <- coef(reg_fit_Credit_full_val, id = i)
+         pred <- test_mat[, names(coefi)] %*% coefi
+         val_errors[i] <- mean((Credit$Bbalance[test] - pred)^2)
+ }
```

We find that the best model is the one that contains five variables.

```
> val_errors
 [1] 51754.40 24326.78 12905.05 11616.12 11582.29 12176.73 12041.24
 [8] 12024.54 11980.66 11930.86 11932.30
> which.min(val_errors)
[1] 5
> coef(reg_fit_Credit_full_val, which.min(val_errors))
 (Intercept)       Income        Limit        Cards          Age
-443.5105267   -7.8334235    0.2659310   22.0262551   -0.8361878
  StudentYes
 454.7860565
```

We can capture our steps above and write our own predict method.

```
> predict_regsubsets <- function(object, newdata, id, ...) {
+         form <- as.formula(object$call[[2]])
+         mat <- model.matrix(form, newdata)
+         coefi <- coef(object, id = id)
+         xvars <- names(coefi)
+         mat[, xvars] %*% coefi
+ }
```

Finally, we perform best subset selection on the full data set, and select the best five-variable model. It is important that we make use of the full data set in order to obtain more accurate coefficient estimates.

<u>Note</u>: We perform best subset selection on the full data set and select the best five-variable model, rather than simply using the variables that were obtained from the training set, because the best five-variable model on the full data set may differ from the corresponding model on the training set.

```
> reg_fit_Credit_full_val_best <- regsubsets(Balance ~ ., Credit, nvmax = 11)
> coef(reg_fit_Credit_full_val_best, 5)
 (Intercept)       Income        Limit       Rating        Cards
-526.1555233   -7.8749239    0.1944093    1.0879014   17.8517307
  StudentYes
 426.8501456
```

we see that the best five-variable model has a different set of variables than the best five-variable model on the training set.

We now try to choose among the models of different sizes using cross-validation. First, we create a vector that allocates each observation to one of $k = 10$ folds, and we create a matrix in which we will store the results.

```
> k <- 10
> n <- nrow(Credit)
> set.seed(1)
> folds <- sample(rep(1:k, length = n))
> cv_errors <- matrix(NA, k, 11, dimnames = list(NULL, paste(1:11)))
```

Now we write a for loop that performs cross-validation. In the jth fold, the elements of folds that equal j are in the test set, and the remainder are in the training set.

```
> for (j in 1:k) {
+         best_fit <- regsubsets(Balance ~ ., data = Credit[folds != j, ],
          nvmax = 11)
+         for (i in 1:11) {
+                 pred <- predict_regsubsets(best_fit, Credit[folds == j, ],
                  id = i)
+                 cv_errors[j, i] <- mean((Credit$Balance[folds == j] - pred)^2)
+         }
+ }
```

This has given us a 10×11 matrix, of which the (j, i)th element corresponds to the test MSE for the jth cross-validation fold for the best i-variable model. We use the apply() function to average over the columns of this matrix in order to obtain a vector for which the ith element is the cross-validation error for the i-variable model.

```
> mean_cv_errors <- apply(cv_errors, 2, mean)
> mean_cv_errors
       1        2        3        4        5        6        7 
54664.42 26824.23 10927.28 10149.03 10285.49 10008.45 10118.66 
       8        9       10       11 
10177.63 10169.47 10125.08 10065.36 
```

We see that cross-validation selects a 6-variable model. We now perform best subset selection on the full data set in order to obtain the 6-variable model.

```
> reg_fit_Credit_full_cv_best <- regsubsets(Balance ~ ., Credit, nvmax = 11)
> coef(reg_fit_Credit_full_cv_best, 6)
 (Intercept)        Income         Limit        Rating         Cards
-493.7341870    -7.7950824     0.1936914     1.0911874    18.2118976
         Age    StudentYes
  -0.6240560   425.6099369
```

Shrinkage Methods

- The subset selection methods use least squares to fit a linear model that contains a subset of the predictors.
- As an alternative, we can fit a model containing all p predictors using a technique that constrains or regularizes the coefficient estimates, or equivalently, that shrinks the coefficient estimates towards zero.
- It may not be immediately obvious why such a constraint should improve the fit, but it turns out that shrinking the coefficient estimates can significantly reduce their variance.
- The two best-known techniques for shrinking the regression coefficients towards zero are ridge regression and the lasso.

Ridge Regression:

- Recall that the least squares fitting procedure estimates $\beta_0, \beta_1, ..., \beta_p$ using the values that minimize

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

- In contrast, the ridge regression coefficient estimates $\hat{\beta}^R$ are the values that minimize

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

where $\lambda \geq 0$ is a tuning parameter, to be determined separately.

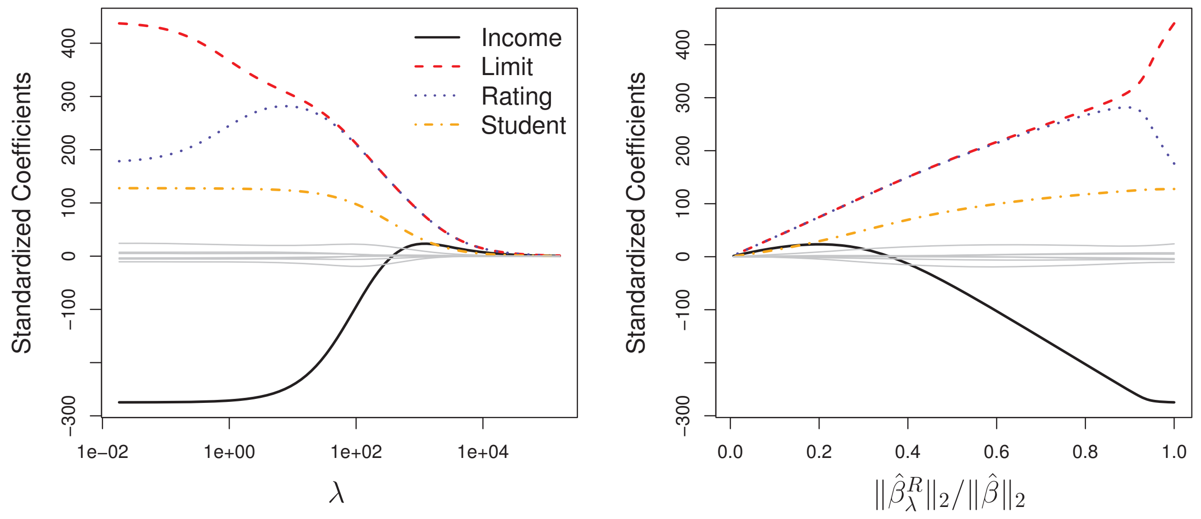
- As with least squares, ridge regression seeks coefficient estimates that fit the data well, by making the RSS small.

- However, the second term, $\lambda\sum_{j=1}^{p}\beta_j^2$, called a shrinkage penalty, is small when $\beta_1, ..., \beta_p$ are close to zero, and so it has the effect of shrinking the estimates of β_j towards zero.

- The tuning parameter λ serves to control the relative impact of these two terms on the regression coefficient estimates.

when $\lambda = 0$, the penalty term has no effect, and ridge regression will produce the least squares estimates. However, as $\lambda \to \infty$, the impact of the shrinkage penalty grows, and the ridge regression coefficient estimates will approach zero.

- Unlike least squares, which generates only one set of coefficient estimates, ridge regression will produce a different set of coefficient estimates, $\hat{\beta}_\lambda^R$, for each value of λ. Selecting a good value for λ is critical; cross-validation is used for this.



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.

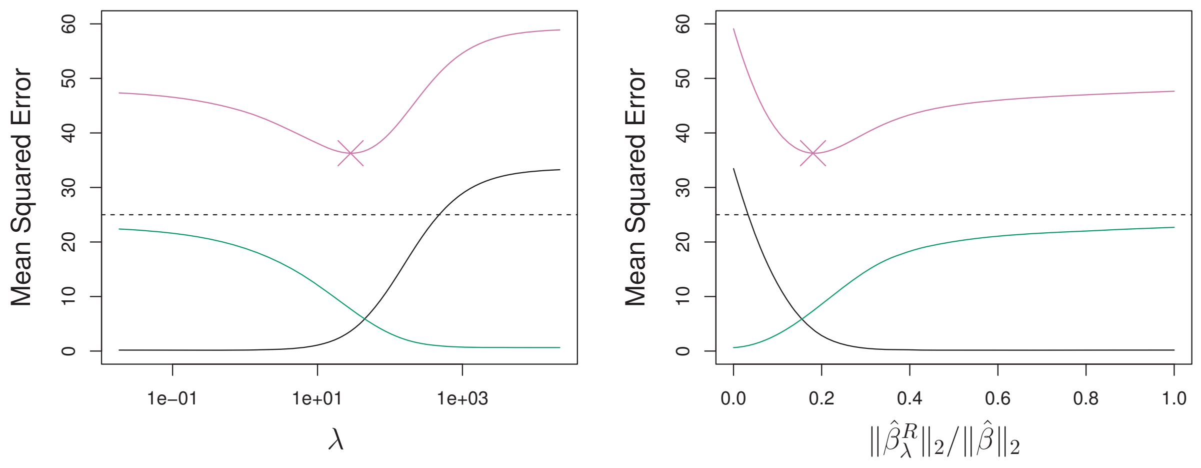
- Therefore, it is best to apply ridge regression after standardizing the predictors, using the formula

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

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 $\|\hat{\beta}^R_\lambda\|_2/\|\hat{\beta}\|_2$. 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 $\|\beta\|_1 = \sum|\beta_j|$.

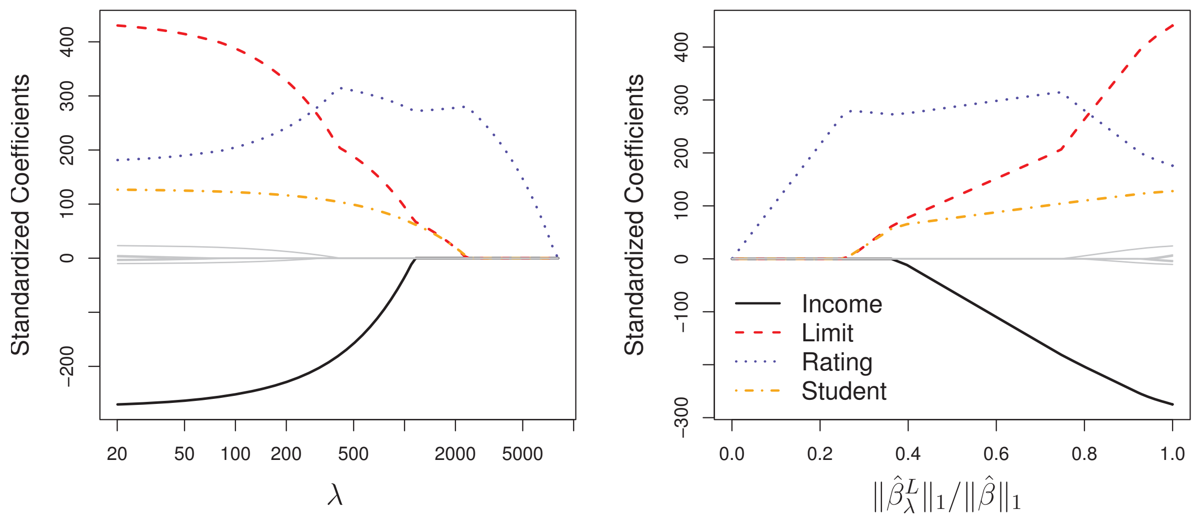
- As with ridge regression, the lasso shrinks the coefficient estimates towards zero.

- 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 $\|\hat{\beta}^L_\lambda\|_1/\|\hat{\beta}\|_1$.

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?

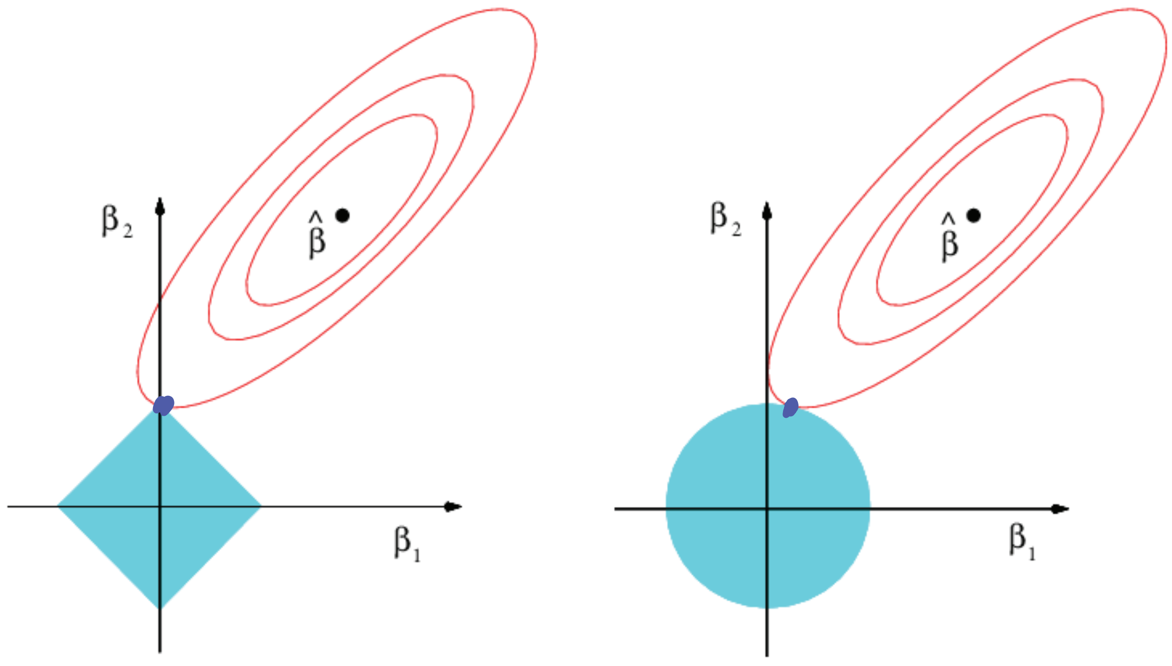
One can show that the lasso and ridge regression coefficient estimates solve the problems

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

and

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

respectively.



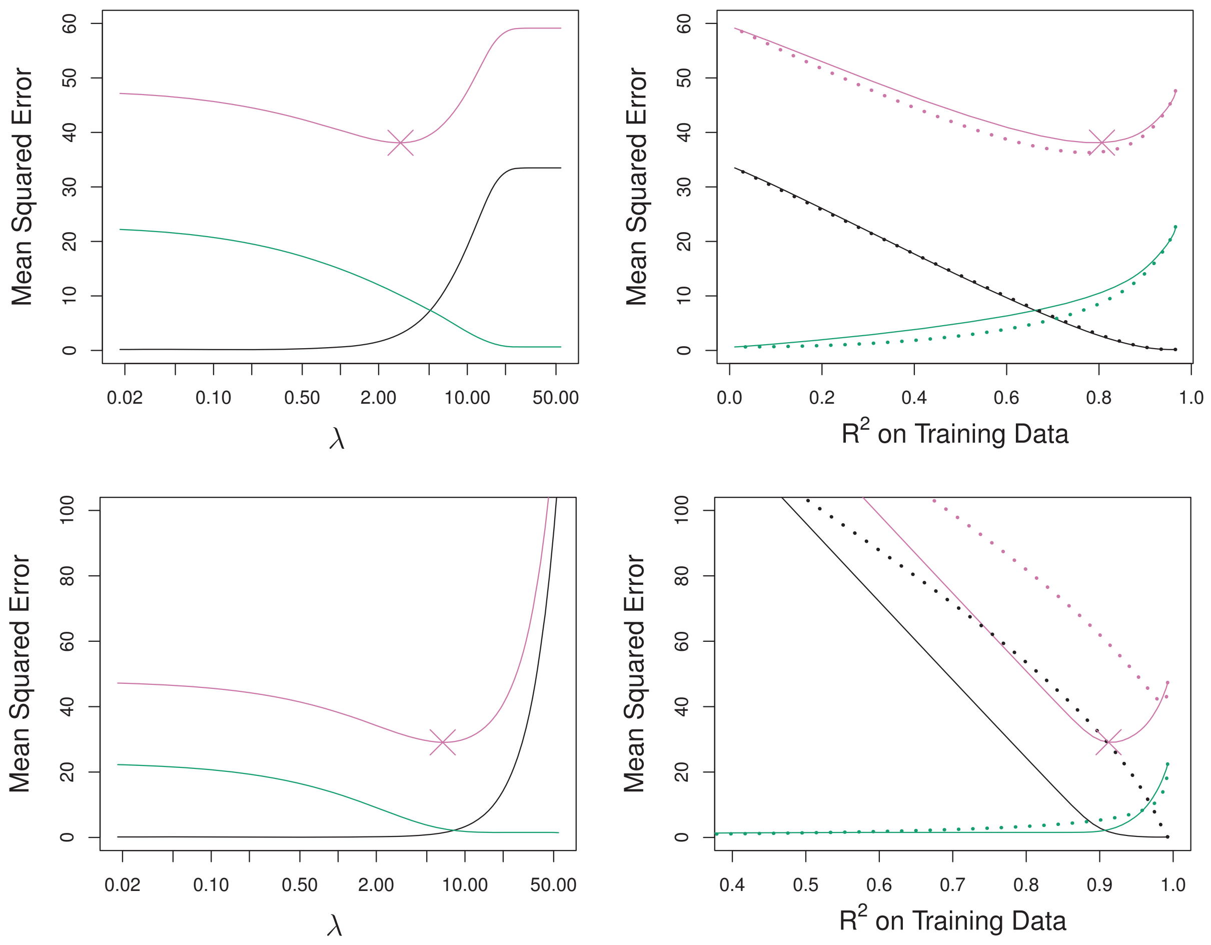
Contours of the error and constraint functions for the lasso (left) and ridge regression (right). The solid blue areas are the constraint regions, $|\beta_1|+|\beta_2| \leq s$ and $\beta_1^2 + \beta_2^2 \leq s$, while the red ellipses are the contours of the RSS.

$\hat{\beta}$: 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.

Comparing the Lasso and Ridge Regression



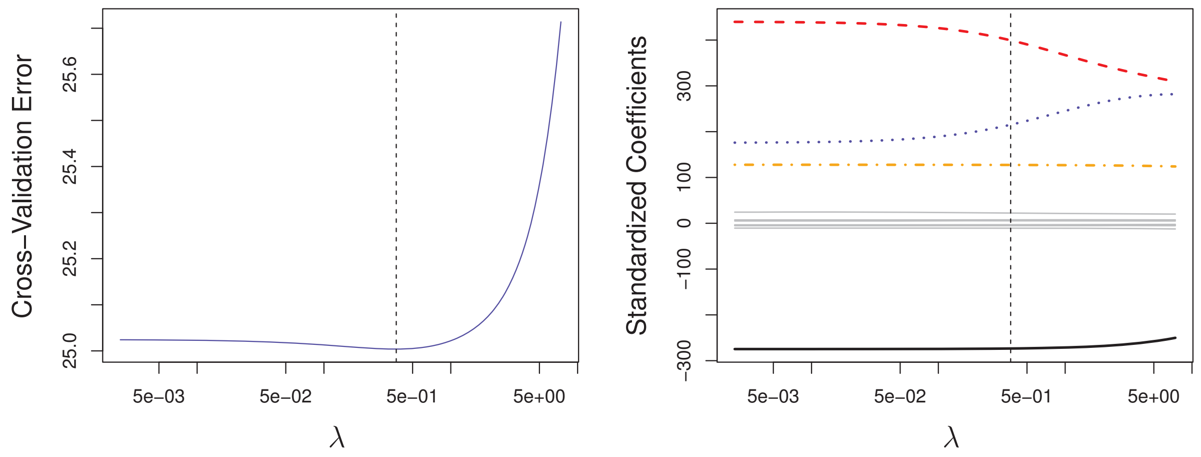
Simulated data with $n = 50$ observations, $p = 45$ predictors. Top: All predictors are related to the response so they have nonzero coefficients. Bottom: Only 2 out of 45 predictors are related to the response. Left: Plots of squared bias (black), variance (green), and test MSE (purple) for the lasso 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:

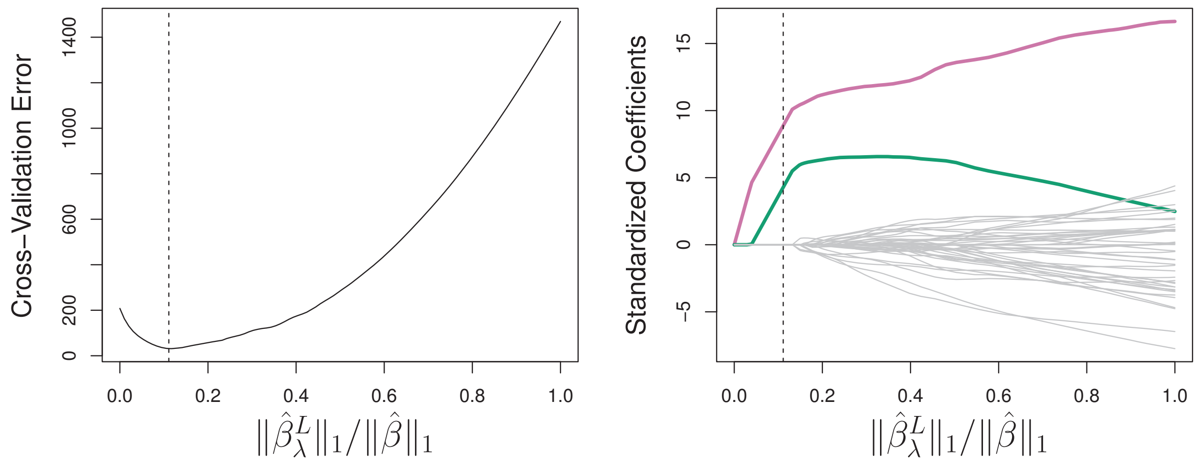
- These two examples illustrate that neither ridge regression nor the lasso will universally dominate the other.
- In general, one might expect the lasso to perform better in a setting where a relatively small number of predictors have substantial coefficients, and the remaining predictors have coefficients 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
```

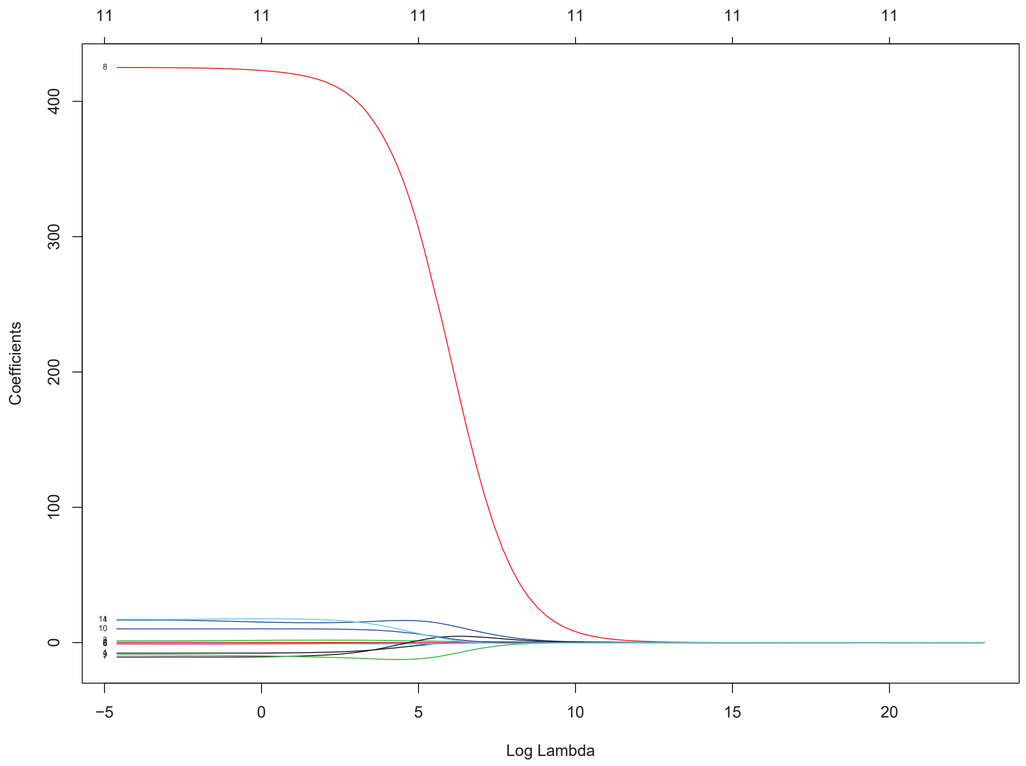
The model.matrix() function is particularly useful for creating x; not only does it produce a matrix corresponding to the 10 predictors but it also automatically transforms any qualitative variables into dummy variables. The latter property is important because glmnet() can only take numerical, quantitative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

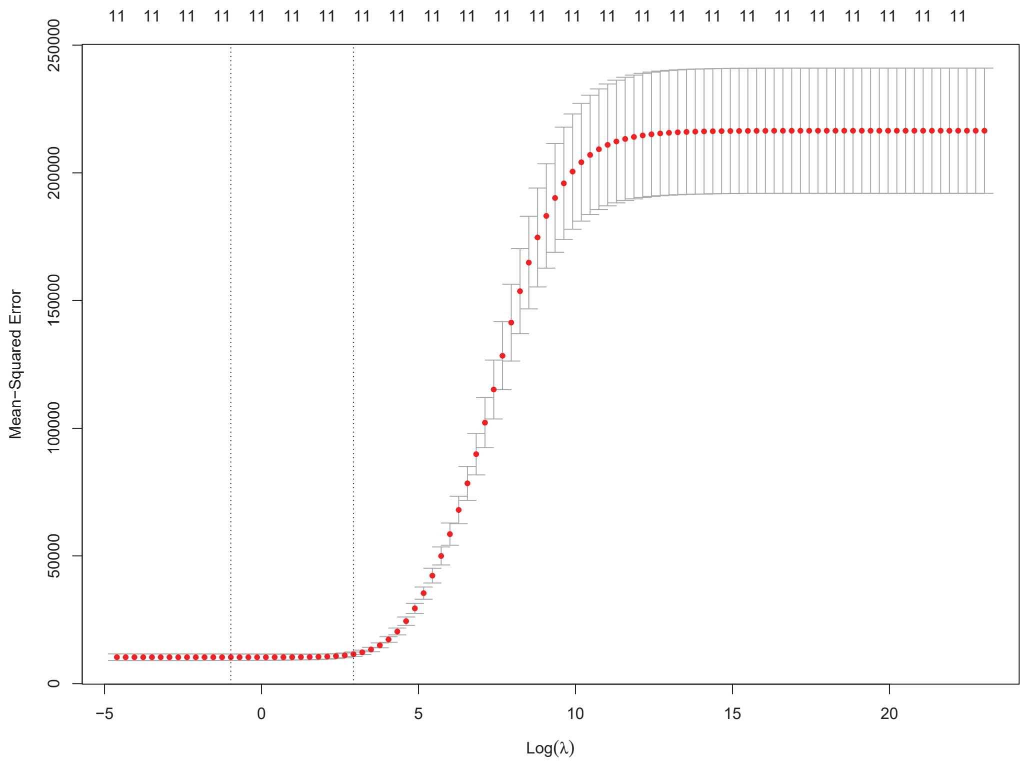
We now check whether there is any benefit to performing ridge regression with $\lambda = 6$ instead of just performing least squares regression. Recall that least squares is simply ridge regression with $\lambda = 0$.

```
> 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
```

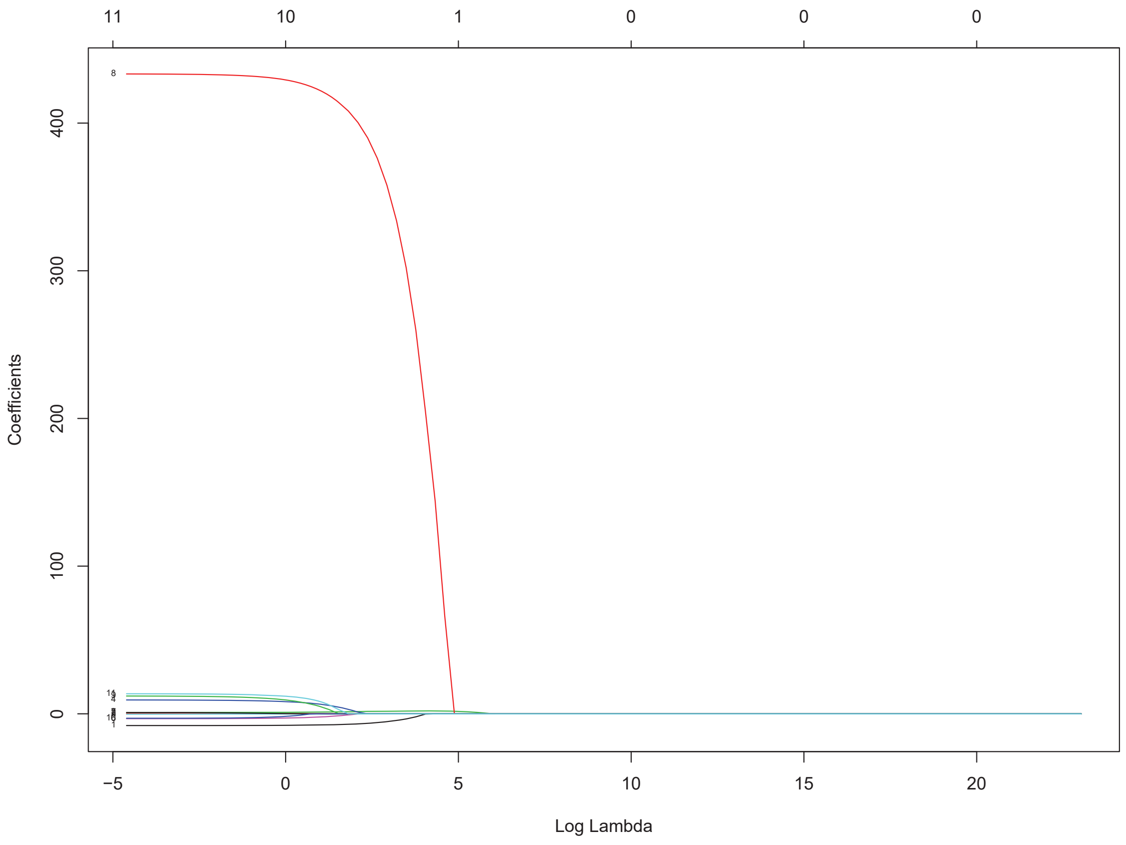
Finally, we refit our ridge regression model on the full data set, using the value of λ chosen by cross-validation, and examine the coefficient estimates.

```
> 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.

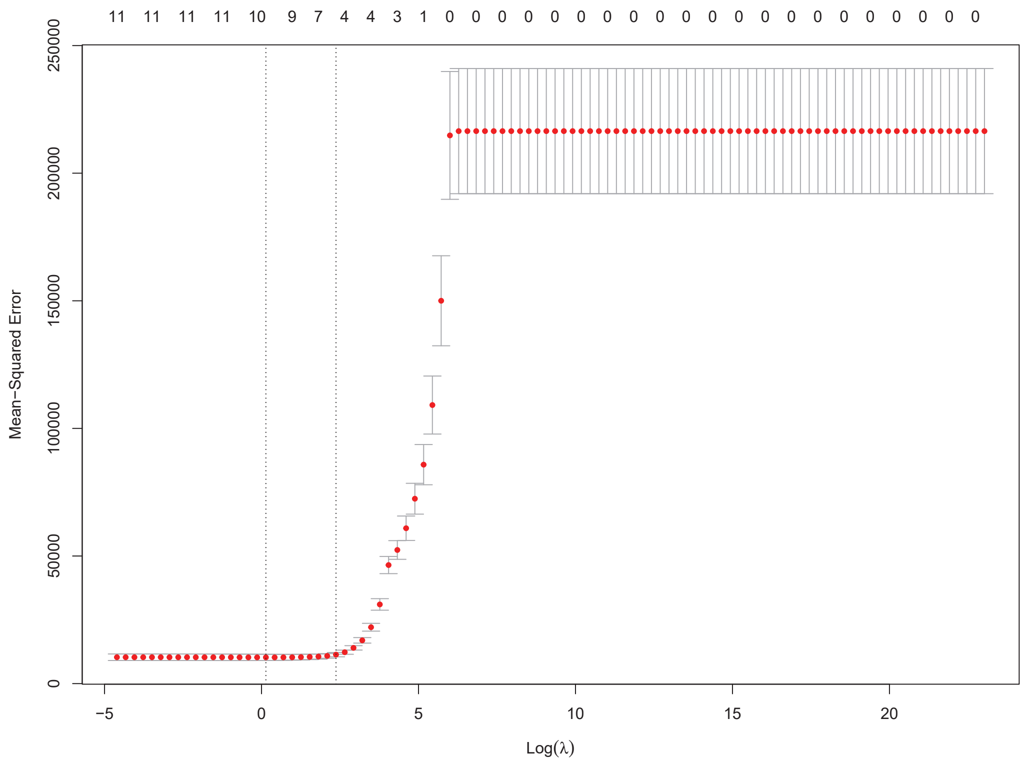
In order to fit a lasso model, we once again use the glmnet() function; however, this time we use the argument alpha=1. Other than that change, we proceed just as we did in fitting a ridge model.

```
> 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 
```