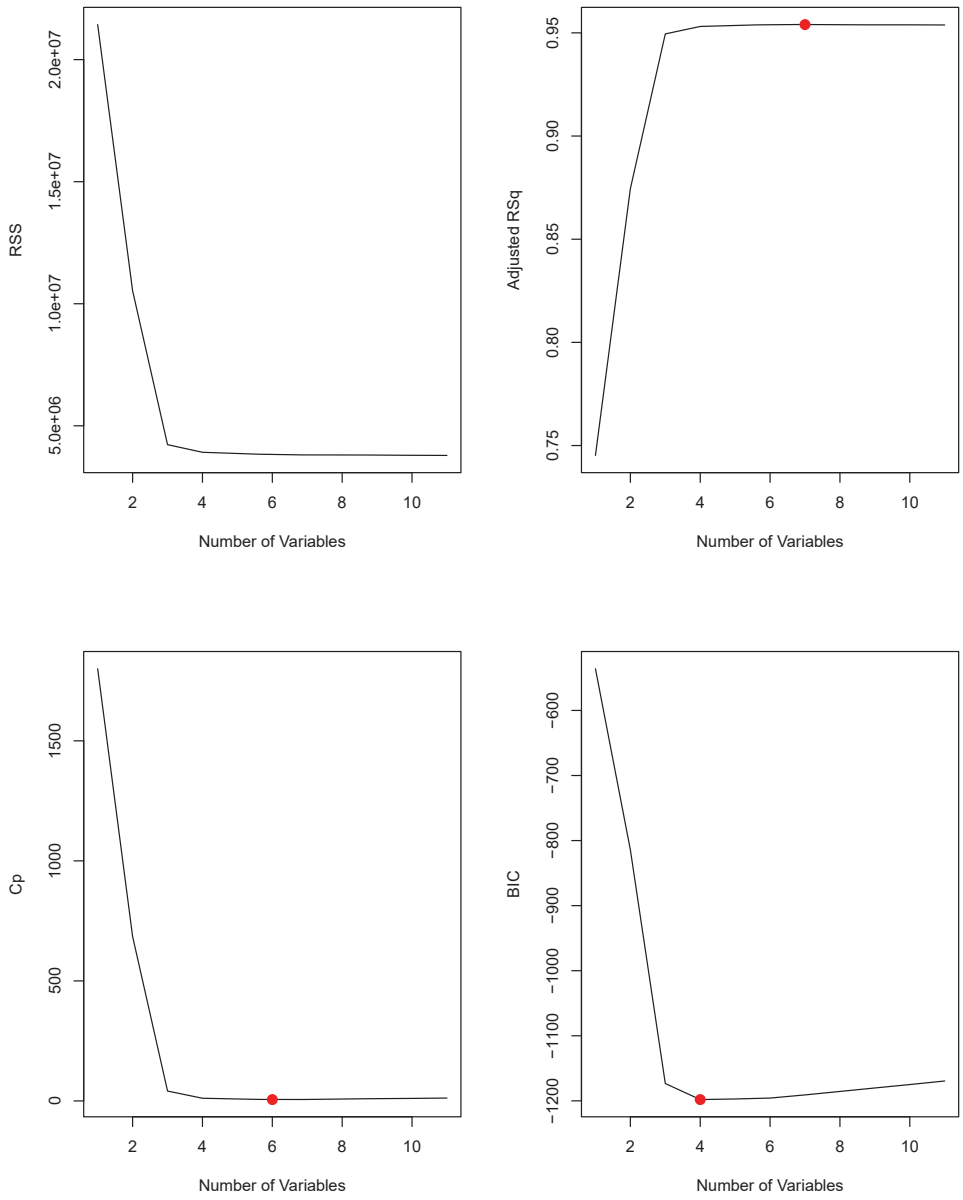
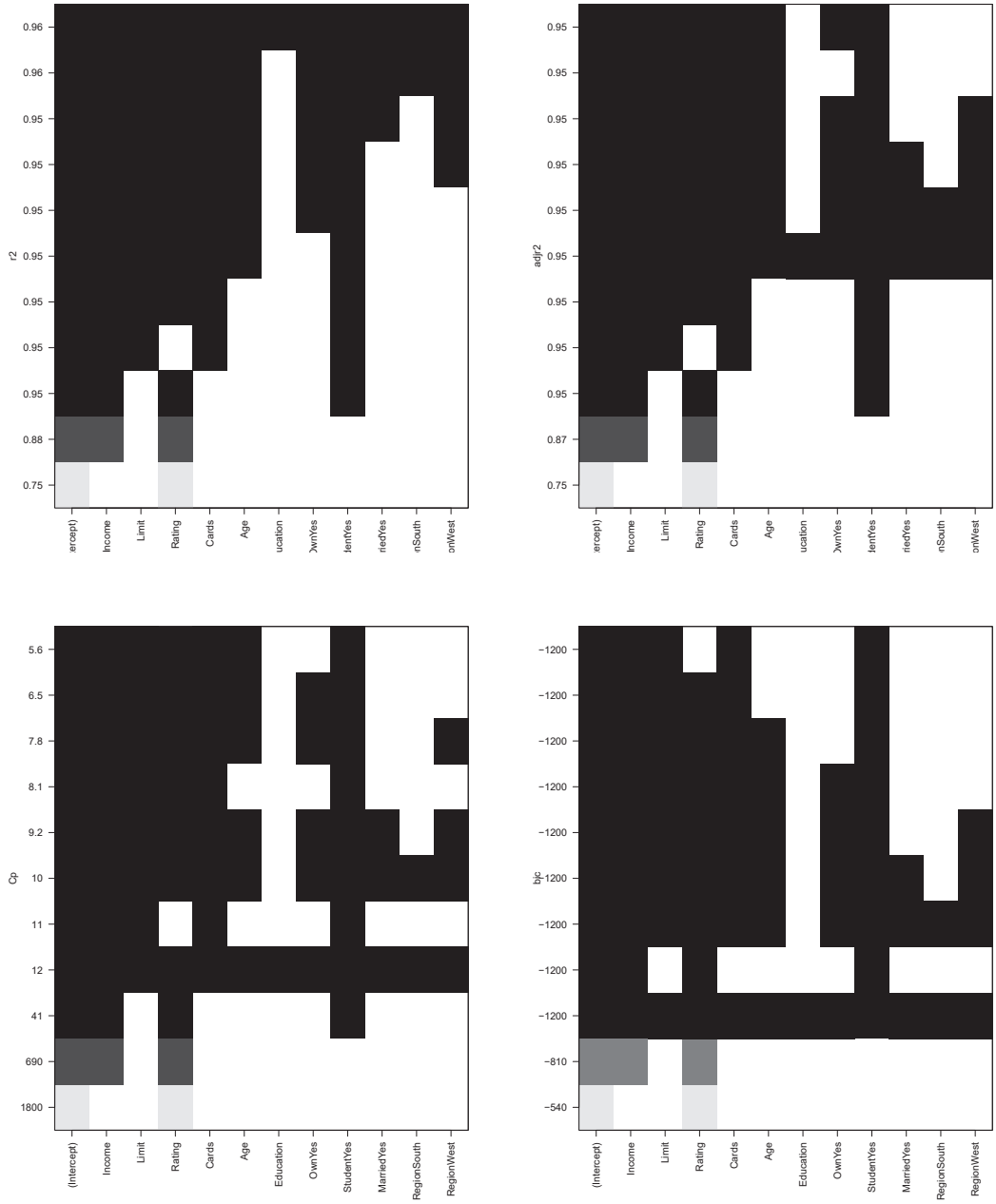
```
   type = "l")
> which.min(reg_summary$bic)
[1] 4
> points(which.min(reg_summary$bic),
   reg_summary$bic[which.min(reg_summary$bic)], col = "red", cex = 2, pch = 20)
```



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.

Note: 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

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

  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.