# Module 6: Recommended Exercises

Statistical Learning V2021

alexaoh

29 mars, 2021

## Recommended Exercise 1

1. Show that the least square estimator of a standard linear model is given by $\hat{\boldsymbol{\beta}} = (X^T X)^{-1} X\boldsymbol{Y}$.

I will use linear algebra and the projection theorem to show this easily. The least squares estimator wants to minimize the quantity $\|\boldsymbol{Y} - \hat{\boldsymbol{Y}}\|_2 = \|\boldsymbol{Y} - X\hat{\boldsymbol{\beta}}\|_2$. Since we know that $X\hat{\boldsymbol{\beta}} \in \text{Col}(X)$, the projection theorem gives that the closest point to $\boldsymbol{Y}$ in $\text{Col}(X)$ is the orthogonal projection of $\boldsymbol{Y}$ onto $\text{Col}(X)$. This means that

$$
\begin{aligned}
& X\hat{\boldsymbol{\beta}} = \text{proj}_{\text{Col}(X)}\boldsymbol{Y} \\
\implies & \boldsymbol{Y} - X\hat{\boldsymbol{\beta}} \in (\text{Col}(X))^{\perp} \\
\implies & \boldsymbol{v}^T(\boldsymbol{Y} - X\hat{\boldsymbol{\beta}}) = 0, \forall \boldsymbol{v} \in \text{Col}(X) \\
\implies & X^T(\boldsymbol{Y} - X\hat{\boldsymbol{\beta}}) = 0 \\
\implies & X^T\boldsymbol{Y} = X^T X\hat{\boldsymbol{\beta}}.
\end{aligned}
$$

Now, if $X$ has full rank, this implies that $\hat{\boldsymbol{\beta}} = (X^T X)^{-1} X\boldsymbol{Y}$, which completes the proof.

2. Show that the maximum likelihood estimator is equal to the least square estimator for the standard linear model.

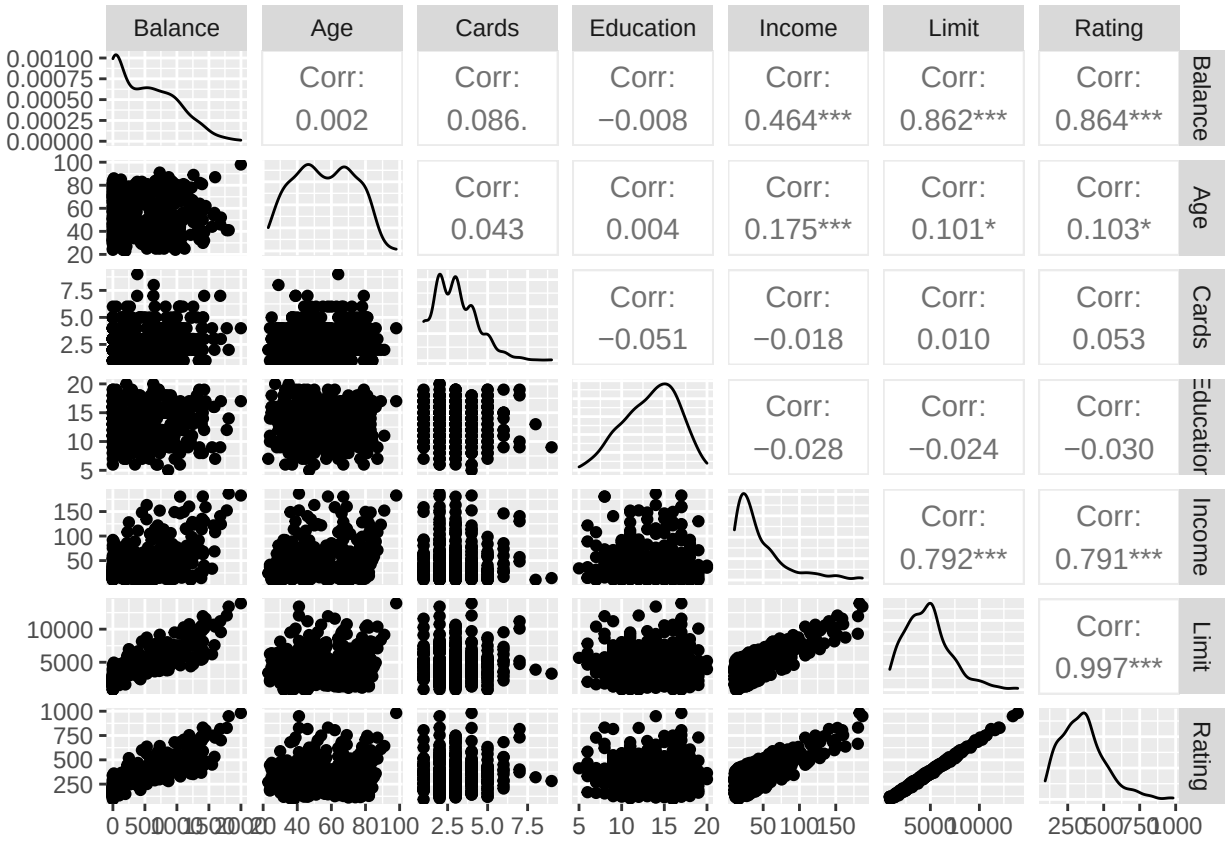
We assume that $\boldsymbol{Y} \sim N(X\hat{\boldsymbol{\beta}}, \sigma^2 I)$, which means that the log-likelihood function becomes

$$
\ln(\mathcal{L}(\boldsymbol{\beta}, \sigma^2)) = \ln\left(\frac{1}{(2\pi)^{n/2}\sigma^{n/2}} \exp\left(-\frac{1}{2}(\boldsymbol{Y} - X\boldsymbol{\beta})^T(\sigma^2 I)^{-1}(\boldsymbol{Y} - X\boldsymbol{\beta})\right)\right),
$$

which should be maximized. This is equivalent to minimizing the exponent $(\boldsymbol{Y} - X\boldsymbol{\beta})^T(\boldsymbol{Y} - X\boldsymbol{\beta}) = \|\boldsymbol{Y} - X\boldsymbol{\beta}\|_2^2$, which is what was done in 1. This shows that the MLE is equal to OLS for the standard linear model.

## Recommended Exercise 2

```
library(ISLR)
data(Credit)
library(GGally)
data <- Credit[, c("Balance", "Age", "Cards", "Education", "Income",
    "Limit", "Rating")]
ggpairs(data)
```




## Recommended Exercise 3

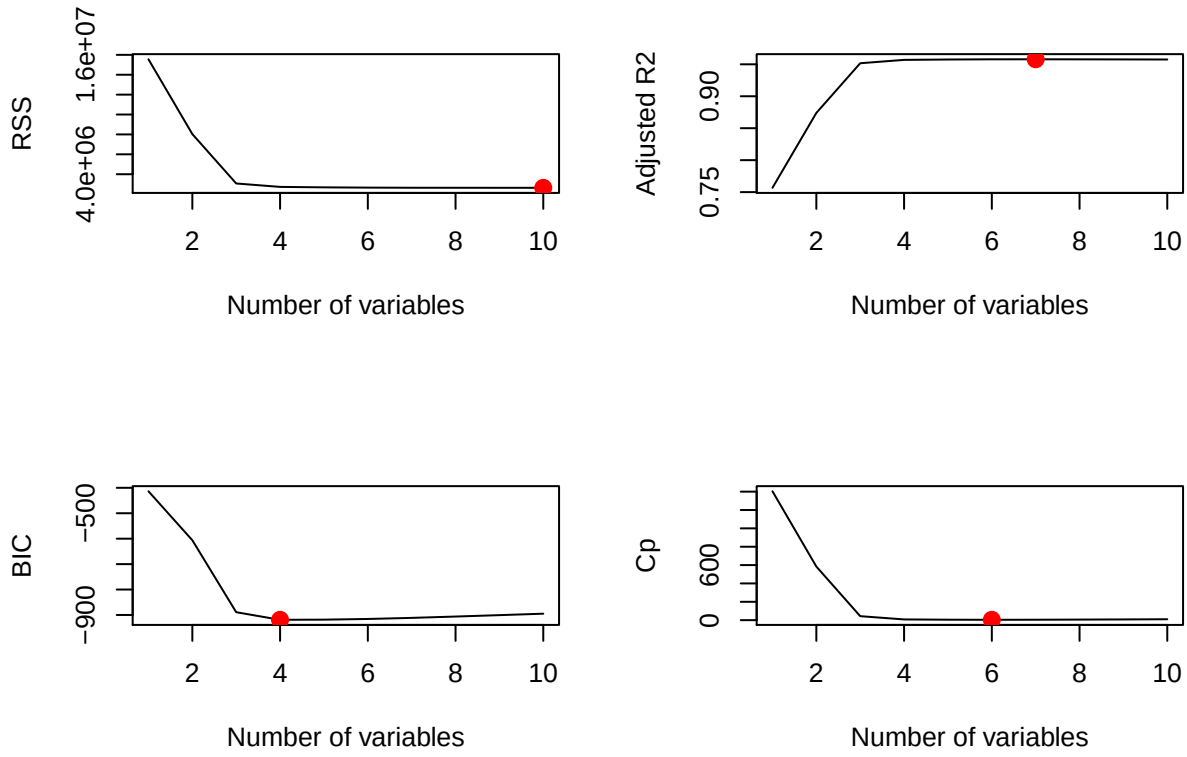
1. For the Credit dataset, pick the best model using Best Subset Selection according to $C_p$, $BIC$ and Adjusted $R^2$.

```
library(leaps)
sum(is.na(Credit))  # No data is missing!
```

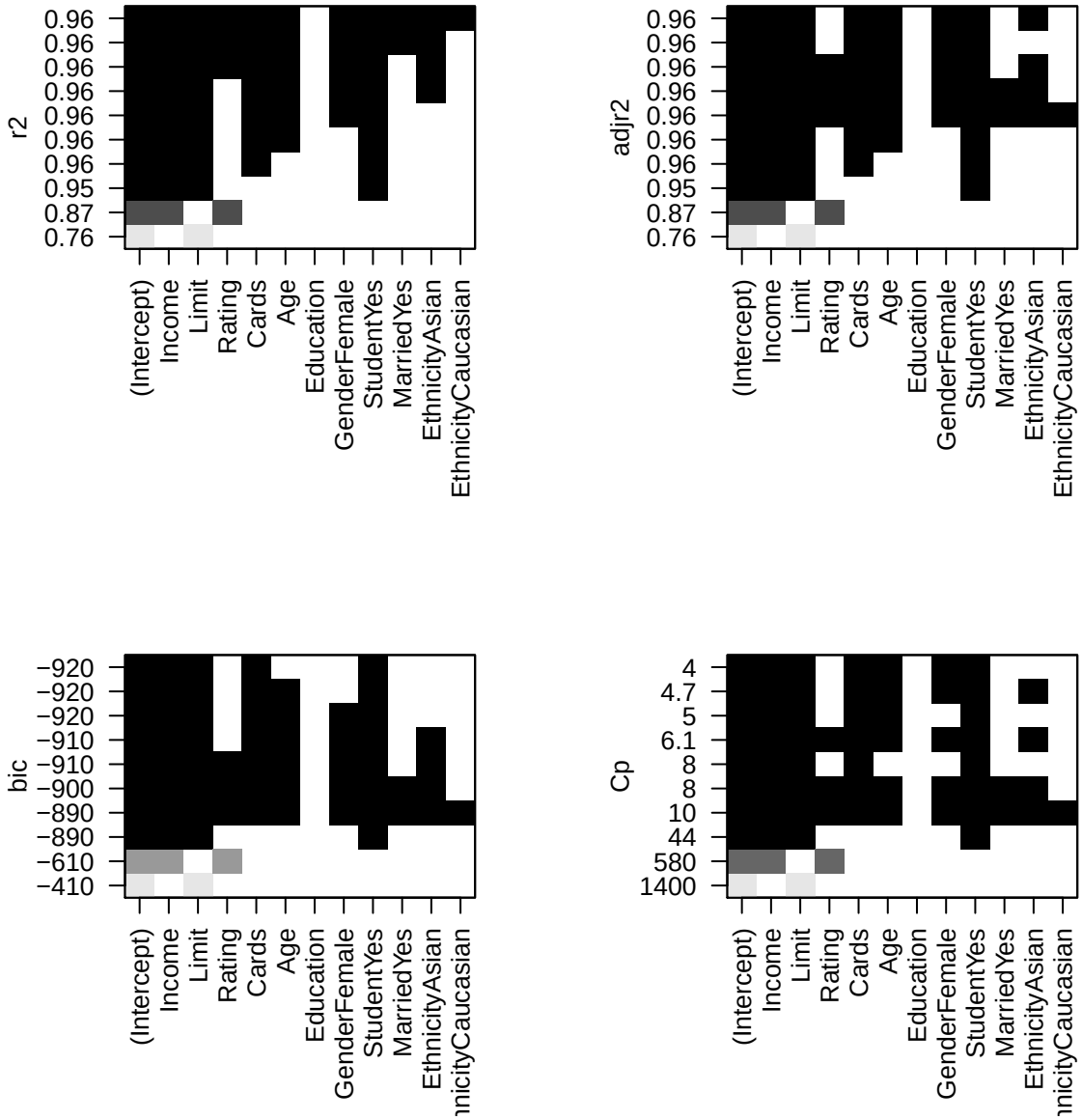
```
#> [1] 0
```

```
credit <- Credit[, -1]  # Remove the ID-column.
set.seed(1)
train_perc <- 0.75
credit_data_train_index <- sample(1:nrow(credit), nrow(credit) * train_perc)
credit_data_test_index <- (-credit_data_train_index)
credit_data_training <- credit[credit_data_train_index, ]
credit_data_testing <- credit[credit_data_test_index, ]

n <- ncol(credit_data_training) - 1

regfit.full <- regsubsets(Balance ~ ., data = credit_data_training, nvmax = n)
reg.summary <- summary(regfit.full)

# For plotting best points.
best.adjr2 <- which.max(reg.summary$adjr2)
best.rss <- which.min(reg.summary$rss)
best.cp <- which.min(reg.summary$cp)
```

```r
best.bic <- which.min(reg.summary$bic)

# Plot manually.
par(mfrow = c(2, 2))

plot(reg.summary$rss, xlab = "Number of variables", ylab = "RSS", type = "l")
points(best.rss, reg.summary$rss[best.rss], col = "red", cex = 2, pch = 20)

plot(reg.summary$adjr2, xlab = "Number of variables", ylab = "Adjusted R2",
    type = "l")
points(best.adjr2, reg.summary$adjr2[best.adjr2], col = "red", cex = 2,
    pch = 20)

plot(reg.summary$bic, xlab = "Number of variables", ylab = "BIC", type = "l")
points(best.bic, reg.summary$bic[best.bic], col = "red", cex = 2, pch = 20)

plot(reg.summary$cp, xlab = "Number of variables", ylab = "Cp", type = "l")
points(best.cp, reg.summary$cp[best.cp], col = "red", cex = 2, pch = 20)
```




```r
par(mfrow = c(2, 2))
plot(regfit.full, scale = "r2")
plot(regfit.full, scale = "adjr2")
plot(regfit.full, scale = "bic")
plot(regfit.full, scale = "Cp")
```

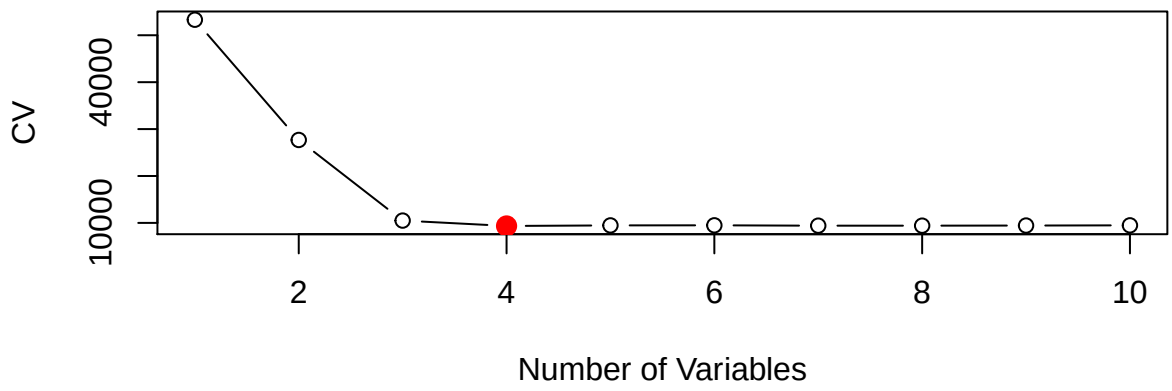



The above plots show which variables are selected in the optimal models, based on each of the different model selection criteria. A black square means that the variable is selected in the model that yields the given value of the criterium shown on the vertical axis. The optimal models are shown in the uppermost row of each of the plots.

2. Pick the best model using Best Subset Selection according to 10-fold CV.

```
set.seed(1)
# Create a prediction function to make predictions for regsubsets
# with id predictors included.
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
}

# Create indices to divide the data between folds.
k <- 10
folds <- sample(1:k, nrow(credit_data_training), replace = TRUE)
cv.errors <- matrix(NA, k, n, dimnames = list(NULL, paste(1:n)))

# Perform CV.
for (j in 1:k) {
    bss.obj <- regsubsets(Balance ~ ., data = credit_data_training[folds !=
        j, ], nvmax = n)
    for (i in 1:n) {
        pred <- predict(bss.obj, credit_data_training[folds == j, ],
            id = i)  # Skal ikke noe test-data brukes her?
        cv.errors[j, i] <- mean((credit_data_training$Balance[folds ==
            j] - pred)^2)
    }
}

# Compute mean cv errors for each model size.
mean.cv.errors = apply(cv.errors, 2, mean)
bss.cv <- which.min(mean.cv.errors)

# Plot the mean cv errors.
plot(mean.cv.errors, xlab = "Number of Variables", ylab = "CV", type = "b")
points(bss.cv, mean.cv.errors[bss.cv], col = "red", cex = 2, pch = 20)
```




```
par(mfrow = c(1, 1))

# Best model, based on amount of variables chosen by cv.
reg.best <- regsubsets(Balance ~ ., data = credit)
```

```r
coef(reg.best, bss.cv)  # Selected variables.
```

```
#>  (Intercept)       Income        Limit        Cards   StudentYes
#> -499.7272117   -7.8392288    0.2666445   23.1753794  429.6064203
```

```r
fit <- lm(Balance ~ Income + Limit + Cards + Student, data = credit_data_training)
summary(fit)
```

```
#>
#> Call:
#> lm(formula = Balance ~ Income + Limit + Cards + Student, data = credit_data_training)
#>
#> Residuals:
#>     Min      1Q  Median      3Q     Max
#> -160.26  -76.81  -11.21   48.15  350.49
#>
#> Coefficients:
#>               Estimate Std. Error t value Pr(>|t|)
#> (Intercept) -5.216e+02  1.758e+01 -29.670  < 2e-16 ***
#> Income      -7.856e+00  2.651e-01 -29.627  < 2e-16 ***
#> Limit        2.706e-01  4.001e-03  67.622  < 2e-16 ***
#> Cards        2.426e+01  3.981e+00   6.094 3.43e-09 ***
#> StudentYes   4.196e+02  1.782e+01  23.542  < 2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 96.14 on 295 degrees of freedom
#> Multiple R-squared:  0.9575, Adjusted R-squared:  0.9569
#> F-statistic:  1661 on 4 and 295 DF,  p-value: < 2.2e-16
```

```r
pred.regbest <- predict(fit, newdata = credit_data_testing)
mse.regbest <- mean((pred.regbest - credit_data_testing$Balance)^2)  # Test Mean Square Error.
mse.regbest
```

```
#> [1] 12243.75
```

```r
######### A nicer way of doing it (even though I do not understand this
######### completely!)  Create info for lm call
variables <- names(coef(regfit.full, id = bss.cv))
variables <- variables[!variables %in% "(Intercept)"]
bsm_formula <- as.formula(regfit.full$call[[2]])
bsm_design_matrix <- model.matrix(bsm_formula, credit_data_training)[,
    variables]
bsm_data_train <- data.frame(Balance = credit_data_training$Balance,
    bsm_design_matrix)
# Fit a standard linear model using only the selected# predictors on
# the training data
model_best_subset_method <- lm(formula = bsm_formula, bsm_data_train)
summary(model_best_subset_method)
```

```
#>
#> Call:
#> lm(formula = bsm_formula, data = bsm_data_train)
#>
#> Residuals:
#>     Min      1Q  Median      3Q     Max
```

```
#> -160.26  -76.81  -11.21   48.15  350.49
#>
#> Coefficients:
#>               Estimate Std. Error t value Pr(>|t|)
#> (Intercept) -5.216e+02  1.758e+01 -29.670  < 2e-16 ***
#> Income      -7.856e+00  2.651e-01 -29.627  < 2e-16 ***
#> Limit        2.706e-01  4.001e-03  67.622  < 2e-16 ***
#> Cards        2.426e+01  3.981e+00   6.094 3.43e-09 ***
#> StudentYes   4.196e+02  1.782e+01  23.542  < 2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 96.14 on 295 degrees of freedom
#> Multiple R-squared:  0.9575, Adjusted R-squared:  0.9569
#> F-statistic:  1661 on 4 and 295 DF,  p-value: < 2.2e-16
```

3. Compare the results from the previous steps.

```
fit.adjr2 <- lm(Balance ~ Income + Limit + Cards + Age + Gender + Student +
    Ethnicity, data = credit_data_training)
pred.adjr2 <- predict(fit.adjr2, newdata = credit_data_testing)
mse.adjr2 <- mean((pred.adjr2 - credit_data_testing$Balance)^2)  # Test Mean Square Error.
mse.adjr2
```

```
#> [1] 12659.23
```

```
fit.BIC <- lm(Balance ~ Income + Limit + Cards + Student, data = credit_data_training)
pred.BIC <- predict(fit.BIC, newdata = credit_data_testing)
mse.BIC <- mean((pred.BIC - credit_data_testing$Balance)^2)  # Test Mean Square Error.
mse.BIC
```

```
#> [1] 12243.75
```

```
fit.Cp <- lm(Balance ~ Income + Limit + Cards + Age + Gender + Student,
    data = credit_data_training)
pred.Cp <- predict(fit.Cp, newdata = credit_data_testing)
mse.Cp <- mean((pred.Cp - credit_data_testing$Balance)^2)  # Test Mean Square Error.
mse.Cp
```

```
#> [1] 12505.6
```

```
# Table for MSE from the different models.
msrate = rbind(c(mse.adjr2), c(mse.BIC), c(mse.Cp), c(mse.regbest))
rownames(msrate) = c("Adjr2", "BIC", "Cp", "10 fold CV")
colnames(msrate) = c("Test MSE")
msrate
```

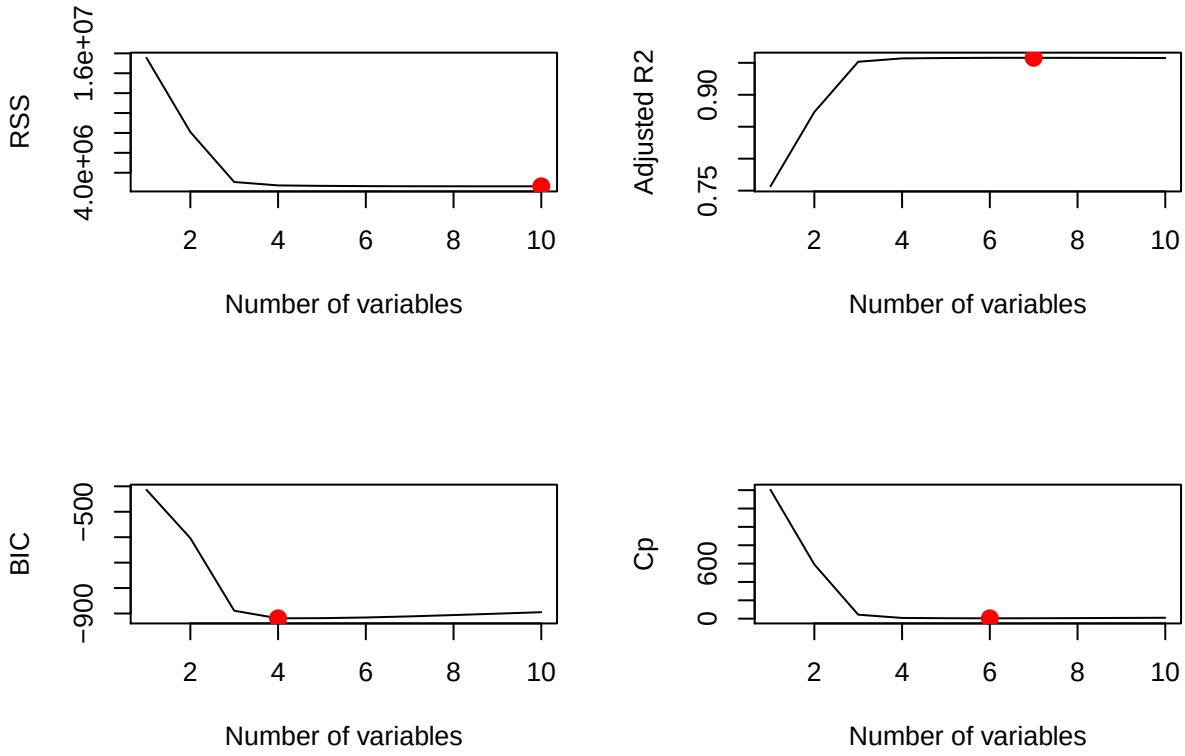
```
#>            Test MSE
#> Adjr2      12659.23
#> BIC        12243.75
#> Cp         12505.60
#> 10 fold CV 12243.75
```

We can see that 10 fold CV and BIC (since they are the same model in this case), give the best test MSE.

## Recommended Exercise 4

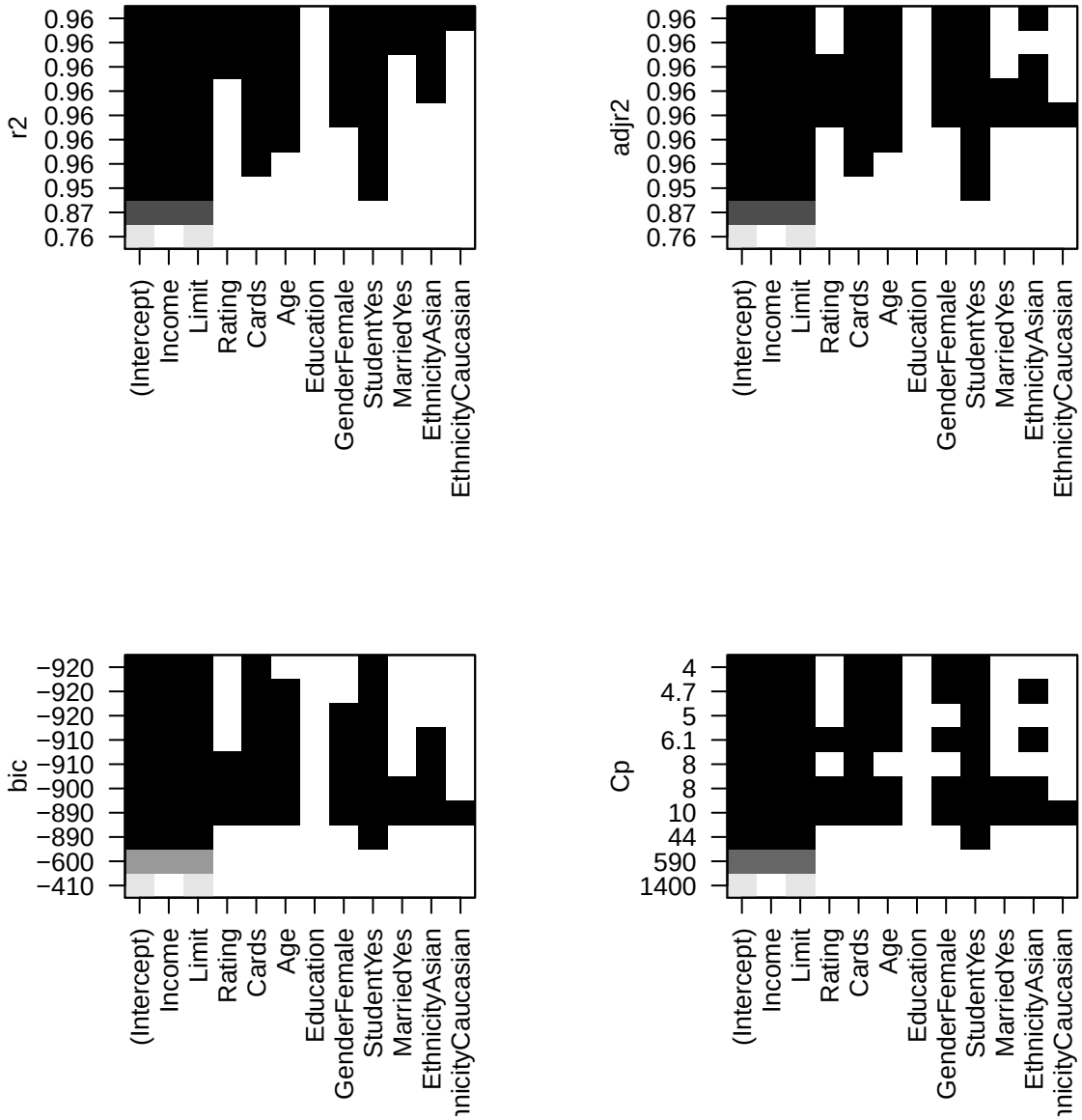
1. Select the best model for the Credit dataset using Forward, Backward and Hybrid (sequential replacement) Stepwise Selection.

```
# Forward selection.
regfit.fwd <- regsubsets(Balance ~ ., data = credit_data_training, nvmax = n,
    method = "forward")
reg.fwd.summary <- summary(regfit.fwd)

# For plotting best points.
best.adjr2.fwd <- which.max(reg.fwd.summary$adjr2)
best.rss.fwd <- which.min(reg.fwd.summary$rss)
best.cp.fwd <- which.min(reg.fwd.summary$cp)
best.bic.fwd <- which.min(reg.fwd.summary$bic)

# Plot manually.
par(mfrow = c(2, 2))

plot(reg.fwd.summary$rss, xlab = "Number of variables", ylab = "RSS",
    type = "l")
points(best.rss.fwd, reg.fwd.summary$rss[best.rss.fwd], col = "red",
    cex = 2, pch = 20)

plot(reg.fwd.summary$adjr2, xlab = "Number of variables", ylab = "Adjusted R2",
    type = "l")
points(best.adjr2.fwd, reg.fwd.summary$adjr2[best.adjr2.fwd], col = "red",
    cex = 2, pch = 20)

plot(reg.fwd.summary$bic, xlab = "Number of variables", ylab = "BIC",
    type = "l")
points(best.bic.fwd, reg.fwd.summary$bic[best.bic.fwd], col = "red",
    cex = 2, pch = 20)

plot(reg.fwd.summary$cp, xlab = "Number of variables", ylab = "Cp", type = "l")
points(best.cp.fwd, reg.fwd.summary$cp[best.cp.fwd], col = "red", cex = 2,
    pch = 20)
```



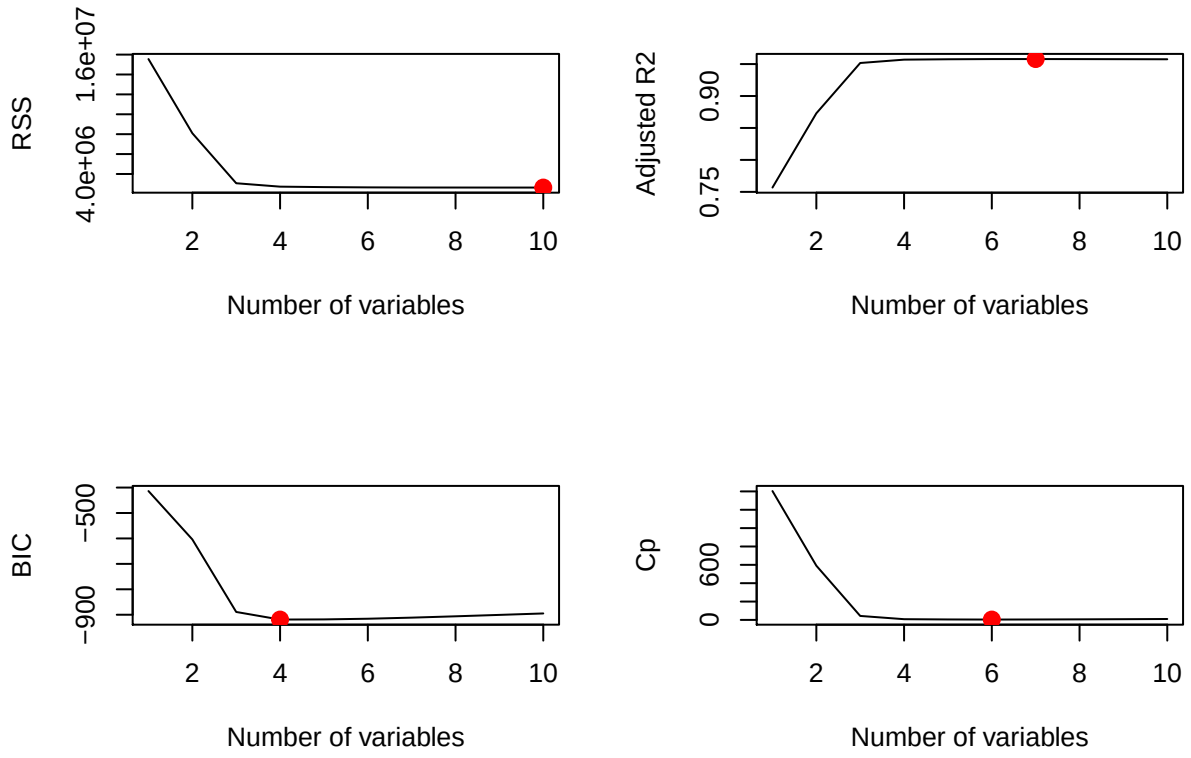

The variables selected in Forward Stepwise Selection are

```
par(mfrow = c(2, 2))
plot(regfit.fwd, scale = "r2")
plot(regfit.fwd, scale = "adjr2")
plot(regfit.fwd, scale = "bic")
plot(regfit.fwd, scale = "Cp")
```




```r
# Backward selection.
regfit.bwd <- regsubsets(Balance ~ ., data = credit_data_training, nvmax = n,
    method = "forward")
reg.bwd.summary <- summary(regfit.bwd)

# For plotting best points.
best.adjr2.bwd <- which.max(reg.bwd.summary$adjr2)
best.rss.bwd <- which.min(reg.bwd.summary$rss)
best.cp.bwd <- which.min(reg.bwd.summary$cp)
best.bic.bwd <- which.min(reg.bwd.summary$bic)
```

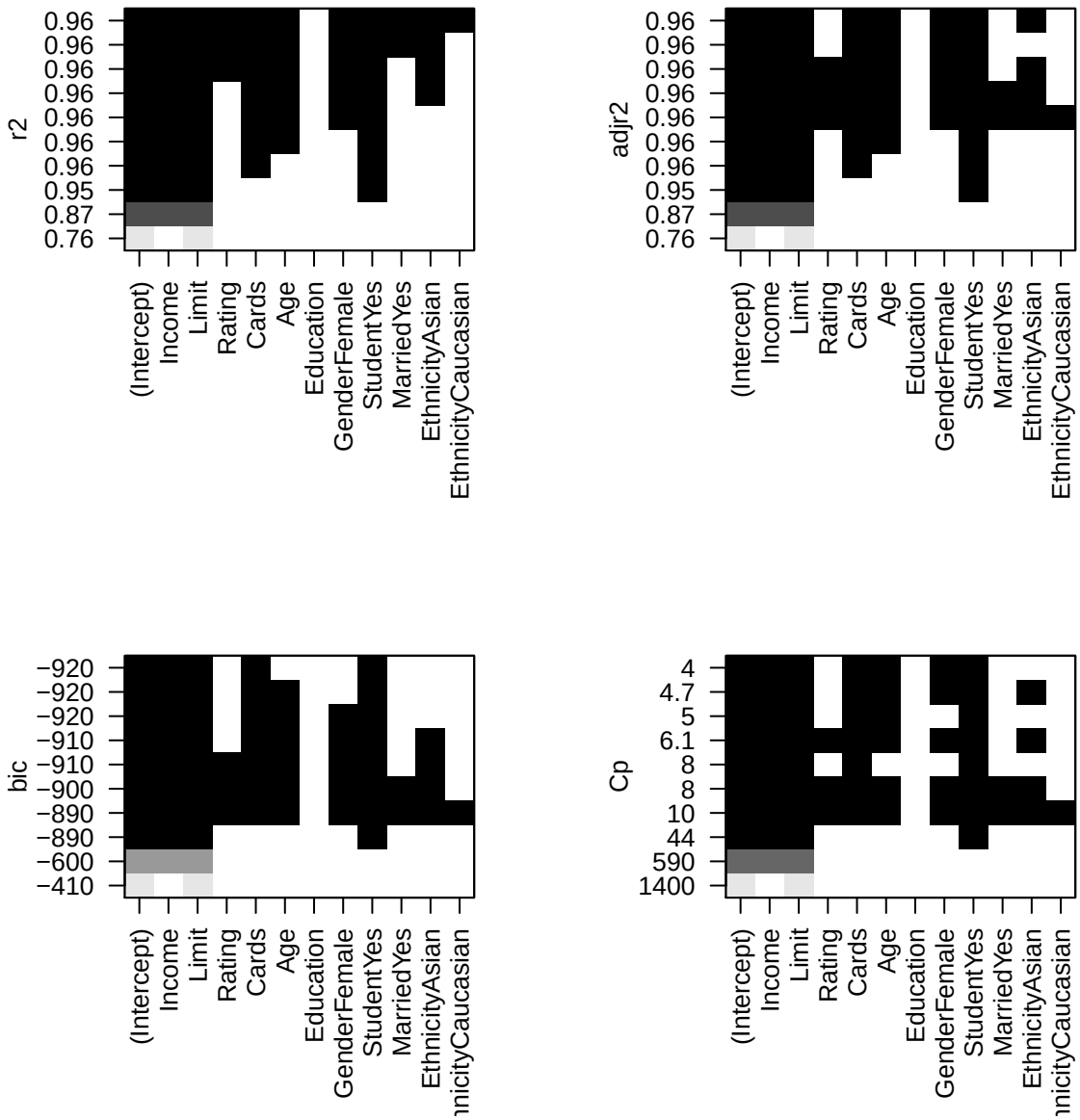
```
par(mfrow = c(2, 2))

plot(reg.bwd.summary$rss, xlab = "Number of variables", ylab = "RSS",
    type = "l")
points(best.rss.bwd, reg.bwd.summary$rss[best.rss.bwd], col = "red",
    cex = 2, pch = 20)

plot(reg.bwd.summary$adjr2, xlab = "Number of variables", ylab = "Adjusted R2",
    type = "l")
points(best.adjr2.bwd, reg.bwd.summary$adjr2[best.adjr2.bwd], col = "red",
    cex = 2, pch = 20)

plot(reg.bwd.summary$bic, xlab = "Number of variables", ylab = "BIC",
    type = "l")
points(best.bic.bwd, reg.bwd.summary$bic[best.bic.bwd], col = "red",
    cex = 2, pch = 20)

plot(reg.bwd.summary$cp, xlab = "Number of variables", ylab = "Cp", type = "l")
points(best.cp.bwd, reg.bwd.summary$cp[best.cp.bwd], col = "red", cex = 2,
    pch = 20)
```




The variables selected in Backward Stepwise Selection are

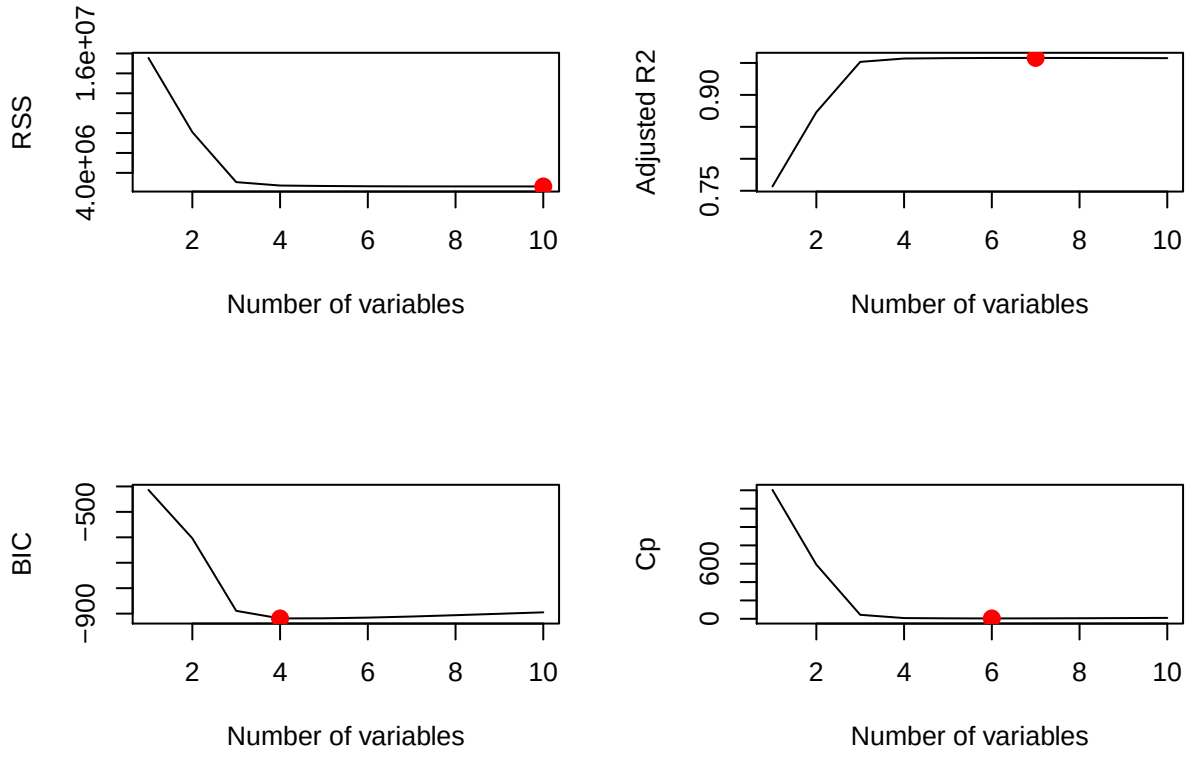
```
par(mfrow = c(2, 2))
plot(regfit.bwd, scale = "r2")
plot(regfit.bwd, scale = "adjr2")
```

```
plot(regfit.bwd, scale = "bic")
plot(regfit.bwd, scale = "Cp")
```




```
# Hybrid selection.
regfit.h <- regsubsets(Balance ~ ., data = credit_data_training, nvmax = n,
    method = "forward")
reg.h.summary <- summary(regfit.h)

# For plotting best points.
best.adjr2.h <- which.max(reg.h.summary$adjr2)
best.rss.h <- which.min(reg.h.summary$rss)
```

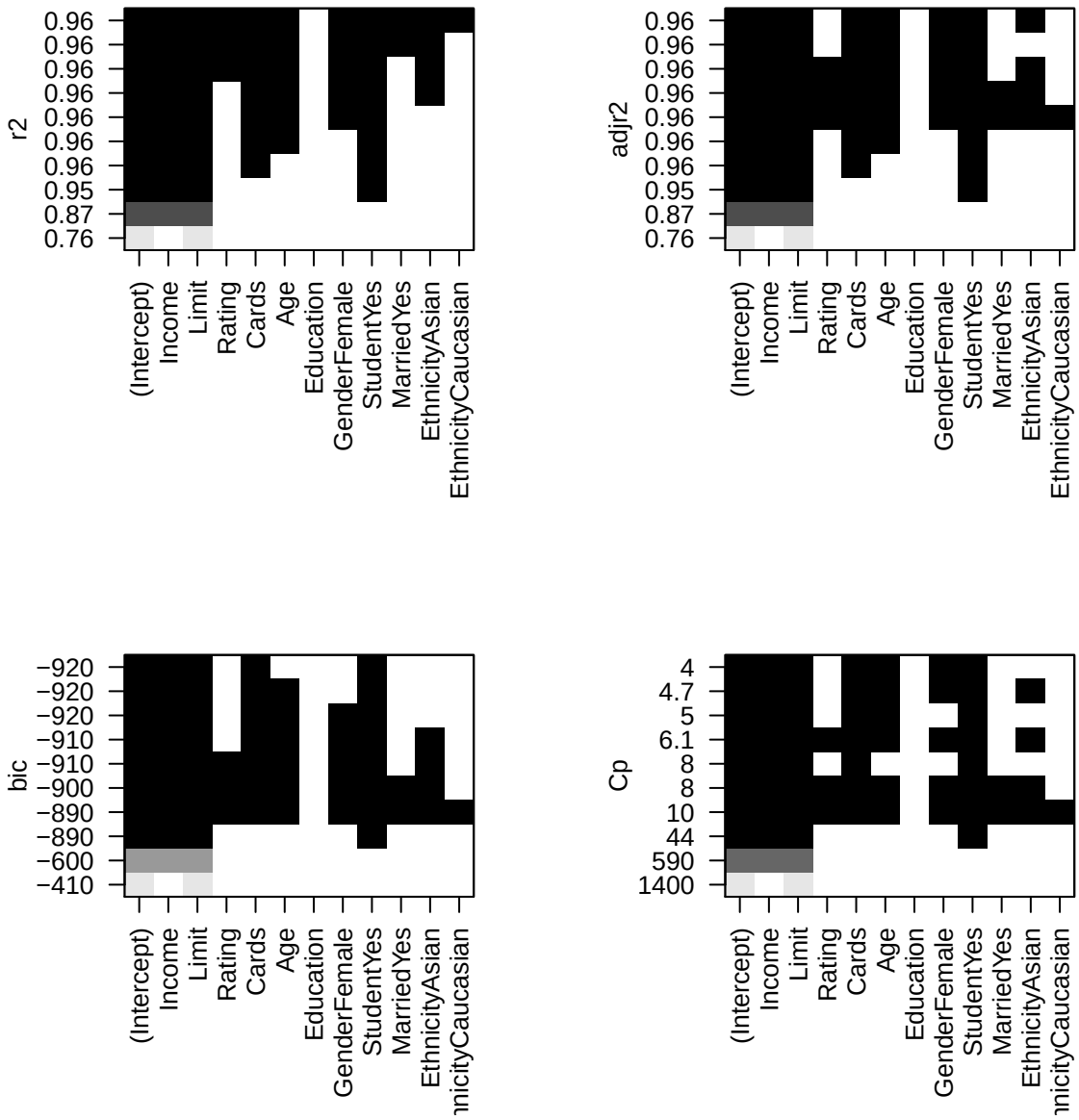
```
best.cp.h <- which.min(reg.h.summary$cp)
best.bic.h <- which.min(reg.h.summary$bic)

par(mfrow = c(2, 2))

plot(reg.h.summary$rss, xlab = "Number of variables", ylab = "RSS", type = "l")
points(best.rss.h, reg.h.summary$rss[best.rss.h], col = "red", cex = 2,
    pch = 20)

plot(reg.h.summary$adjr2, xlab = "Number of variables", ylab = "Adjusted R2",
    type = "l")
points(best.adjr2.h, reg.h.summary$adjr2[best.adjr2.h], col = "red",
    cex = 2, pch = 20)

plot(reg.h.summary$bic, xlab = "Number of variables", ylab = "BIC", type = "l")
points(best.bic.h, reg.h.summary$bic[best.bic.h], col = "red", cex = 2,
    pch = 20)

plot(reg.h.summary$cp, xlab = "Number of variables", ylab = "Cp", type = "l")
points(best.cp.h, reg.h.summary$cp[best.cp.h], col = "red", cex = 2,
    pch = 20)
```




The variables selected in Hybrid Stepwise Selection are

```
par(mfrow = c(2, 2))
plot(regfit.h, scale = "r2")
```

```
plot(regfit.h, scale = "adjr2")
plot(regfit.h, scale = "bic")
plot(regfit.h, scale = "Cp")
```

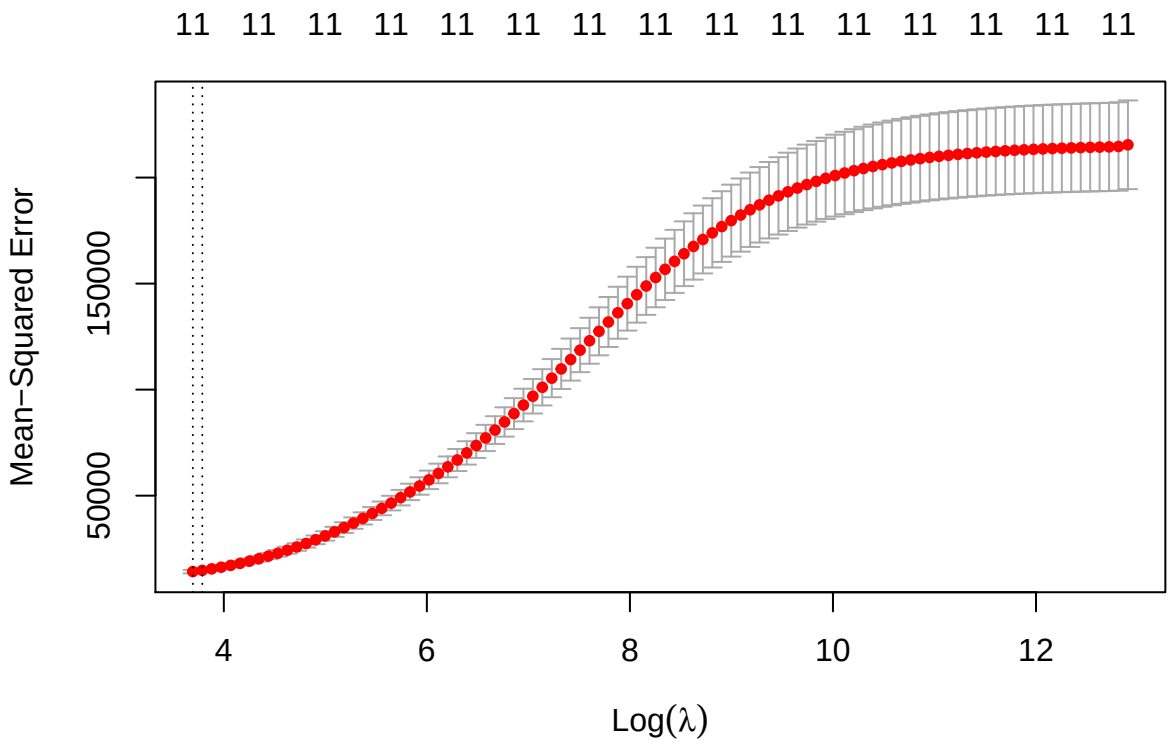



The same models are selected with all three methods and with Best Subset Selection!

## Recommended Exercise 5

1. Apply Ridge regression to the Credit dataset.

```r
library(glmnet)
x.train <- model.matrix(Balance ~ ., data = credit_data_training)[, -1]  # Remove the intercept.
y.train <- credit_data_training$Balance

x.test <- model.matrix(Balance ~ ., data = credit_data_testing)[, -1]  # Remove the intercept.
y.test <- credit_data_testing$Balance

# grid <- 10^seq(10, -2, length = 100)
ridge.mod <- glmnet(x.train, y.train, alpha = 0)  #, lambda = grid) # alpha = 0 --> Ridge.

set.seed(1)
cv.ridge <- cv.glmnet(x.train, y.train, alpha = 0)
plot(cv.ridge)
```




```r
best.lambda.ridge <- cv.ridge$lambda.min
best.lambda.ridge
```

```
#> [1] 40.24862
```

2. Compare the results with the standard linear regression.

```r
lin.reg <- lm(y.train ~ x.train, )

# Lin reg predictions.
lin.reg.pred <- predict(lin.reg, newx = x.test)
lin.reg.mse <- mean((lin.reg.pred - y.test)^2)
lin.reg.mse
```

```
#> [1] 411841.1
```

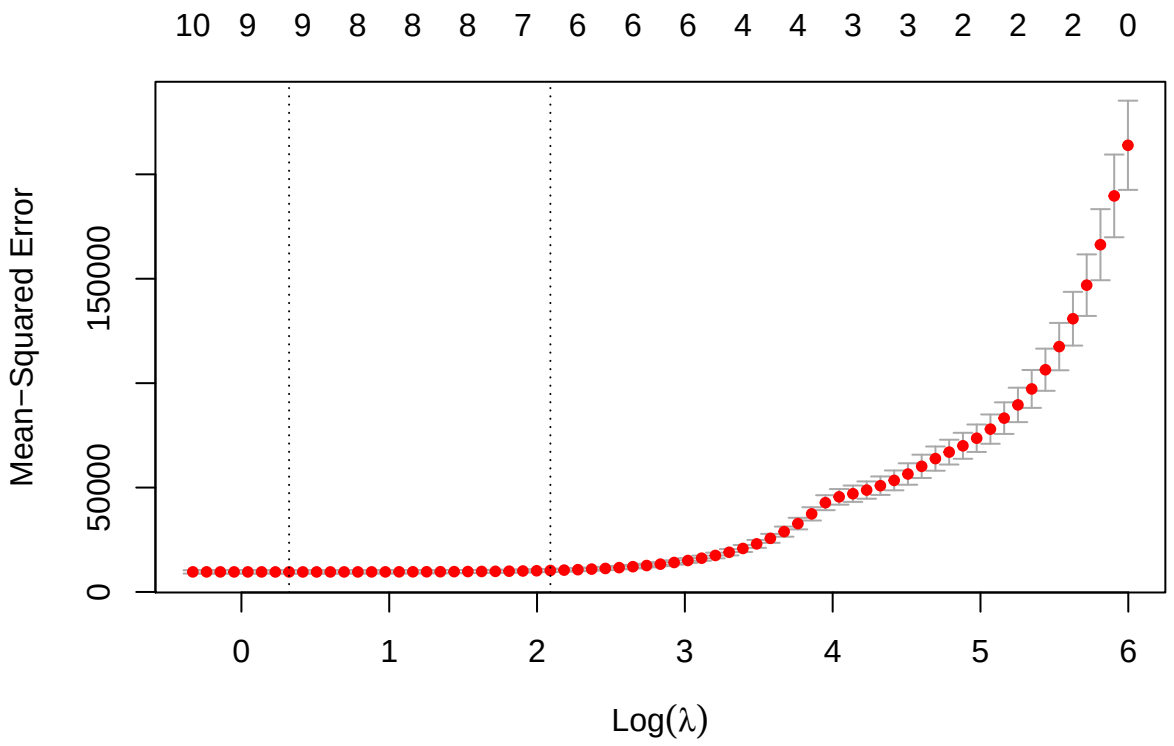
```
# Ridge predictions.
ridge.pred <- predict(ridge.mod, s = best.lambda.ridge, newx = x.test)
ridge.mse <- mean((ridge.pred - y.test)^2)
ridge.mse
```

```
#> [1] 15742.83
```

### Recommended Exercise 6

1. Apply Lasso regression to the Credit dataset.

```
lasso.mod <- glmnet(x.train, y.train, alpha = 1)  #, lambda = grid) # alpha = 0 --> Ridge.

set.seed(1)
cv.lasso <- cv.glmnet(x.train, y.train, alpha = 1)
plot(cv.lasso)
```




```
best.lambda.lasso <- cv.lasso$lambda.min
best.lambda.lasso
```

```
#> [1] 1.380717
```

2. Compare the results with the standard linear regression and the Ridge regression.

```
lin.reg.mse
```

```
#> [1] 411841.1
```

```r
# Lasso predictions.
lasso.pred <- predict(lasso.mod, s = best.lambda.lasso, newx = x.test)
lasso.mse <- mean((lasso.pred - y.test)^2)
lasso.mse
```

```
#> [1] 12077.15
```

Lasso gives the smallest test MSE among the linear regression, Ridge regression and Lasso regression methods.

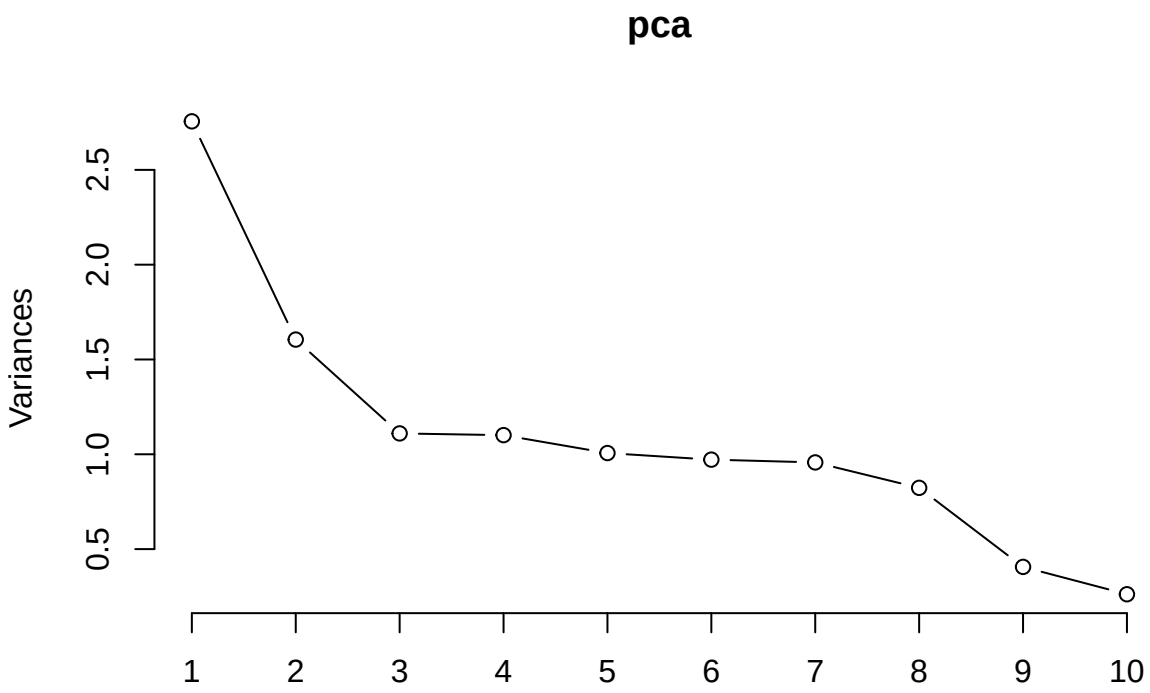
## Recommended Exercise 7

We should use 8 principal components for the Credit dataset because of the R-output below. It is debatable how many should be used, but since each of the principal components until PC8 explain between 7.4% and 25%, while PC9 only explains 3.7%, PC8 is thought of as the final most important component.

```r
x <- model.matrix(Balance ~ ., data = credit)[, -1]
pca <- prcomp(x, scale = TRUE, center = TRUE)
# print(pca)
summary(pca)
```

```
#> Importance of components:
#>                           PC1    PC2    PC3    PC4    PC5     PC6     PC7
#> Standard deviation     1.6601 1.2669 1.0536 1.0493 1.0032 0.98577 0.97831
#> Proportion of Variance 0.2505 0.1459 0.1009 0.1001 0.0915 0.08834 0.08701
#> Cumulative Proportion  0.2505 0.3964 0.4973 0.5974 0.6889 0.77727 0.86427
#>                           PC8     PC9    PC10    PC11
#> Standard deviation     0.90715 0.63723 0.51174 0.04618
#> Proportion of Variance 0.07481 0.03691 0.02381 0.00019
#> Cumulative Proportion  0.93908 0.97600 0.99981 1.00000
```

```r
plot(pca, type = "l")
```




### Recommended exercise 8

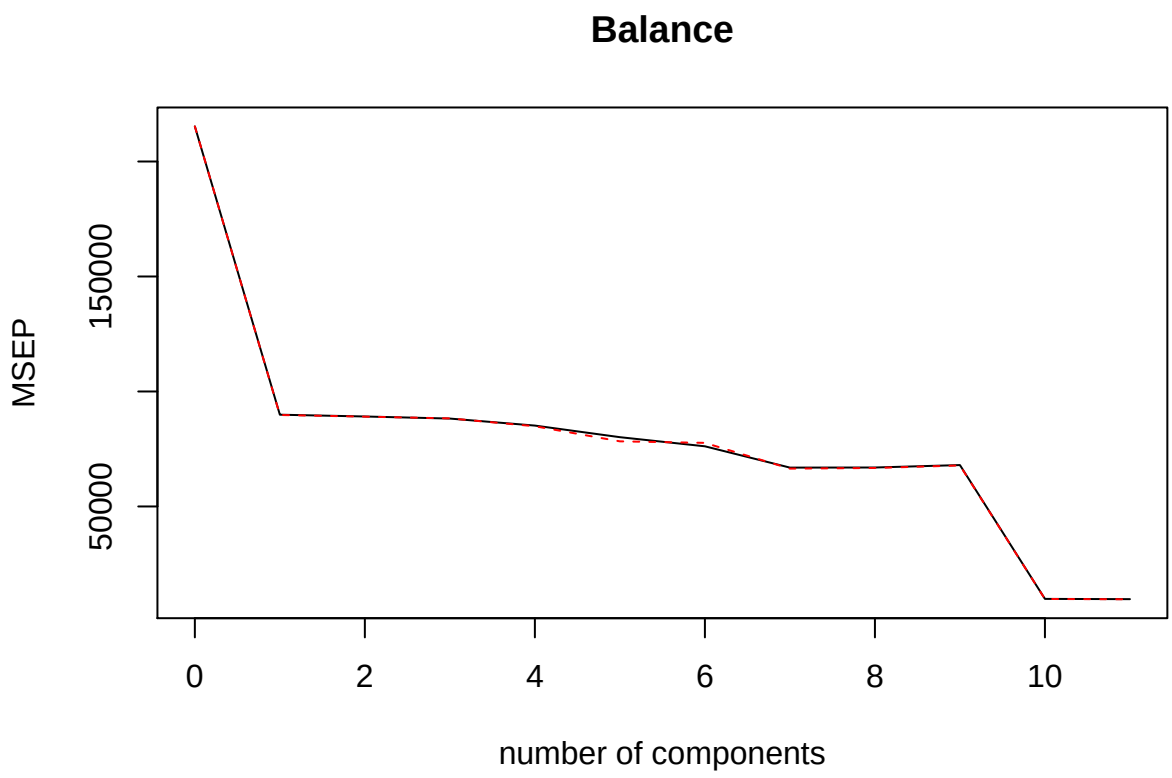
Apply PCR on the Credit dataset and compare the results with the previous methods used in this module.

```
library(pls)
set.seed(1)
pcr.fit <- pcr(Balance ~ ., data = credit_data_training, scale = TRUE,
    validation = "CV")
summary(pcr.fit)
```

```
#> Data:    X dimension: 300 11
#>  Y dimension: 300 1
#> Fit method: svdpc
#> Number of components considered: 11
#>
#> VALIDATION: RMSEP
#> Cross-validated using 10 random segments.
#>        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
#> CV             464    299.8    298.5    297.0    291.8    283.1    276.0
#> adjCV          464    299.6    298.4    297.2    291.4    279.9    278.7
#>        7 comps  8 comps  9 comps  10 comps  11 comps
#> CV       258.6    258.7    260.8     99.08     98.45
#> adjCV    257.8    258.3    260.4     98.87     98.21
#>
#> TRAINING: % variance explained
#>          1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
#> X          25.29    40.07    50.73    60.39    69.38    78.09    86.55    94.25
```

```
#> Balance    58.41    58.92    59.96    61.79    66.09    66.12    70.84    70.86
#>          9 comps  10 comps  11 comps
#> X          97.81     99.98    100.00
#> Balance    70.88     95.77     95.89
```

```
validationplot(pcr.fit, val.type = "MSEP")
```




```
pcr.pred <- predict(pcr.fit, credit_data_testing, ncomp = 10)
mse.pcr <- mean((pcr.pred - credit_data_testing$Balance)^2)
mse.pcr
```

```
#> [1] 11578.1
```

### Recommended exercise 9

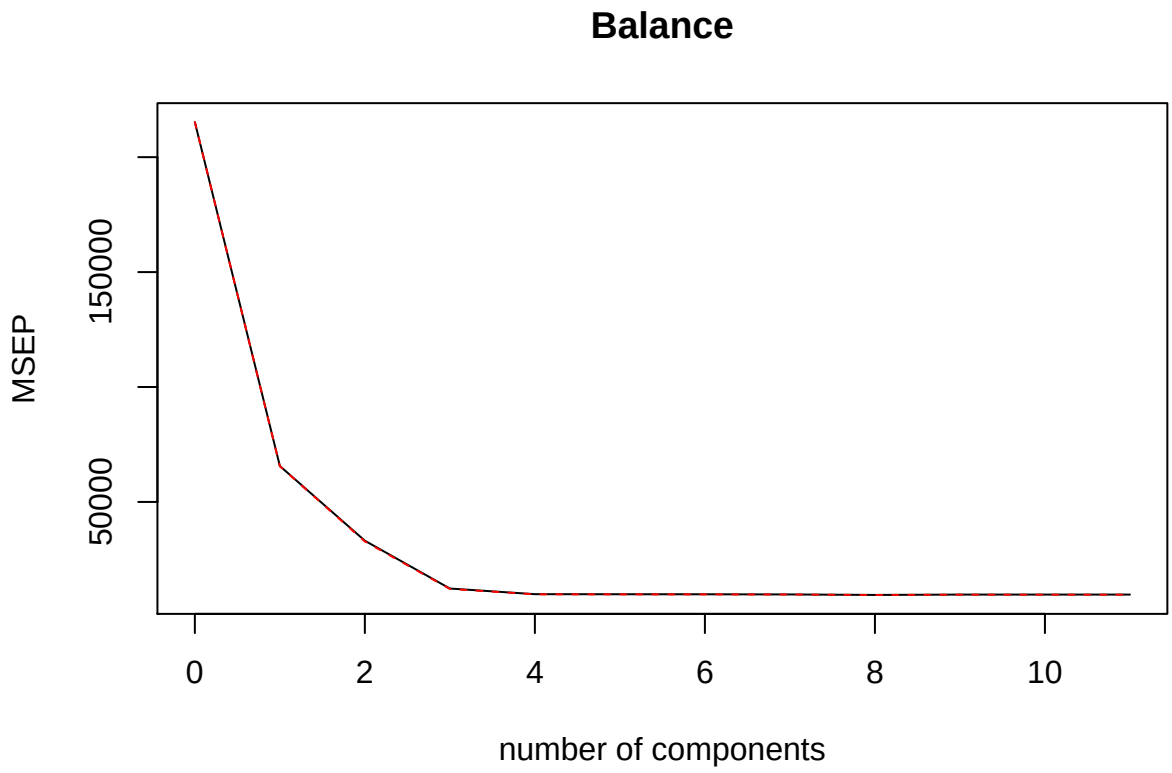
Apply PLS on the Credit dataset and compare the results with the previous methods used in this module.

```
set.seed(1)
pls.fit <- plsr(Balance ~ ., data = credit_data_training, scale = TRUE,
    validation = "CV")
summary(pls.fit)
```

```
#> Data:    X dimension: 300 11
#>  Y dimension: 300 1
#> Fit method: kernelpls
#> Number of components considered: 11
#>
#> VALIDATION: RMSEP
```

```
#> Cross-validated using 10 random segments.
#>        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
#> CV             464    256.2    182.0    111.0    99.33    99.17    99.08
#> adjCV          464    256.0    181.1    110.2    99.01    98.93    98.87
#>        7 comps  8 comps  9 comps  10 comps  11 comps
#> CV       98.74    97.61    98.48     98.46     98.45
#> adjCV    98.58    97.58    98.19     98.21     98.21
#>
#> TRAINING: % variance explained
#>          1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
#> X          24.78    33.24    38.78    50.84    61.04    68.88    74.31    81.10
#> Balance    70.31    85.90    94.70    95.74    95.77    95.77    95.78    95.81
#>          9 comps  10 comps  11 comps
#> X          83.50     91.30    100.00
#> Balance    95.89     95.89     95.89
```

```
validationplot(pls.fit, val.type = "MSEP")
```




```
pls.pred <- predict(pls.fit, credit_data_testing, ncomp = 3)
mse.pls <- mean((pls.pred - credit_data_testing$Balance)^2)
mse.pls
```

```
#> [1] 12476.32
```

One final comparison table is made, between all the different methods used in this exercise.

```
# Table for MSE from the different models.
msvalues <- rbind(c(mse.regbest), c(ridge.mse), c(lasso.mse), c(mse.pcr),
```

```
    c(mse.pls))
rownames(msvalues) <- c("BSS (10 fold CV)", "Ridge", "Lasso", "PCR",
    "PLS")
colnames(msvalues) <- c("Test MSE")
msvalues
```

```
#>                  Test MSE
#> BSS (10 fold CV) 12243.75
#> Ridge            15742.83
#> Lasso            12077.15
#> PCR              11578.10
#> PLS              12476.32
```

Have a look in the solutions for how to make boxplots for the test MSE for all the methods with ggplot (very nice!).