

TMA4268 Statistical Learning V2020

Module 6: Solution sketches

Thiago G. Martins, Department of Mathematical Sciences, NTNU

Spring 2020

Recommended exercise 1

For the least square estimator, the solution can be found in the first session here.

For the maximum likelihood estimator, the solution can be found here.

Recommended exercise 2

```
library(ISLR) # Package with data for an Introduction to Statistical
              # Learning with Applications in R
```

```
# Load Credit dataset
data(Credit)
```

```
# Check column names
names(Credit)
```

```
## [1] "ID"      "Income"  "Limit"   "Rating"  "Cards"
## [6] "Age"     "Education" "Gender"  "Student" "Married"
## [11] "Ethnicity" "Balance"
```

```
# Check dataset shape
dim(Credit)
```

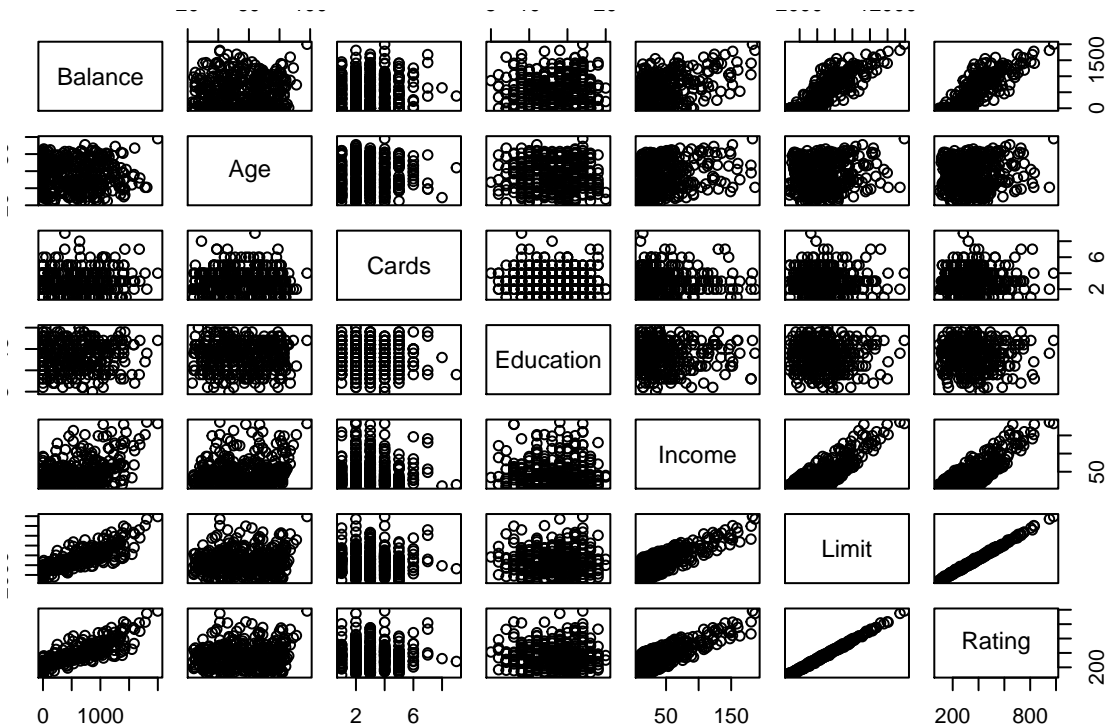
```
## [1] 400 12
```

```
head(Credit)
```

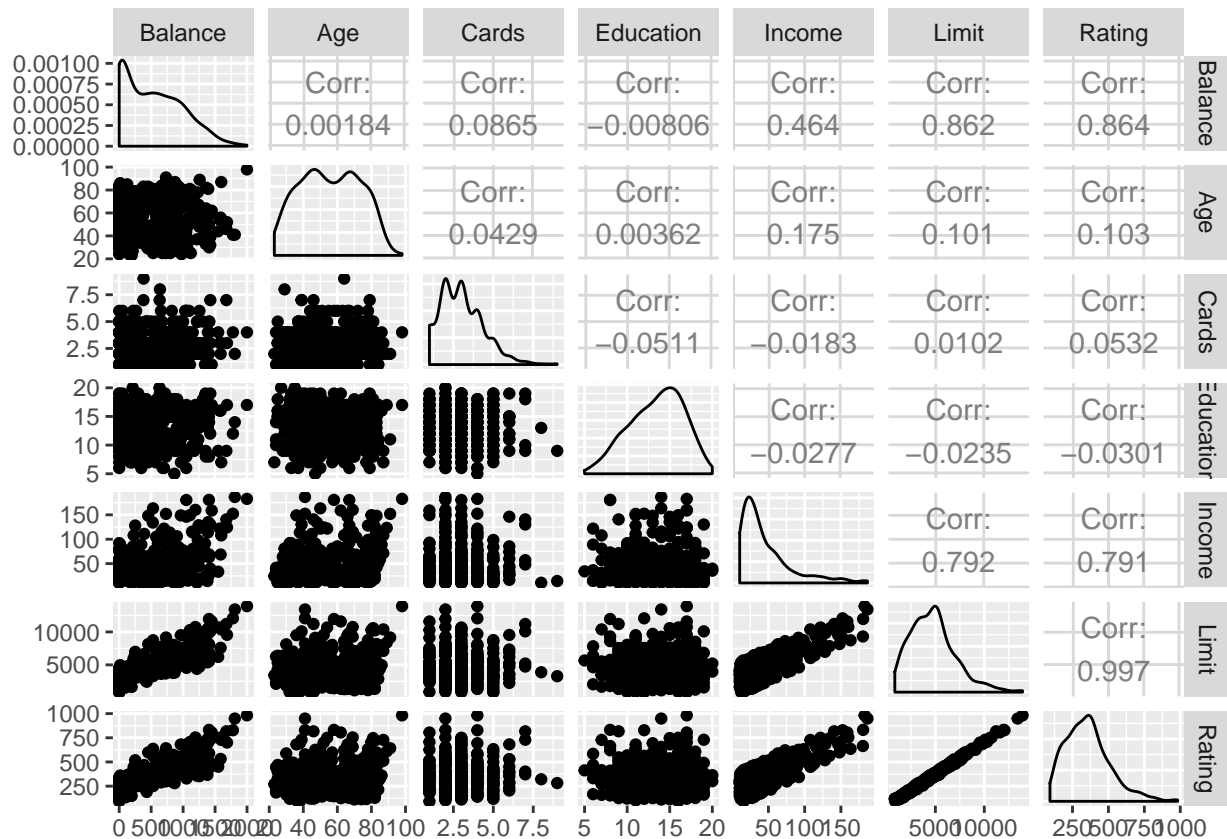
```
##   ID  Income Limit Rating Cards Age Education Gender Student Married
## 1  1  14.891  3606   283     2  34         11  Male      No      Yes
## 2  2 106.025  6645   483     3  82         15 Female    Yes      Yes
## 3  3 104.593  7075   514     4  71         11  Male      No      No
## 4  4 148.924  9504   681     3  36         11 Female    No      No
## 5  5  55.882  4897   357     2  68         16  Male      No      Yes
## 6  6  80.180  8047   569     4  77         10  Male      No      No
##   Ethnicity Balance
## 1 Caucasian    333
## 2   Asian     903
## 3   Asian     580
## 4   Asian     964
## 5 Caucasian    331
## 6 Caucasian   1151
```

```
# Select variable to plot
pairwise_scatter_data <- Credit[,c("Balance", "Age", "Cards", "Education",
                                   "Income", "Limit", "Rating")]
```

```
# Simplest possible pairwise scatter plot
pairs(pairwise_scatter_data)
```



```
# More interesting but slower pairwise plot from package GGally
library(GGally)
ggpairs(data=pairwise_scatter_data)
```



Check here for quick get started to ggpairs

Recommended exercise 3

```
# Exclude 'ID' column
credit_data <- subset(Credit, select=-c(ID))

# Counting the dummy variables as well
credit_data_number_predictors <- 11

# Take a look at the data
head(credit_data)
```

```
##      Income Limit Rating Cards Age Education Gender Student Married
## 1  14.891  3606    283    2  34         11  Male      No      Yes
## 2 106.025  6645    483    3  82         15 Female    Yes      Yes
## 3 104.593  7075    514    4  71         11  Male      No      No
## 4 148.924  9504    681    3  36         11 Female    No      No
## 5  55.882  4897    357    2  68         16  Male      No      Yes
## 6  80.180  8047    569    4  77         10  Male      No      No

##      Ethnicity Balance
## 1 Caucasian      333
## 2   Asian      903
## 3   Asian      580
## 4   Asian      964
## 5 Caucasian      331
```

```
## 6 Caucasian      1151
```

```
# Summary statistics  
summary(credit_data)
```

```
##      Income      Limit      Rating      Cards  
## Min.   : 10.35   Min.    : 855   Min.    : 93.0   Min.    :1.000  
## 1st Qu.: 21.01   1st Qu.: 3088   1st Qu.:247.2   1st Qu.:2.000  
## Median : 33.12   Median : 4622   Median :344.0   Median :3.000  
## Mean   : 45.22   Mean    : 4736   Mean    :354.9   Mean    :2.958  
## 3rd Qu.: 57.47   3rd Qu.: 5873   3rd Qu.:437.2   3rd Qu.:4.000  
## Max.   :186.63   Max.    :13913   Max.    :982.0   Max.    :9.000  
##      Age      Education      Gender      Student      Married  
## Min.   :23.00   Min.    : 5.00   Male :193   No :360   No :155  
## 1st Qu.:41.75   1st Qu.:11.00   Female:207   Yes: 40   Yes:245  
## Median :56.00   Median :14.00  
## Mean   :55.67   Mean    :13.45  
## 3rd Qu.:70.00   3rd Qu.:16.00  
## Max.   :98.00   Max.    :20.00  
##      Ethnicity      Balance  
## African American: 99   Min.    : 0.00  
## Asian           :102   1st Qu.: 68.75  
## Caucasian       :199   Median : 459.50  
##                Mean    : 520.01  
##                3rd Qu.: 863.00  
##                Max.    :1999.00
```

```
# Create train and test set indexes
```

```
set.seed(1)  
train_perc <- 0.75  
credit_data_train_index <- sample(1:nrow(credit_data), nrow(credit_data)*train_perc)  
credit_data_test_index <- (-credit_data_train_index)
```

```
# Create train and test set
```

```
credit_data_training <- credit_data[credit_data_train_index, ]  
credit_data_testing <- credit_data[credit_data_test_index, ]
```

```
library(leaps)
```

```
# Perform best subset selection using all the predictors and the training data
```

```
best_subset_method=regsubsets(Balance~.,credit_data_training,nvmax=credit_data_number_predictors)
```

```
# Save summary obj
```

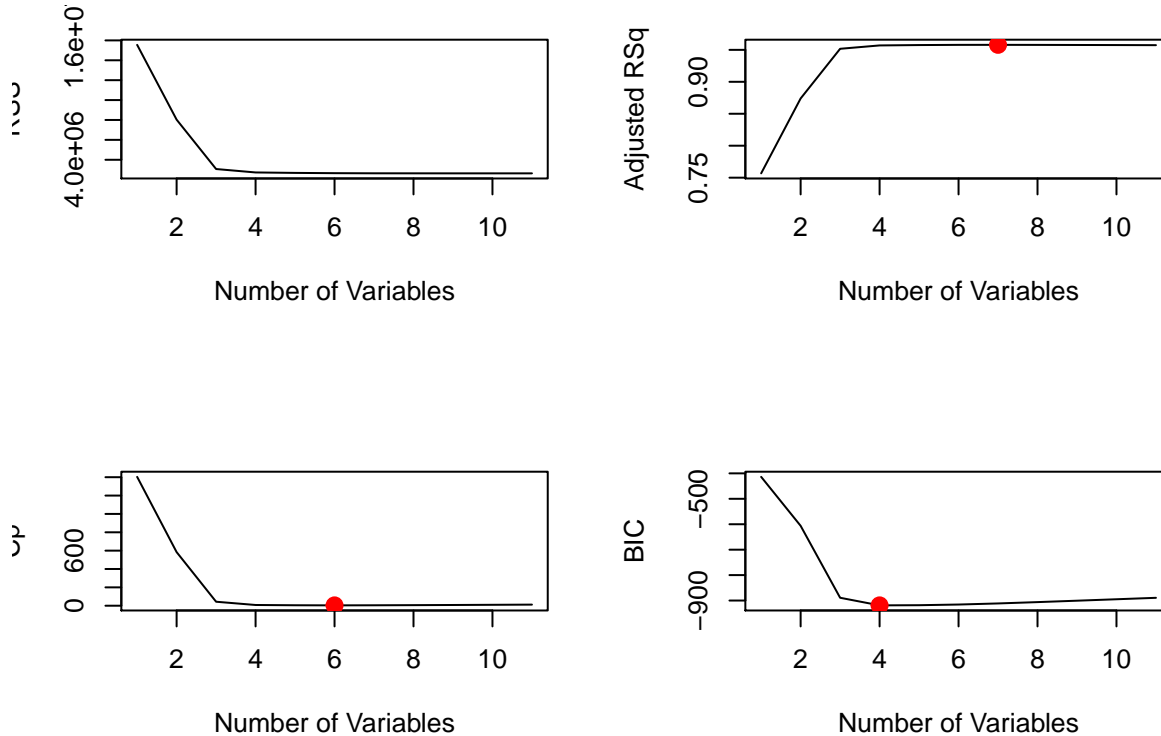
```
best_subset_method_summary=summary(best_subset_method)
```

```
# Plot RSS, Adjusted R2, Cp and BIC
```

```
par(mfrow=c(2,2))  
plot(best_subset_method_summary$rss,xlab="Number of Variables",ylab="RSS",type="l")  
plot(best_subset_method_summary$adjr2,xlab="Number of Variables",ylab="Adjusted RSq",type="l")  
bsm_best_adjr2 = which.max(best_subset_method_summary$adjr2)  
  
points(bsm_best_adjr2,best_subset_method_summary$adjr2[bsm_best_adjr2], col="red",cex=2,pch=20)  
plot(best_subset_method_summary$cp,xlab="Number of Variables",ylab="Cp",type='l')  
bsm_best_cp=which.min(best_subset_method_summary$cp)
```

```
points(bsm_best_cp,best_subset_method_summary$cp[bsm_best_cp],col="red",cex=2,pch=20)
bsm_best_bic=which.min(best_subset_method_summary$bic)

plot(best_subset_method_summary$bic,xlab="Number of Variables",ylab="BIC",type='l')
points(bsm_best_bic,best_subset_method_summary$bic[bsm_best_bic],col="red",cex=2,pch=20)
```



```
# 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
}

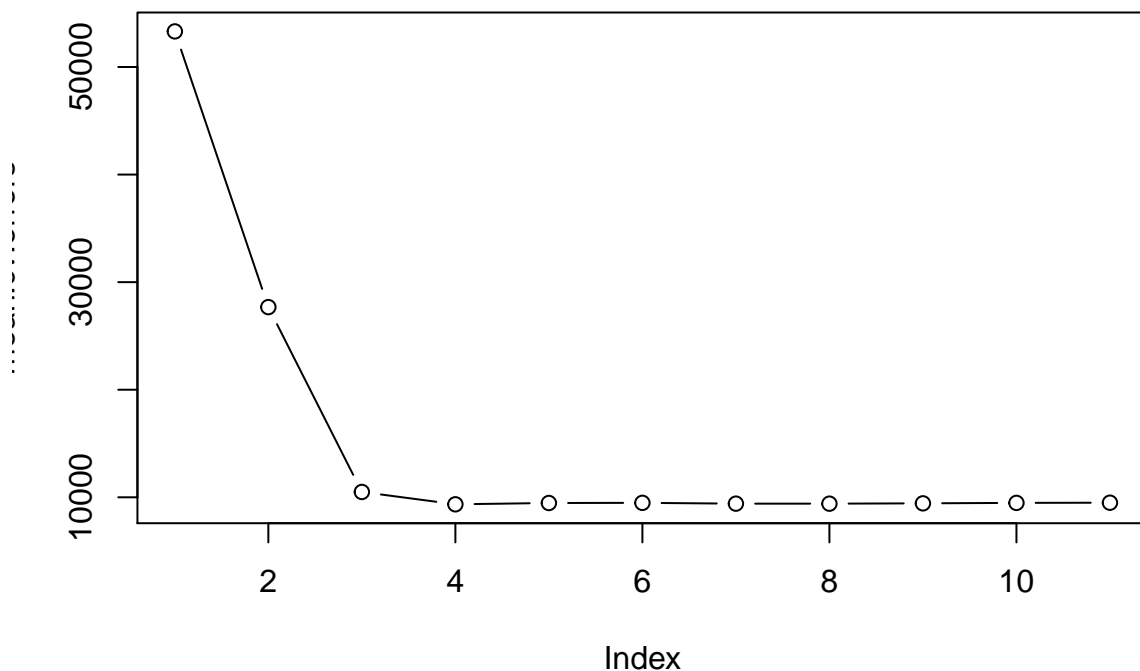
# Create indexes to divide the data between folds
k=10
set.seed(1)
folds=sample(1:k,nrow(credit_data_training),replace=TRUE)
cv.errors=matrix(NA,k,credit_data_number_predictors, dimnames=list(NULL, paste(1:credit_data_number_predictors)))

# Perform CV
for(j in 1:k){
  best_subset_method=regsubsets(Balance~.,data=credit_data_training[folds!=j,],nvmax=credit_data_number_predictors)
  for(i in 1:credit_data_number_predictors){
    pred=predict(best_subset_method,credit_data_training[folds==j,],id=i)
    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)
mean.cv.errors
```

```
##          1          2          3          4          5          6          7
## 53308.978 27681.063 10497.276 9349.190 9468.743 9484.566 9410.272
##          8          9         10         11
## 9409.024 9437.443 9480.517 9496.783
```

```
# Plot the mean cv errors
par(mfrow=c(1,1))
plot(mean.cv.errors,type='b')
```



```
# Fit the selected model using the whole training data
# and compute test error
```

```
# models selected
```

```
number_predictors_selected <- 4
```

```
# Create info for lm call
```

```
variables <- names(coef(best_subset_method,id=number_predictors_selected))
```

```
variables <- variables[!variables %in% "(Intercept)"]
```

```
bsm_formula <- as.formula(best_subset_method$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

# Make predictions on the test set
bsm_design_matrix_test <- model.matrix(bsm_formula,credit_data_testing)[, variables]
bsm_predictions <- predict(object = model_best_subset_method, newdata = as.data.frame(bsm_design_matrix_test))

# Compute test squared errors
bsm_squared_errors <- (credit_data_testing$Balance-bsm_predictions)^2
squared_errors <- data.frame(bsm_squared_errors=bsm_squared_errors)

# test MSE
mean(bsm_squared_errors)

## [1] 12243.75
```

Recommended exercise 4

Similar analysis as previous exercise, simply replace Best Subset Selection (`best_subset_method=regsubsets(Balance~.,credit_data,nvmax=credit_data_number_predictors)`) by Forward Stepwise Selection (`regfit.fwd=regsubsets(Balance~.,credit_data,nvmax=credit_data_number_predictors)`), Backward Stepwise Selection (`regfit.bwd=regsubsets(Balance~.,credit_data,nvmax=credit_data_number_predictors)`) and Hybrid Stepwise Selection (`regfit.fwd=regsubsets(Balance~.,credit_data,nvmax=credit_data_number_predictors)`).

Recommended exercise 5

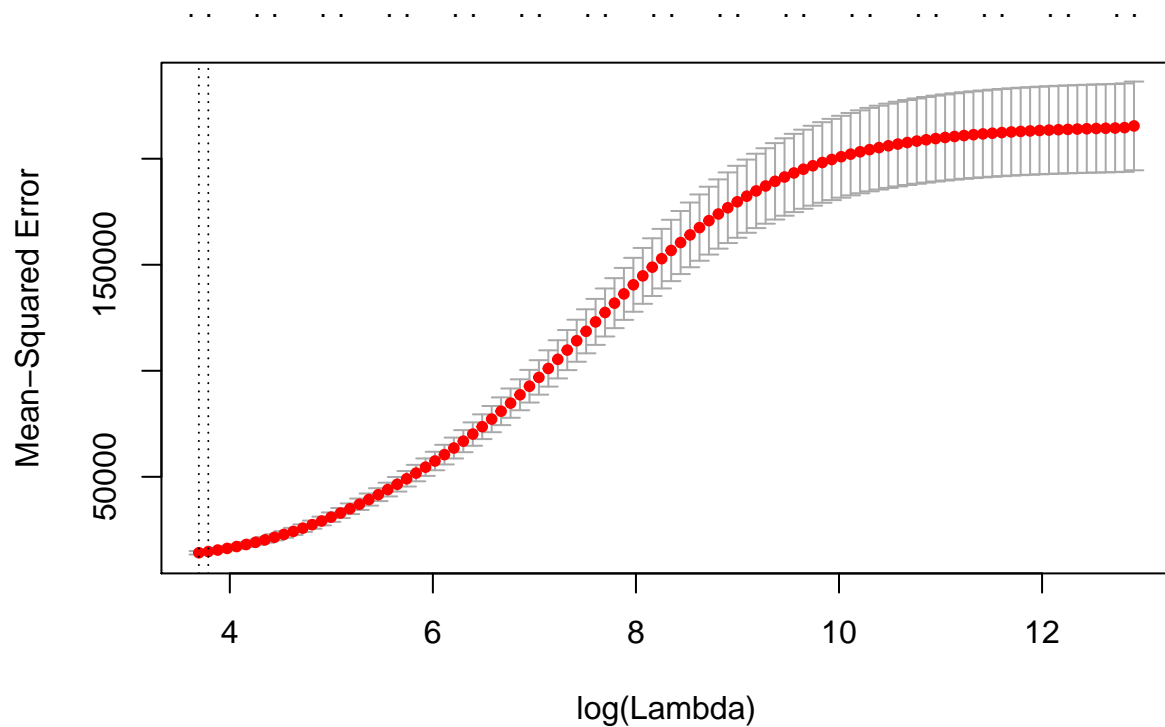
```
library(glmnet) # Package Lasso and Elastic-Net Regularized
                # Generalized Linear Models

x_train <- model.matrix(Balance~.,credit_data_training)[,-1]
y_train <- credit_data_training$Balance

x_test  <- model.matrix(Balance~.,credit_data_testing)[,-1]
y_test  <- credit_data_testing$Balance

ridge_mod <- glmnet(x_train,y_train,alpha=0)
```

```
set.seed(1)
cv.out=cv.glmnet(x_train, y_train,alpha=0)
plot(cv.out)
```



```
best_lambda_ridge <- cv.out$lambda.min
best_lambda_ridge
```

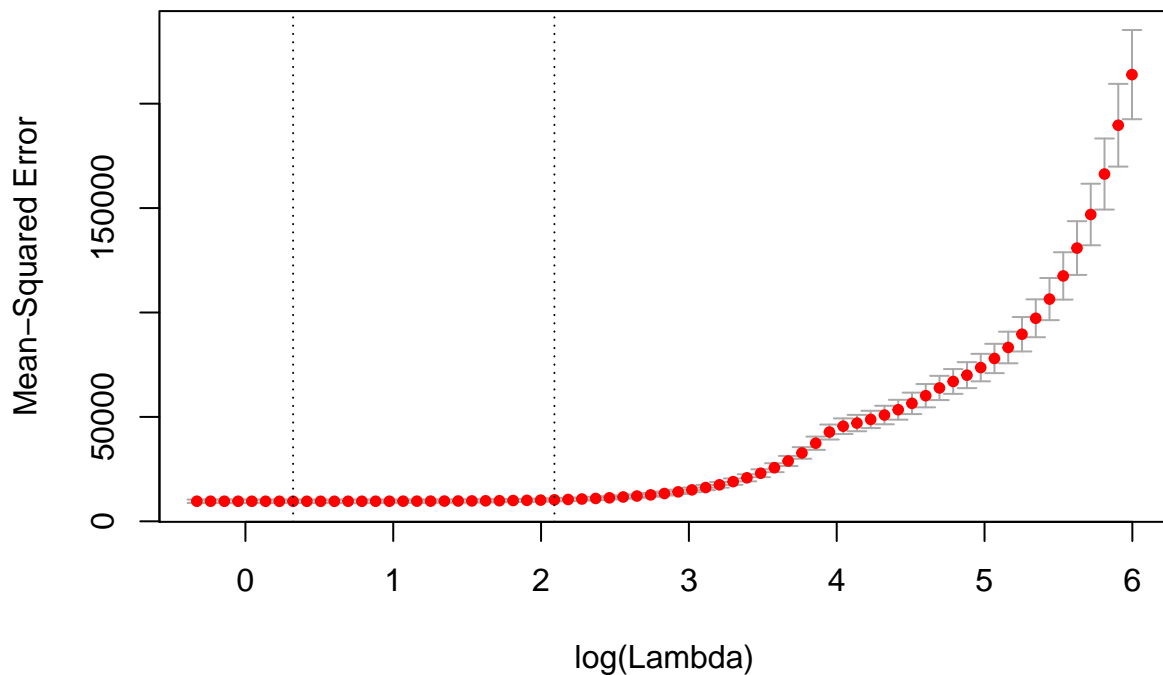
```
## [1] 40.24862
```

```
ridge_predictions = predict(ridge_mod,s=best_lambda_ridge,newx=x_test)
ridge_square_errors <- as.numeric((ridge_predictions-y_test)^2)
squared_errors <- data.frame(ridge_square_errors = ridge_square_errors, squared_errors)
```

Recommended exercise 6

```
lasso_mod <- glmnet(x_train,y_train,alpha=1)

set.seed(1)
cv.out=cv.glmnet(x_train, y_train,alpha=1)
plot(cv.out)
```

```
best_lambda_lasso <- cv.out$lambda.min
best_lambda_lasso

## [1] 1.380717

lasso_predictions = predict(lasso_mod,s=best_lambda_lasso,newx=x_test)
lasso_square_errors <- as.numeric((lasso_predictions-y_test)^2)
squared_errors <- data.frame(lasso_square_errors = lasso_square_errors, squared_errors)
```

Recommended exercise 7

```
x <- model.matrix(Balance~.,credit_data)[,-1]

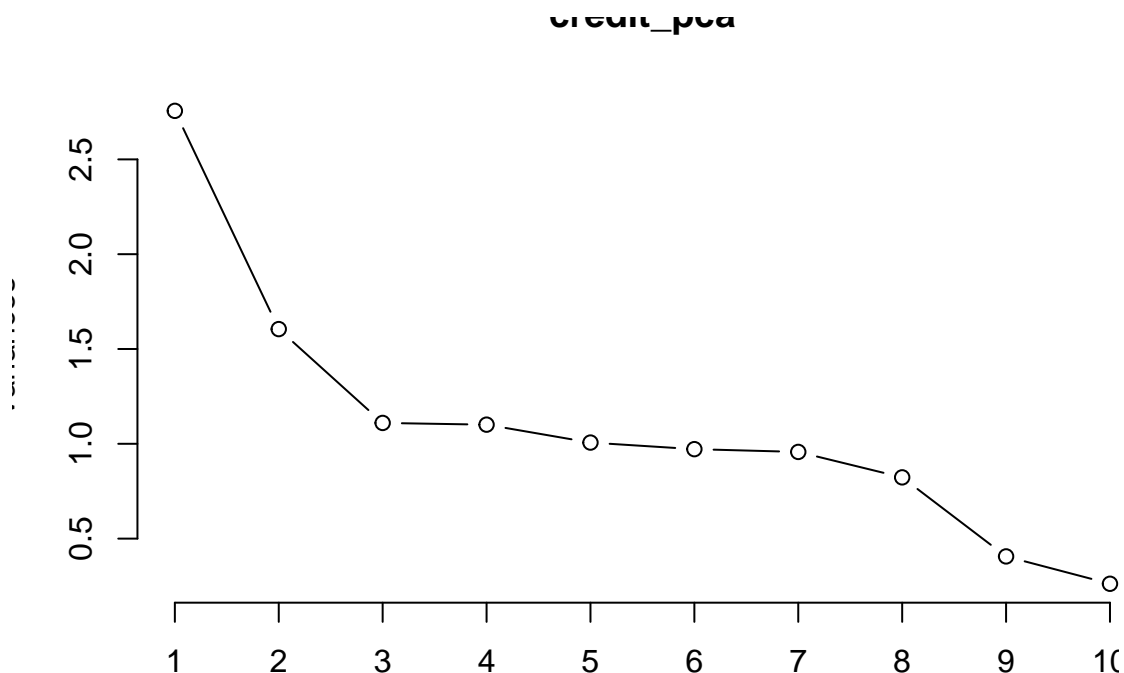
credit_pca <- prcomp(x, center = TRUE, scale. = TRUE)

print(credit_pca)

## Standard deviations (1, .., p=11):
## [1] 1.66007642 1.26685832 1.05356810 1.04926273 1.00322222 0.98576693
## [7] 0.97830708 0.90714714 0.63722533 0.51174012 0.04617646
##
## Rotation (n x k) = (11 x 11):
##
##          PC1          PC2          PC3          PC4
## Income    -0.542206953    0.029036783   -0.033270648   -6.564051e-05
## Limit     -0.586332930    0.017502630   -0.024351723    4.678929e-02
## Rating    -0.586751867    0.014971105   -0.004630758    3.687909e-02
## Cards     -0.019086978   -0.008549632    0.479005750   -2.720228e-01
## Age       -0.122783390   -0.071116603    0.107188498   -4.787335e-01
## Education  0.026797471    0.096557225   -0.475418336    1.990653e-01
## GenderFemale -0.002519860    0.052811098   -0.334014058   -4.207748e-02
```

```
## StudentYes      0.002276904  0.125422970 -0.618650527 -2.963169e-01
## MarriedYes      -0.026218561  0.094278214  0.125718135  7.389864e-01
## EthnicityAsian   0.032769895  0.696759512  0.105703127  6.686132e-03
## EthnicityCaucasian -0.004070799 -0.686505857 -0.100240068  1.338718e-01
##                PC5          PC6          PC7          PC8
## Income          -0.02816858  0.02297156 -0.04086888  0.03502243
## Limit           0.02393728  0.06109959  0.02753603 -0.07998103
## Rating          0.03044748  0.04901285  0.06298342 -0.07474080
## Cards           0.07450235 -0.28313105  0.77070237 -0.10917776
## Age            -0.29468570 -0.58353604 -0.35860755  0.41270188
## Education       -0.58335540 -0.40244676  0.21601791 -0.41794930
## GenderFemale    0.74620452 -0.51375214 -0.10203846 -0.22746095
## StudentYes      0.05874438  0.20236658  0.42777847  0.53366278
## MarriedYes      0.04850438 -0.32419986  0.13571418  0.53676497
## EthnicityAsian  0.02125450  0.01284830 -0.04334986  0.01824866
## EthnicityCaucasian 0.04400214 -0.02306227  0.10322555  0.06987098
##                PC9          PC10         PC11
## Income          -0.016018928  0.836411394  0.0017092799
## Limit           -0.010697575 -0.379489022  0.7053633132
## Rating          -0.005366527 -0.373834509 -0.7081335719
## Cards           0.005357720  0.059511066  0.0305564113
## Age            -0.048994454 -0.102540342  0.0005901693
## Education       -0.021973159  0.014172918 -0.0036133922
## GenderFemale    0.014513597  0.027300122  0.0001327203
## StudentYes      0.022068488 -0.032119354  0.0044219212
## MarriedYes      0.119017609 -0.018248384  0.0051766487
## EthnicityAsian  -0.706522468 -0.014783578 -0.0035849536
## EthnicityCaucasian -0.694731116  0.008145839 -0.0004464620
```

```
plot(credit_pca, type = "l")
```



```
summary(credit_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
```

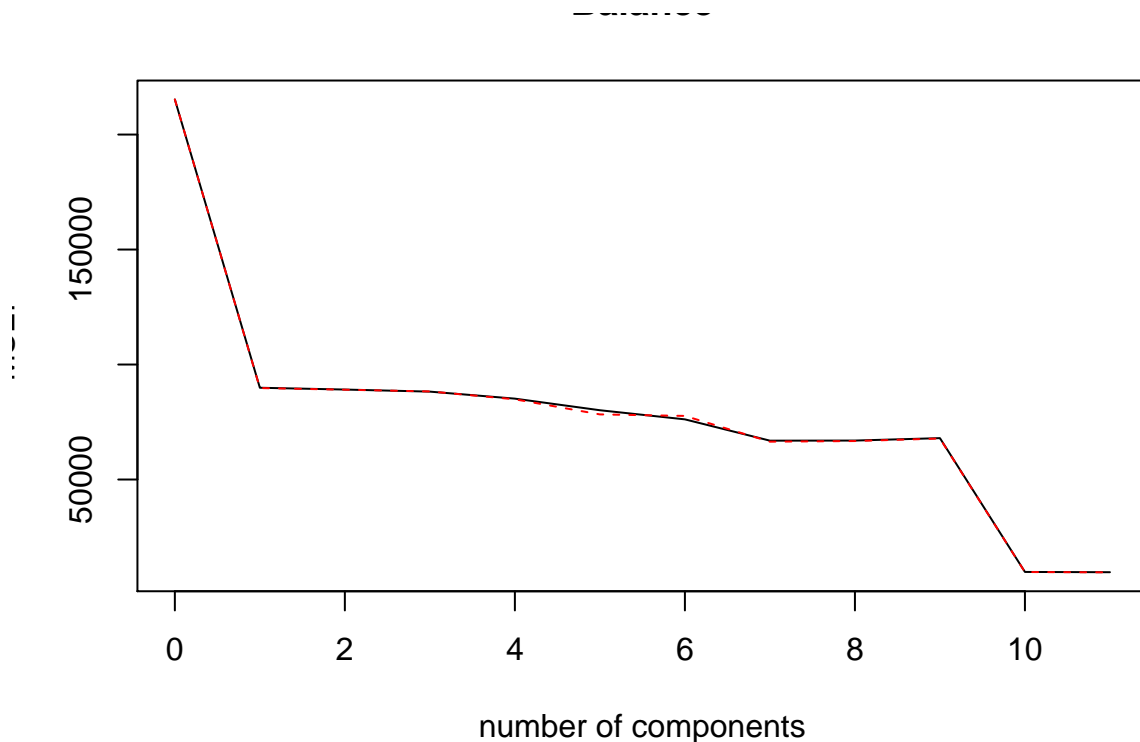
The first PC explain along 25% of the variability in the data. Then the second PC explain an extra 15% of the variability in the data. From the third PC until 8th PC the extra variability explained per PC varies between 7.5% to 10%, dropping to 3.6% on the 9th PCA. So I would likely use 8 PCs for the Credit dataset.

Recommended exercise 8

```
library(pls)

set.seed(1)

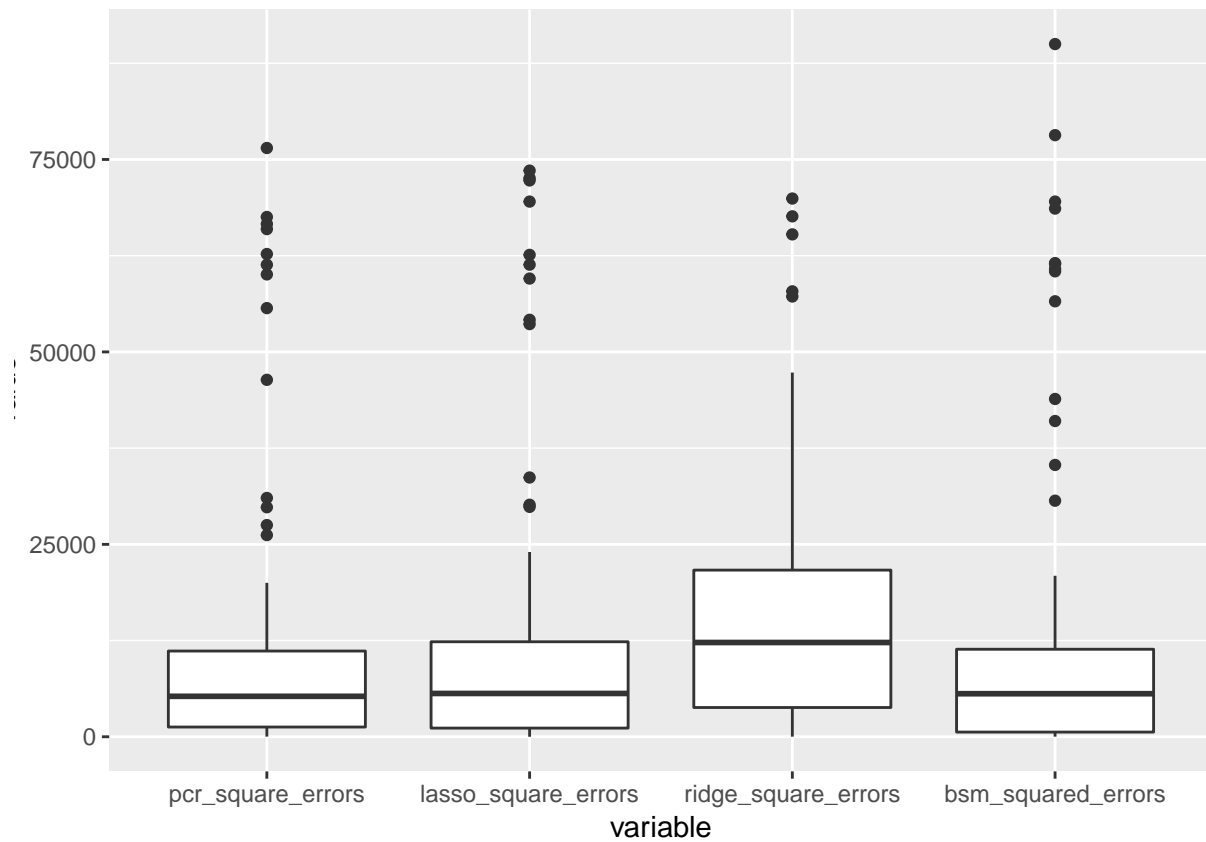
pcr_model <- pcr(Balance~., data=credit_data_training,scale=TRUE, validation="CV")
validationplot(pcr_model,val.type="MSEP")
```



```
pcr_predictions = predict(pcr_model,credit_data_testing,ncomp=10)
pcr_square_errors <- as.numeric((pcr_predictions-credit_data_testing$Balance)^2)
squared_errors <- data.frame(pcr_square_errors = pcr_square_errors, squared_errors)
mean(pcr_square_errors)
```

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

```
library(ggplot2)
library(reshape2)
ggplot(melt(squared_errors)) + geom_boxplot(aes(variable, value))
```

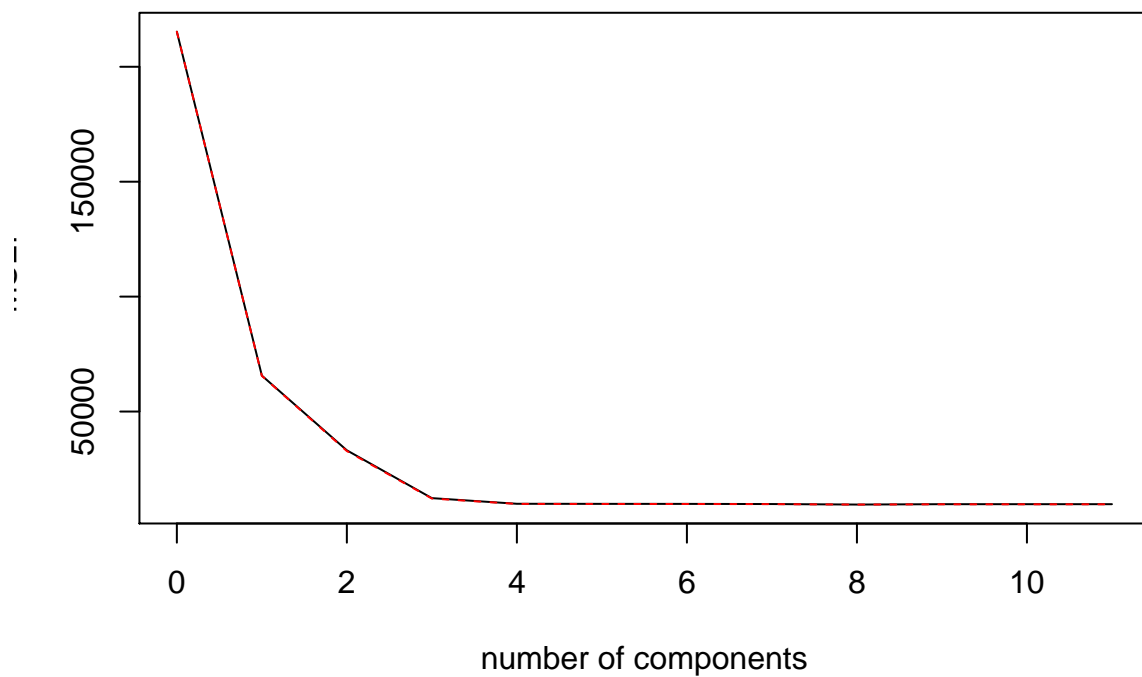


Recommended exercise 9

```
library(pls)

set.seed(1)

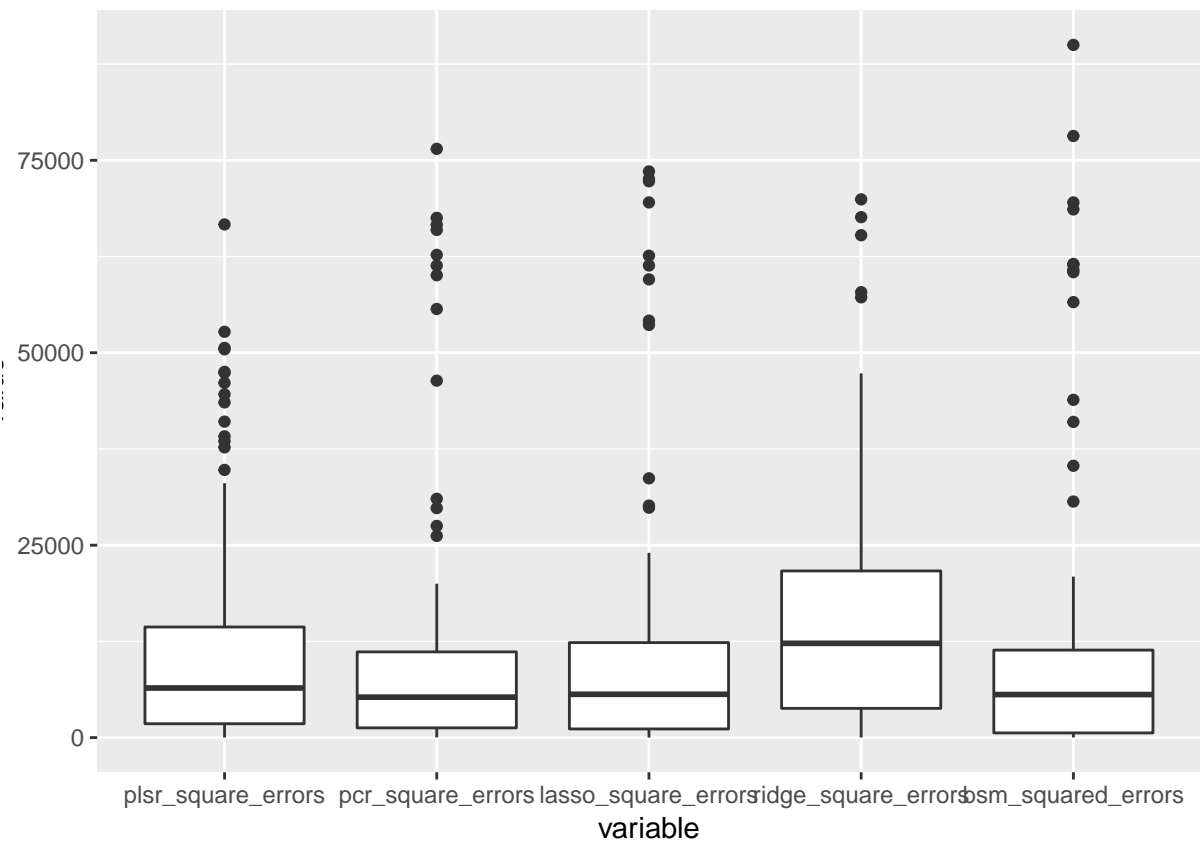
plsr_model <- plsr(Balance~., data=credit_data_training, scale=TRUE, validation="CV")
validationplot(plsr_model, val.type="MSEP")
```



```
plsr_predictions = predict(plsr_model,credit_data_testing,ncomp=3)
plsr_square_errors <- as.numeric((plsr_predictions-credit_data_testing$Balance)^2)
squared_errors <- data.frame(plsr_square_errors = plsr_square_errors, squared_errors)
mean(plsr_square_errors)
```

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

```
ggplot(melt(squared_errors)) + geom_boxplot(aes(variable, value))
```



```
colMeans(squared_errors)
```

```
## plsr_square_errors pcr_square_errors lasso_square_errors
##      12476.32      11578.10      12077.15
## ridge_square_errors bsm_squared_errors
##      15742.83      12243.75
```