Advances Computational Econometrics. Chapter 3: Model selection

Thierry Denoeux

4/11/2022

Exercise 1

Question 1

We start by loading the data:

```
movie<-read.csv('/Users/Thierry/Documents/R/Data/Economics/Greene/movie_buzz.csv',
                header=TRUE,sep=",")
```

We redefining some variables as factors:

```
movie$MPRATING<-as.factor(movie$MPRATING)
levels(movie$MPRATING)<-c("G","PG","PG13","R")
movie$SEQUEL<-as.factor(movie$SEQUEL)
movie$ACTION<-as.factor(movie$ACTION)
movie$COMEDY<-as.factor(movie$COMEDY)
movie$ANIMATED<-as.factor(movie$ANIMATED)
movie$HORROR<-as.factor(movie$HORROR)
```

We apply a logarithmic transformation to variables `BOX` and `BUDGET` to symmetrize their distributions:

```
movie$BOX<-log(movie$BOX)
movie$BUDGET<-log(movie$BUDGET)
```

We then create a design matrix `X`, which will be used later:

```
X<-model.matrix(BOX~.,data=movie)
```

Finally, we split the data randomly into training and test sets:

```
set.seed(11042022)
n<-nrow(movie)
ntrain<-40
ntest<-n-ntrain
train<-sample(n,ntrain)
movie.train<-movie[train,]
movie.test<-movie[-train,]
X.train<-X[train,]
X.test<-X[-train,]
```

Question 2

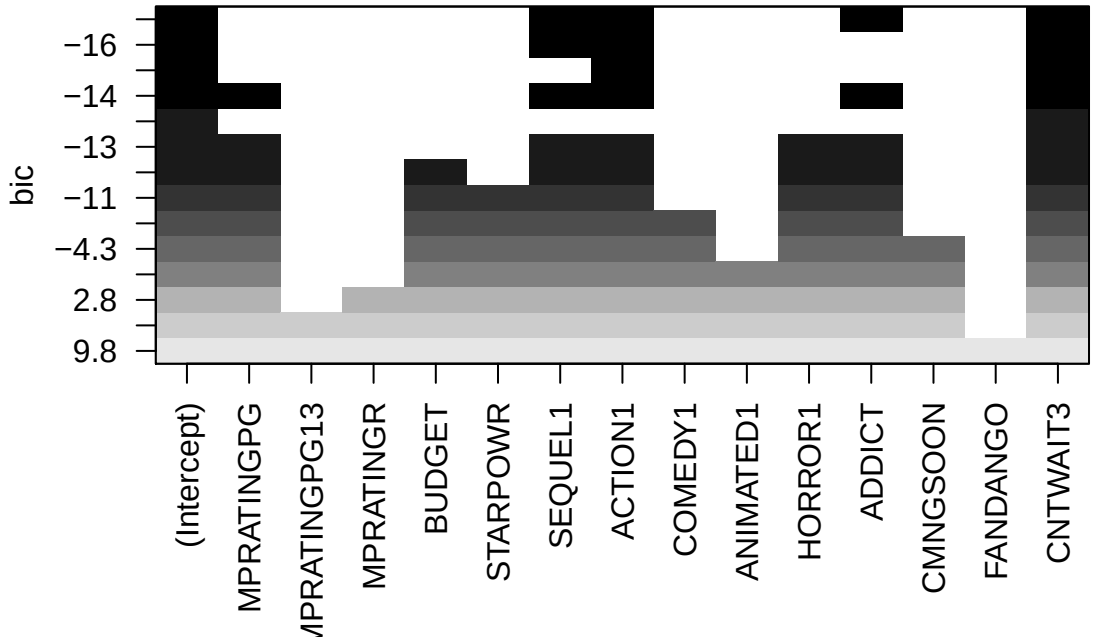
Subset selection methods

We first consider subset selection methods in package `leaps`. We start with forward stepwise selection:

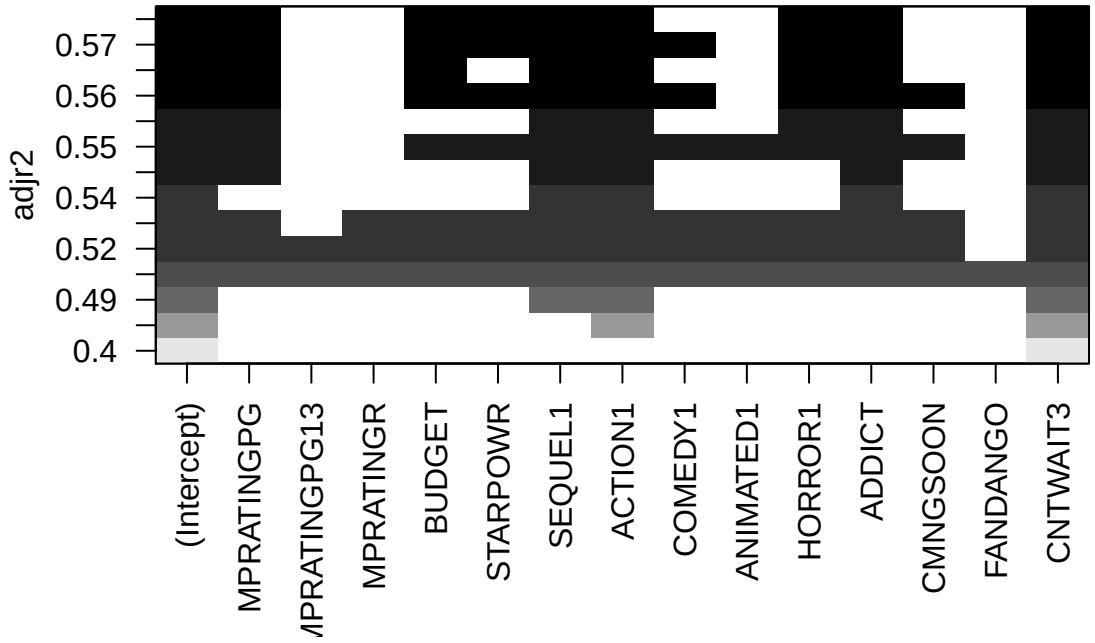
```
library("leaps")
```

```
## Warning: package 'leaps' was built under R version 4.0.2
```

```
reg.forward<-regsubsets(BOX~.,data=movie.train,method='forward',nvmax=15)
plot(reg.forward,scale="bic")
```




```
plot(reg.forward,scale="adjr2")
```



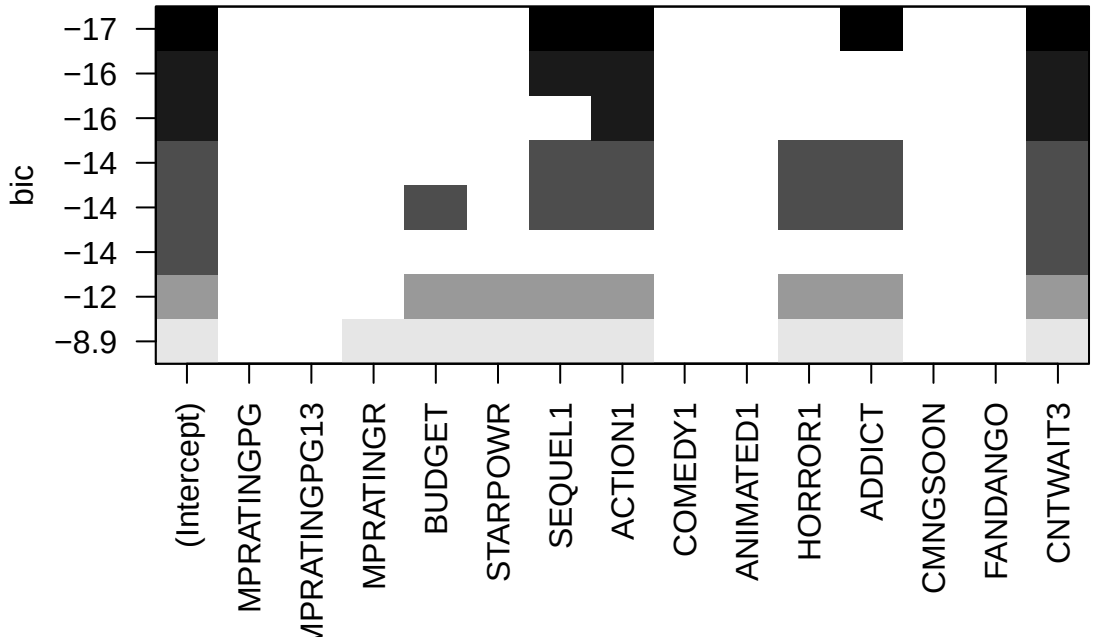
We compute the test MSE for the best models according to BIC and the adjusted R^2:

```
res<-summary(reg.forward)
best.bic<-which.min(res$bic) # Best model according to BIC
ypred<-X.test[,res$which[best.bic,]]%*%coef(reg.forward,best.bic)
mse_forward_bic<-mean((ypred-movie.test$BOX)^2)
best.adjr2<-which.max(res$adjr2) # Best model according to adjust. R2
ypred<-X.test[,res$which[best.adjr2,]]%*%coef(reg.forward,best.adjr2)
mse_forward_adjr2<-mean((ypred-movie.test$BOX)^2)
print(c(mse_forward_bic,mse_forward_adjr2))
```

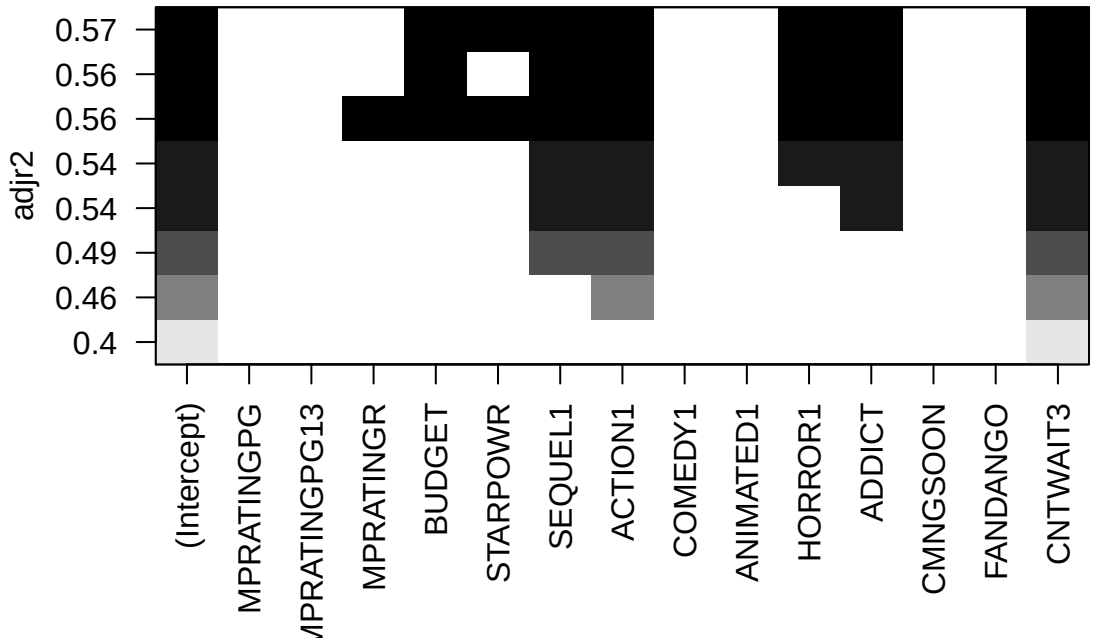
```
## [1] 0.6259606 0.7266837
```

We now use backward selection:

```
reg.backward<-regsubsets(BOX~.,data=movie.train,method='backward')
plot(reg.backward,scale="bic")
```




```
plot(reg.backward,scale="adjr2")
```



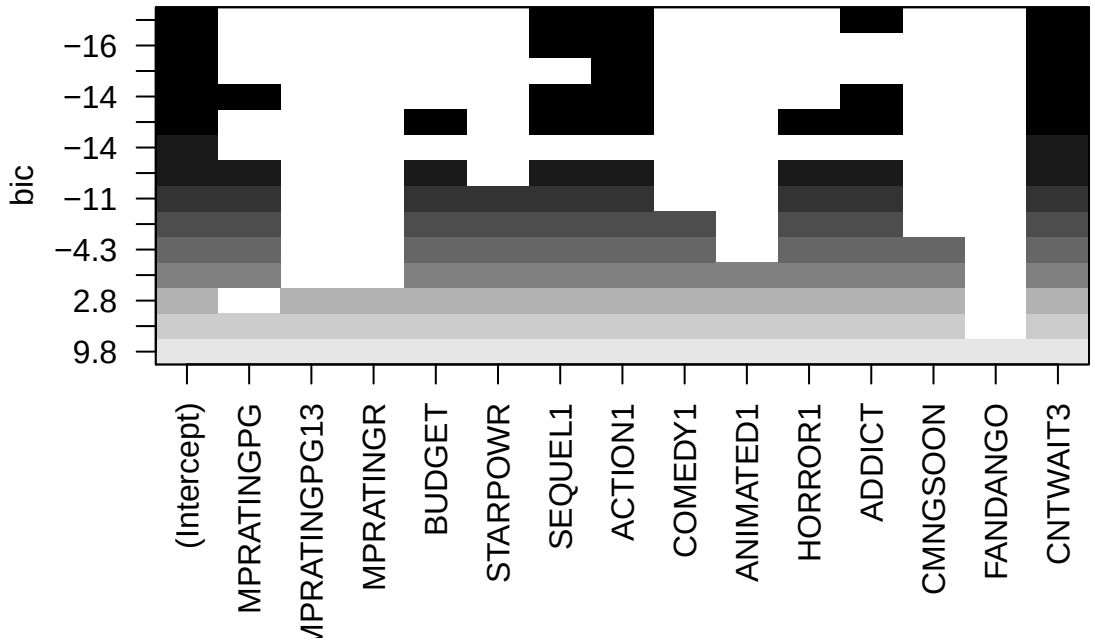
The best model according to BIC is identical to that found by forward selection, but according to the adjusted R^2 we have a different solution (with one less predictor). We compute its test MSE:

```
res<-summary(reg.backward)
best.adjr2<-which.max(res$adjr2)
ypred<-X.test[,res$which[best.adjr2,]]%*%coef(reg.backward,best.adjr2)
mse_backward_adjr2<-mean((ypred-movie.test$BOX)^2)
print(c(mse_forward_bic,mse_forward_adjr2,mse_backward_adjr2))
```

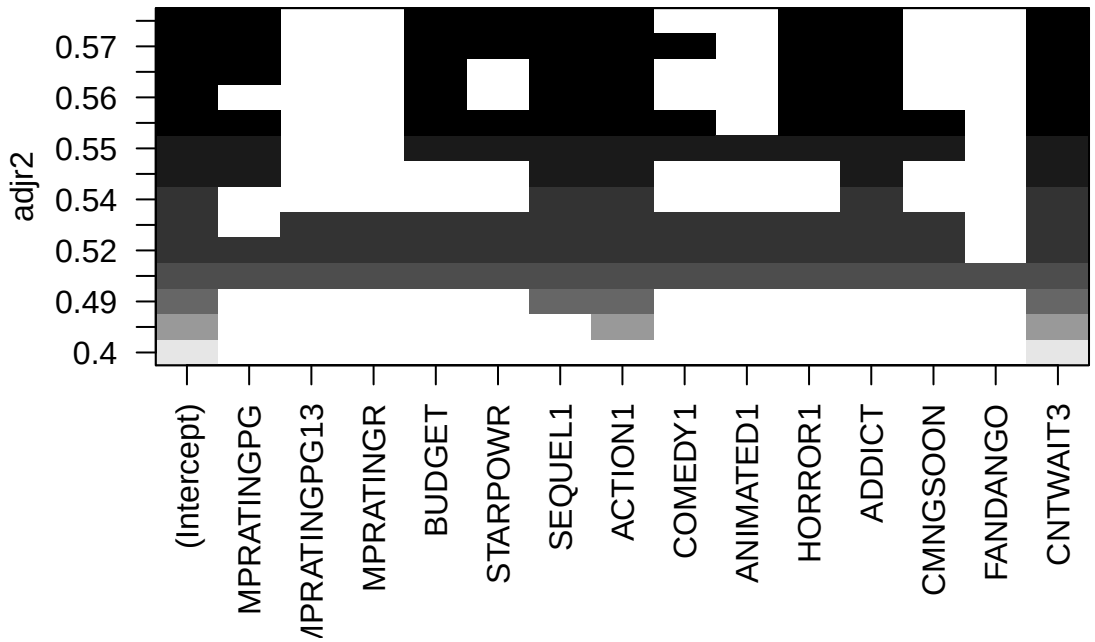
```
## [1] 0.6259606 0.7266837 0.6811740
```

Finally, we try the best subset approach (exhaustive search):

```
reg.exhaustive<-regsubsets(BOX~.,data=movie.train,method='exhaustive',nvmax=15)
plot(reg.exhaustive,scale="bic")
```




```
plot(reg.exhaustive,scale="adjr2")
```



We obtain the same solutions as those found by forward stepwise selection.

Regularization methods

For ridge, lasso and elastic net, we need the `glmnet` package:

```
library(glmnet)
```

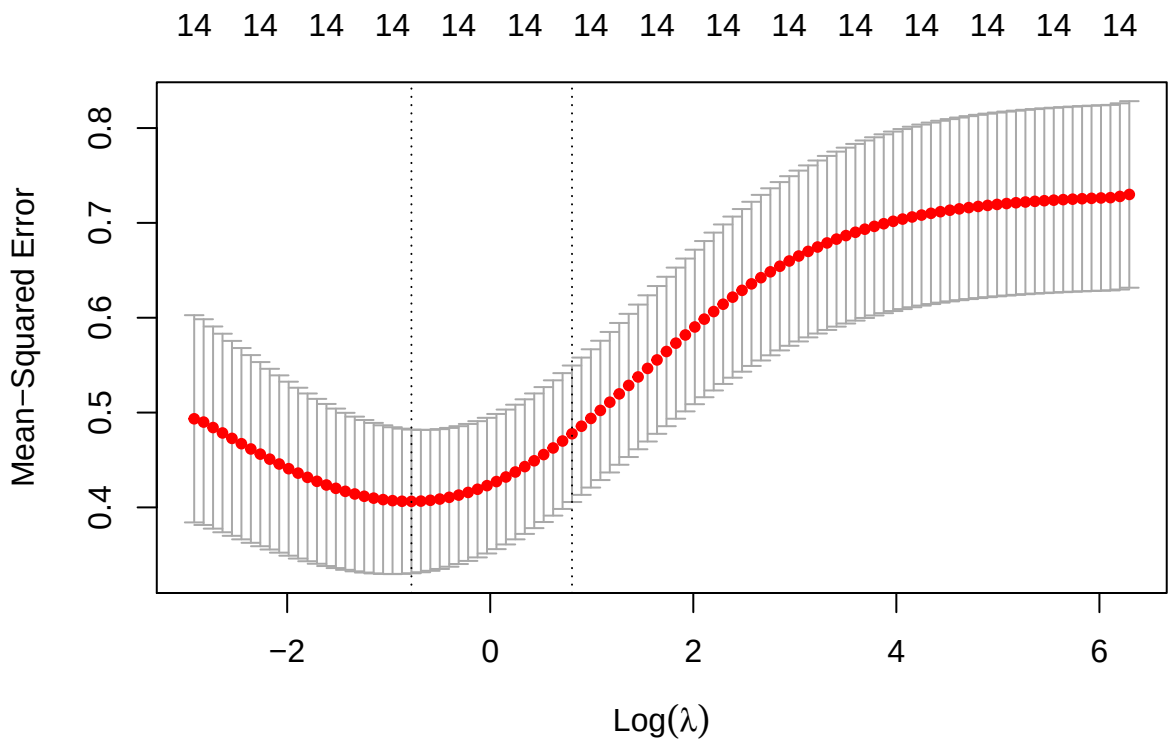
```
## Warning: package 'glmnet' was built under R version 4.0.2
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-1
```

Let us start with ridge regression. Function `cv.glmnet` computes the cross-validation MSE for different values of hyperparameter λ:

```
cv.out<-cv.glmnet(X.train[,-1],movie.train$BOX,alpha=0,standardize=TRUE)
plot(cv.out)
```



There are two methods to select a value of λ: the minimum cross-validation error method and the "one-standard-error" method.We start with the former method:

```
fit1<-glmnet(X.train[,-1],movie.train$BOX,lambda=cv.out$lambda.min,
             alpha=0,standardize=TRUE)
ridge.pred1<-predict(fit1,newx=X.test[,-1])
mse_ridge1<-mean((movie.test$BOX-ridge.pred1)^2)
```

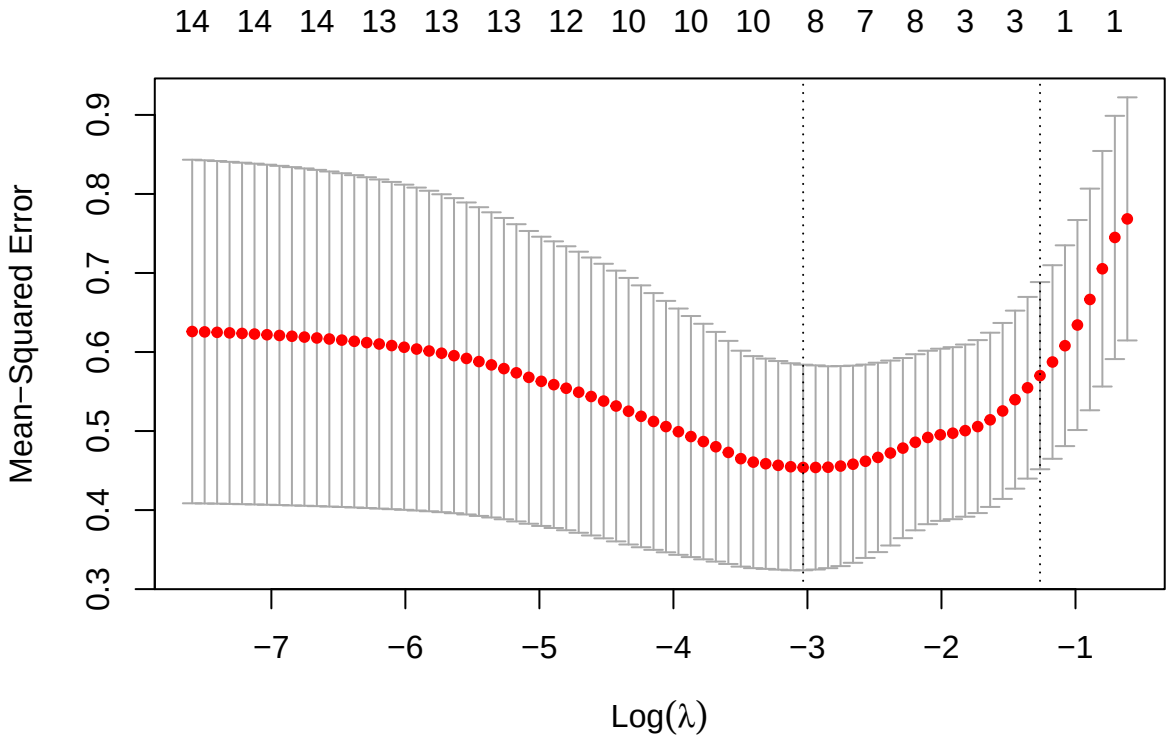
Now, we use the 1-se rule:

```
fit2<-glmnet(X.train[,-1],movie.train$BOX,lambda=cv.out$lambda.1se,
             alpha=0,standardize=TRUE)
ridge.pred2<-predict(fit2,newx=X.test[,-1])
mse_ridge2<-mean((movie.test$BOX-ridge.pred2)^2)
print(c(mse_ridge1,mse_ridge2))
```

```
## [1] 0.8081093 0.9088157
```

We observe that ridge regression does not peroform as well as subset selection. Let us now try the lasso, using again the two model methods for tuning the regularization coefficient:

```
cv.out<-cv.glmnet(X.train[,-1],movie.train$BOX,alpha=1,standardize=TRUE)
plot(cv.out)
```




```
fit1<-glmnet(X.train[,-1],movie.train$BOX,lambda=cv.out$lambda.min,
             alpha=1,standardize=TRUE)
fit2<-glmnet(X.train[,-1],movie.train$BOX,lambda=cv.out$lambda.1se,
             alpha=1,standardize=TRUE)
lasso.pred1<-predict(fit1,newx=X.test[,-1])
mse_lasso1<-mean((movie.test$BOX-lasso.pred1)^2)
lasso.pred2<-predict(fit2,newx=X.test[,-1])
mse_lasso2<-mean((movie.test$BOX-lasso.pred2)^2)
print(c(mse_forward_bic,mse_forward_adjr2,mse_backward_adjr2,
        mse_ridge1,mse_ridge2, mse_lasso1,mse_lasso2))
```

```
## [1] 0.6259606 0.7266837 0.6811740 0.8081093 0.9088157 0.6687930 0.8306950
```

Lasso with minimum CV error performs better than ridge, as similarly to subset selection. Printing the coefficients will allow us to see which predictors have been selected:

```
print(fit1$beta)
```

```
## 14 x 1 sparse Matrix of class "dgCMatrix"
##                            s0
## MPRATINGPG     1.781524e-01
## MPRATINGPG13   .
## MPRATINGR      .
## BUDGET         1.819911e-01
## STARPOWR      -2.671682e-03
## SEQUEL1        4.633897e-01
## ACTION1       -3.476794e-01
## COMEDY1        .
## ANIMATED1      .
## HORROR1        3.543641e-01
## ADDICT         3.268191e-05
## CMNGSOON       .
```

```
## FANDANGO      .
## CNTWAIT3      2.249277e+00
```

Question 2

We now no longer split the data into a training set and a test set. Instead, we will estimate prediction error by corss-validation. In the case of penalization, we will thus implement nested corss-validation. Here, we will only apply the method for the two best methods found previously, i.e., forward selection with BIC, and the Lasso with minimum CV error. We start by randomly selecting the $K = 10$ subsets:

```
K<-10
folds=sample(1:K,n,replace=TRUE)
```

We then compute the cross-validation MSE for forward subset selection with BIC:

```
CV_FS<-0
for(i in (1:K)){
  reg.forward<-regsubsets(BOX~.,data=movie[folds!=i,],method='forward',nvmax=15)
  res<-summary(reg.forward)
  best.bic<-which.min(res$bic)
  ypred<-X[folds==i,res$which[best.bic,]]%*%coef(reg.forward,best.bic)
  mse_forward_bic<-mean((ypred-movie$BOX[folds==i])^2)
  CV_FS<-CV_FS+ sum((ypred-movie$BOX[folds==i])^2)
  }
CV_FS<-CV_FS/n
print(CV_FS)
```

```
## [1] 0.5635497
```

We now apply the method to Lasso regularization. Here, tehre will be two nested cross-validation loop (one of which is performed inside function `cv.glmnet`):

```
CV_Lasso<-0
for(i in (1:K)){
  cv.out<-cv.glmnet(X[folds!=i,-1],movie$BOX[folds!=i],alpha=1,standardize=TRUE)
  fit<-glmnet(X[folds!=i,-1],movie$BOX[folds!=i],lambda=cv.out$lambda.min,
              alpha=1,standardize=TRUE)
  lasso.pred<-predict(fit,newx=X[folds==i,-1])
  CV_Lasso<-CV_Lasso+ sum((movie$BOX[folds==i]-lasso.pred)^2)
}
CV_Lasso<-CV_Lasso/n
print(CV_Lasso)
```

```
## [1] 0.5619631
```

We can see again that both methods have similar performances on this dataset.

Exercise 2

Question 1

We start by reading the data and declaring the factors:

```
credit<-read.csv('/Users/Thierry/Documents/R/Data/Economics/default_credit_card.csv',
                 sep=";",header=TRUE)
```

```
credit$X2<-as.factor(credit$X2)
levels(credit$X2)<-c("M","F")
credit$X3<-as.factor(credit$X3)
credit$X4<-as.factor(credit$X3)
credit$Y<-as.factor(credit$Y)
```

We then split the data into training and test sets:

```
n<-nrow(credit)
ntrain<-20000
ntest<-n-ntrain
train<-sample(n,ntrain)
credit.train<-credit[train,]
credit.test<-credit[-train,]
```

Question 2

We will need packages `MASS` and `naivebayes`:

```
library(MASS)
library(naivebayes)
```

```
## naivebayes 0.9.7 loaded
```

We randomly select the subsets for 10$fold cross-validation:

```
K<-10
folds=sample(1:K,ntrain,replace=TRUE)
CV<-matrix(0,K,4)
```

When then compute the CV error rates of the four models:

```
for(k in (1:K)){
  fit.lda<- lda(Y~.,data=credit.train[folds!=k,-c(2,3,4)])
  pred.lda<-predict(fit.lda,newdata=credit.train[folds==k,])
  CV[k,1]<-sum(pred.lda$class!=credit.train$Y[folds==k])
  fit.qda<- qda(Y~.,data=credit.train[folds!=k,-c(2,3,4)])
  pred.qda<-predict(fit.qda,newdata=credit.train[folds==k,])
  CV[k,2]<-sum(pred.qda$class!=credit.train$Y[folds==k])
  fit.nb<- naive_bayes(Y~.,data=credit.train[folds!=k,])
  pred.nb<-predict(fit.nb,newdata=credit.train[folds==k,],type="class")
  CV[k,3]<-sum(pred.nb!=credit.train$Y[folds==k])
  fit.logreg<- glm(Y~.,data=credit.train[folds!=k,],family=binomial)
  pred.logreg<-predict(fit.logreg,newdata=credit.train[folds==k,],type='response')
  CV[k,4]<-sum(as.factor(as.numeric(pred.logreg>0.5))!=credit.train$Y[folds==k])
}
err_cv=colSums(CV)/ntrain
print(err_cv)
```

```
## [1] 0.18815 0.43460 0.26970 0.18910
```

LDA and logistic regression have similar CV error rates.

Question 3

We first fit LDA on the whole training set:

```
fit.lda<- lda(Y~.,data=credit.train[,-c(2,3,4)])
```

We then compute the test error rate for this model:

```
pred.lda<-predict(fit.lda,newdata=credit.test)
err.lda<-mean(pred.lda$class!=credit.test$Y)
print(err.lda)
```

```
## [1] 0.1919
```