Exercise 2

Konstantinos Vakalopoulos 12223236

2022-11-10

Premilinary work

First, the ISLR package was installed

```
install.packages("ISLR")
```

and then the data was loaded with.

```
library("ISLR")
data(College,package="ISLR")
```

For addition information about the data, the following commands were used.

```
?College
str(College)
```

The observations which contain missing values and the variables Accept and Enroll were removed from the original data.

```
College <- na.omit(College) #Check for NA observations
College <- College[,-c(3,4)] #Remove the Accept and Enroll variables
```

The values of the Apps variable has been changed to log transformed.

```
College$Apps <- log(College$Apps) #log transformation
```

Finally, the data was split randomly into training and test data (about 2/3 and 1/3).

```
#Split the data into train and test
set.seed(12223236)
n <- nrow(College)
train <- sample(1:n,round(n*2/3))
test <- (1:n)[-train]
```

Question 1

First, the cvTools package was installed.

```
install.packages("cvTools")
```

The full model from the previous exercise was created with the training set.

```
fit <- lm(Apps~., data = College, subset = train)
```

Afterwards, for the model evaluation, the library cvTools was introduced and the function cvFit() was used in order to be implemented the cross validation. The parameters was set as follow: cost=rmspe, K=5, R=100, seed=12223236. K equals to 5 indicates the 5-fold cross validation and R equals to 100 the number of replications.

```
library(cvTools)
```

```
cross.validation.fit <- cvFit(fit, data=College[train,],
                        y=College[train,]$Apps, cost=rmspe,
                        K=5,R=100,seed=12223236)
cross.validation.fit
```

```
## 5-fold CV results:
##        CV
## 0.5686902
```

In figures 1 and 2 is shown the boxplot and the distribution of the resulting error measures, respectively.

```
plot(cross.validation.fit, method = "bw")
```

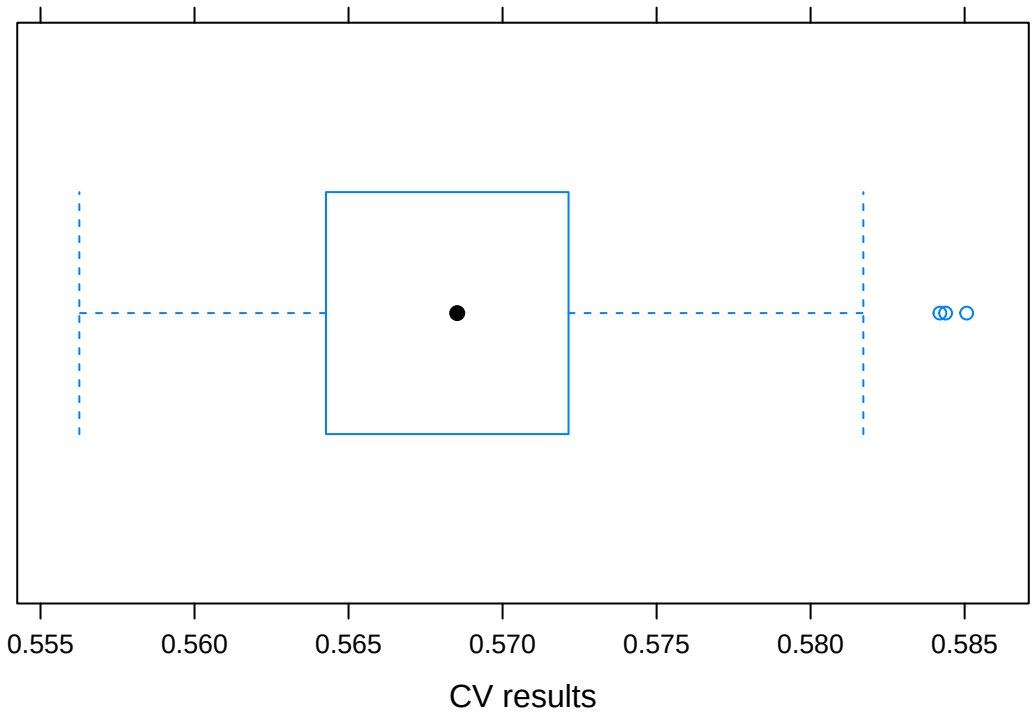

Figure 1: Cross Validation results

```
plot(cross.validation.fit, method = "density")
```

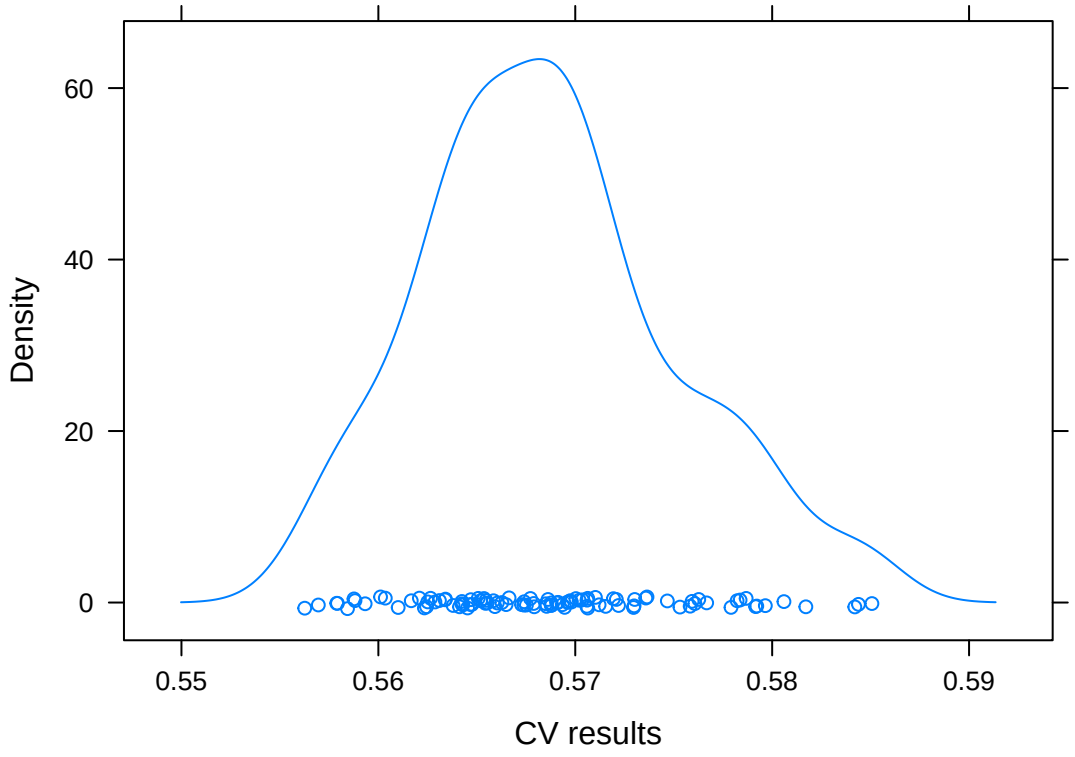

Figure 2: Cross Validation distribution

According to the boxplot, the error measures from the cross validation range between 0.556 and 0.585 and the mean value of all 0.569. All the results are approximately the same and that could be seen in the density plot. The results are normally distributed.

Question 2(a)

In this part of the exercise the best subset regression was implemented. In R the library(leaps) and the function regsubsets() was used. To find the best 3 models of each size the parameter nbest was set to 3 and for the model size of 10 regressors, the parameter nvmax was set to 10.

```
library(leaps)
sub.reg <- regsubsets(College$Apps~., data=College,  subset=train,
                          nbest=3,nvmax=10)
```

Question 2(b)

In figure 3, the results from the best subset regression are presented.

```
plot(sub.reg)
```

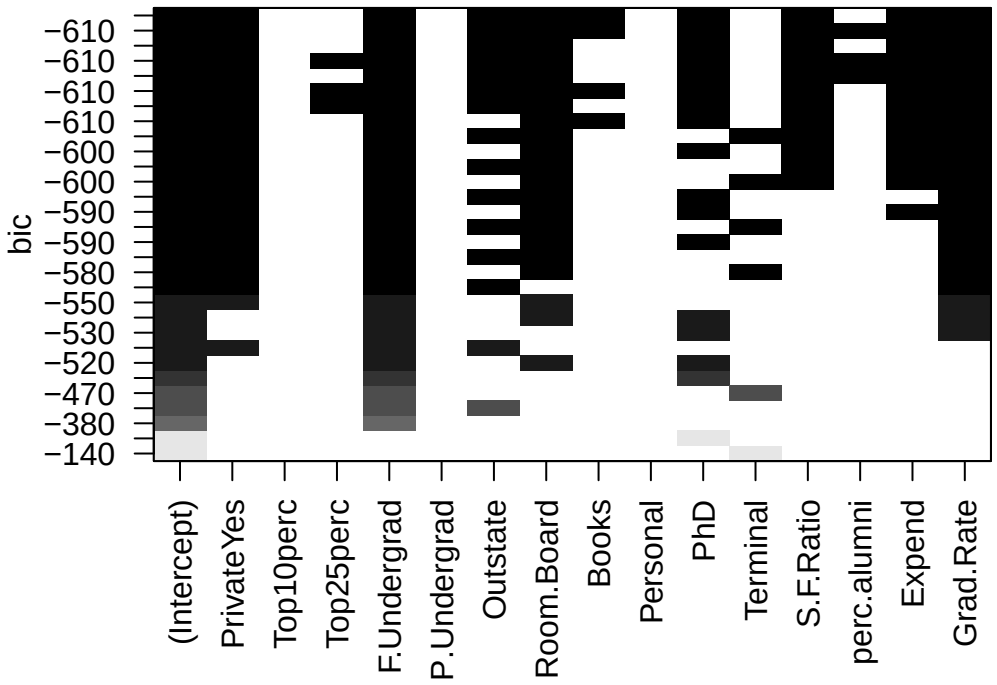

Figure 3: BIC values

The plot describes the reduced models and their BIC values. The BIC values are not linear presented in the plot. The optimal model is chosen from the black part of the plot with the lower BIC, in our case approximately -610, and Simultaneously has the smallest number of variables. Thus the optimal model, according to the plot, has 7 variables and these variables are: "Private", "F.Undergrad", "Room.Board", "PhD", "S.F.ratio", "Expend" and "Grad.Rate".

Question 2(c)

First the resulting summary was saved as another object and then using the function str(), the structure of this object was displayed.

```
sub.summary <- summary(sub.reg)
BIC <- sub.summary$bic
str(sub.summary)
```

```
## List of 8
```

```
##  $ which : logi [1:30, 1:16] TRUE TRUE TRUE TRUE TRUE TRUE ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:30] "1" "1" "1" "2" ...
##   .. ..$ : chr [1:16] "(Intercept)" "PrivateYes" "Top10perc" "Top25perc" ...
##  $ rsq   : num [1:30] 0.529 0.267 0.251 0.618 0.609 ...
##  $ rss   : num [1:30] 267 416 425 217 222 ...
##  $ adjr2 : num [1:30] 0.528 0.265 0.249 0.616 0.607 ...
##  $ cp    : num [1:30] 376 872 902 210 227 ...
##  $ bic   : num [1:30] -377 -148 -137 -479 -467 ...
##  $ outmat: chr [1:30, 1:15] " " " " " " " " " " " " ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:30] "1  ( 1 )" "1  ( 2 )" "1  ( 3 )" "2  ( 1 )" ...
##   .. ..$ : chr [1:15] "PrivateYes" "Top10perc" "Top25perc" "F.Undergrad" ...
##  $ obj   :List of 28
##   ..$ np       : int 16
##   ..$ nrbar    : int 120
##   ..$ d        : num [1:16] 5.18e+02 1.54e+05 6.30e+04 1.06e+10 4.02e+04 ...
##   ..$ rbar     : num [1:120] 27.2 22.5 3608.7 55.6 4360.5 ...
##   ..$ thetab   : num [1:16] 7.390204 0.020573 -0.022489 0.000151 0.009133 ...
##   ..$ first    : int 2
##   ..$ last     : int 16
##   ..$ vorder   : int [1:16] 1 3 14 5 4 8 16 12 15 9 ...
##   ..$ tol      : num [1:16] 1.14e-08 5.35e-07 5.98e-07 1.32e-04 9.54e-07 ...
##   ..$ rss      : num [1:16] 567 502 470 228 225 ...
##   ..$ bound    : num [1:16] 1.00e+35 4.25e+02 2.23e+02 1.97e+02 1.86e+02 ...
##   ..$ nvmax    : int 11
##   ..$ ress     : num [1:11, 1:3] 567 267 217 195 179 ...
##   ..$ ir       : int 11
##   ..$ nbest    : int 3
##   ..$ lopt     : int [1:66, 1:3] 1 1 5 1 11 5 1 11 16 5 ...
##   ..$ il       : int 66
##   ..$ ier      : int 0
##   ..$ xnames   : chr [1:16] "(Intercept)" "PrivateYes" "Top10perc" "Top25perc" ...
##   ..$ method   : chr "exhaustive"
##   ..$ force.in : Named logi [1:16] TRUE FALSE FALSE FALSE FALSE FALSE ...
##   .. ..- attr(*, "names")= chr [1:16] "" "PrivateYes" "Top10perc" "Top25perc" ...
##   ..$ force.out: Named logi [1:16] FALSE FALSE FALSE FALSE FALSE FALSE ...
##   .. ..- attr(*, "names")= chr [1:16] "" "PrivateYes" "Top10perc" "Top25perc" ...
##   ..$ sserr    : num 151
##   ..$ intercept: logi TRUE
##   ..$ lindep   : logi [1:16] FALSE FALSE FALSE FALSE FALSE FALSE ...
##   ..$ nullrss  : num 567
##   ..$ nn       : int 518
##   ..$ call     : language regsubsets.formula(College$Apps ~ ., data = College, subset = train, nbest
##   ..- attr(*, "class")= chr "regsubsets"
##  - attr(*, "class")= chr "summary.regsubsets"
```

In figure 4 is shown the size of the models against the BIC values.

```
#create the number of variables of each model
X<-c(1,1,1,2,2,2,3,3,3,4,4,4,5,5,5,6,6,6,7,7,7,8,8,8,9,9,9,10,10,10)
#plot the BIC values
plot(X,BIC)
axis(1, at=c(1:10))
```

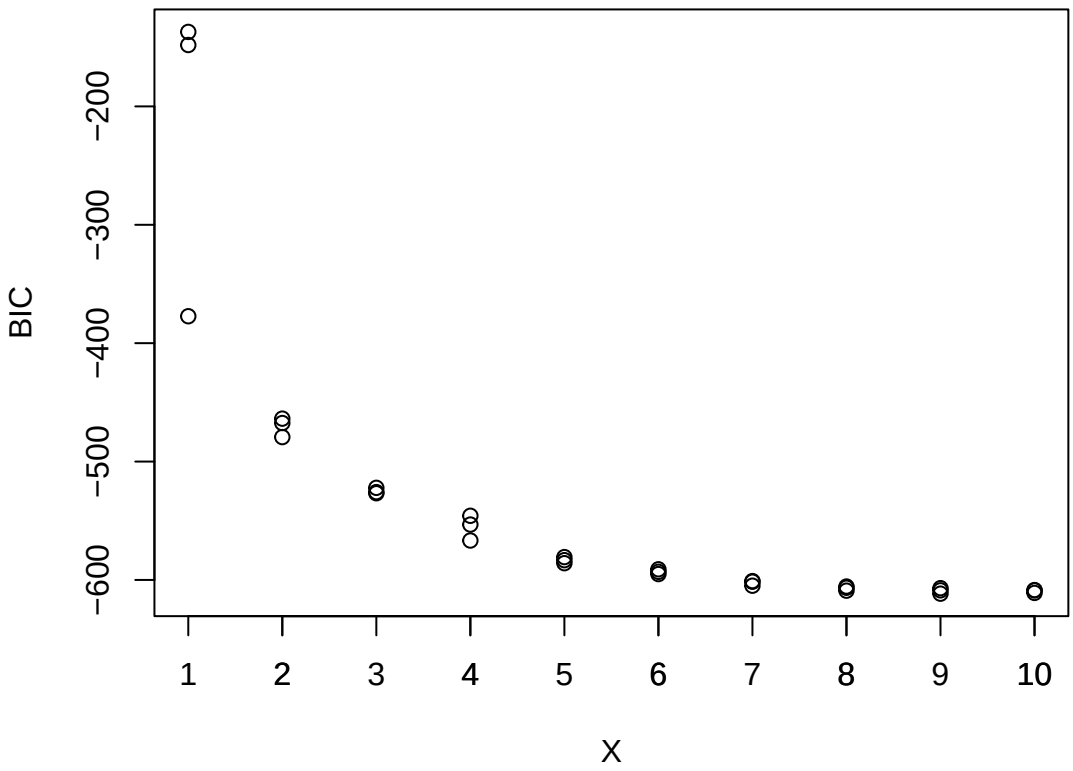

Figure 4: BIC values with different number of variables

Thus, according to the plot with the BIC values and the number of variables, the optimal model is with 7 variables as mentioned in the question 2b. Using the lm() function, a new linear regression reduced model was created with the 7 variables: Private", "F.Undergrad", "Room.Board", "PhD", "S.F.ratio", "Expend" and "Grad.Rate".

```
reduced.model <- lm(Apps~Private+F.Undergrad+Room.Board+
                      PhD+S.F.Ratio+Expend+Grad.Rate,
                data = College, subset = train)
summary(reduced.model)
```

```
##
## Call:
## lm(formula = Apps ~ Private + F.Undergrad + Room.Board + PhD +
##     S.F.Ratio + Expend + Grad.Rate, data = College, subset = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.24841 -0.31587  0.03628  0.37856  1.82247
##
## Coefficients:
```

```
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.281e+00  2.135e-01  20.049  < 2e-16 ***
## PrivateYes  -5.366e-01  8.475e-02  -6.332 5.32e-10 ***
## F.Undergrad  1.174e-04  6.915e-06  16.975  < 2e-16 ***
## Room.Board   1.716e-04  2.775e-05   6.184 1.28e-09 ***
## PhD          8.805e-03  1.793e-03   4.912 1.22e-06 ***
## S.F.Ratio    3.607e-02  8.495e-03   4.246 2.58e-05 ***
## Expend       3.413e-05  6.398e-06   5.335 1.44e-07 ***
## Grad.Rate    1.287e-02  1.744e-03   7.378 6.54e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5604 on 510 degrees of freedom
## Multiple R-squared:  0.7175, Adjusted R-squared:  0.7137
## F-statistic: 185.1 on 7 and 510 DF,  p-value: < 2.2e-16
```

According to the summary table, all the variables have very small p value (lower than a=0.05) which indicates that all input variables are significant in the model. Furthermore, the R-squared is quite good which means that the reduced model is fitting the data particularly well. Finally, the function cvFit() was used to the reduced model in order to be compared the results from the original foul model.

```
cross.validation.fit.reduced <- cvFit(reduced.model, data=College[train,],
                        y=College[train,]$Apps, cost=rmspe,
                        K=5,R=100,seed=12223236)
cross.validation.fit.reduced
```

```
## 5-fold CV results:
##        CV
## 0.5725895
```

As a result, the mean cross validation error from the reduced model is slightly higher than the result from the full model.

Question 3(a)

The Principal component regression (PCR) was applied using for the library(pls) the function pcr(). In the function the parameter scale was set to TRUE on order the data to be scaled. Furthermore, the validation parameter was set to "CV" and the segments equal to 10 for the 10-Fold cross validation to be implemented.

```
library(pls)
```

```
pcr_model <- pcr(Apps~., data=College, scale=TRUE, subset=train,
                  validation="CV", segments=10, segment.type="random")
```

Question 3(b)

In figure 5 are presented the obtained prediction errors from cross-validation.

```
plot(pcr_model, plottype = "validation", val.type = "RMSEP", legend = "topright")
axis(1, at=c(1:15))
```

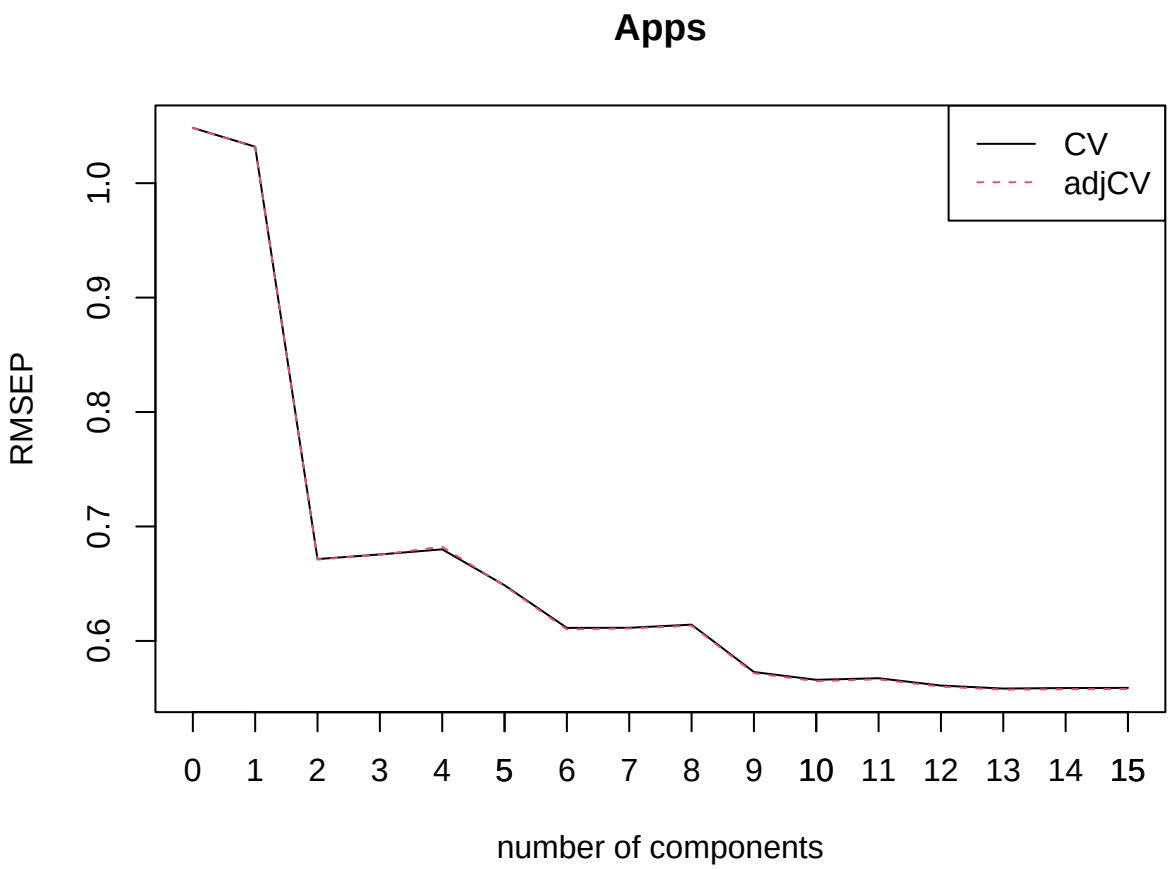

Figure 5: RMSE on different number of components

According to figure 5, 9 components seem to be optimal. The resulting RMSE is:

```
pred_pcr <- predict(pcr_model,newdata=College[train,],ncomp=9)
sqrt(mean((College$Apps[train]-pred_pcr)^2))
```

```
## [1] 0.5587204
```

Question 3(c)

The function predplot() was used to plot the measured y values against the cross-validated y values considering the model with 10 components.

```
predplot(pcr_model,ncomp=9,asp=1, line=TRUE)
```

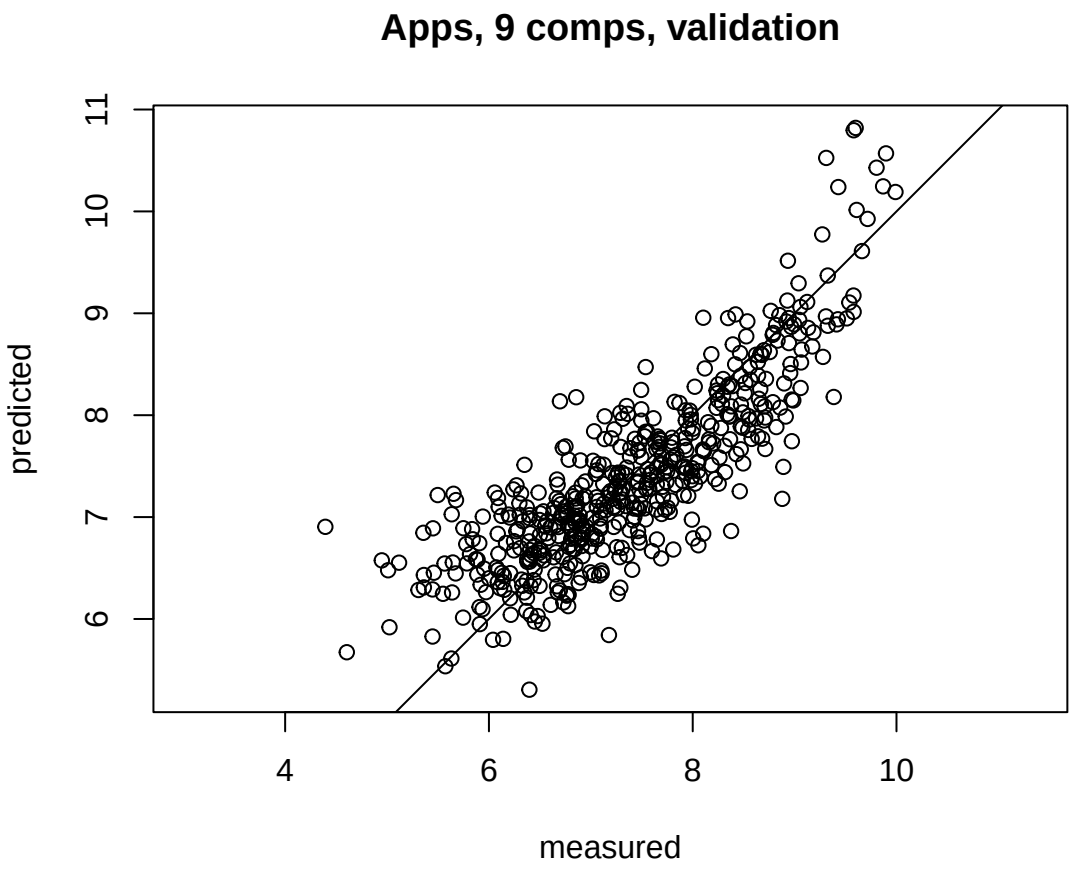

Figure 6: Measured vs Predicted values

Question 3(d)

In this question in order to plot the measured y values against the cross validated y values, the predict() function was used for the test data using the previous model.

```
pred_pcr <- predict(pcr_model,newdata = College[test,],ncomp=9)
```

The following figure shows the measured and the predicted values for the test data.

```
plot(College[test,]$Apps, pred_pcr, xlab="measured", ylab="predicted", main = "Apps")
abline(0,1)
```

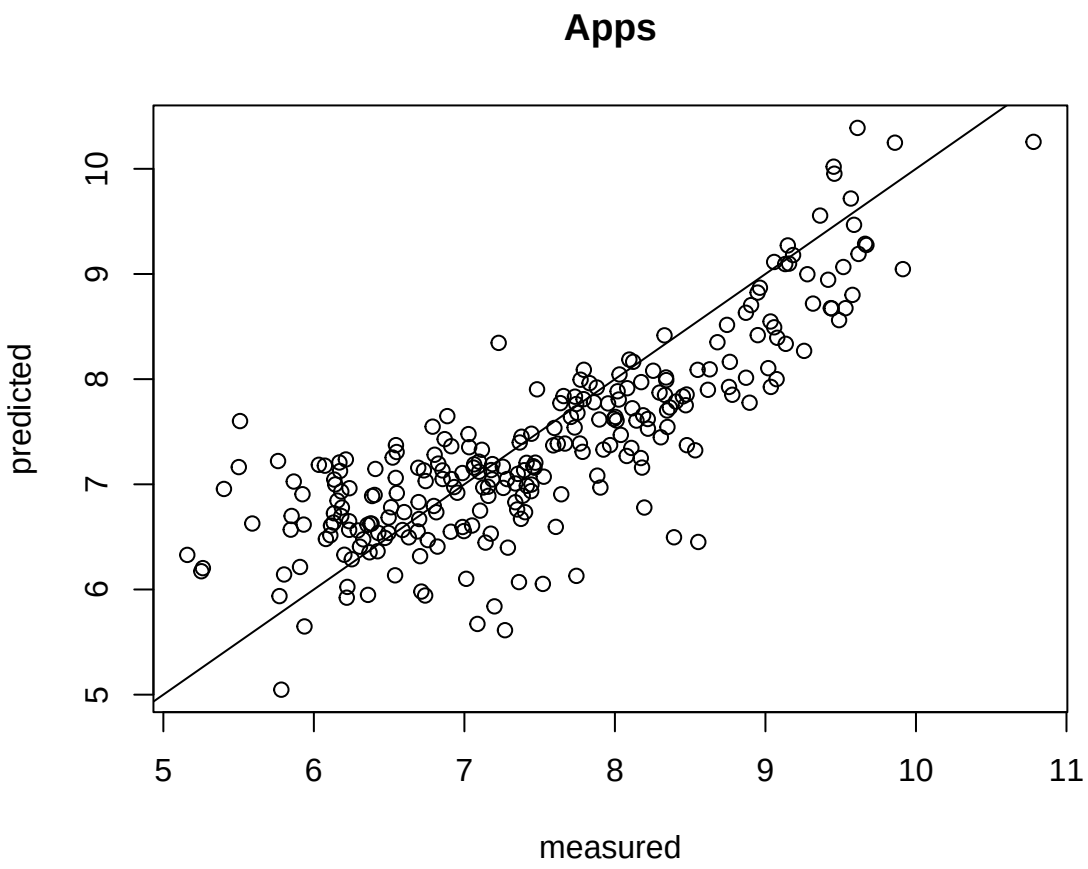

Figure 7: Measured vs Predicted values

Finally, the RMSE for the test data is:

```
sqrt(mean((College$Apps[test]-pred_pcr)^2))
```

```
## [1] 0.6447848
```

Question 4(a)

The Partial least squares regression (PLS) was applied using for the library(pls). This time, the plsr() function was used and the parameters was set as in Question 3a/

```
pls_model <- plsr(College$Apps~., data=College, scale=TRUE, subset=train,
                  validation="CV", segments=10, segment.type="random")
```

Question 4(b)

In figure 7 are presented the obtained prediction errors from cross-validation.

```
plot(pls_model, plottype = "validation", val.type = "RMSEP", legend = "topright")
axis(1, at=c(1:15))
```

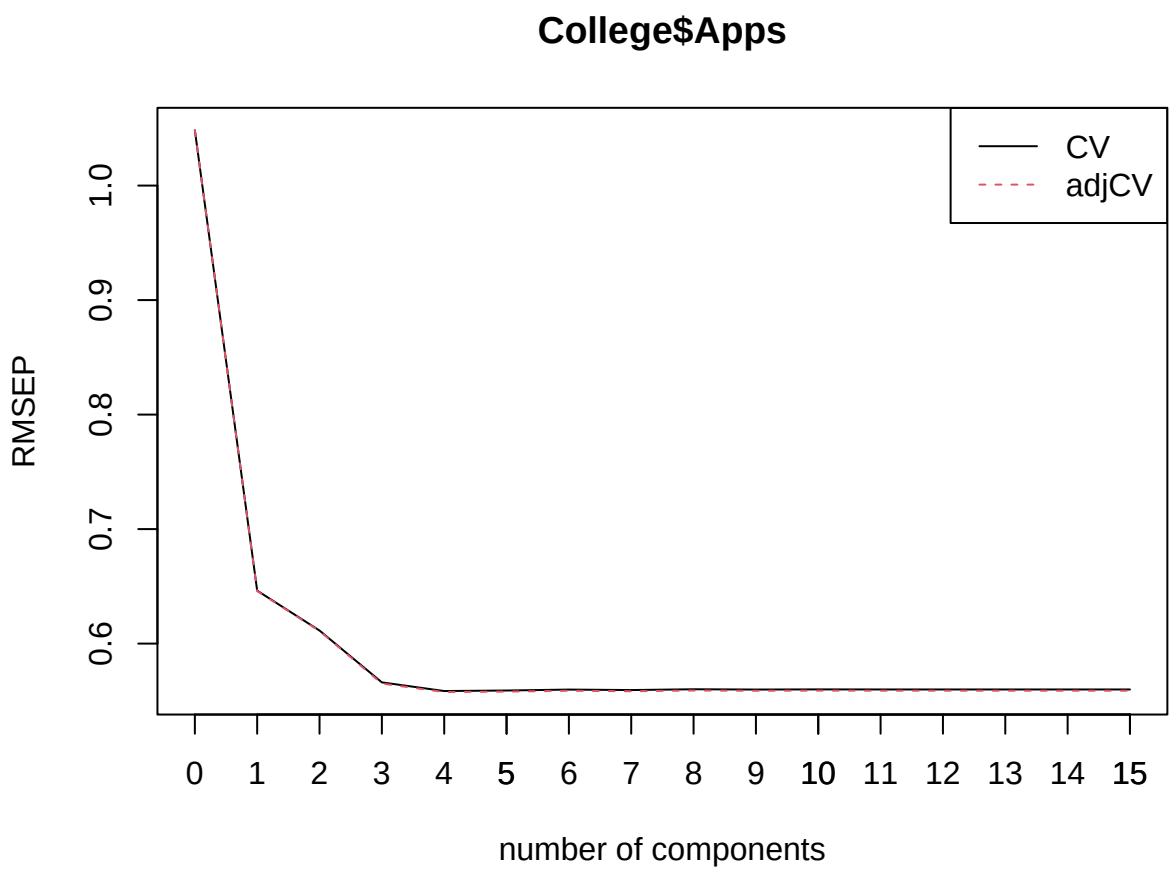

Figure 8: RMSE on different number of components

According to figure 7, 3 and 4 components seem to be optimal because the RMSE is approximately the same. Despite the small difference, 3 components was chosen in order the smaller model to be as more representative to the the full model.

The resulting RMSE is:

```
pred_pls <- predict(pls_model,newdata=College[train,],ncomp=3)
sqrt(mean((College$Apps[train]-pred_pls)^2))
```

```
## [1] 0.5480557
```

Question 4(c)

The function predplot() was used to plot the measured y values against the cross-validated y values considering the model with 4 components.

```
predplot(pls_model,ncomp=3,asp=1, line=TRUE)
```

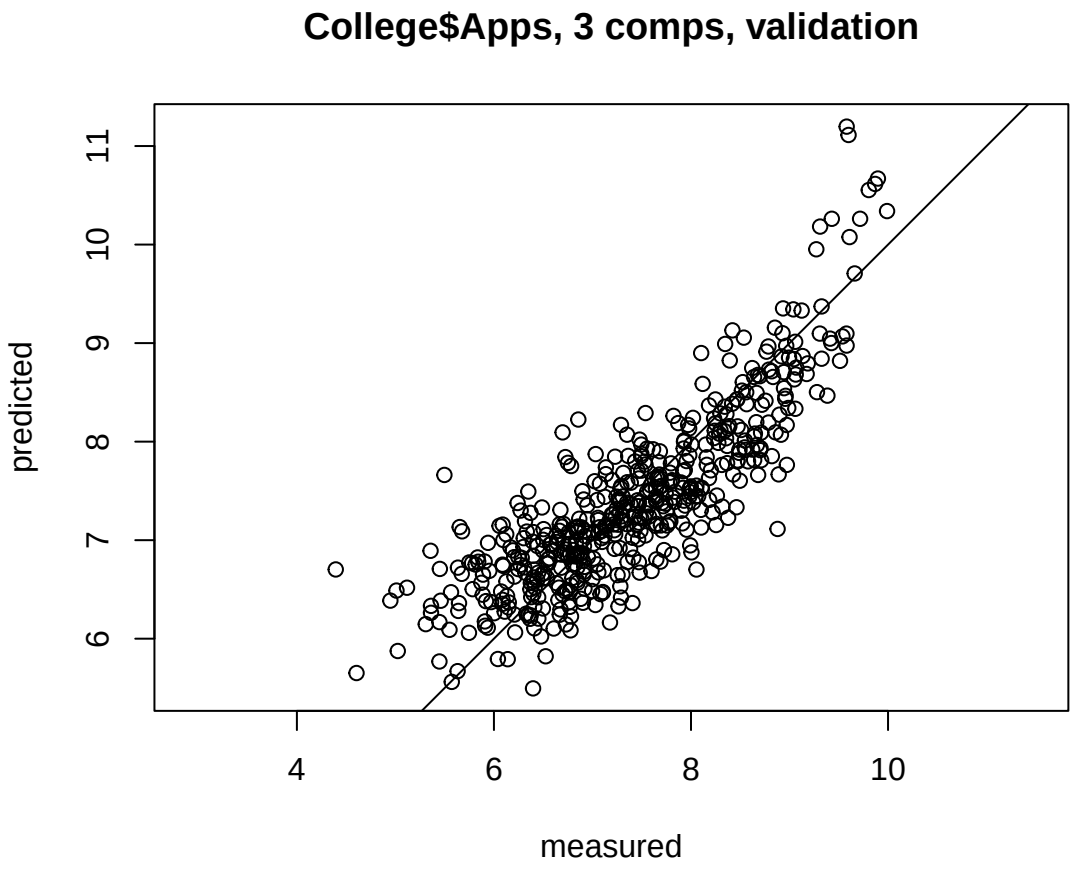

Figure 9: Measured vs Predicted values

Question 4(d)

In this question in order to plot the measured y values against the cross validated y values, the predict() function was used for the test data using the previous model.

```
pred_pls <- predict(pls_model,newdata=College[test,],ncomp=3)
```

The following figure shows the measured and the predicted values for the test data.

```
plot(College[test,]$Apps, pred_pls, xlab="measured", ylab="predicted",main = "Apps")
abline(0,1)
```

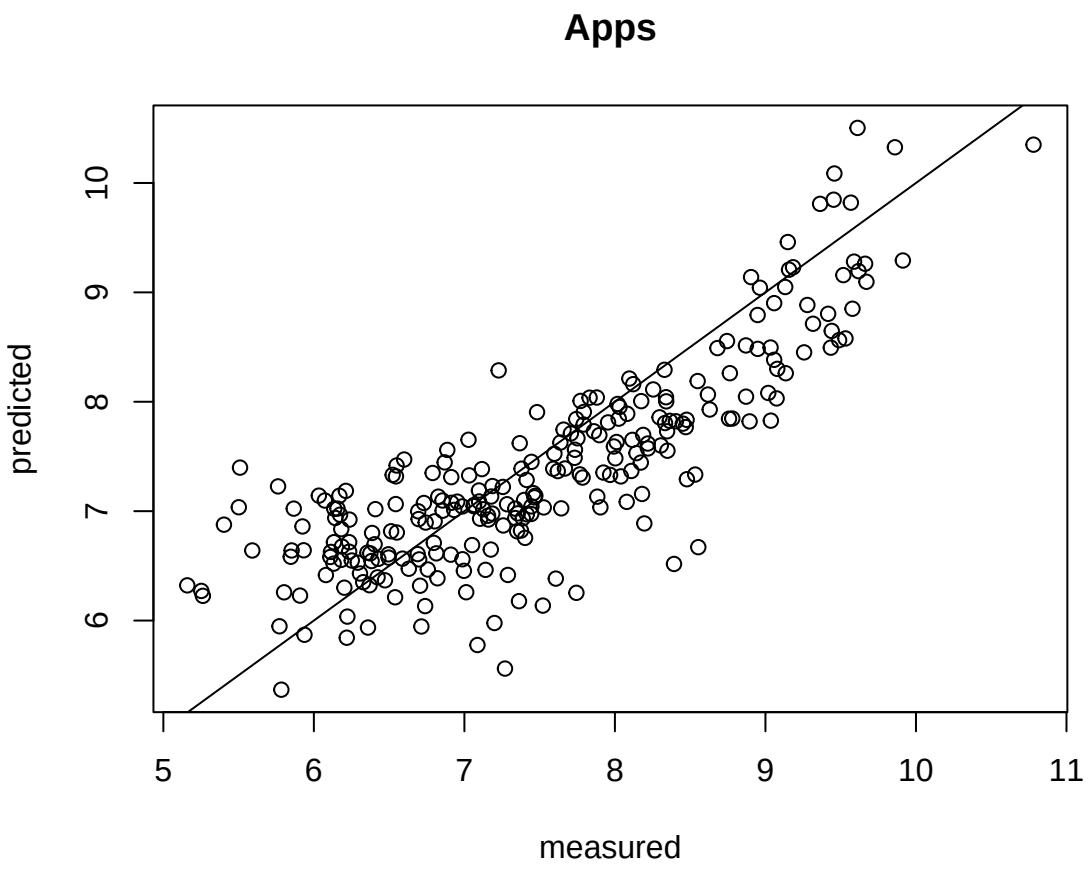

Figure 10: Measured vs Predicted values

Finally, the RMSE for the test data is:

```
sqrt(mean((College$Apps[test]-pred_pls)^2))
```

```
## [1] 0.6288325
```

The major difference between the PCR and the PLS is the number of components that they were chosen. In PCR the components are 9 and in PLS the components are 3. Regarding the RMSE for the train and test data, both algorithms have roughly the same results.

```
#Train data
sqrt(mean((College$Apps[train]-pred_pcr)^2))
```

```
## [1] 0.5587204
```

```
sqrt(mean((College$Apps[train]-pred_pls)^2))
```

```
## [1] 0.5480557
```

```
#Test data
sqrt(mean((College$Apps[test]-pred_pcr)^2))
```

```
## [1] 0.6447848
```

```
sqrt(mean((College$Apps[test]-pred_pls)^2))
```

```
## [1] 0.6288325
```

Question 5

In this question, the principal component regression was implemented by hand. First the response variable Apps was excluded from the data set and the Private variable was replaced to a binary variable with zeros and ones. The zero value replaced the "Yes" value and the one the "No" value. This replacement was done because to implement principal component analysis, all the variables need to be numerical.

```
X <- College[,-c(2)]
X$Private <- as.numeric(as.factor(X$Private))-1
```

Afterwards, the data was scaled and the principal component analysis (PCA) was performed using the function princomp().

```
X <- data.frame(scale(X))
pca <- princomp(~.,data=X,subset=train,scores=TRUE)
```

According to question 3(b), the number of components was 9. Thus, from the table of PCA scores ($scores), the first 9 columns were chosen.

```
scores <- data.frame(pca$scores)[1:9]
```

The same thing was applied to the table of loadings, which indicates the $\mathbf{V}$ matrix, in order to create the $\mathbf{Z}$ matrix for the test data. However, first the unclass function was used and the 9 first columns was loaded to the variable loadings.

```
loadings <- data.frame(unclass(pca$loadings))[1:9]
```

Consequently, the linear regression model was created and the $\mathbf{Z}$ matrix was calculated according to the formula $\mathbf{Z}=\mathbf{XV}$

```
model <- lm(College$Apps[train]~., data = scores)
Z <- data.frame(as.matrix(X[test,]) %*% as.matrix(loadings))
```

Finally the predict() function was used on the test and the RMSE for the test data is:

```
pred <- predict(model,Z)
sqrt(mean((College$Apps[test]-pred)^2))
```

```
## [1] 0.63846
```

Comparing the result with the RMSE on the test data from the question 3(d)

```
sqrt(mean((College$Apps[test]-pred_pcr)^2))
```

```
## [1] 0.6447848
```

they are approximately the same.