Task 2

for Advanced Methods for Regression and Classification

Teodor Chakarov

07.11.2022

Contents

Exercise 1

Compute Cross Validation

Let's load our College data for this exercise.

```
library("cvTools")
```

```
## Loading required package: lattice
```

```
## Loading required package: robustbase
```

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

```
##
## Attaching package: 'pls'
```

```
## The following object is masked from 'package:stats':
##
##     loadings
```

```r
data(College ,package="ISLR")
data_col <- College
str(data_col)
```

```
## 'data.frame':    777 obs. of  18 variables:
##  $ Private    : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 2 2 2 2 2 ...
##  $ Apps       : num  1660 2186 1428 417 193 ...
##  $ Accept     : num  1232 1924 1097 349 146 ...
##  $ Enroll     : num  721 512 336 137 55 158 103 489 227 172 ...
##  $ Top10perc  : num  23 16 22 60 16 38 17 37 30 21 ...
##  $ Top25perc  : num  52 29 50 89 44 62 45 68 63 44 ...
##  $ F.Undergrad: num  2885 2683 1036 510 249 ...
##  $ P.Undergrad: num  537 1227 99 63 869 ...
##  $ Outstate   : num  7440 12280 11250 12960 7560 ...
##  $ Room.Board : num  3300 6450 3750 5450 4120 ...
##  $ Books      : num  450 750 400 450 800 500 500 450 300 660 ...
##  $ Personal   : num  2200 1500 1165 875 1500 ...
##  $ PhD        : num  70 29 53 92 76 67 90 89 79 40 ...
##  $ Terminal   : num  78 30 66 97 72 73 93 100 84 41 ...
##  $ S.F.Ratio  : num  18.1 12.2 12.9 7.7 11.9 9.4 11.5 13.7 11.3 11.5 ...
##  $ perc.alumni: num  12 16 30 37 2 11 26 37 23 15 ...
##  $ Expend     : num  7041 10527 8735 19016 10922 ...
##  $ Grad.Rate  : num  60 56 54 59 15 55 63 73 80 52 ...
```

Train and Test split

We need to predict the attribute ***Apps*** (which will be our dependent variable). First we will get rid of the columns ***"Accept"*** and ***"Enroll"***. We have to convert our categorical variables to numeric one.

```r
data_col2 <- data_col[ , -which(names(data_col) %in% c("Accept", "Enroll"))]
data_col2$Private <- as.numeric(data_col2$Private)
str(data_col2)
```

```
## 'data.frame':    777 obs. of  16 variables:
##  $ Private    : num  2 2 2 2 2 2 2 2 2 2 ...
##  $ Apps       : num  1660 2186 1428 417 193 ...
##  $ Top10perc  : num  23 16 22 60 16 38 17 37 30 21 ...
##  $ Top25perc  : num  52 29 50 89 44 62 45 68 63 44 ...
##  $ F.Undergrad: num  2885 2683 1036 510 249 ...
##  $ P.Undergrad: num  537 1227 99 63 869 ...
##  $ Outstate   : num  7440 12280 11250 12960 7560 ...
##  $ Room.Board : num  3300 6450 3750 5450 4120 ...
##  $ Books      : num  450 750 400 450 800 500 500 450 300 660 ...
##  $ Personal   : num  2200 1500 1165 875 1500 ...
##  $ PhD        : num  70 29 53 92 76 67 90 89 79 40 ...
##  $ Terminal   : num  78 30 66 97 72 73 93 100 84 41 ...
##  $ S.F.Ratio  : num  18.1 12.2 12.9 7.7 11.9 9.4 11.5 13.7 11.3 11.5 ...
##  $ perc.alumni: num  12 16 30 37 2 11 26 37 23 15 ...
##  $ Expend     : num  7041 10527 8735 19016 10922 ...
##  $ Grad.Rate  : num  60 56 54 59 15 55 63 73 80 52 ...
```

I am picking 2/3rd random data indexes of the data for the training set and 1/3 for the testing.

```
set.seed(16)

## 2/3 of the sample size
smp_size <- floor(round(nrow(data_col2)*2/3))
train_ind <- sample(seq_len(nrow(data_col2)), size = smp_size)
smp_size
```

```
## [1] 518
```

Getting the sample size. Lets now split the data into train and test while also separating the dependent variable with the independent once.

```
train <- data_col2[train_ind, ]
test <- data_col2[-train_ind, ]

# Setting the y to be "Apps"
y_train = train[ , which(names(train) %in% c("Apps"))]
y_test = test[ , which(names(test) %in% c("Apps"))]

# Removing the predictive variable from the training and testing sets.
x_train = train[ , -which(names(train) %in% c("Apps"))]
x_test = test[ , -which(names(test) %in% c("Apps"))]
```

```
dim(x_train)
```

```
## [1] 518  15
```

```
dim(x_test)
```

```
## [1] 259  15
```

Linear Model Fit with Cross Validation

Now we will fit our training data to the linear model while using Cross Validation technique for the training process.

```
lin_reg <- lm(y_train ~ ., data = x_train)

cv_model <- cvFit(lm, formula=y_train ~ ., data=x_train, y=y_train, cost=rmspe, K=5, seed = 16)
```

```
cv_model
```

```
## 5-fold CV results:
##       CV
## 1379.353
```

This is the average RMSPE error which is little higher than the previous model.

```r
compute_rmse <- function(y_true, y_pred) {

   return(sqrt(mean((y_true-y_pred)^2)))
}
```

```r
pred_lm <- predict(lin_reg, x_train)
compute_rmse(y_train, pred_lm)
```

```
## [1] 1314.409
```

```r
pred_lm_test <- predict(lin_reg, x_test)
compute_rmse(y_test, pred_lm_test)
```

```
## [1] 2802.396
```

This is the RMSE on the linear regression alone.

Best Subset Regression

```r
subs_reg <- regsubsets(y_train ~., data=x_train, nbest = 3, nvmax=10)
summary(subs_reg)
```

```
## Subset selection object
## Call: regsubsets.formula(y_train ~ ., data = x_train, nbest = 3, nvmax = 10)
## 15 Variables  (and intercept)
##             Forced in Forced out
## Private         FALSE      FALSE
## Top10perc       FALSE      FALSE
## Top25perc       FALSE      FALSE
## F.Undergrad     FALSE      FALSE
## P.Undergrad     FALSE      FALSE
## Outstate        FALSE      FALSE
## Room.Board      FALSE      FALSE
## Books           FALSE      FALSE
## Personal        FALSE      FALSE
## PhD             FALSE      FALSE
## Terminal        FALSE      FALSE
## S.F.Ratio       FALSE      FALSE
## perc.alumni     FALSE      FALSE
## Expend          FALSE      FALSE
## Grad.Rate       FALSE      FALSE
## 3 subsets of each size up to 10
## Selection Algorithm: exhaustive
##           Private Top10perc Top25perc F.Undergrad P.Undergrad Outstate
## 1  ( 1 )  " "     " "       " "       "*"         " "         " "
## 1  ( 2 )  "*"     " "       " "       " "         " "         " "
## 1  ( 3 )  " "     " "       " "       " "         "*"         " "
## 2  ( 1 )  " "     " "       " "       "*"         " "         " "
## 2  ( 2 )  " "     " "       " "       "*"         " "         "*"
```

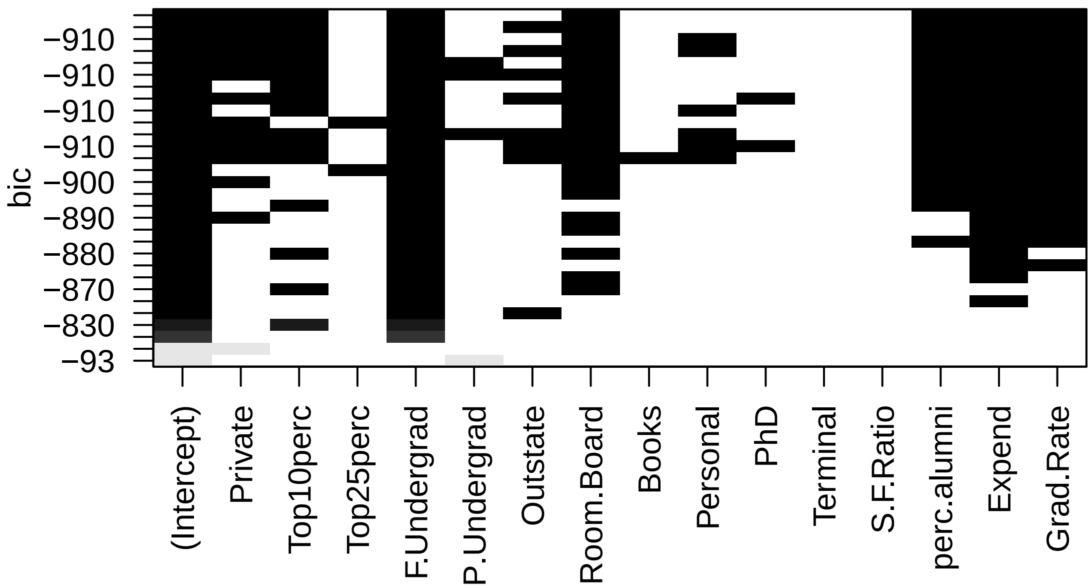
```
## 2  ( 3 )  " "     "*"       " "       "*"         " "         " "
## 3  ( 1 )  " "     " "       " "       "*"         " "         " "
## 3  ( 2 )  " "     " "       " "       "*"         " "         " "
## 3  ( 3 )  " "     "*"       " "       "*"         " "         " "
## 4  ( 1 )  " "     " "       " "       "*"         " "         " "
## 4  ( 2 )  " "     " "       " "       "*"         " "         " "
## 4  ( 3 )  " "     "*"       " "       "*"         " "         " "
## 5  ( 1 )  " "     " "       " "       "*"         " "         " "
## 5  ( 2 )  " "     "*"       " "       "*"         " "         " "
## 5  ( 3 )  "*"     " "       " "       "*"         " "         " "
## 6  ( 1 )  " "     "*"       " "       "*"         " "         " "
## 6  ( 2 )  " "     " "       "*"       "*"         " "         " "
## 6  ( 3 )  "*"     " "       " "       "*"         " "         " "
## 7  ( 1 )  "*"     "*"       " "       "*"         " "         " "
## 7  ( 2 )  " "     "*"       " "       "*"         " "         " "
## 7  ( 3 )  "*"     " "       "*"       "*"         " "         " "
## 8  ( 1 )  "*"     "*"       " "       "*"         " "         "*"
## 8  ( 2 )  "*"     "*"       " "       "*"         " "         " "
## 8  ( 3 )  "*"     "*"       " "       "*"         "*"         " "
## 9  ( 1 )  "*"     "*"       " "       "*"         " "         "*"
## 9  ( 2 )  "*"     "*"       " "       "*"         "*"         "*"
## 9  ( 3 )  "*"     "*"       " "       "*"         " "         "*"
## 10  ( 1 ) "*"     "*"       " "       "*"         "*"         "*"
## 10  ( 2 ) "*"     "*"       " "       "*"         " "         "*"
## 10  ( 3 ) "*"     "*"       " "       "*"         " "         "*"
##           Room.Board Books Personal PhD Terminal S.F.Ratio perc.alumni Expend
## 1  ( 1 )  " "        " "   " "      " " " "      " "       " "         " "
## 1  ( 2 )  " "        " "   " "      " " " "      " "       " "         " "
## 1  ( 3 )  " "        " "   " "      " " " "      " "       " "         " "
## 2  ( 1 )  " "        " "   " "      " " " "      " "       " "         "*"
## 2  ( 2 )  " "        " "   " "      " " " "      " "       " "         " "
## 2  ( 3 )  " "        " "   " "      " " " "      " "       " "         " "
## 3  ( 1 )  " "        " "   " "      " " " "      " "       " "         "*"
## 3  ( 2 )  "*"        " "   " "      " " " "      " "       " "         "*"
## 3  ( 3 )  "*"        " "   " "      " " " "      " "       " "         " "
## 4  ( 1 )  "*"        " "   " "      " " " "      " "       " "         "*"
## 4  ( 2 )  " "        " "   " "      " " " "      " "       "*"         "*"
## 4  ( 3 )  "*"        " "   " "      " " " "      " "       " "         "*"
## 5  ( 1 )  "*"        " "   " "      " " " "      " "       "*"         "*"
## 5  ( 2 )  " "        " "   " "      " " " "      " "       "*"         "*"
## 5  ( 3 )  "*"        " "   " "      " " " "      " "       " "         "*"
## 6  ( 1 )  "*"        " "   " "      " " " "      " "       "*"         "*"
## 6  ( 2 )  "*"        " "   " "      " " " "      " "       "*"         "*"
## 6  ( 3 )  "*"        " "   " "      " " " "      " "       "*"         "*"
## 7  ( 1 )  "*"        " "   " "      " " " "      " "       "*"         "*"
## 7  ( 2 )  "*"        " "   "*"      " " " "      " "       "*"         "*"
## 7  ( 3 )  "*"        " "   " "      " " " "      " "       "*"         "*"
## 8  ( 1 )  "*"        " "   " "      " " " "      " "       "*"         "*"
## 8  ( 2 )  "*"        " "   "*"      " " " "      " "       "*"         "*"
## 8  ( 3 )  "*"        " "   " "      " " " "      " "       "*"         "*"
## 9  ( 1 )  "*"        " "   "*"      " " " "      " "       "*"         "*"
## 9  ( 2 )  "*"        " "   " "      " " " "      " "       "*"         "*"
## 9  ( 3 )  "*"        " "   " "      "*" " "      " "       "*"         "*"
## 10  ( 1 ) "*"        " "   "*"      " " " "      " "       "*"         "*"
```

```
## 10  ( 2 ) "*"         " "   "*"       "*" " "       " "        "*"          "*"
## 10  ( 3 ) "*"         "*"   "*"       " " " "       " "        "*"          "*"
##           Grad.Rate
## 1  ( 1 )  " "
## 1  ( 2 )  " "
## 1  ( 3 )  " "
## 2  ( 1 )  " "
## 2  ( 2 )  " "
## 2  ( 3 )  " "
## 3  ( 1 )  "*"
## 3  ( 2 )  " "
## 3  ( 3 )  " "
## 4  ( 1 )  "*"
## 4  ( 2 )  "*"
## 4  ( 3 )  " "
## 5  ( 1 )  "*"
## 5  ( 2 )  "*"
## 5  ( 3 )  "*"
## 6  ( 1 )  "*"
## 6  ( 2 )  "*"
## 6  ( 3 )  "*"
## 7  ( 1 )  "*"
## 7  ( 2 )  "*"
## 7  ( 3 )  "*"
## 8  ( 1 )  "*"
## 8  ( 2 )  "*"
## 8  ( 3 )  "*"
## 9  ( 1 )  "*"
## 9  ( 2 )  "*"
## 9  ( 3 )  "*"
## 10  ( 1 ) "*"
## 10  ( 2 ) "*"
## 10  ( 3 ) "*"
```

Now we can compare which attributes gives us the bes models for each sizes. For example: - Variables for model with 1 attribute are **F.Undergrad (model1),P.Undergrad(model2), Private(model3) -** Variables for model with 2 attribute are **F.Undergrad and Expend (model1), F.Undergrad and Room.Board (model2) and etc.**

This output wont help use because we dont have information base on metric which model is actually the best one.

```
plot(subs_reg)
```



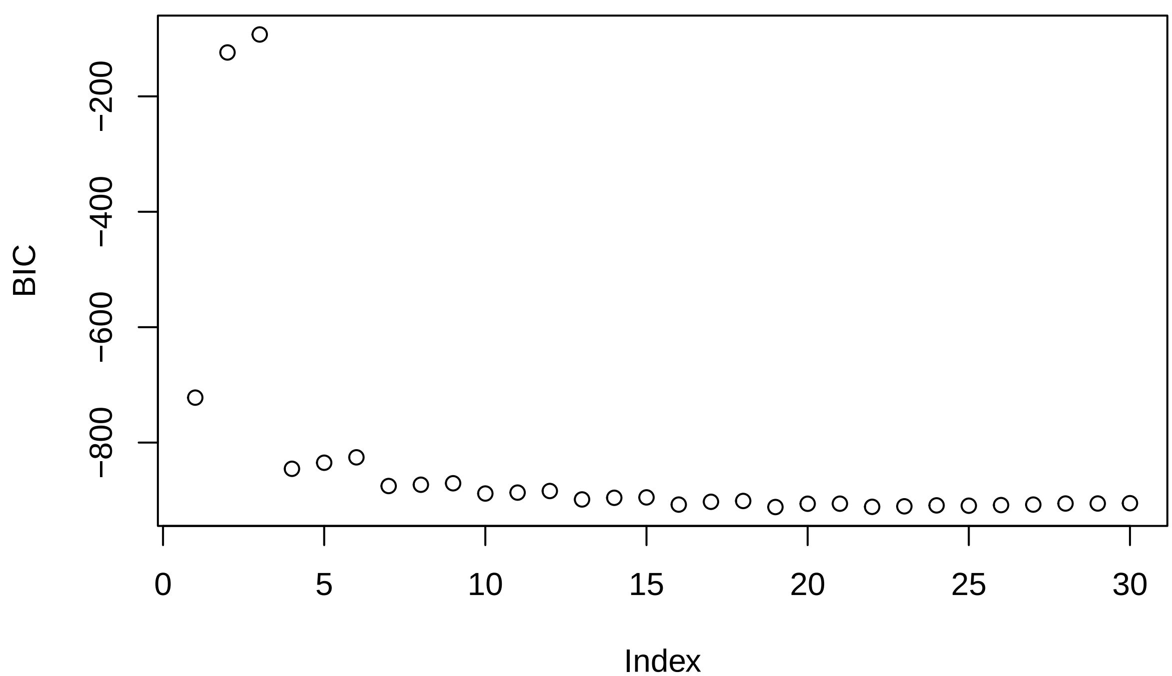

```r
subs_reg_summary <- summary(subs_reg)

str(subs_reg)
```

```
## List of 28
##  $ np       : int 16
##  $ nrbar    : int 120
##  $ d        : num [1:16] 518 83344 6935 106853 11195204 ...
##  $ rbar     : num [1:120] 23.4 14.1 79.7 542.3 72.7 ...
##  $ thetab   : num [1:16] 2922.27 -32.26 52.75 104.66 2.68 ...
##  $ first    : int 2
##  $ last     : int 16
##  $ vorder   : int [1:16] 1 14 13 12 9 11 15 4 16 3 ...
##  $ tol      : num [1:16] 1.14e-08 6.24e-07 2.58e-07 1.34e-06 8.84e-06 ...
##  $ rss      : num [1:16] 5.92e+09 5.84e+09 5.82e+09 4.65e+09 4.57e+09 ...
##  $ bound    : num [1:16] 1.00e+35 4.83e+09 1.16e+09 1.05e+09 1.01e+09 ...
##  $ nvmax    : int 11
##  $ ress     : num [1:11, 1:3] 5.92e+09 1.43e+09 1.12e+09 1.04e+09 1.00e+09 ...
##  $ ir       : int 11
##  $ nbest    : int 3
##  $ lopt     : int [1:66, 1:3] 1 1 5 1 5 15 1 5 15 16 ...
##  $ il       : int 66
##  $ ier      : int 0
##  $ xnames   : chr [1:16] "(Intercept)" "Private" "Top10perc" "Top25perc" ...
##  $ method   : chr "exhaustive"
```

```
##  $ force.in : Named logi [1:16] TRUE FALSE FALSE FALSE FALSE FALSE ...
##   ..- attr(*, "names")= chr [1:16] "" "Private" "Top10perc" "Top25perc" ...
##  $ force.out: Named logi [1:16] FALSE FALSE FALSE FALSE FALSE FALSE ...
##   ..- attr(*, "names")= chr [1:16] "" "Private" "Top10perc" "Top25perc" ...
##  $ sserr    : num 8.95e+08
##  $ intercept: logi TRUE
##  $ lindep   : logi [1:16] FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ nullrss  : num 5.92e+09
##  $ nn       : int 518
##  $ call     : language regsubsets.formula(y_train ~ ., data = x_train, nbest = 3, nvmax = 10)
##  - attr(*, "class")= chr "regsubsets"
```

```
BIC <- summary(subs_reg)$bic
plot(BIC)
```



Based on the Scatter plot we can select the index with the lowest BIC. Which is the model with the 4 attrbibutes (index 10): **F.Undergrad and Room.Board, Exprend and Grad.Rate**

```
chosen_data <- data_col[ , which(names(data_col) %in% c("F.Undergrad", "Room.Board", "Exprend", "Grad.Ra
str(chosen_data)
```

```
## 'data.frame':    777 obs. of  4 variables:
##  $ Apps       : num  1660 2186 1428 417 193 ...
##  $ F.Undergrad: num  2885 2683 1036 510 249 ...
##  $ Room.Board : num  3300 6450 3750 5450 4120 ...
##  $ Grad.Rate  : num  60 56 54 59 15 55 63 73 80 52 ...
```

We will do the train test split again and fit the data in the linear model with cross validation.

```
train_2 <- chosen_data[train_ind, ]
test_2 <- chosen_data[-train_ind, ]

# Setting the y to be "Apps"
y_train_2 = train_2[ , which(names(train_2) %in% c("Apps"))]
y_test_2 = test_2[ , which(names(test_2) %in% c("Apps"))]

# Removing the predictive variable from the training and testing sets.
x_train_2 = train_2[ , -which(names(train_2) %in% c("Apps"))]
x_test_2 = test_2[ , -which(names(test_2) %in% c("Apps"))]
```

```
cvFit(lm, formula=y_train ~ ., data=x_train_2, y=y_train_2, cost=rmspe, K=5, seed = 16)
```

```
## 5-fold CV results:
##       CV
## 1474.894
```

We get higher number as before. But we reduced our variables to 3.

```
lm_2 <- lm(y_train ~ ., data=x_train_2)
summary(lm_2)
```

```
##
## Call:
## lm(formula = y_train ~ ., data = x_train_2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8706.5  -759.9  -181.5   465.2  6937.2
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.080e+03  3.178e+02  -9.691  < 2e-16 ***
## F.Undergrad  6.046e-01  1.292e-02  46.804  < 2e-16 ***
## Room.Board   5.051e-01  6.698e-02   7.541 2.13e-13 ***
## Grad.Rate    2.396e+01  4.031e+00   5.944 5.15e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1464 on 514 degrees of freedom
## Multiple R-squared:  0.8141, Adjusted R-squared:  0.813
## F-statistic: 750.3 on 3 and 514 DF,  p-value: < 2.2e-16
```

Here we can see that all of the attributes as significant for our model.

```
pred_train_2 <- predict(lm_2, x_train_2)

compute_rmse(y_train_2, pred_train_2)
```

```
## [1] 1457.85
```

Compute Partial component regression

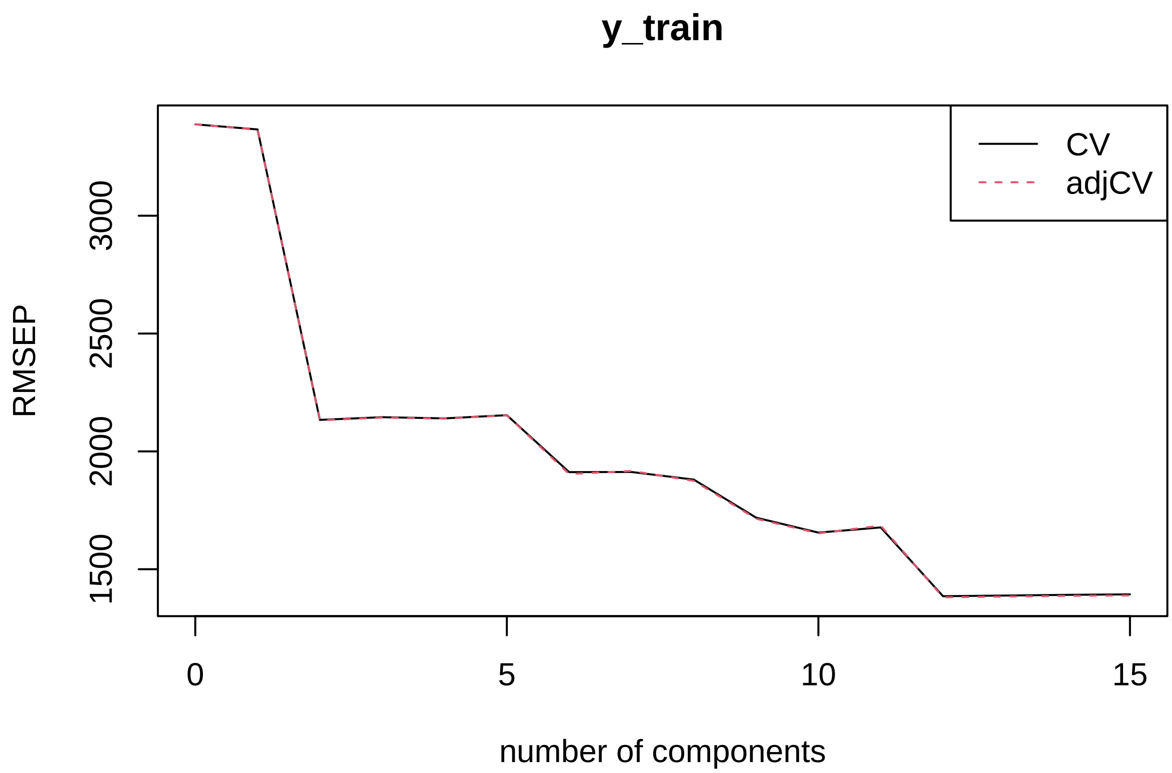
Now we will try to compute the PCR, getting rid of the co linearity of the data while we do also dimesionalitiy reduction.

```
pcr_model <- pcr(y_train ~., data=x_train, scale=TRUE,
                 validation="CV", segments=10, segment.type="random")
summary(pcr_model)
```

```
## Data:    X dimension: 518 15
##  Y dimension: 518 1
## Fit method: svdpc
## Number of components considered: 15
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV            3388     3366     2134     2145     2140     2154     1912
## adjCV         3388     3365     2132     2143     2139     2153     1904
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV        1913     1880     1719      1655      1677      1386      1388
## adjCV     1917     1875     1714      1652      1685      1381      1384
##        14 comps  15 comps
## CV         1391      1393
## adjCV      1387      1389
##
## TRAINING: % variance explained
##          1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X         36.142    55.66    63.51    69.94    75.55    80.05    84.03    87.86
## y_train    1.665    60.79    60.90    61.25    61.30    69.84    70.06    71.51
##          9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
## X          90.79     93.32     95.48     97.31     98.51     99.44    100.00
## y_train    76.20     77.75     77.77     84.80     84.82     84.83     84.89
```

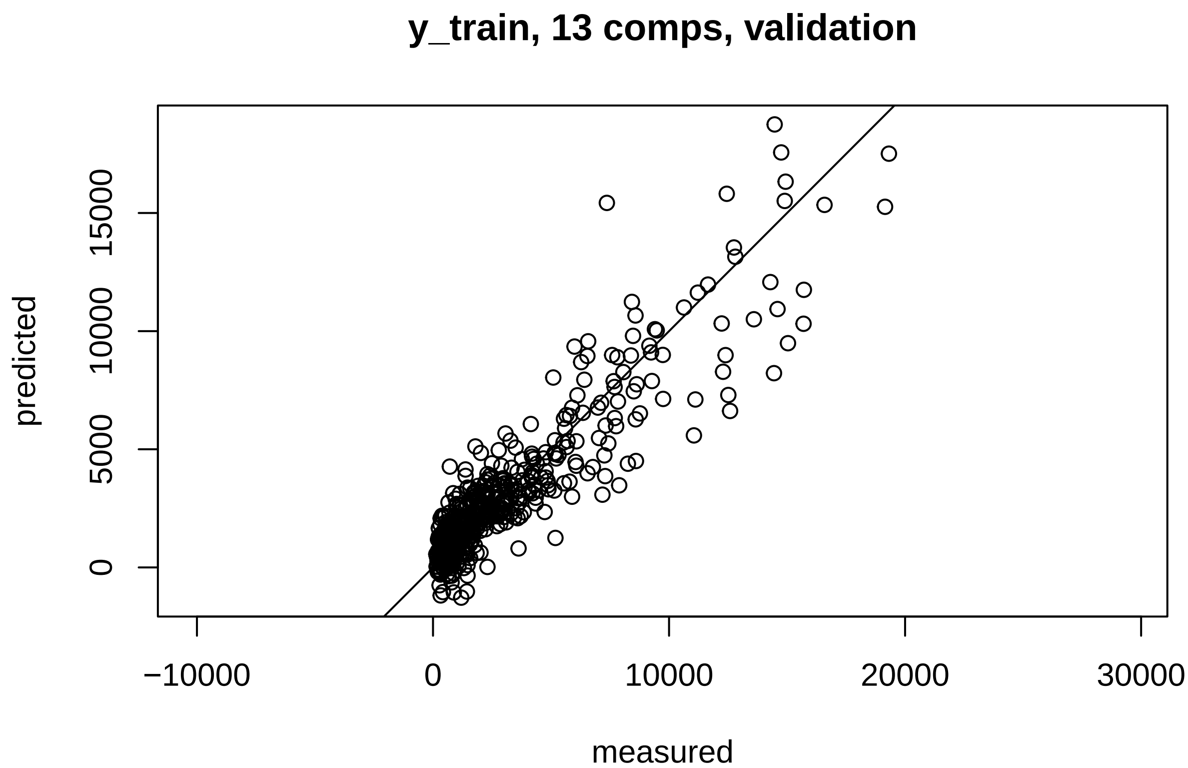
As we can see we have 15 components and 10 cross-validations are being done. The metric is root-mean-squared-error. The data is being scaled. With this dataset we can inspect that the first 9 components are explaining 90% of the data.

```
plot(pcr_model, plottype = "validation", val.type = "RMSEP", legend = "topright")
```



We can see that the RMSEP error is smaller at 13-14 attributes as we have the smallest RMSEP CV error 2124. Even though we can have 90% of the information gain based on the first 9 components.

```
predplot(pcr_model, ncomp=13, asp=1, line=TRUE)
```



The predicted RMSE for the train data is:

```
preds_pcr_train <- predict(pcr_model, x_train, ncomp=13)
compute_rmse(y_train, preds_pcr_train)
```

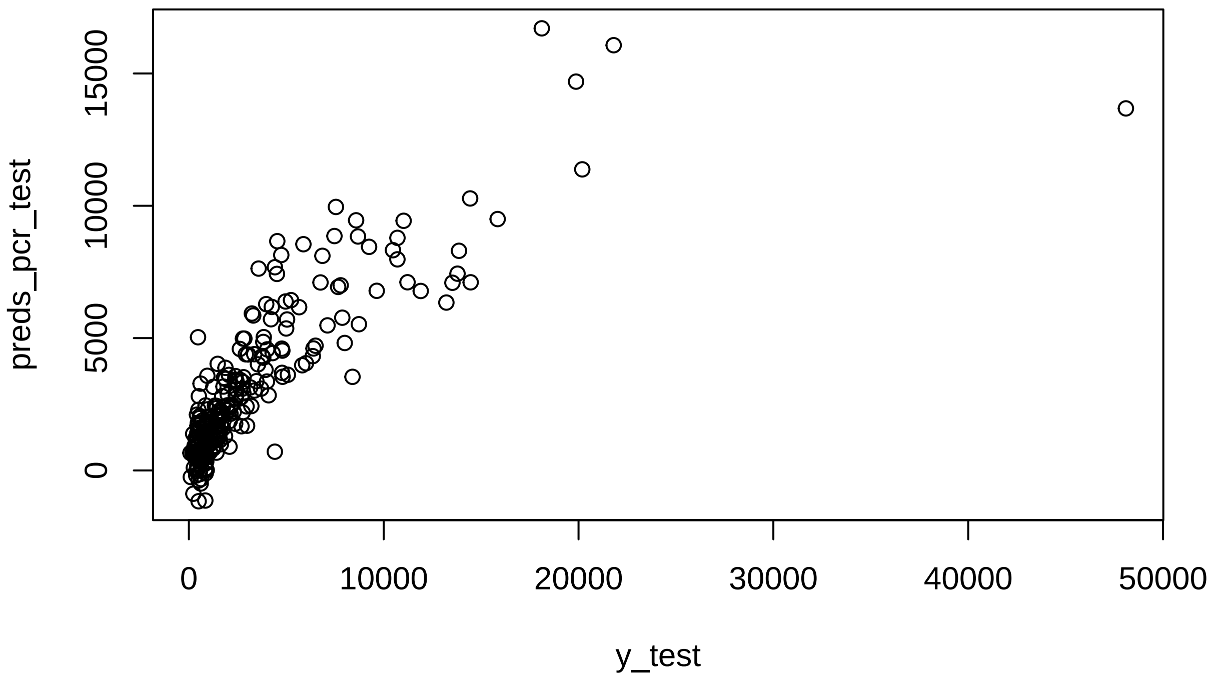
```
## [1] 1317.288
```

The predicted RMSE for the testing data is:

```
preds_pcr_test <- predict(pcr_model,newdata=x_test,ncomp=13)
#  RMSE
compute_rmse(y_test, preds_pcr_test)
```

```
## [1] 2803.985
```

```
plot(y_test, preds_pcr_test)
```



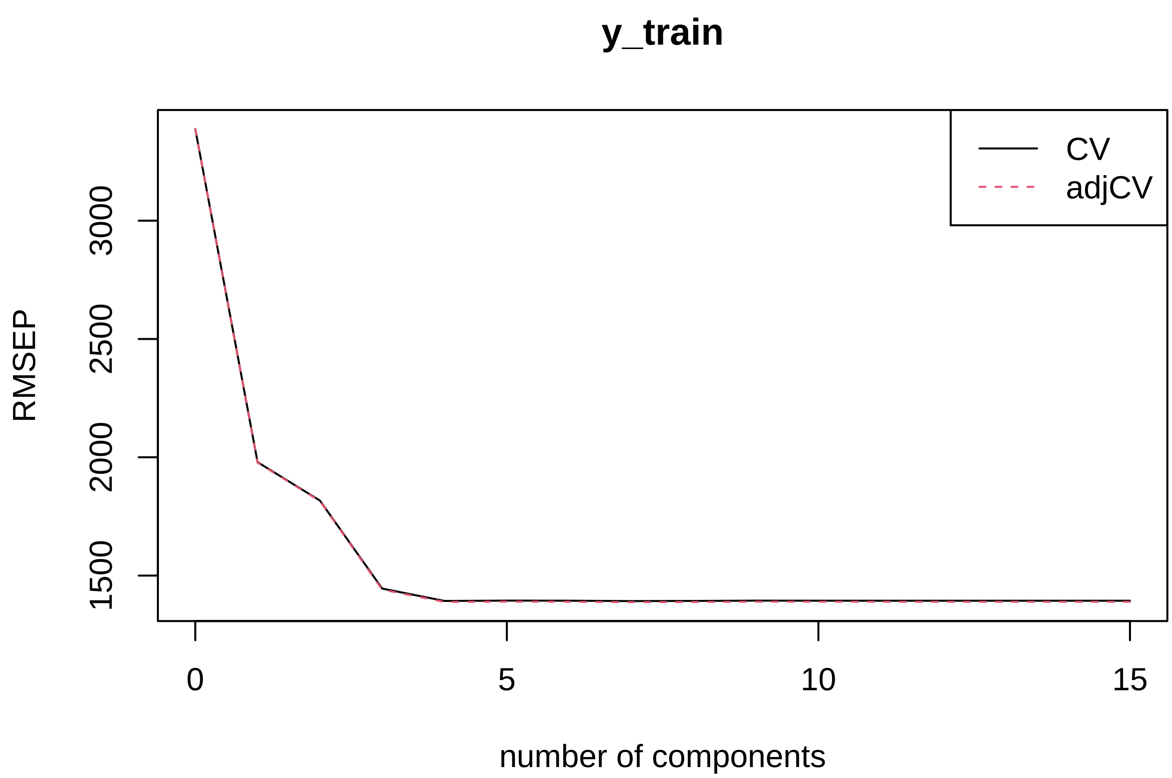
Partial least squares regression

```
plsr_model <- plsr(y_train ~., data=x_train, scale=TRUE,
                  validation="CV", segments=10, segment.type="random")
summary(plsr_model)
```

```
## Data:    X dimension: 518 15
##  Y dimension: 518 1
## Fit method: kernelpls
## Number of components considered: 15
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV            3388     1979     1817     1445     1393     1394     1394
## adjCV         3388     1977     1815     1441     1389     1390     1389
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV        1392     1393     1394      1394      1394      1394      1394
## adjCV     1388     1388     1390      1389      1389      1389      1389
##        14 comps  15 comps
## CV         1394      1394
## adjCV      1389      1389
##
```

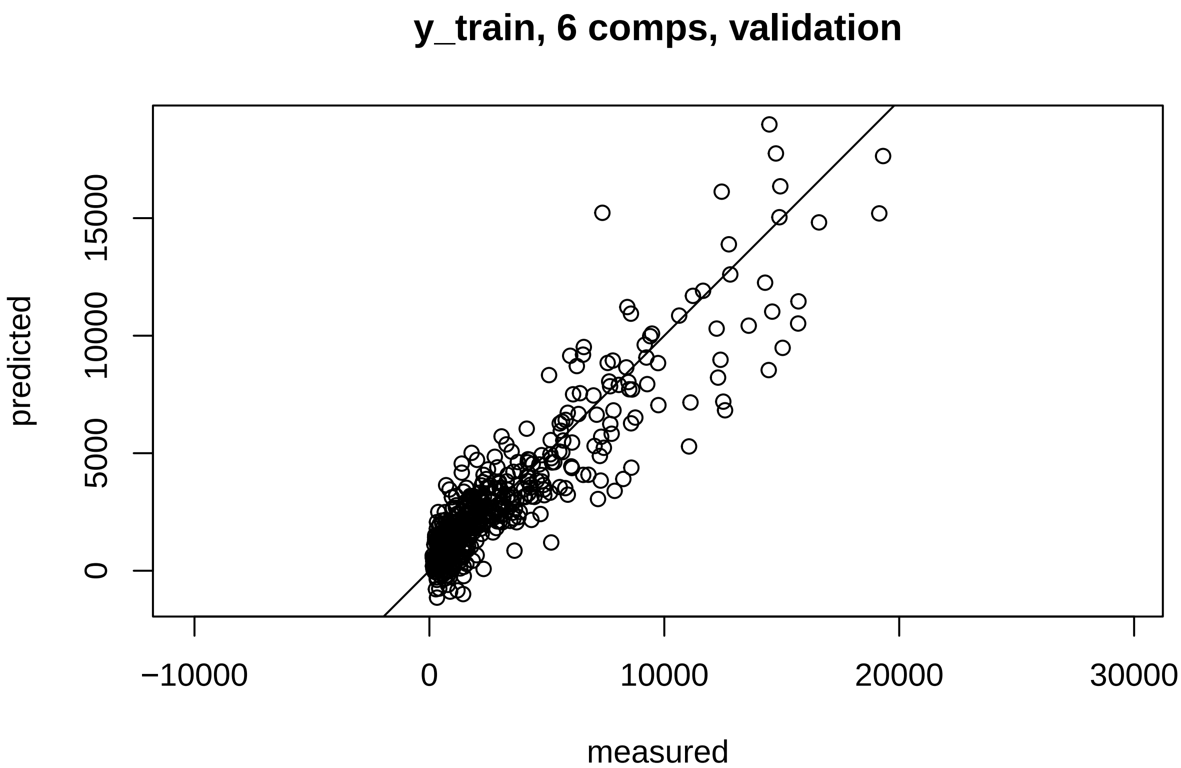
```
## TRAINING: % variance explained
##          1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X          20.78    51.79    59.52    63.43    69.18    73.52    78.78    80.93
## y_train    66.89    72.50    83.05    84.62    84.83    84.87    84.88    84.89
##          9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
## X          83.92     88.23     91.00     92.85     95.62     97.86    100.00
## y_train    84.89     84.89     84.89     84.89     84.89     84.89     84.89
```

```
plot(plsr_model, plottype = "validation", val.type = "RMSEP", legend = "topright")
```



We can see in comparison to the PCR model, this models converges faster and the RMSEP is minimizing way earlier. Around 5 components is hitting 2148 as error. The error is a little bit more than the PCR but uses less attributes.

```
predplot(plsr_model, ncomp=6, asp=1, line=TRUE)
```



We can see that the data is still heteroscedastic with outlyers.

The predicted RMSE for the train data is:

```
preds_plsr_train <- predict(plsr_model, x_train, ncomp=6)
compute_rmse(y_train, preds_plsr_train)
```

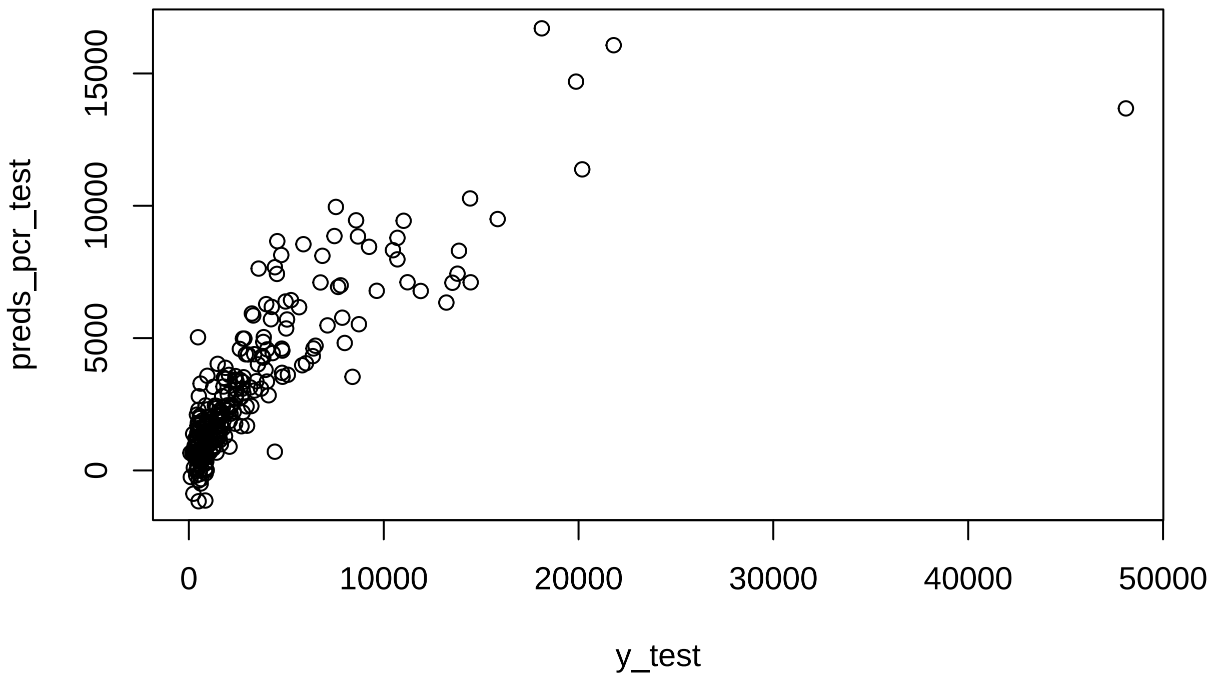
```
## [1] 1315.11
```

The predicted RMSE for the testing data is:

```
preds_plsr_test <- predict(plsr_model,newdata=x_test,ncomp=6)
#  RMSE
compute_rmse(y_test, preds_plsr_test)
```

```
## [1] 2801.622
```

```
plot(y_test, preds_pcr_test)
```



As for final conclusion we can say that PLSR method was better in therms of reducing the dimensions and also scoring less for the test data as the basic Linear model and PCR.

PCR by hand

```
x_scaled <- scale(x_train, scale = TRUE, center = TRUE)
lm_scaled <- lm(y_train ~ ., data=as.data.frame(x_scaled))
```

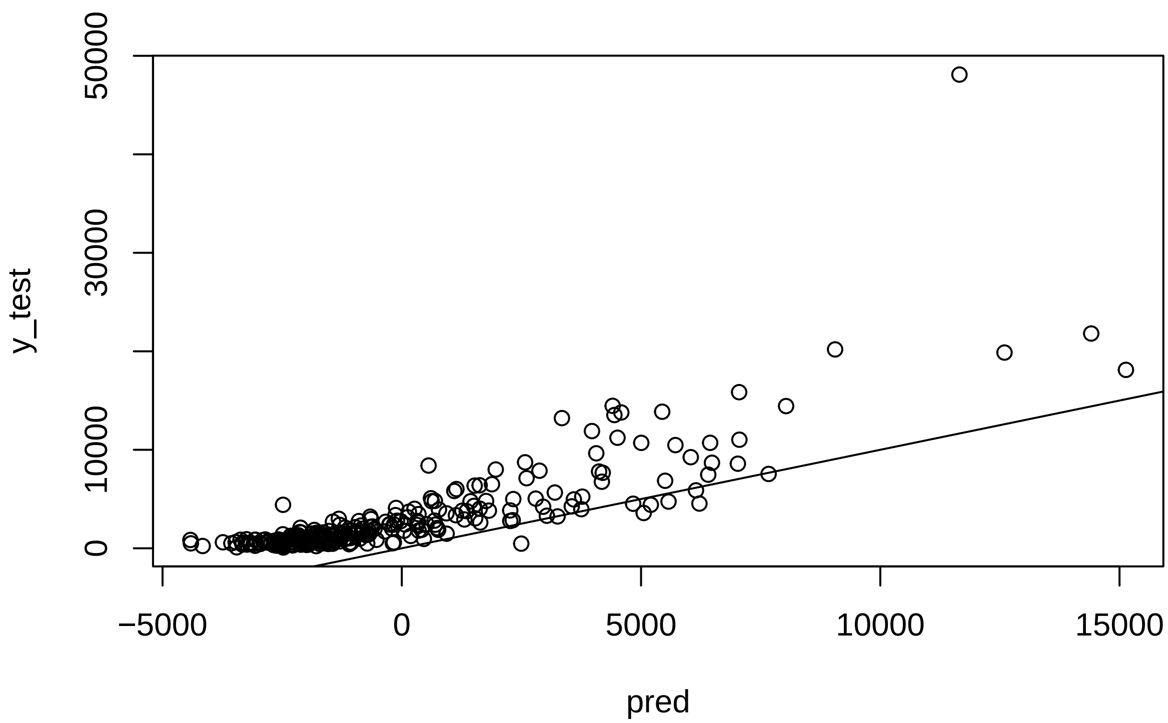
```
X <- model.matrix(lm_scaled)

X[, 1] <- numeric(length = nrow(X))
princ <- princomp(X)
princ
```

```
## Call:
## princomp(x = X)
##
## Standard deviations:
##    Comp.1    Comp.2    Comp.3    Comp.4    Comp.5    Comp.6    Comp.7    Comp.8
## 2.3261177 1.7092532 1.0846513 0.9806747 0.9169325 0.8204632 0.7715087 0.7578753
##    Comp.9   Comp.10   Comp.11   Comp.12   Comp.13   Comp.14   Comp.15   Comp.16
## 0.6615545 0.6158089 0.5686385 0.5231453 0.4247307 0.3721949 0.2900427 0.0000000
##
##  16  variables and  518 observations.
```

```r
scores <- princ$scores[, 1:13]
loadings <- princ$loadings[, 1:13]
Z_t <- scores
theta <- solve(t(Z_t) %*% Z_t) %*% t(Z_t) %*% y_train

x_test_scaled <- scale(x_test, scale = TRUE, center = TRUE)
X_t <- model.matrix(y_test ~ ., as.data.frame(x_test_scaled))
X_t[, 1] <- numeric(length = nrow(X_t))
Z_t <- X_t %*% princ$loadings
pred <- Z_t[, 1:13] %*% theta
plot(pred, y_test)
abline(a = 0, b = 1)
```




```r
rmse_pcr_man <- compute_rmse(y_test, pred)
rmse_pcr_man
```

```
## [1] 4175.606
```