Exercise 3

Nikolaus Czernin

```
# install.packages("ISLR")
library("ISLR")
library("tidyverse")
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.6     v purrr   0.3.4
## v tibble  3.1.7     v dplyr   1.0.9
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library("knitr")
# install.packages("pls")
library("pls")
```

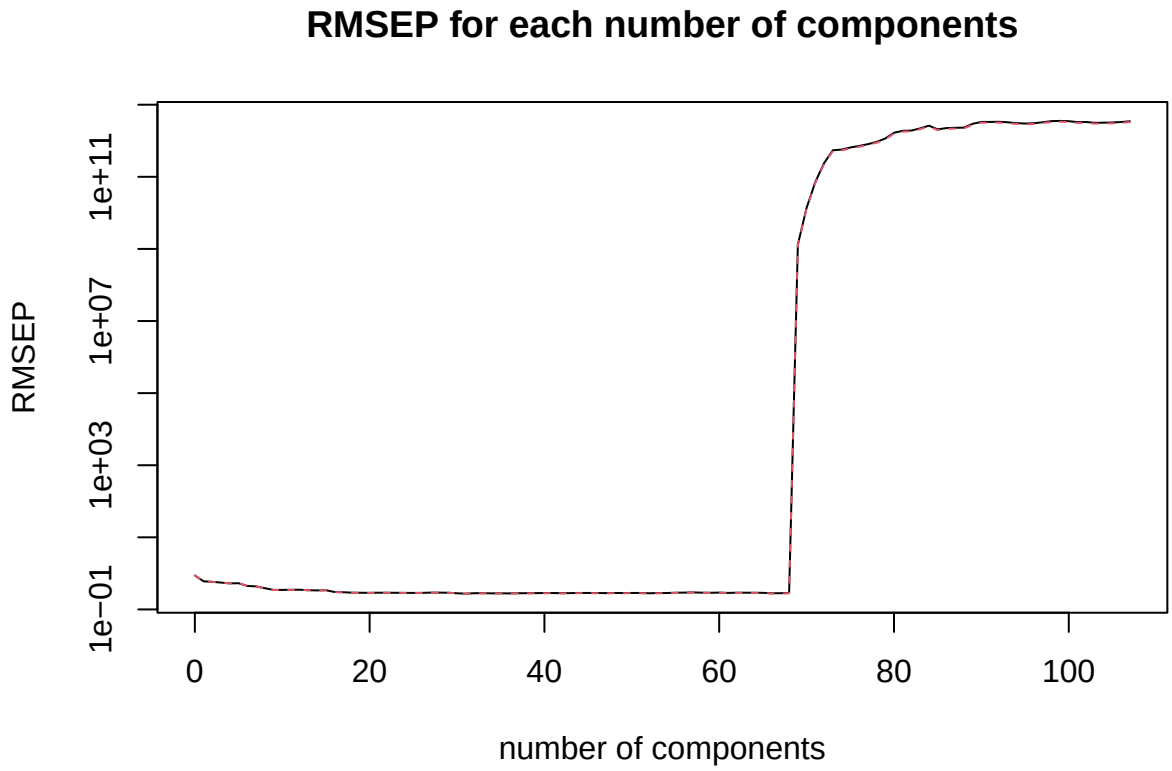
```
##
## Attaching package: 'pls'

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

```
load("building.RData")
# df %>% head()
N <- nrow(df)
train_ids <- sample(1:N, (N %/% 3) * 2)
train <- df[train_ids, ]
test <- df[-train_ids, ]
dim(df)
```

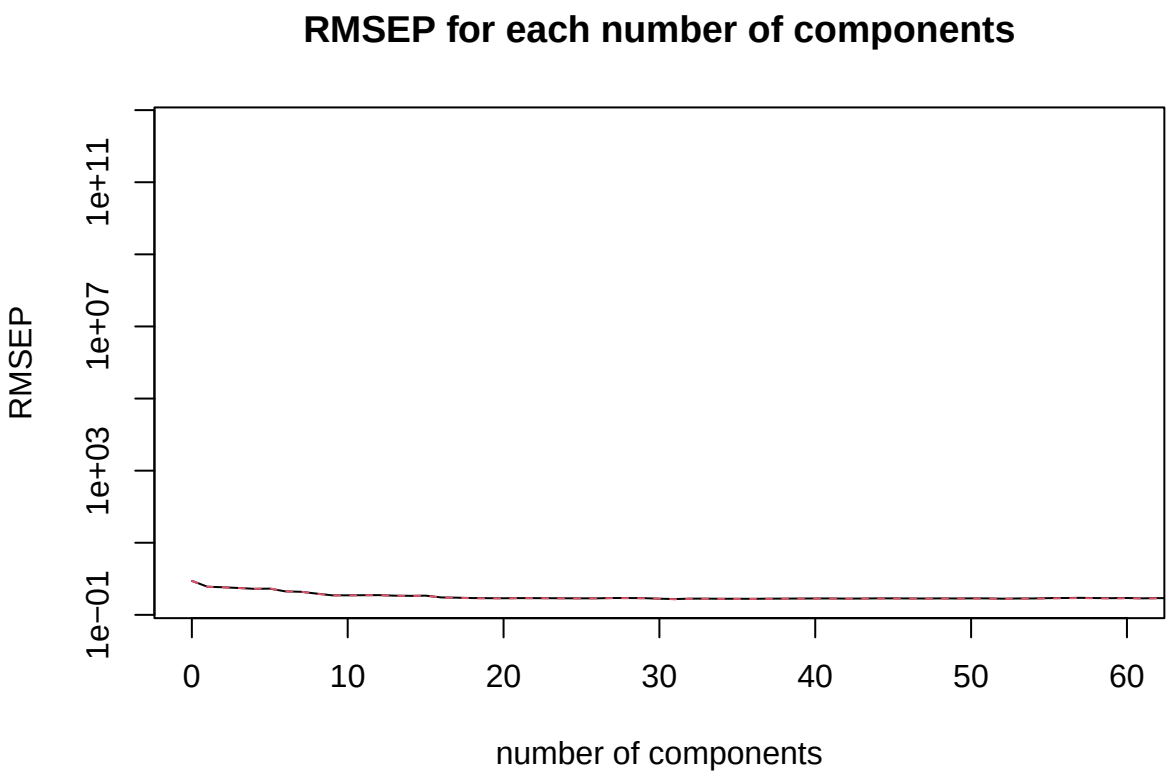
```
## [1] 372 108
```

1: PCA

```r
# ?pcr

model.pcr <- pcr(y ~ ., data=train, scale=T ,
                 validation="CV", segments=10)
# summary(model.pcr)
validationplot(model.pcr, val.type = "RMSE", main = "RMSEP for each number of components", log = "y")
```

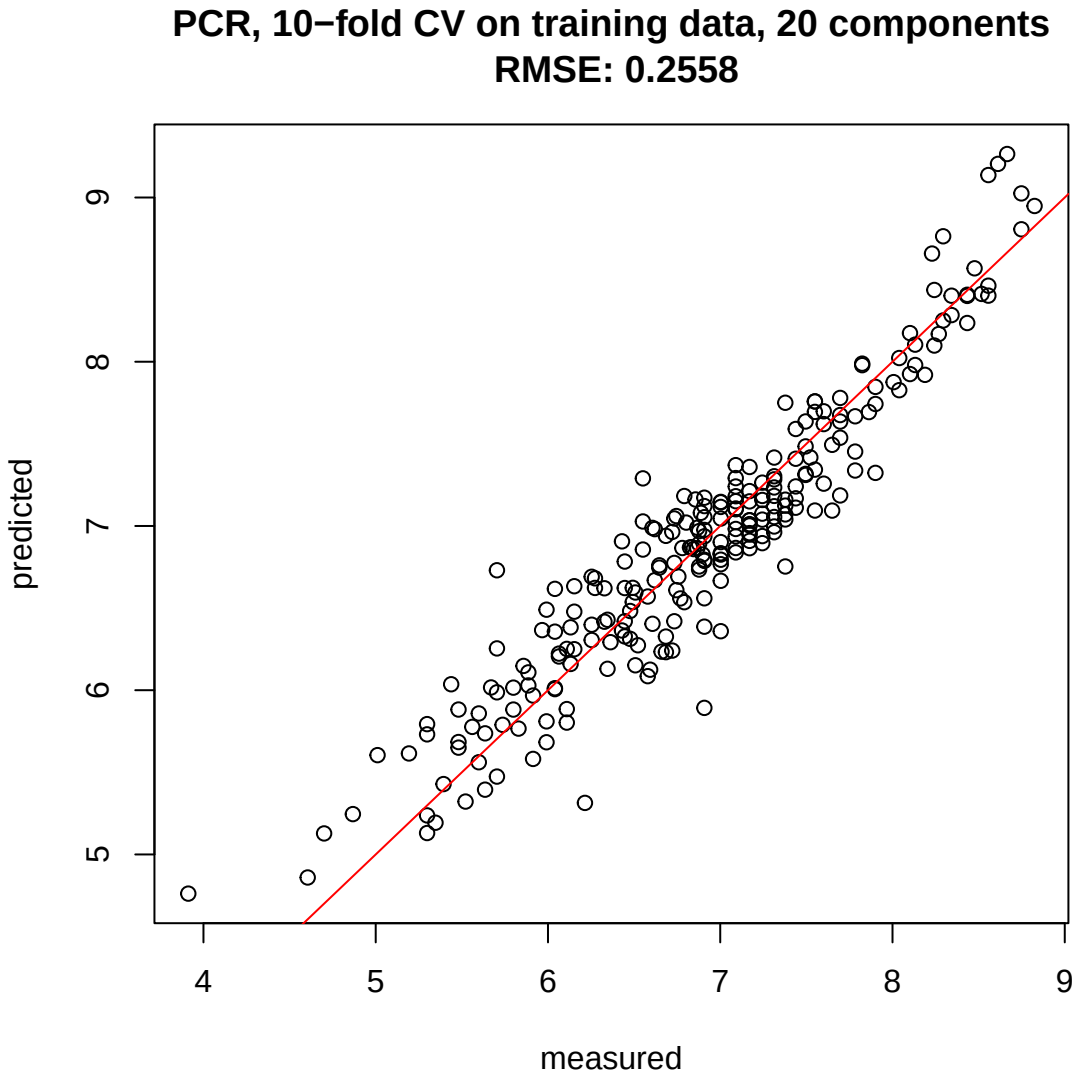



```r
validationplot(model.pcr, val.type = "RMSE", main = "RMSEP for each number of components", log = "y", xl
```

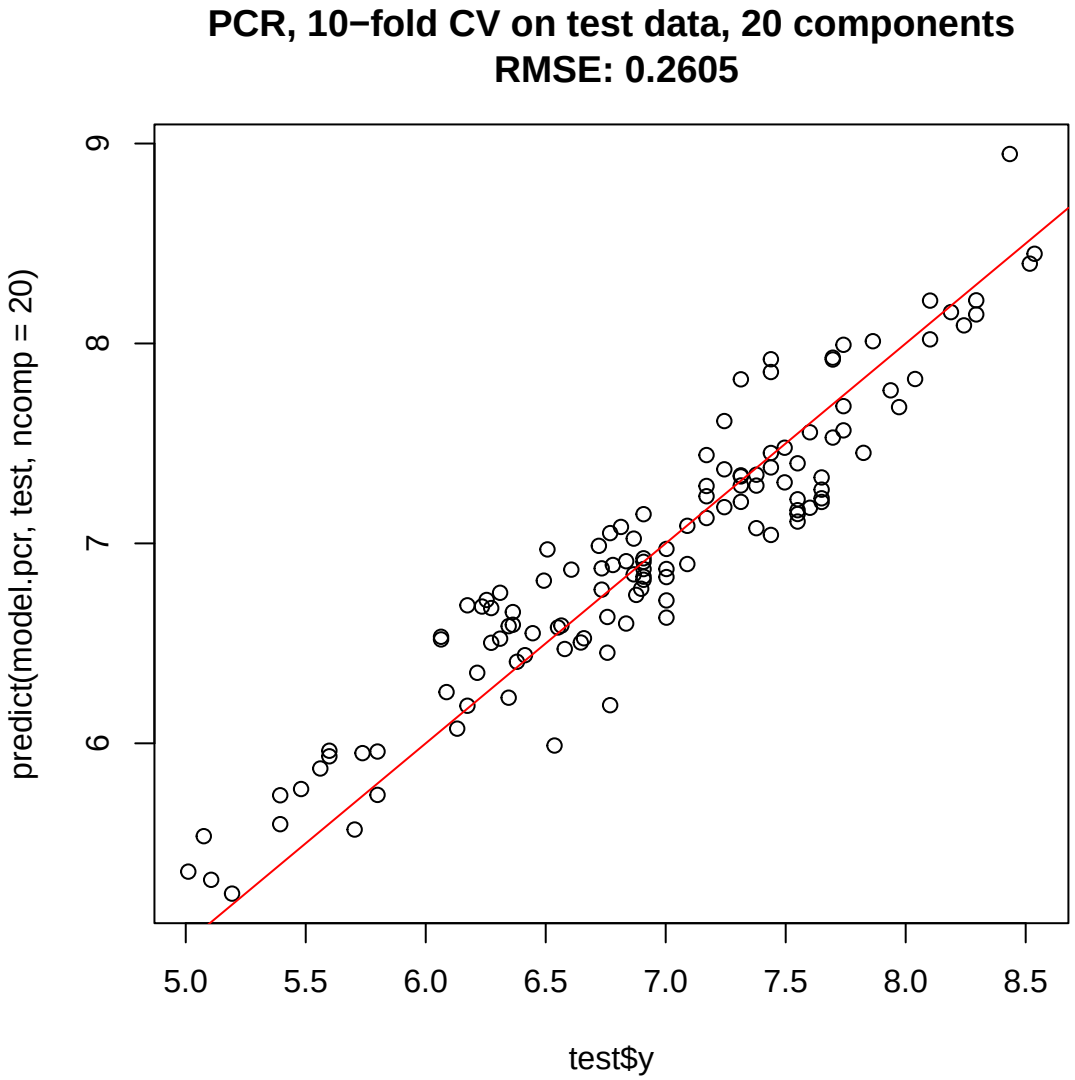


From lookin at the validation plot, I feel like anything between 20 and 65 components is probably fine, but since we are trying to minimize the number of variables, I am going to go with ~20 components.

```
rmse <- function(y, yhat, r=4){
  sqrt(mean((y-yhat)^2)) %>% round(4)
}


predplot(model.pcr, ncomp = 20, main="PCR, 10-fold CV on training data, 20 components\n RMSE:" %>% paste

abline(coef = c(0,1), col="red")
```

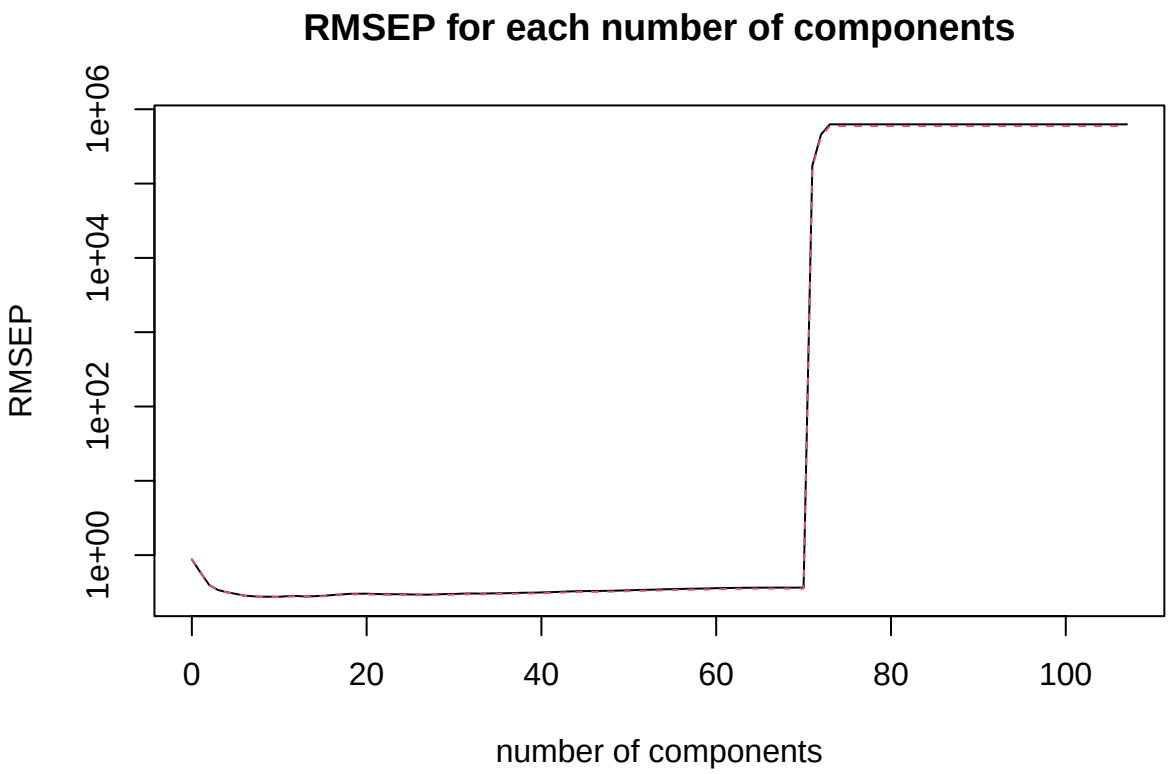



```
plot(test$y, predict(model.pcr, test, ncomp=20), main="PCR, 10-fold CV on test data, 20 components\n RMS

abline(coef = c(0,1), col="red")
```

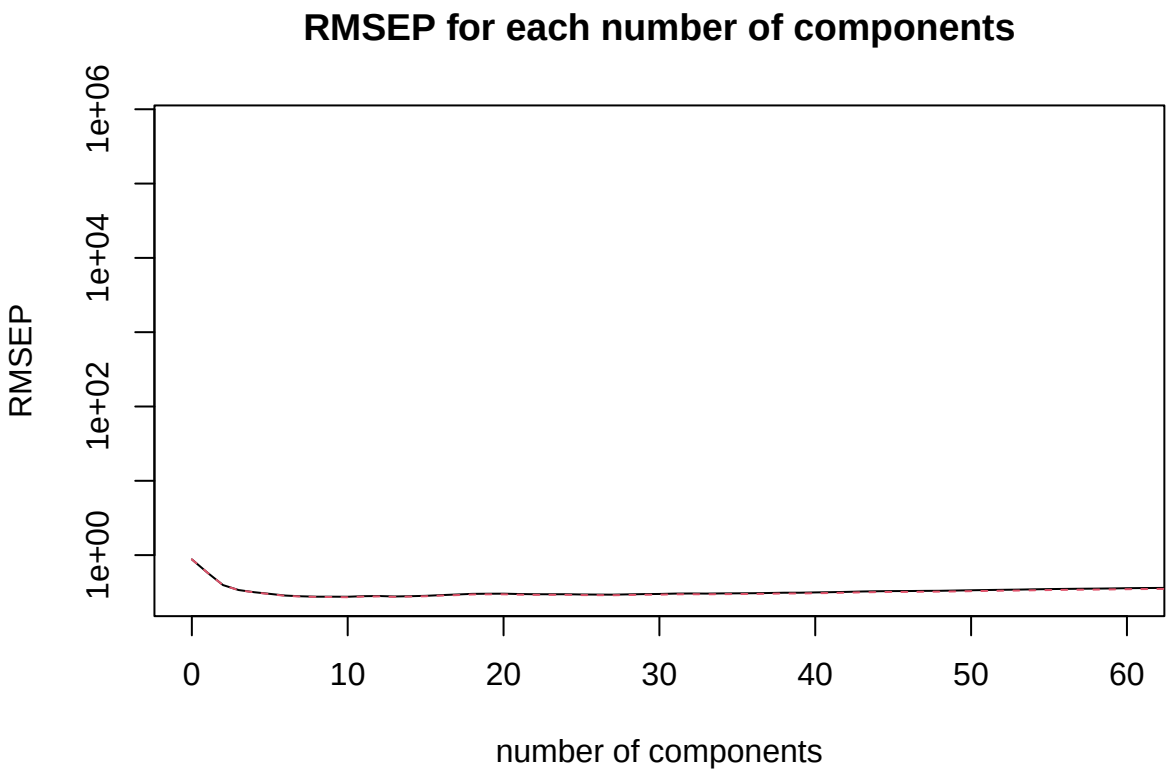


2: PLS

```
model.pls <- plsr(y ~ .,  data=train, scale=T ,
                  validation="CV", segments=10)
# summary(model.pls)
validationplot(model.pls, val.type = "RMSE", main = "RMSEP for each number of components", log = "y")
```

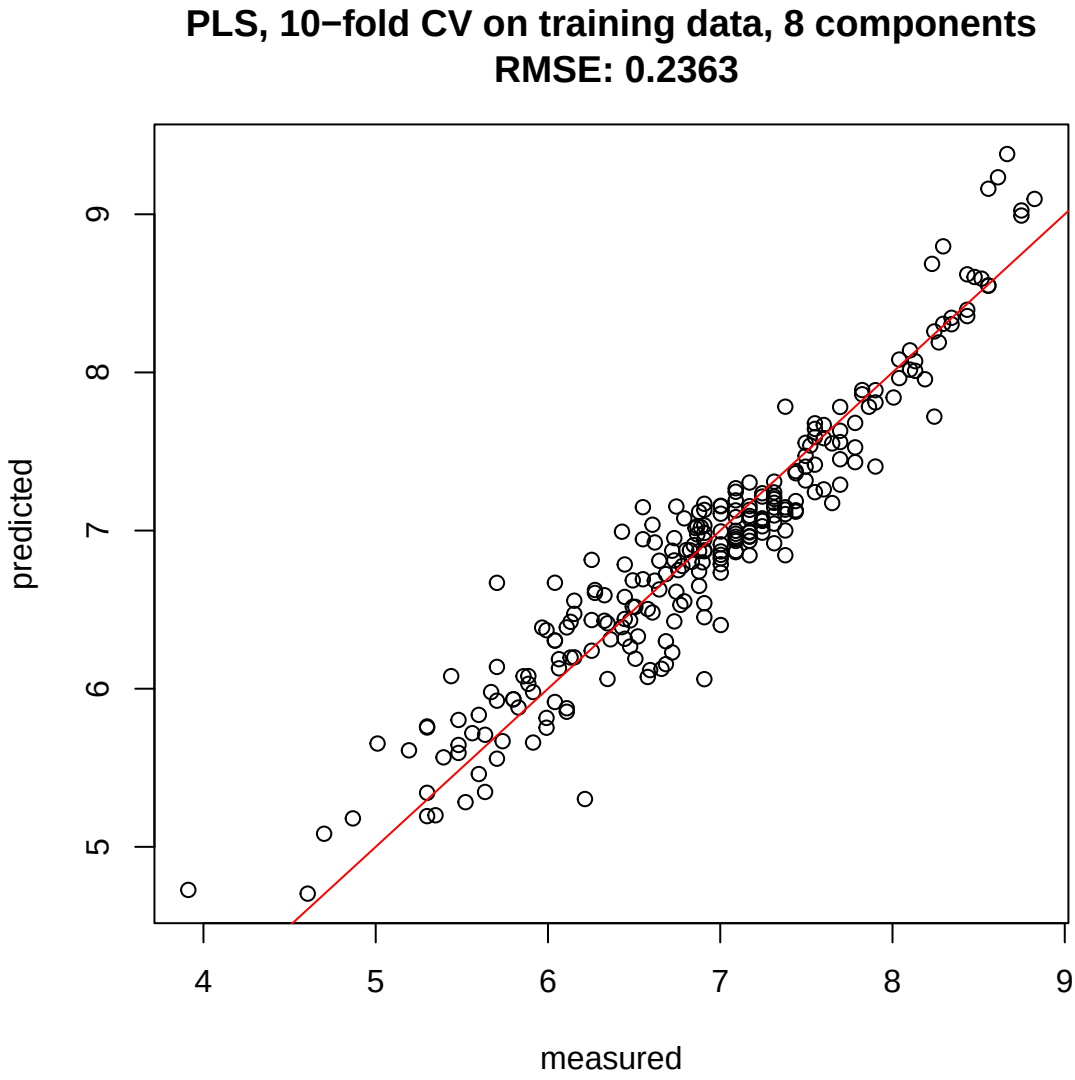



```
validationplot(model.pls, val.type = "RMSE", main = "RMSEP for each number of components", log = "y", xl
```



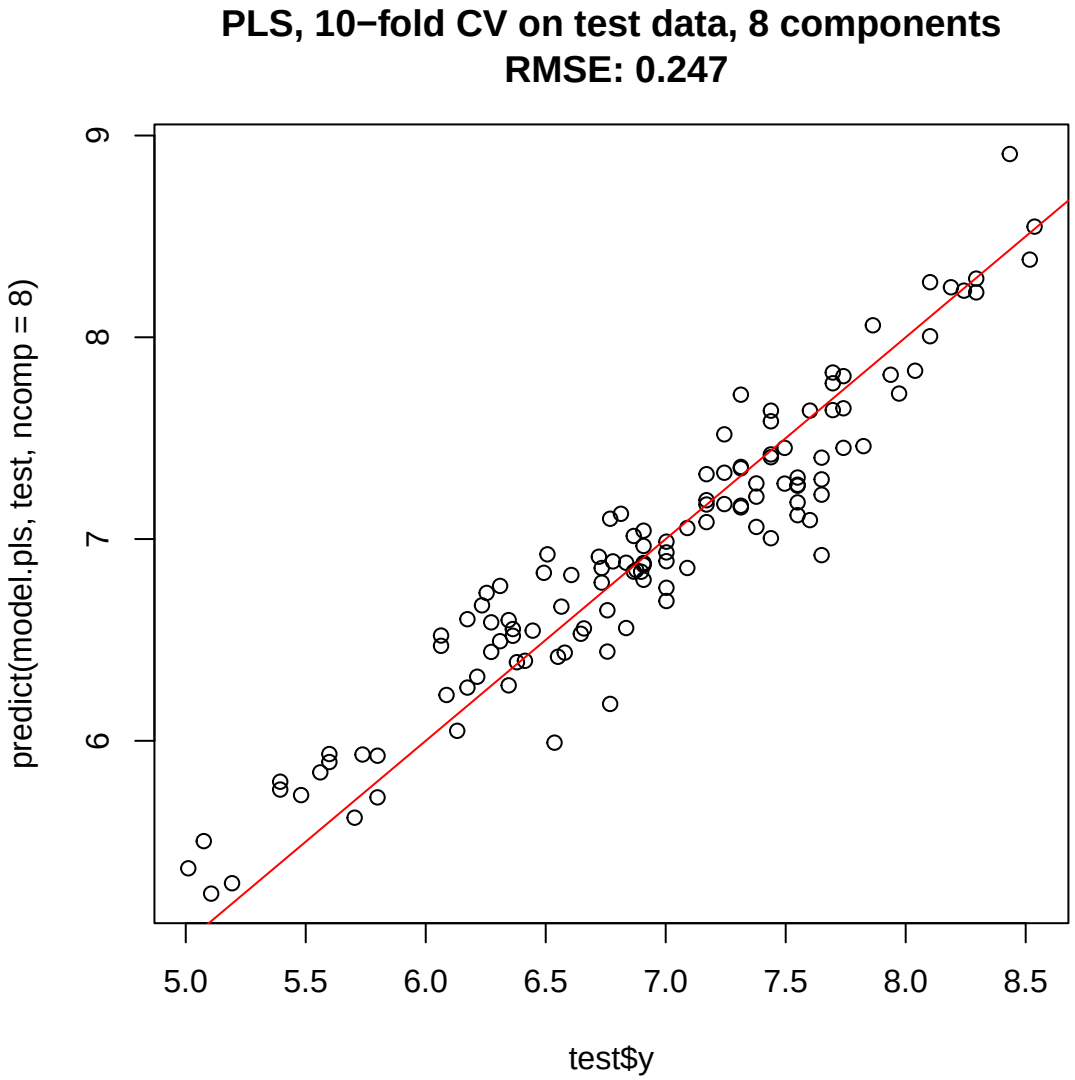
Again, the validation plot is more easily leggible when limiting the axis view. Visually, it seems that a very low number of components, even as low as 7 ort 8 may be optimal.

```
predplot(model.pls, ncomp = 8, main="PLS, 10-fold CV on training data, 8 components\n RMSE:" %>% paste(

abline(coef = c(0,1), col="red")
```



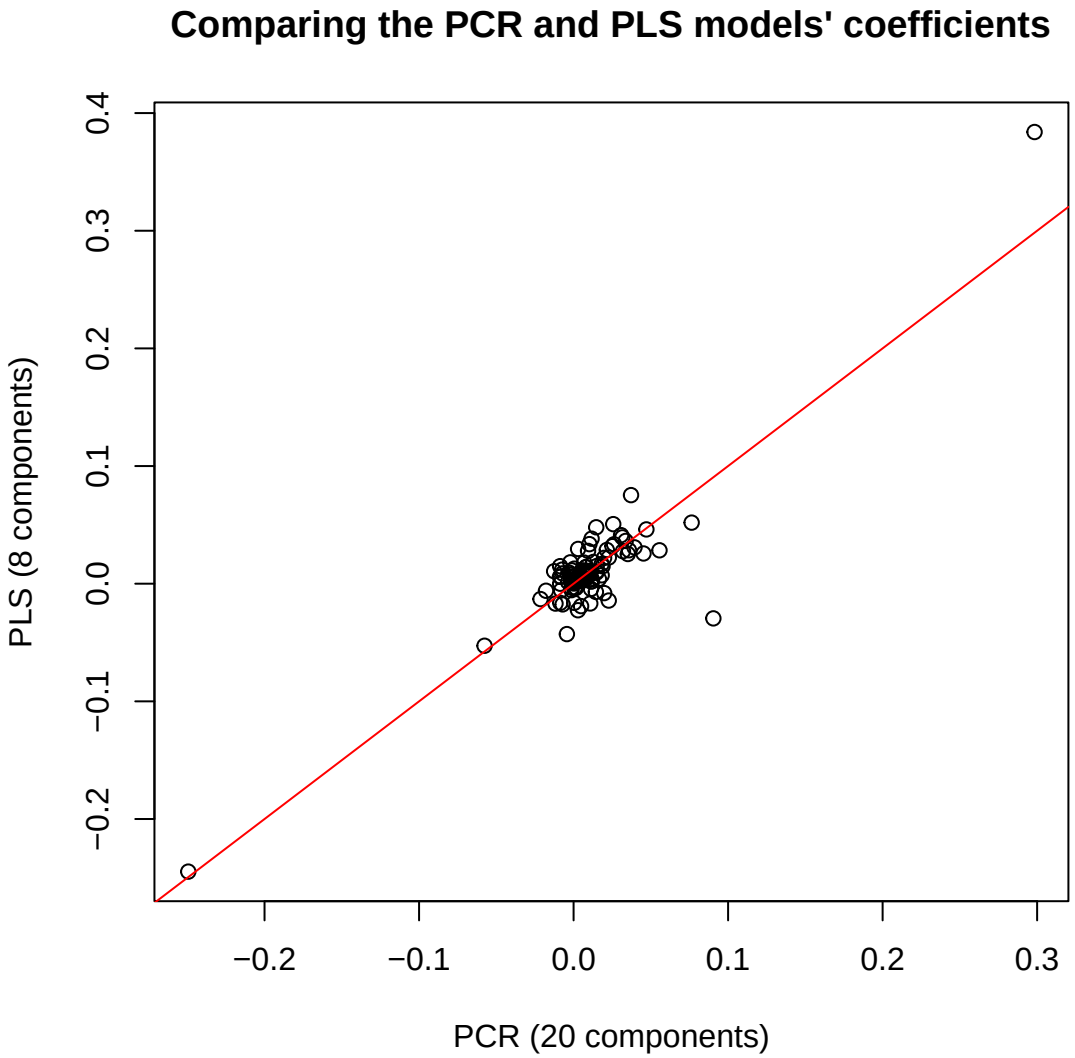
The fit of the PLS prediction plot looks a little better than that of the PCR model.

```
plot(test$y, predict(model.pls, test, ncomp=8), main="PLS, 10-fold CV on test data, 8 components\n RMSE

abline(coef = c(0,1), col="red")
```



I do not see an obvious improvement of the test prediction from changing model from PCR to PLS.

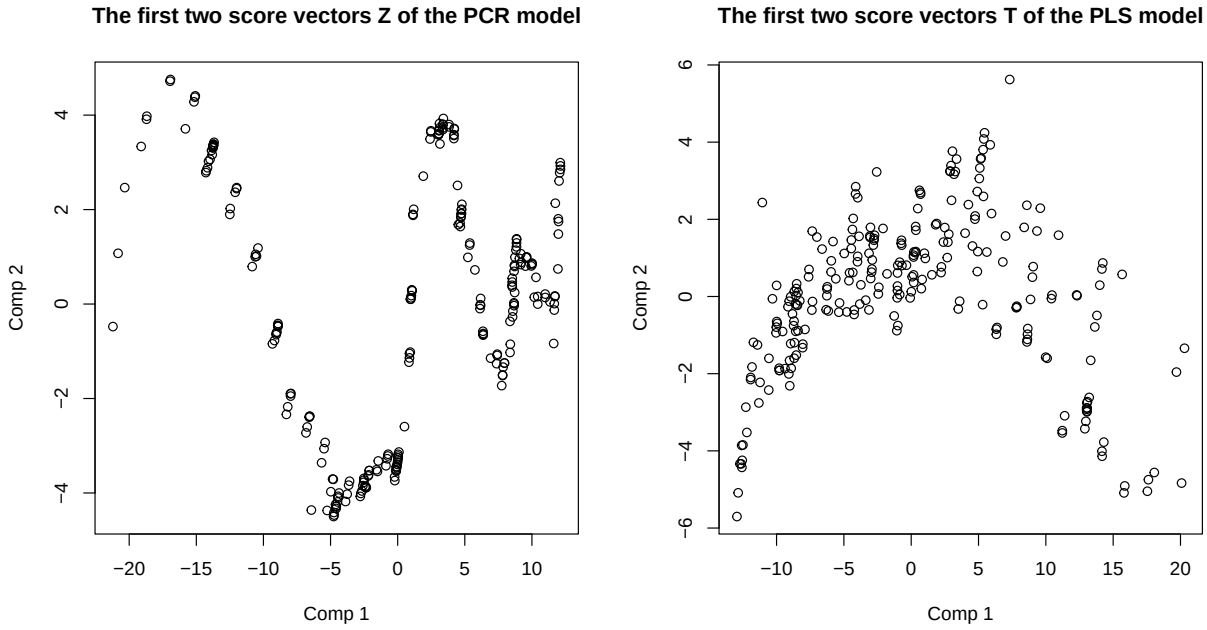
```
plot(
  model.pcr %>% coef(ncomp = 20),
  model.pls %>% coef(ncomp = 8),
  main="Comparing the PCR and PLS models' coefficients",
  xlab="PCR (20 components)", ylab="PLS (8 components)"
  )
abline(coef = c(0,1), col="red")
```



I plotted a scatterplot with the PCR coefficients on the x-axis and the PLS coefficients on the y-axis. The red line is where the model agree. Since most points are pretty close to the red line, I assume that the models did not yield results too differently. Notably, there are coefficient way higher (absolute value) than the others, which PLS estimate a little higher than PCR. Otherwise, I see a general agreement.

Scores

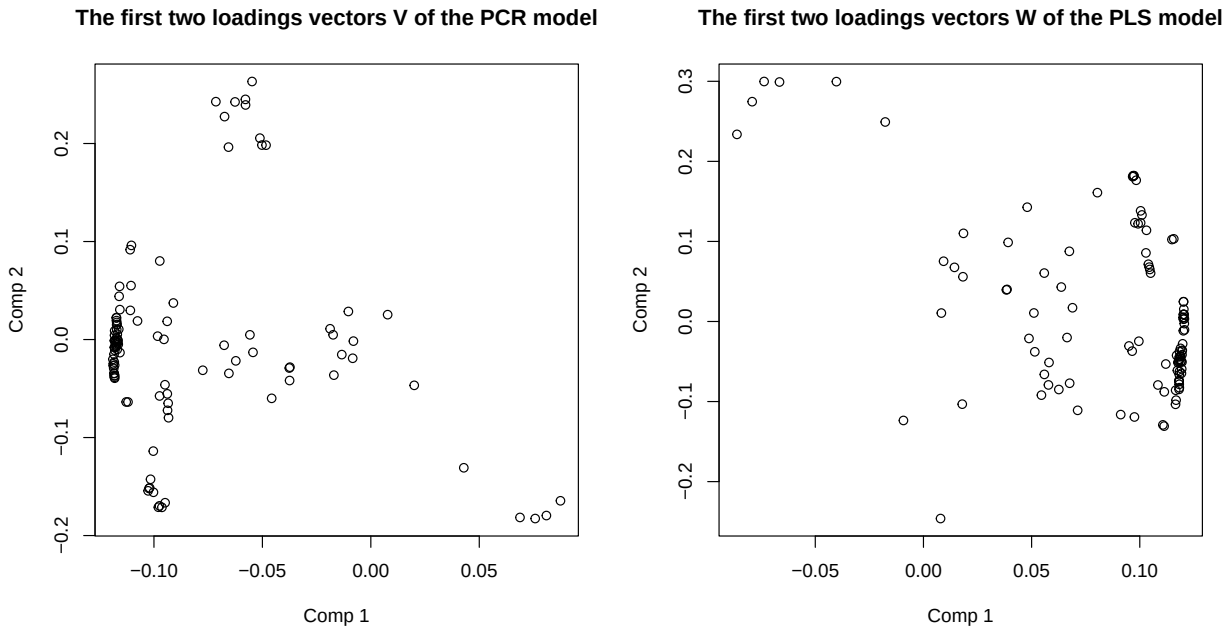
```
par(mfrow = c(1, 2))  # Arrange plots in a 2x2 grid
plot(model.pcr$scores[,1:2], main="The first two score vectors Z of the PCR model")
plot(model.pls$scores[,1:2], main="The first two score vectors T of the PLS model")
```



The scores are the actual values of the transformation vector to multiply the variable values of the observed matrix with. When plotting the scores of PCR for 1 component and 2 components against each other, they drawn points almost appear to create an oscillating curve function.

Loadings

```
par(mfrow = c(1, 2))  # Arrange plots in a 2x2 grid
plot(model.pcr$loadings[,1:2], main="The first two loadings vectors V of the PCR model")
plot(model.pls$loadings[,1:2], main="The first two loadings vectors W of the PLS model")
```



The loadings outline the contribution to explaining a true models variance of the components in question. The loadings of the PCR and PLS model for a single component kind of cluster around -0.1 and 0.1 respectively.