Advanced Methods for Regression and Classification

Exercise 3

Bosse Behrens , st.id: 12347333

Task 1

```r
load("building.RData")
library(pls)
```

```
##
## Attache Paket: 'pls'

## Das folgende Objekt ist maskiert 'package:stats':
##
##     loadings
```

Setting up the training and testing data split the same as in exercise 2.

```r
set.seed(1234)
n <- nrow(df)
train <- sample(1:n, round((2/3) * n))
test<-(1:n)[-train]
```

a)

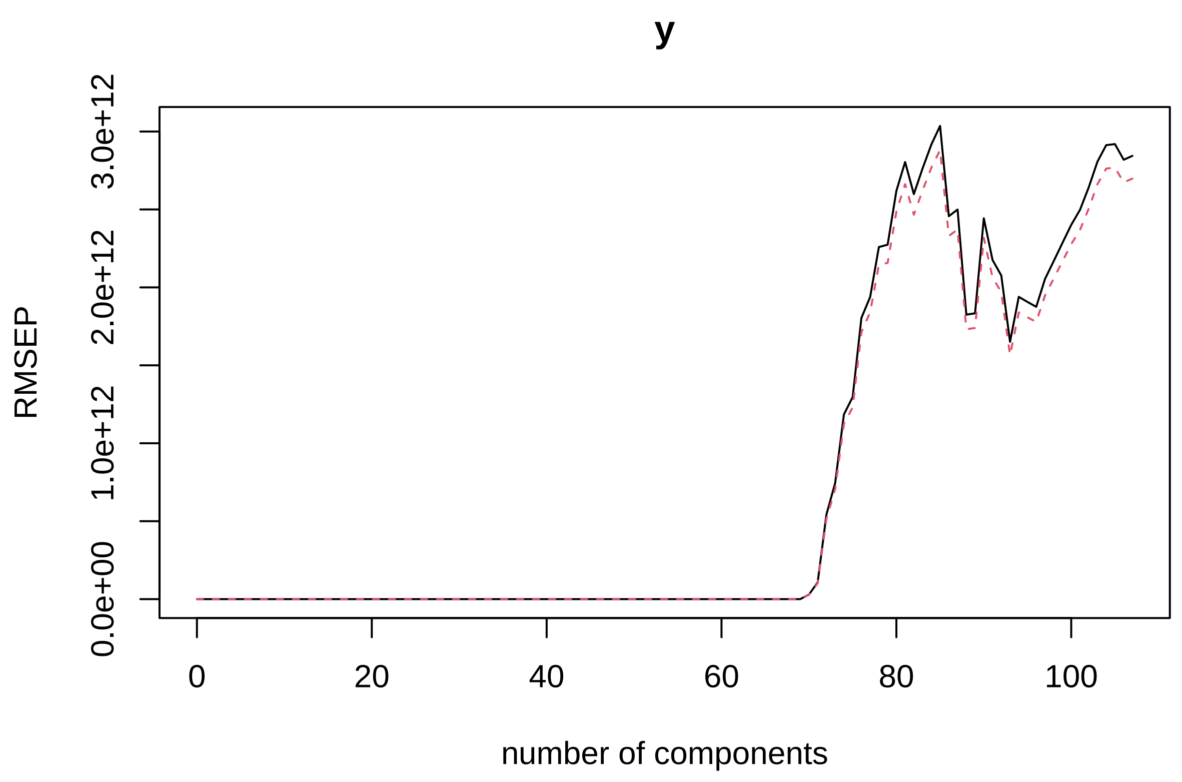
Creating the principal component regression model.

```r
model_pcr <- pcr(y ~ ., data=df, subset=train, scale=TRUE,validation="CV",segments=10,segment.type="ran
```

b)

Now we are plotting the obatined prediction errors from the cross-validations for the different number of principla components.

```r
validationplot(model_pcr, val.type = "RMSEP")
```



As we can see, it is difficult to say on which number of components exactly the RMSE gets extremely large since until about 70 it is close to zero and then it suddenly has values in the range of 10^{12}. We therefore extract the RMSE values from 65 to 75 to take a closer look at them.

```
cv_rmse <- RMSEP(model_pcr, estimate = "CV")$val
rmse_subset <- cv_rmse[65:75]

rmse_results <- data.frame(Component = 65:75, RMSE = rmse_subset)
print(rmse_results)
```

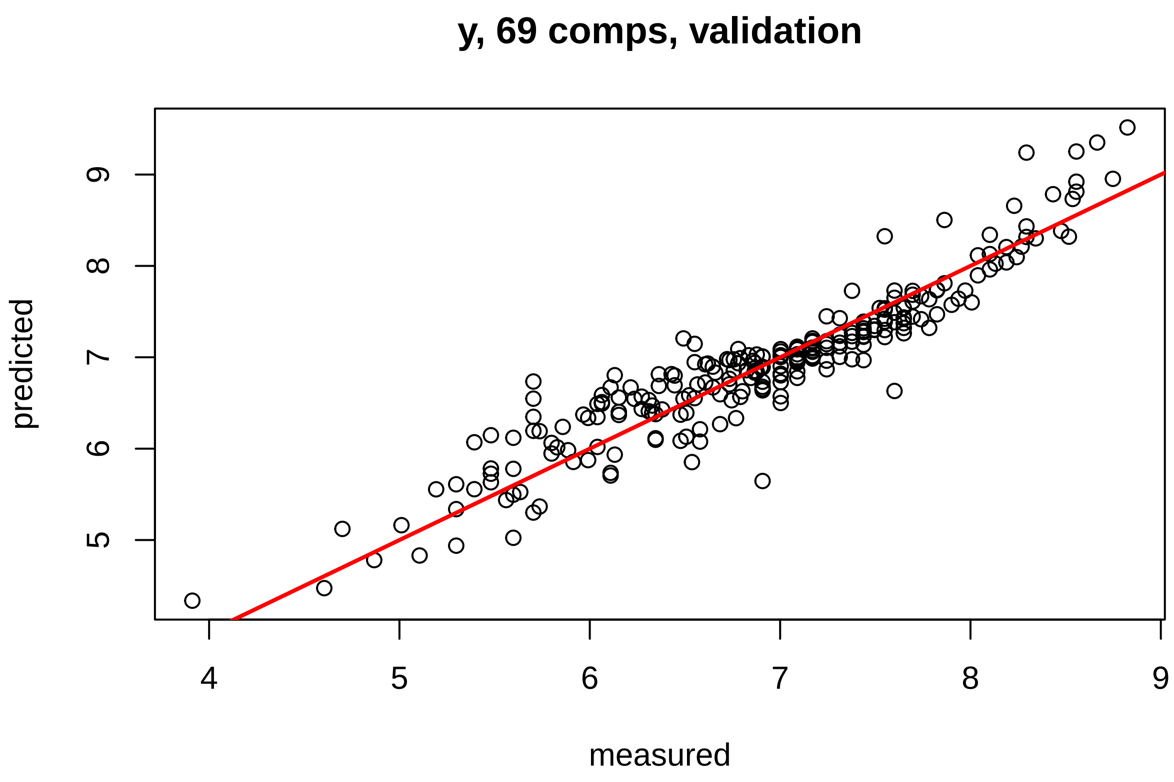
```
##    Component         RMSE
## 1         65 2.824277e-01
## 2         66 2.851095e-01
## 3         67 2.842372e-01
## 4         68 2.920903e-01
## 5         69 3.004906e-01
## 6         70 3.130520e-01
## 7         71 2.907373e+10
## 8         72 1.075212e+11
## 9         73 5.427718e+11
## 10        74 7.455047e+11
## 11        75 1.184718e+12
```

As we can see for 70 components the RMSE is at ≈ 0.313 and at 71 components at $> 2.9 \cdot 10^{10}$. Additionally in this list the intercept is interpreted as one component, so the optimal number we want to extract is 69 (plus the intercept) with the RMSE of ≈ 0.313.

c)

We now plot the measured values against the cross-validated values of y using predplot() with 69 components.

```r
predplot(model_pcr, ncomp = 69)
abline(0, 1, col = "red", lwd = 2)
```

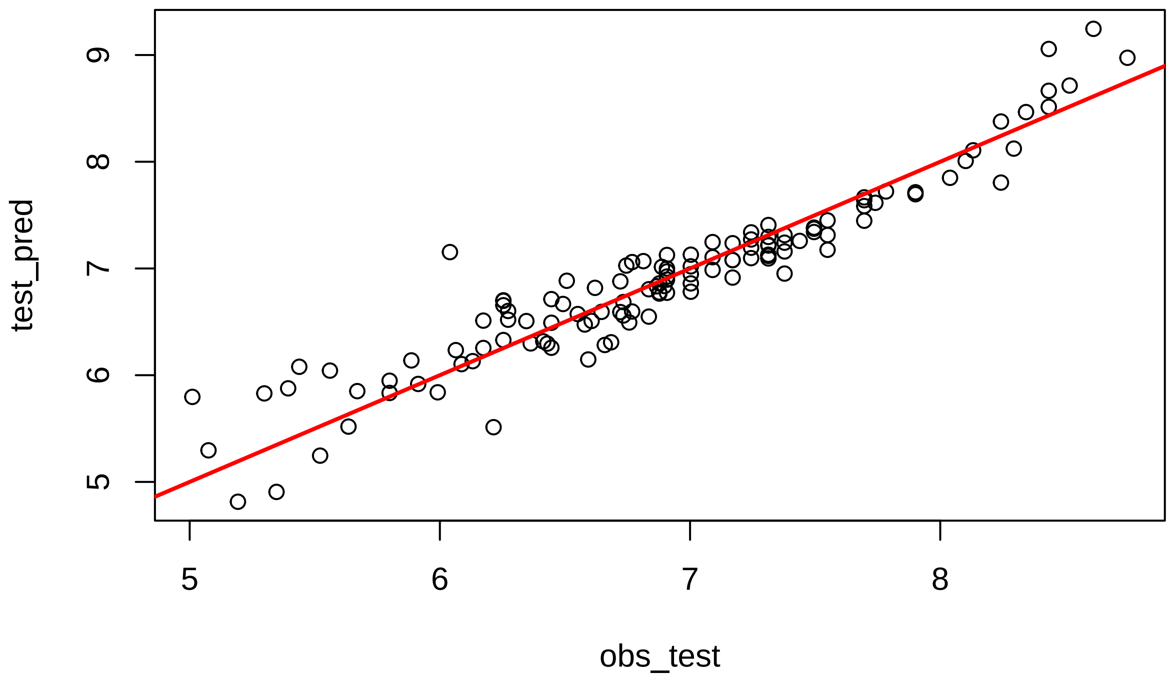


As we can see the model seems now well-suited at least to the training data and the look small and random.

d)

We now also plot the measured values of y against the predicted ones using our pcr-model for the testing data.

```r
test_pred <- predict(model_pcr, newdata=df[test, ], ncomp=69)
obs_test <- df[test, "y"]

plot(obs_test, test_pred)
abline(0, 1, col = "red", lwd = 2)
```



The plot looks similar to the one for the training data so the model seems to be well-suited to the data now. We now also compute the RMSE for the test data.

```
n_test <- length(obs_test)

RMSE_test <- sqrt((1/n_test)*sum((obs_test - test_pred)^2))
RMSE_test
```

```
## [1] 0.2671537
```

The RMSE for the test data also is small enough to evaluate the model as well-suited, being even slightly smaller than for the training set.

Task 2

a)

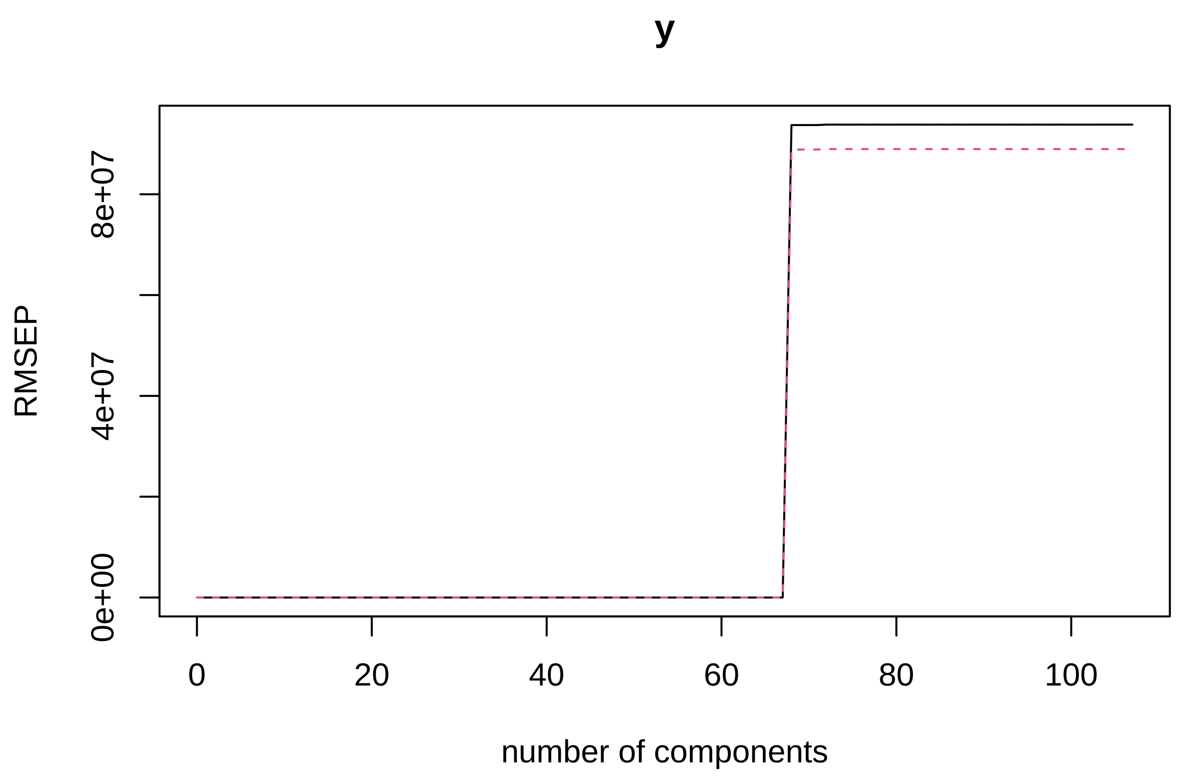
We create a second model similar to $1(a)$ now with partial least squares by using the pls() function.

```
model_pls <- plsr(y ~ ., data=df, scale=TRUE, subset=train, validation="CV", segments=10, segment.type=
```

b-d)

We do the same steps as in $1(b) - (d)$ with the pls-model.

```
validationplot(model_pls, val.type = "RMSEP")
```



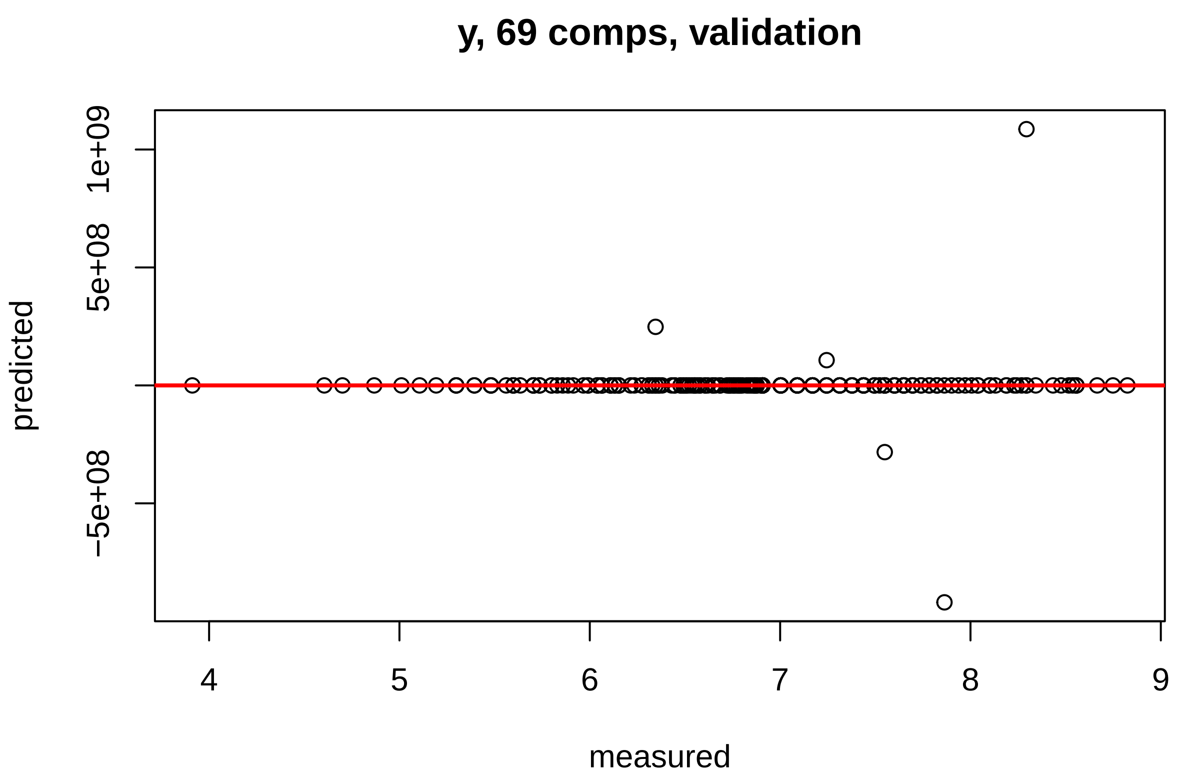
```
cv_rmse_pls <- RMSEP(model_pls, estimate = "CV")$val
rmse_subset_pls <- cv_rmse[65:75]

rmse_results_pls <- data.frame(Component = 65:75, RMSE = rmse_subset_pls)
print(rmse_results_pls)
```

```
##    Component         RMSE
## 1         65 2.824277e-01
## 2         66 2.851095e-01
## 3         67 2.842372e-01
## 4         68 2.920903e-01
## 5         69 3.004906e-01
## 6         70 3.130520e-01
## 7         71 2.907373e+10
## 8         72 1.075212e+11
## 9         73 5.427718e+11
## 10        74 7.455047e+11
## 11        75 1.184718e+12
```

Again it is somewhat difficult to get the optimal number of components only from looking at the CV-error plot, so we again print out the exact RMSE values for 65 to 75 components. This shows that for the pls-model the optimal number is also 69 (70 including the intercept) with an RMSE of again ≈ 0.313. The

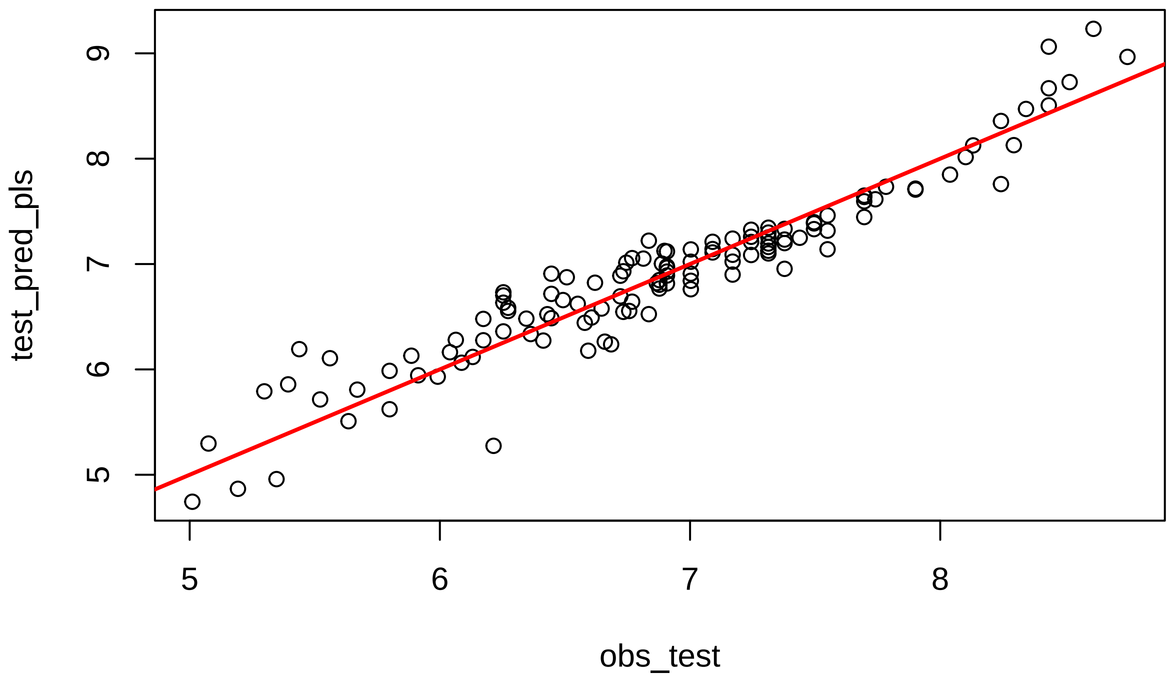
CV-error plot looks somewhat similar to the one from task 1 with the RMSE being close to zero for up to 69 components and getting extremely large starting at 70 though this time the values stay more or less constant for larger nummers of components, while in the pcr-model it was still going up and down a bit. We now plot the measured vs the cross-validated values of y again.

```r
predplot(model_pls, ncomp = 69)
abline(0, 1, col = "red", lwd = 2)
```



This plot also looks very similar to the one of the pcr-model with small and random erros, except for some slight outliers for the larger values. We now plot the same for the testing data.

```r
test_pred_pls <- predict(model_pls, newdata=df[test, ], ncomp=69)
obs_test <- df[test, "y"]

plot(obs_test, test_pred_pls)
abline(0, 1, col = "red", lwd = 2)
```



Again the errors of the observed vs predicted y-values are small and random, suggesting a model well-suited to the data. Now we also compute the RMSE of the test data.

```r
n_test <- length(obs_test)

RMSE_test_pls <- sqrt((1/n_test)*sum((obs_test - test_pred_pls)^2))
RMSE_test_pls
```

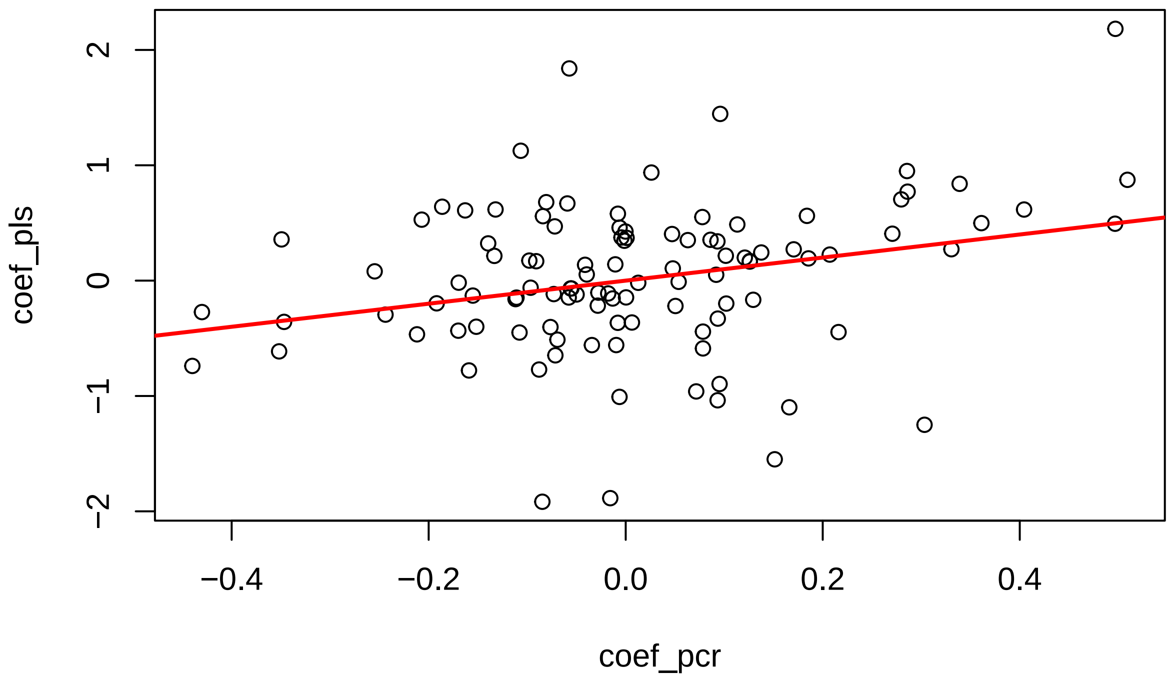
```
## [1] 0.2534891
```

The RMSE for the test data on the pls-model is also small, suggesting the same result as the evalutaion from the training data. Overall the two models seem to be performing very similar.

e)

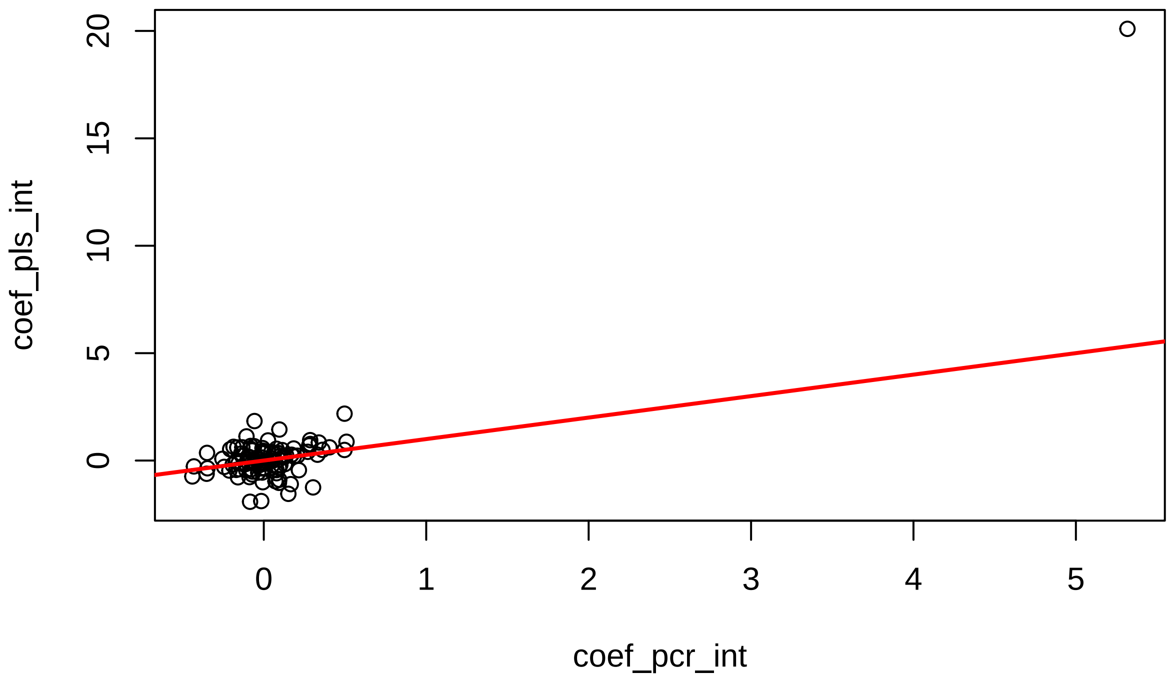
We now want to compare the coefficients of the pcr- and pls-model. We plot them against each other once excluding the intercept and once including it.

```r
coef_pcr <- coef(model_pcr, ncomp = 69, intercept = FALSE)
coef_pls <- coef(model_pls, ncomp = 69, intercept = FALSE)

plot(coef_pcr, coef_pls)
abline(0, 1, col = "red", lwd = 2)
```



When plotting the resulting coefficients against each other (excluding the intercept), they do differ (else they would all be on one line), but no pattern can be discerned from the plot as they seem to be randomly scattered around the $x = y$ line. NOw we do the same but include the intercept.

```
coef_pcr_int <- coef(model_pcr, ncomp = 69, intercept = TRUE)
coef_pls_int <- coef(model_pls, ncomp = 69, intercept = TRUE)

plot(coef_pcr_int, coef_pls_int)
abline(0, 1, col = "red", lwd = 2)
```



As we can see in this plot the intercept seems to be the only coefficient where a big difference can be observed. For the pcr-model it is ≈ 5 whie for the pls-model it is ≈ 20. Therefore we could in this used seed conclude that there seems to be an underlying difference in how both models work, in this case especially regarding the intercept, but since both models performed similarily well on the data we can not really say which method works better in this case.

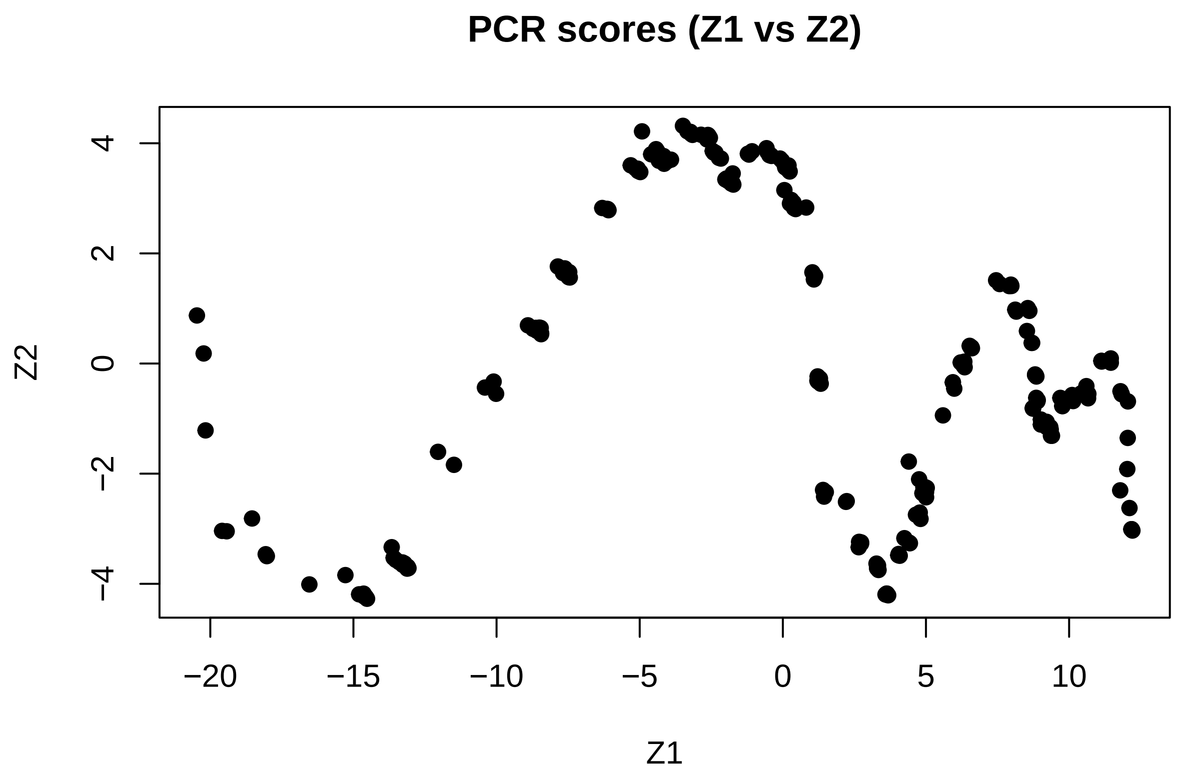
Task 3

From the previously created two models we extract the first two score and loading vectors of the pcr-model (Z_1, Z_2) and (V_1, V_2) and the same for the pls-model with (T_1, T_2) and (W_1, W_2).

```
scores_pcr <- model_pcr$scores[, 1:2]
loadings_pcr <- model_pcr$loadings[, 1:2]

scores_pls <- model_pls$scores[, 1:2]
loadings_pls <- model_pls$loadings[, 1:2]
```

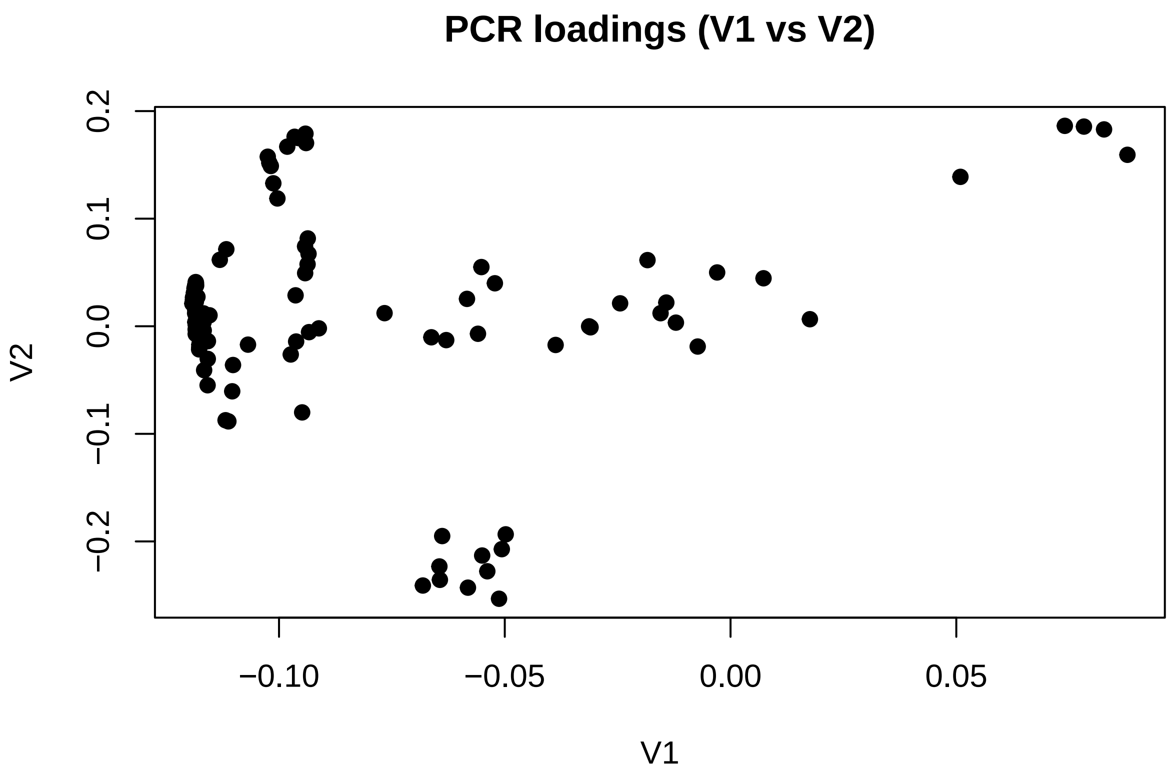
Now we plot each of the two first vectors respectively against each other in scastterplots.

```
plot(scores_pcr[, 1], scores_pcr[, 2],
     xlab = "Z1", ylab = "Z2",
     main = "PCR scores (Z1 vs Z2)", pch = 19)
```



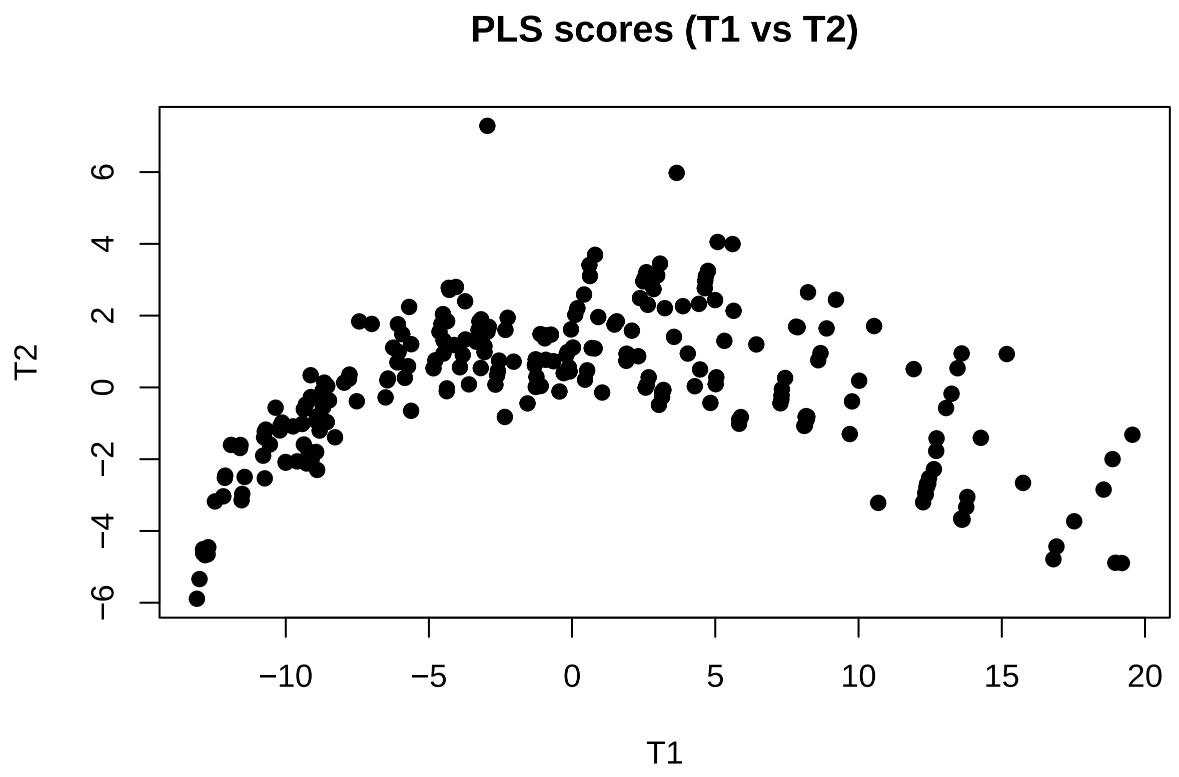
The plot for the first two score vectors of the pcr-model shows a specific pattern. The values follow some kind of oscilating line that gets narrower in the interval with larger values of Z1. First the general amount of variance these first two score vectores capture seems to be significant due to the values ranges. Furthermore the oscilating line suggest some kind of periodic/repeating pattern. This means at least that there is some underlying structure in the data that gets captured by this non-linear relationship of the first two score vectors. Also at some areas the points seem clustered more than in others along hte line.

```r
plot(loadings_pcr[, 1], loadings_pcr[, 2],
     xlab = "V1", ylab = "V2",
     main = "PCR loadings (V1 vs V2)", pch = 19)
```



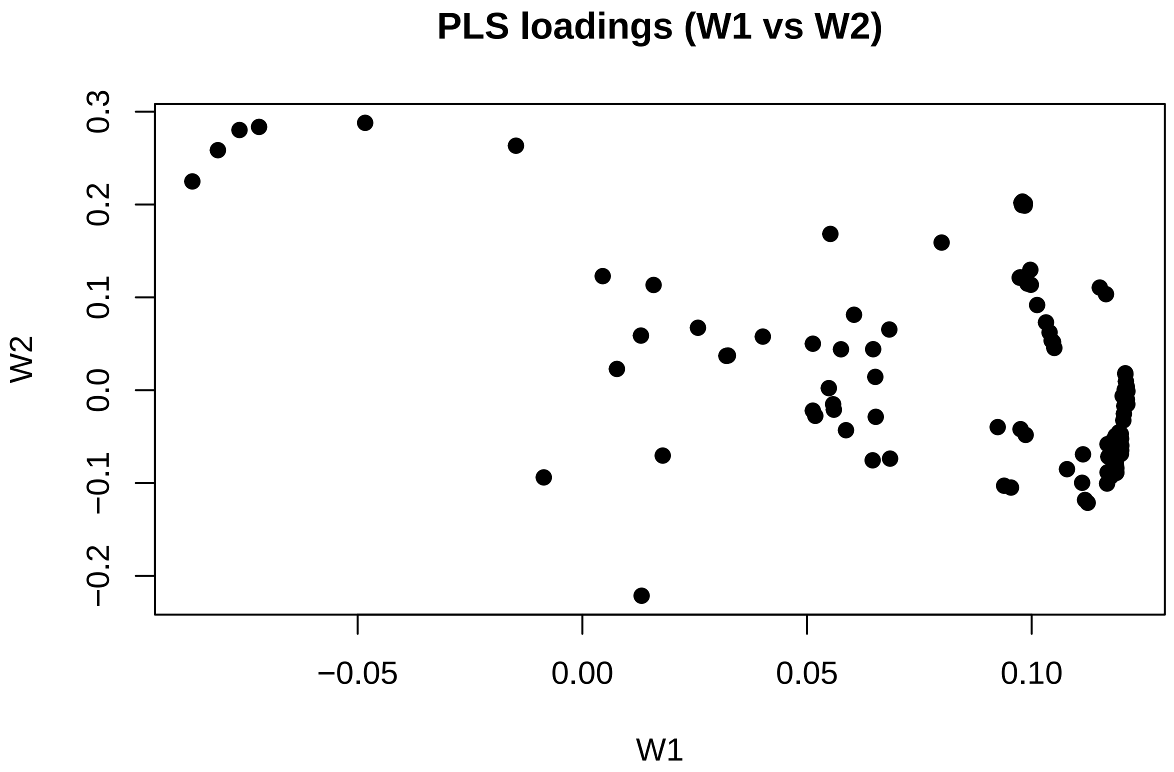
The plot of the first two loading vectors for the pcr-model shows not a clear functional pattern as for the score vectors, but there are definitely highly clustered areas of data points. First the spread on the x- and y-axis indicate that the contribution of the original predictors to these first two components vary much. The clusters can suggest that there are groups of original predictors that that contribute similarily to the components. Also variables with similar loadings suggest they could be correlated or have similar effects. The plot can also help identify which variables have a greater impact on the first two components than others.

```r
plot(scores_pls[, 1], scores_pls[, 2],
     xlab = "T1", ylab = "T2",
     main = "PLS scores (T1 vs T2)", pch = 19)
```



The data points for the first two score vectors of the pls-model follow also a curved line, but more in the terms of something like a parabolic or logistic function line but spreading out more around that line for greater values of T1. This suggests that there is again an underlying strucutre and non-linear relationship between the score vectors. The spread is more centered around zero this time. SInce pls is also maximizing covariance with the repsonse variable y, this could mean there is also some specific pattern reagrding the respsonse varaible.

```r
plot(loadings_pls[, 1], loadings_pls[, 2],
     xlab = "W1", ylab = "W2",
     main = "PLS loadings (W1 vs W2)", pch = 19)
```



The plot of the loading vectors for the pls-model is in so far similar as that there is no clear functional pattern discernible. There are again some clusters visible. Since in pls the loadings have more to do with the predicitiong power of the predicotrs in regard to the response, this could mean that certain groups of predictors have a similar impcat on the response and in general similar predicitng power. Higher values owuld mean in this regard which original variables have more influence on the score vectors in regard to the covarinace with the response.