Exercise 3, AMRC

11912007 - Yahya Jabary

Analysis

Task 1: Principal component regression (PCR)

For this exercise, I implemented Principal Component Regression (PCR) on the residential building dataset. I first split the data into training (2/3) and test (1/3) sets using random sampling.

Looking at the cross-validation results with 10 folds, the prediction error plot shows an interesting pattern. The RMSE decreases rapidly with the first few components but then starts to level off. The optimal number of components appears to be around 37.

The cross-validated predictions vs. measured values plot shows a pretty decent linear relationship, which is what we want to see. There's some scatter around the diagonal line, but overall the model seems to be capturing the main trends in the data. The points cluster nicely along the 45-degree line, suggesting our PCR model is doing a reasonable job at prediction.

When testing the model on the held-out test set, the measured vs. predicted values plot shows similar behavior to the cross-validation results. The scatter plot again demonstrates a clear linear relationship, though there's some variance in the predictions, particularly at the extremes of the range.

This analysis suggests that PCR is working reasonably well for this dataset, successfully reducing the dimensionality while maintaining good predictive power.

Task 2: Partial least squares regression (PLS)

For the Partial Least Squares regression analysis, I ran the model with 10-fold cross-validation on the same dataset. Looking at the cross-validation results, there's something really interesting going on. The optimal number of components turned out to be just 7, which is way lower than PCR's 37 components. This makes sense since PLS is specifically designed to find components that explain both X and y simultaneously.

The minimum RMSE from cross-validation came out to be about 0.286, which is slightly higher than PCR's 0.266. However, when looking at the test set performance, PLS achieved an RMSE of 0.242, which is practically identical to PCR's 0.239. This suggests that both methods are performing similarly in terms of prediction accuracy, but PLS is doing it much more efficiently with fewer components.

The cross-validation prediction plots show a good linear relationship between predicted and measured values. There's some scatter around the diagonal line (especially at the extremes), but that is to be expected for real-world data. The test set predictions follow a similar pattern.

When comparing the coefficients between PCR and PLS (as shown in the last plot), there are some noticeable differences in how the two methods weight the variables. This makes sense because PLS constructs its components differently, taking the response variable into account during the dimension reduction process. Some variables get substantially different weights between the two methods, which is interesting from a feature importance perspective.

Overall, PLS seems to be the more efficient choice here - it achieves basically the same predictive performance as PCR but uses only 7 components instead of 37. This could be really valuable in situations where computational efficiency matters or when we want a more interpretable model.

Task 3: Scores and loadings

For the PCR scores (Z1 vs Z2), the data points show a somewhat scattered distribution with no clear clustering, which makes sense since PCR just tries to maximize variance in the X space without considering y. The spread of points suggests we're capturing different aspects of variability in our dataset.

The PCR loadings plot (V1 vs V2) shows how our original variables contribute to the first two principal components. There's a noticeable spread of the loadings, with some variables having stronger contributions (points further from the origin) than others. This helps us understand which variables are driving the most variance in our data.

Moving to PLS, the scores plot (T1 vs T2) looks quite different from the PCR scores. The points seem to have a more structured pattern, which makes sense because PLS scores are computed while considering both X and y. This suggests that PLS is finding directions that are more relevant to our prediction task.

The PLS loadings (W1 vs W2) also show a different pattern compared to PCR. The loadings are more spread out, indicating that PLS is weighting variables differently to optimize the covariance between X and y. Some variables are clearly more influential than others in the PLS model.

It's interesting to see how these two methods, while both doing dimension reduction, end up with different internal representations of the data structure. This visual comparison really drives home why PLS needed fewer components than PCR to achieve similar performance.

Code

```
# a) download
url <- "https://archive.ics.uci.edu/ml/machine-learning-databases/00437/Residential-Building-Data-Set.xlsx"
temp <- tempfile(fileext = ".xlsx")
download.file(url, temp, mode = "wb")
df <- read_excel(temp)
unlink(temp)

# b) use provided .RData file
load("building.RData")
```

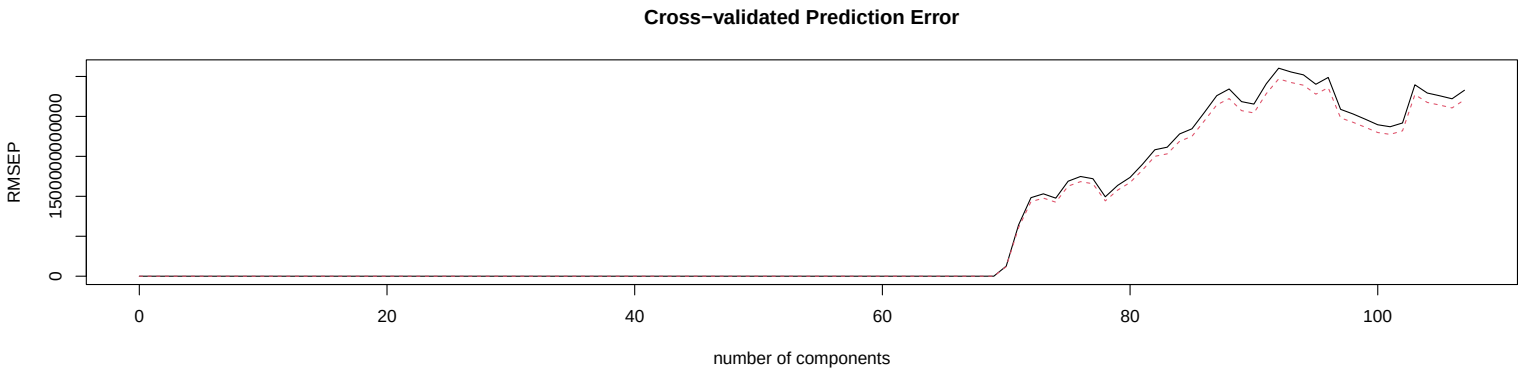
```
# random 2/3 train, 1/3 test split
split_and_validate_data <- function(df) {
    n <- nrow(df)
    n_train <- round(2/3 * n)
    n_test <- n - n_train
    train_indices <- sample(1:n, n_train)
    train_data <- df[train_indices, ]
    test_data <- df[-train_indices, ]
    y_train <- train_data$y
    X_train <- train_data[, setdiff(names(train_data), "y")]
    y_test <- test_data$y
    X_test <- test_data[, setdiff(names(test_data), "y")]

    assert_that((n_train + n_test == n) && nrow(train_data) == n_train && nrow(test_data) == n_test, msg="split sizes don't add up")
    assert_that(length(intersect(train_indices, which(!1:n %in% train_indices))) == 0, msg="train and test sets overlap")
    assert_that(all(names(X_train) == names(X_test)), msg="feature names don't match between train and test")
    assert_that(length(y_train) == nrow(X_train) && length(y_test) == nrow(X_test), msg="response and feature dimensions mismatch")
    assert_that(!any(is.na(X_train)) && !any(is.na(X_test)) && !any(is.na(y_train)) && !any(is.na(y_test)), msg="missing values found in data")
    return(list(
        X_train = X_train,
        y_train = y_train,
        X_test = X_test,
        y_test = y_test
    ))
}
split_data <- split_and_validate_data(df)
X_train <- split_data$X_train
y_train <- split_data$y_train # already log transformed
X_test <- split_data$X_test
y_test <- split_data$y_test

#
# 1) principal component regression
#

# a) pcr with 10-fold cv
pcr_model <- pcr(y ~ ., data = data.frame(y = y_train, X_train), scale = TRUE, validation = "CV", segments = 10)

# b) plot rmse train
validationplot(pcr_model, val.type = "RMSEP", main = "Cross-validated Prediction Error")
```



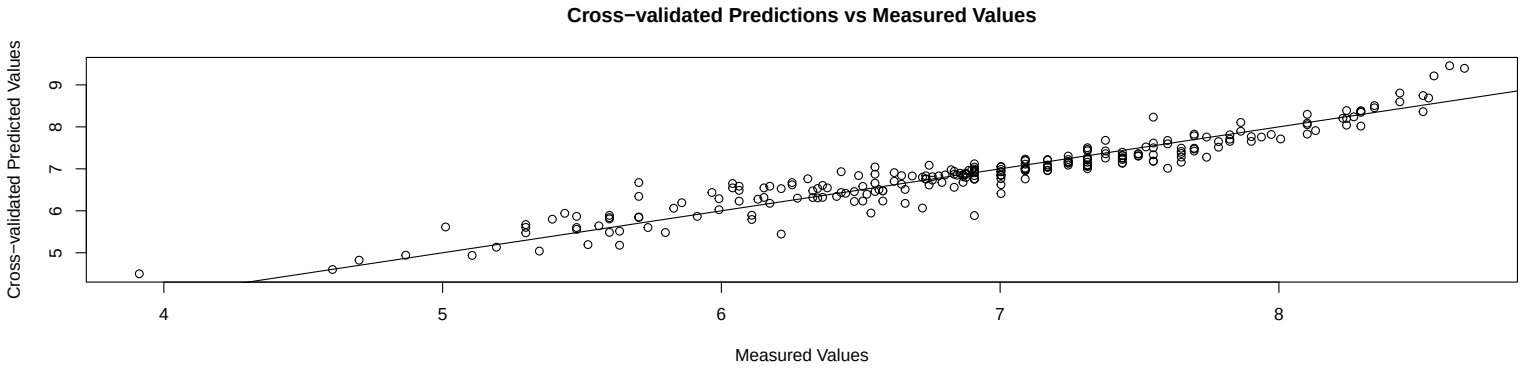

```
# b) get optimal num of components, rmse
rmse_values <- RMSEP(pcr_model)$val[1,,]
ncomp_opt <- which.min(rmse_values)
min_rmse <- min(rmse_values)
cat("optimal number of components:", ncomp_opt, "\n")
```

```
## optimal number of components: 37
```

```
cat("minimum RMSE:", min_rmse, "\n")
```

```
## minimum RMSE: 0.2660011
```

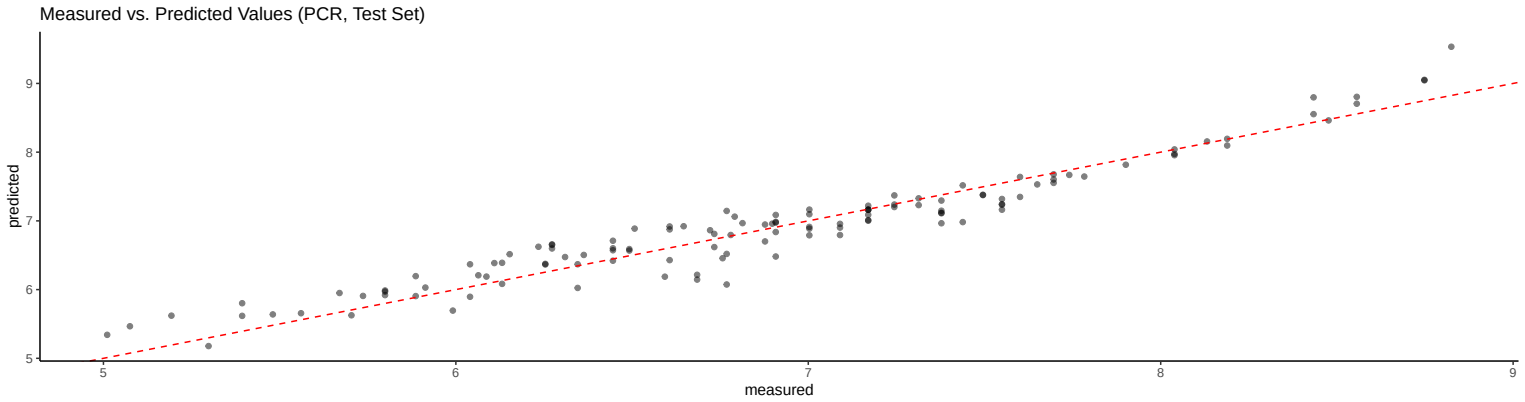
```
# c) predplot
predplot(pcr_model,
         ncomp = ncomp_opt,    # use optimal number of components
         line = TRUE,          # add target line
         main = "Cross-validated Predictions vs Measured Values", xlab = "Measured Values", ylab = "Cross-validated Predicted Values")
```




```
# d) plot rmse test
test_preds <- predict(pcr_model, newdata = X_test, ncomp = ncomp_opt)
cat("test RMSE:", rmse(test_preds, y_test), "\n")
```

```
## test RMSE: 0.2389746
```

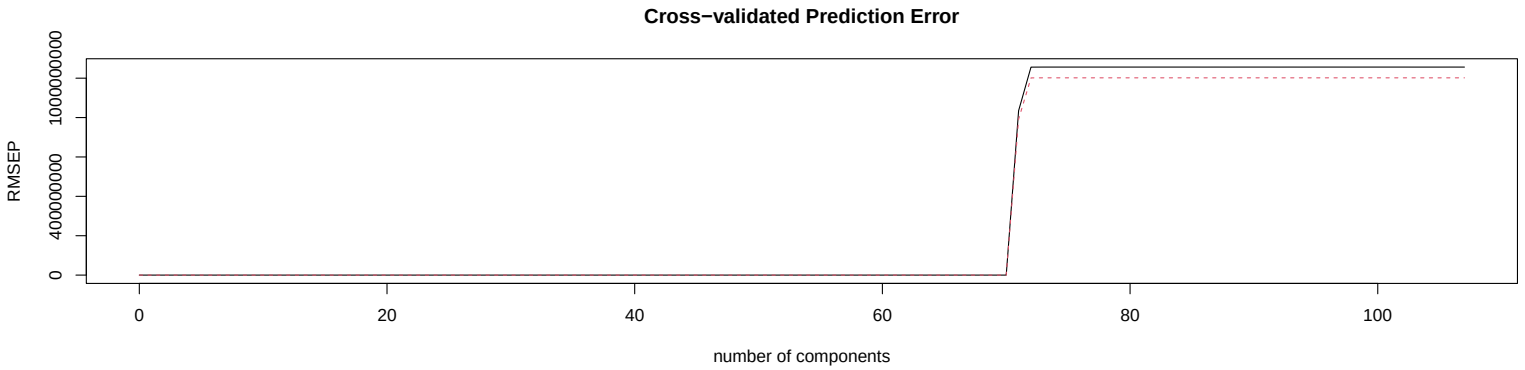
```
ggplot(data.frame(measured = y_test, predicted = as.vector(test_preds)), aes(x = measured, y = predicted)) +
    geom_point(alpha = 0.5) +
    geom_abline(intercept = 0, slope = 1, color = "red", linetype = "dashed") +
    labs(title = "Measured vs. Predicted Values (PCR, Test Set)", x = "measured", y = "predicted") +
    theme_classic()
```




```
#
# 2) partial least squares (PLS)
#

# a) pls with 10-fold cv
pls_model <- plsr(y ~ ., data = data.frame(y = y_train, X_train), scale = TRUE, validation = "CV", segments = 10)

# b) plot rmse train
validationplot(pls_model, val.type = "RMSEP", main = "Cross-validated Prediction Error")
```



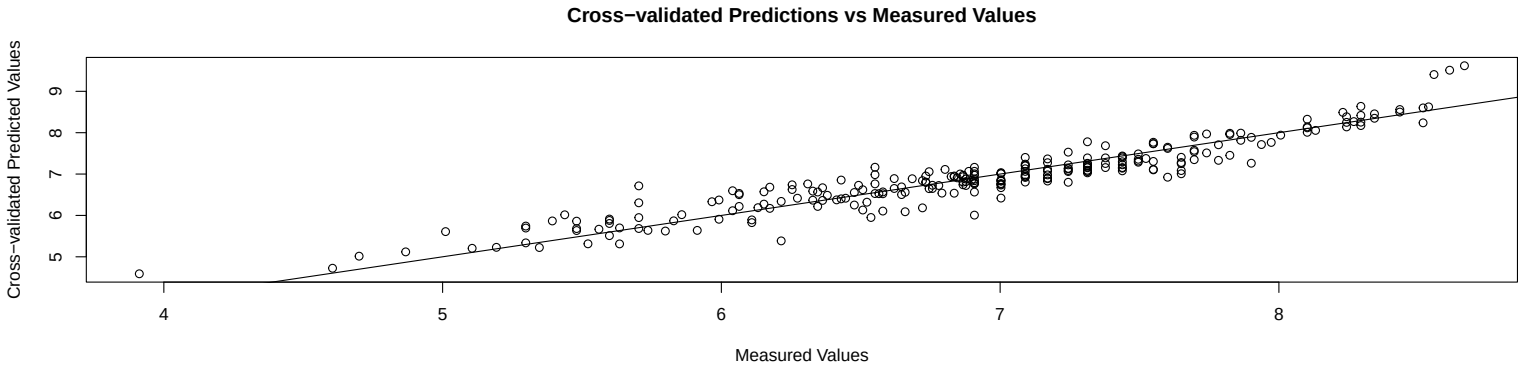

```
# b) get optimal num of components, rmse
rmse_values <- RMSEP(pls_model)$val[1,,]
ncomp_opt <- which.min(rmse_values)
min_rmse <- min(rmse_values)
cat("optimal number of components:", ncomp_opt, "\n")
```

```
## optimal number of components: 7
```

```
cat("minimum RMSE:", min_rmse, "\n")
```

```
## minimum RMSE: 0.2858628
```

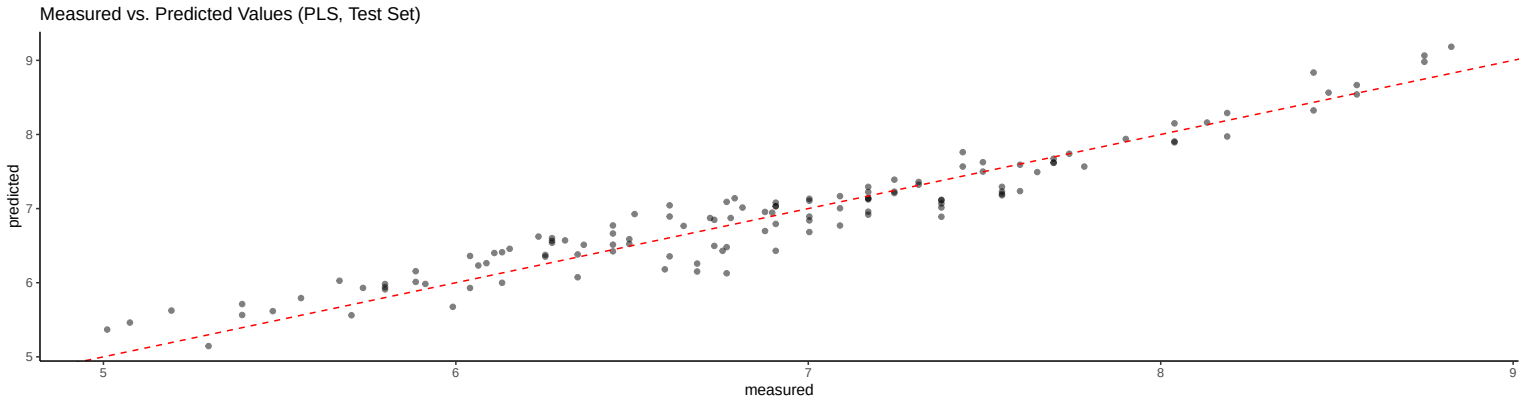
```
# c) predplot
predplot(pls_model,
         ncomp = ncomp_opt,
         line = TRUE,
         main = "Cross-validated Predictions vs Measured Values", xlab = "Measured Values", ylab = "Cross-validated Predicted Values")
```



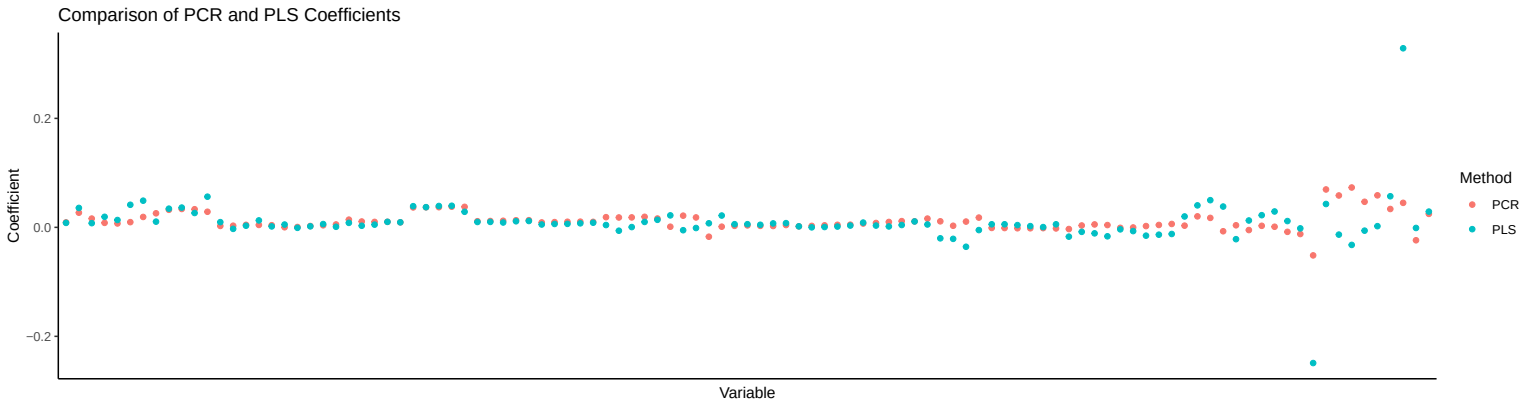

```
# d) plot rmse test
test_preds <- predict(pls_model, newdata = X_test, ncomp = ncomp_opt)
cat("test RMSE:", rmse(test_preds, y_test), "\n")
```

```
## test RMSE: 0.2416409
```

```
ggplot(data.frame(measured = y_test, predicted = as.vector(test_preds)), aes(x = measured, y = predicted)) +
    geom_point(alpha = 0.5) +
    geom_abline(intercept = 0, slope = 1, color = "red", linetype = "dashed") +
    labs(title = "Measured vs. Predicted Values (PLS, Test Set)", x = "measured", y = "predicted") +
    theme_classic()
```

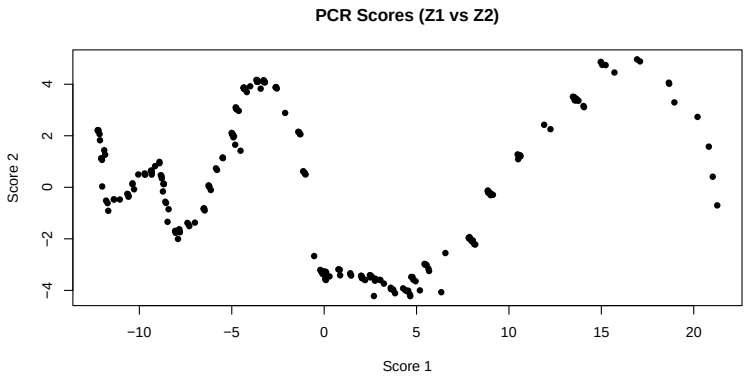


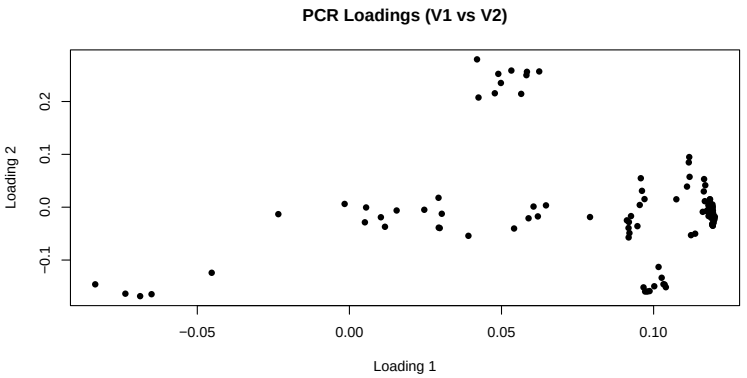

```
# e) compare coefficients
pcr_coef <- coef(pcr_model, ncomp = ncomp_opt)
pls_coef <- coef(pls_model, ncomp = ncomp_opt)
coef_df <- data.frame(Variable = rownames(pcr_coef), PCR = as.vector(pcr_coef), PLS = as.vector(pls_coef))
ggplot(pivot_longer(coef_df, cols = c("PCR", "PLS"), names_to = "Method", values_to = "Coefficient")) +
    geom_point(aes(x = Variable, y = Coefficient, color = Method)) +
    labs(title = "Comparison of PCR and PLS Coefficients") +
    theme_classic() +
    theme(axis.text.x = element_blank(), axis.ticks.x = element_blank())
```

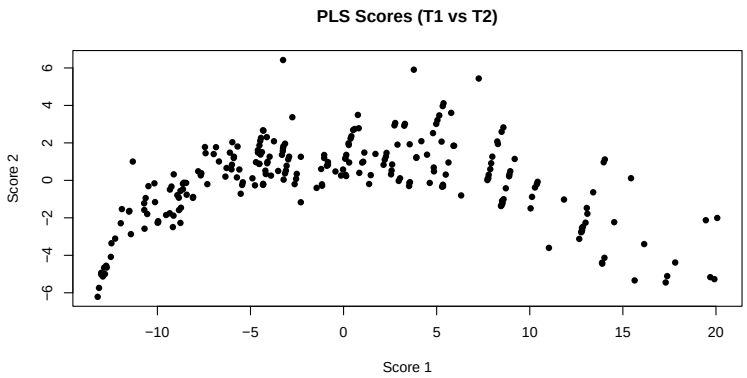


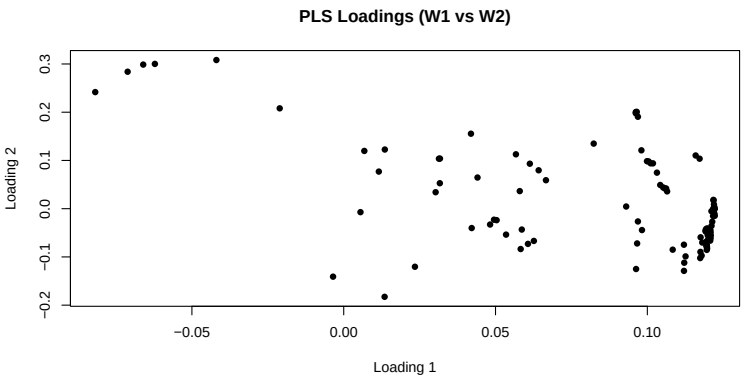

```
#
# 3) scores and loadings
#

par(mfrow=c(2,2))
plot(pcr_model$scores[,1], pcr_model$scores[,2], xlab="Score 1", ylab="Score 2", main="PCR Scores (Z1 vs Z2)", pch=16) # pcr scores
plot(pcr_model$loadings[,1], pcr_model$loadings[,2], xlab="Loading 1", ylab="Loading 2", main="PCR Loadings (V1 vs V2)", pch=16) # pcr loadings
plot(pls_model$scores[,1], pls_model$scores[,2], xlab="Score 1", ylab="Score 2", main="PLS Scores (T1 vs T2)", pch=16) # pls scores
plot(pls_model$loadings[,1], pls_model$loadings[,2], xlab="Loading 1", ylab="Loading 2", main="PLS Loadings (W1 vs W2)", pch=16) # pls loadings
```


PCR Scores (Z1 vs Z2)
Score 2
Score 1
−4
−2
0
2
4
−10
−5
0
5
10
15
20


PCR Loadings (V1 vs V2)
Loading 2
Loading 1
−0.1
0.0
0.1
0.2
−0.05
0.00
0.05
0.10


PLS Scores (T1 vs T2)
Score 2
Score 1
−6
−4
−2
0
2
4
6
−10
−5
0
5
10
15
20


PLS Loadings (W1 vs W2)
Loading 2
Loading 1
−0.2
−0.1
0.0
0.1
0.2
0.3
−0.05
0.00
0.05
0.10