R package `plsdepot`
PLS Regression 1

Gaston Sanchez
www.gastonsanchez.com/plsdepot

1 PLS-R1

PLS Regression is the method that most people think of when hearing the acronym **PLS**. Shortly speaking, PLS Regression is just an algorithm for regression analysis in which we want to analyze one block of response variables Y in terms of another block of predictor variables X. When we only have one response y_1, we talk about PLS-R1. When we have more than one response, we talk about PLS-R2.

The main reason for using PLS regression is to obtain a solution when ordinary least squares can no longer do a good job. But there's more than that. A derived feature of PLS regression is its dimension reduction properties that can be used for data visualization purposes.

1.1 Regression model

In a regression analysis, we model the behavior of a quantitative response y in terms of a set of quantitative predictors X by using a linear model like this one

$$E(y) = x_1 b_1 + x_2 b_2 + \ldots + x_p b_p$$

The goal is to find those coefficients b_j that help us to combine our predictors so that we can have a solution as "close" as possible to the expected value of y.

If our data is well-behaved and follows some general assumptions, ordinary least squares (OLS) is enough to get the job done. However, when the predictors in X are highly correlated, or when we have missing values, or when we have more variables than observations, ordinary least squares is not an option anymore: chances are that the obtained regression coefficients will be unstable, make no sense, and be difficult to interpret. An alternative solution to these problems is to apply **Partial Least Squares Regression**. Let's see an example.

2 Data `cornell`

In this demo we are going to use the data set `cornell` that is discussed in the example of chapter 5 from the textbook *La Regression PLS: Theorie et Pratique* by Michel Tenenhaus. The analyzed data `cornell`, that comes in `plsdepot`, consists of 8 variables measured on 12 chemicals. The values describe the composition of 12 octane mixtures (units measured in proportions). The aim is to model `Octane` in terms of the other mixture ingredients.

```r
# load the package
library(plsdepot)

# load the data
data(cornell)

# let's take a peek
head(cornell)
```

```
##   Distillation Reformed NaphthaTher NaphthaCat Polymer Alkylat NatEssence
## 1            0     0.23           0       0.00    0.00    0.74       0.03
## 2            0     0.10           0       0.00    0.12    0.74       0.04
## 3            0     0.00           0       0.10    0.12    0.74       0.04
## 4            0     0.49           0       0.00    0.12    0.37       0.02
## 5            0     0.00           0       0.62    0.12    0.18       0.08
## 6            0     0.62           0       0.00    0.00    0.37       0.01
##   Octane
## 1   98.7
## 2   97.8
## 3   96.6
## 4   92.0
## 5   86.6
## 6   91.2
```

2.1 Correlations

Before applying a regression model, let's calculate the correlations among the variables. Note that `Octane` is negatively correlated with most of the predictors except for `Polymer` and `Alkalyt`. This is important to keep in mind because we would expect the sign of the regression coefficients to reflect the sign of the correlations.

```r
# matrix of correlations
round(as.dist(cor(cornell)), 3)
```

```
##             Distillation Reformed NaphthaTher NaphthaCat Polymer Alkylat
## Reformed           0.104
## NaphthaTher        1.000    0.101
## NaphthaCat         0.371   -0.537       0.374
## Polymer           -0.548   -0.293      -0.548     -0.211
## Alkylat           -0.805   -0.191      -0.805     -0.646   0.463
## NatEssence         0.603   -0.590       0.607      0.916  -0.274  -0.656
## Octane            -0.836   -0.072      -0.837     -0.707   0.494   0.985
##             NatEssence
## Reformed
## NaphthaTher
## NaphthaCat
## Polymer
```

```
## Alkylat
## NatEssence
## Octane          -0.740
```

Note also that some of the variables are strongly correlated. For instance, `Distillation` and `NaphthaTher` have a correlation of 1. Or what about the correlation between `NaphthaCat` and `NatEssence`? It has a value of 0.916. All these highly correlated variables may negatively affect the results af an OLS regression analysis.

2.2 A not very good regression

Let's run an ordinary least squares regression having `Octane` as the response variable.

```
# ordinary least squares regression
regression = lm(Octane ~ ., data = cornell)

# check summary
summary(regression)

##
## Call:
## lm(formula = Octane ~ ., data = cornell)
##
## Residuals:
##         1         2         3         4         5         6         7
##  1.21e+00 -2.06e-01 -5.86e-01 -6.62e-02  8.04e-01 -3.53e-01  2.47e-02
##         8         9        10        11        12
##  3.06e-01 -3.52e-01  2.15e-02  4.77e-15 -8.00e-01
##
## Coefficients: (1 not defined because of singularities)
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)     112.4       80.9    1.39     0.22
## Distillation    -94.6      172.8   -0.55     0.61
## Reformed        -26.6       80.6   -0.33     0.76
## NaphthaTher      59.1      430.8    0.14     0.90
## NaphthaCat      -34.8       90.0   -0.39     0.72
## Polymer         -24.5       83.8   -0.29     0.78
## Alkylat         -11.9       84.2   -0.14     0.89
## NatEssence         NA         NA      NA       NA
##
## Residual standard error: 0.834 on 5 degrees of freedom
## Multiple R-squared: 0.993,Adjusted R-squared: 0.984
## F-statistic:  111 on 6 and 5 DF,  p-value: 3.75e-05
##
```

If you look at the table of coefficients you should be disturbed by four things:

- the signs of the coefficients don't coincide with those of the correlations
- there is no coefficient value for `NatEssence`

- according to the p-values, none of the coefficients are significant
- even weirder, the `R-squared` has a value of 0.9925!

Obviously these results are not OK. A much better solution can be achieved by applying a PLS regression.

3 PLS Regression

PLS regression comes in different presentations and flavors, but they all share the spirit of applying an iterative algorithm with multiple steps of least squares regerssions to calculate the parameters of the model. I'm not going to explain the details behind PLS regression, but just to mention the overall concepts.

The idea behind PLS regression is to look for components T that allow us not only to decompose the block of predictors

$$X = TP + Residuals$$

but also to predict the response

$$y = TC + error$$

The way PLS regression works is by following an iterative algorithm. The first component t_1 is a weighted sum of predictors: $t_1 = w_{11}x_1 + \ldots + w_{1p}x_p$. What PLS looks for is to get a component that is a good proxy of X, but that also has as much to do with the response y.

The real trick is in the way the weights w_{1j} are calculated. For the first component t_1, the weights are obtained as

$$w_{1j} = \frac{cov(x_j, y)}{\sqrt{\sum_{j=1}^{p} cov^2(x_j, y)}}$$

The denominator is just there to normalize the weights so that $\|w_1\| = 1$. But the relevant part is the covariance between x_j and y. This tells us that we are looking for weights that reflect the strength of the association between a given predictor and the response. The more correlated x_j and y are, the larger the weight w_{1j} is going to be. But that's not all. In addition, we require an extra condition which is that the extracted comopnents have to be orthogonal. In this way we get rid of the potential problem of having strongly correlated variables. To know more about the details behind the PLS regression algorithm, good references are:

- Geladi P., Kowalski B (1986) Partial Least Squares Regression: A tutorial. *Analytica Chimica Acta*, 185: 1-17.
- Tenenhaus M. (1998) *La Regression PLS: Theorie et pratique.* Paris: Editions TECHNIP.
- Helland I.S. (2001) Some theoretical aspects of partial least squares regression. *Chemometrics and Intelligent Laboratory Systems*, 58: 97-107.

4 Function `plsreg1()`

The package `plsdepot` provides the function `plsreg1()` for performing PLS regression with one response variable. This function has 4 arguments: `predictors`, `response`, `comps` and `crosval`. Obviously the first argument `predictors` is the data containing the predictors. This can be either a matrix or a data frame. The second argment, `response`, is the data containg the response variable, which is a vector (or a matrix or a data frame with just one column). The third argument `comps` specifies the number of extracted components. And finally, `crosval` indicates whether to perform cross-validation(`TRUE` by default). Let's apply the `plsreg1()` function on `cornell`, asking for three components

```
# apply plsreg1
my_pls1 = plsreg1(cornell[, 1:7], cornell[, 8, drop = FALSE], comps = 3)

# what's in my_pls1?
my_pls1

##
## PLS Regression 1
## ------------------------------------------
## $x.scores     X-scores (T-components)
## $x.loads      X-loadings
## $y.scores     Y-scores (U-components)
## $y.loads      Y-loadings
## $cor.xyt      score correlations
## $raw.wgs      raw weights
## $mod.wgs      modified weights
## $std.coefs    standard coefficients
## $reg.coefs    regular coefficients
## $R2           R-squared
## $R2Xy         explained variance of X-y by T
## $y.pred       y-predicted
## $resid        residuals
## $T2           T2 hotelling
## $Q2           Q2 cross validation
## ------------------------------------------
##
```

What you get in `my_pls1` is an object of class `"plsreg1"`, and everytime you print an object of such class you get a display with the list of results.

4.1 PLS components

The first two elements in the list are `$x.scores` and `$x.loads` which contains the extracted PLS components and its loadings, respectively.

```
# pls components
head(my_pls1$x.scores)
```

```
##         t1      t2      t3
## 1  2.05133  0.8189  1.5787
## 2  2.47541  0.6520  0.1126
## 3  2.33205  0.9293 -0.1702
## 4  2.03606 -1.5932 -0.4978
## 5 -0.06812 -0.2198 -2.9571
## 6  1.61199 -1.4263  0.9684

# X-loadings
my_pls1$x.loads

##                    p1       p2      p3
## Distillation -0.45358 -0.04064  0.2736
## Reformed      0.03153 -1.00379  0.4494
## NaphthaTher  -0.45438 -0.03713  0.2713
## NaphthaCat   -0.35604  0.27687 -0.5338
## Polymer       0.29439 -0.04363 -0.4937
## Alkylat       0.46207  0.43956  0.1052
## NatEssence   -0.41250  0.47699 -0.3391
```

The third and fourth elements are `$y.scores` and `$y.loads`, which are the U components and loadings associated to the response variable.

```
# U components
head(my_pls1$y.scores)

##        u1      u2      u3
## 1  3.2215  2.0511  3.2694
## 2  2.9344  0.8046  0.4049
## 3  2.5517  0.3850 -1.4445
## 4  1.0845 -1.6680 -0.1984
## 5 -0.6379 -0.9988 -2.0670
## 6  0.8293 -1.3719  0.1442

# Y-loadings
my_pls1$y.loads

##     c1     c2     c3
## 0.4819 0.2749 0.1036
```

4.2 Correlations between variables and components

In order to check how the PLS components are associated the X and y, we need to check `$cor.xyt`

```
# correlations between X-y and T
my_pls1$cor.xyt
```

```
##                    t1       t2      t3
## Distillation -0.90424 -0.03416  0.3165
## Reformed      0.06285 -0.84372  0.5200
## NaphthaTher  -0.90584 -0.03121  0.3139
## NaphthaCat   -0.70979  0.23272 -0.6176
## Polymer       0.58688 -0.03667 -0.5713
## Alkylat       0.92115  0.36946  0.1217
## NatEssence   -0.82234  0.40093 -0.3924
## Octane        0.96061  0.23107  0.1199
```

The first component t_1 is clearly highly correlated with the response, but it's also a good proxy for most of the predictors in X. The second component t_2 captures most of the information of `Reformed` but is less correlated with `Octane`. The third component t_3 also summarizes some information from the predictors although its weekly correlated with `Octane`.

4.3 Modified weights

The modified weights, `$mod.wgs`, are the weights W^* used for calculating the components with the original block of predictors: $X = TW^*$.

```
# pls components
head(my_pls1$mod.wgs)
```

```
##                   w*1     w*2      w*3
## Distillation -0.43670  0.1221  0.37718
## Reformed     -0.03757 -0.6852 -0.03771
## NaphthaTher  -0.43703  0.1268  0.38217
## NaphthaCat   -0.36906 -0.1658 -0.68583
## Polymer       0.25784 -0.3342 -0.68211
## Alkylat       0.51451  0.5693  0.51509
```

You can manually get the pls components like this

```
# scale matrix X
Xs = scale(cornell[, 1:7])

# use modified weights to obtain pls components
head(Xs %*% my_pls1$mod.wgs)
```

```
##        w*1     w*2     w*3
## 1  2.05133  0.8189  1.5787
## 2  2.47541  0.6520  0.1126
## 3  2.33205  0.9293 -0.1702
## 4  2.03606 -1.5932 -0.4978
## 5 -0.06812 -0.2198 -2.9571
## 6  1.61199 -1.4263  0.9684
```

4.4 Regression coefficients

The function `plsreg1()` provides two types of regression coefficients. `$std.coefs` are the coefficients for the standardized data. `$reg.coefs` are the coefficients for the unstandardized data, and because of that there will be an intercept term.

```
# standardized coefficients
my_pls1$std.coefs

## Distillation     Reformed  NaphthaTher   NaphthaCat      Polymer
##     -0.13778     -0.21037     -0.13614     -0.29446     -0.03829
##      Alkylat   NatEssence
##      0.45778     -0.14314


# regular (unstandardized) coefficients
my_pls1$reg.coefs

##    Intercept Distillation     Reformed  NaphthaTher   NaphthaCat
##       92.677       -9.713       -7.000      -16.457       -8.440
##      Polymer      Alkylat   NatEssence
##       -4.362       10.169      -34.392
```

If you want to get the regression model expressed in terms of the predictors in the original scale, you need to use the regular coefficients:

$$\widehat{\texttt{Octane}} = 92.67 - 9.71\texttt{Distillation} - 7\texttt{Reformed} - 16.45\texttt{NaphthaTher}$$

$$-8.43\texttt{NaphthaCat} - 4.36\texttt{Polymer} + 10.17\texttt{Alkalyt} - 34.4\texttt{NatEssence}$$

Let's manually calculate the solution and compare it to the predicted values `$y.pred`

```
# calculate octane prediction
oct.hat = as.matrix(cornell[, 1:7]) %*% my_pls1$reg.coefs[-1] + my_pls1$reg.coefs[1]
data.frame(oct.hat = oct.hat, oct.pred = my_pls1$y.pred)

##    oct.hat oct.pred
## 1    97.56    97.56
## 2    97.60    97.60
## 3    97.46    97.46
## 4    91.80    91.80
## 5    86.00    86.00
## 6    91.76    91.76
## 7    81.53    81.53
## 8    82.61    82.61
## 9    82.56    82.56
## 10   83.30    83.30
## 11   81.97    81.97
## 12   89.05    89.05
```

4.5 R-squared and explained variance

Another important result has to do with the R-squared coefficient and the variance proportion explained by the pls components.

```
# R-squared coefficients for each component
my_pls1$R2

##      t1      t2      t3
## 0.92277 0.05339 0.01437

# cumulative explained variance of X-y by T
my_pls1$R2Xy

##                    t1     t2     t3
## Distillation 0.81765 0.8188 0.9190
## Reformed     0.00395 0.7158 0.9862
## NaphthaTher  0.82054 0.8215 0.9200
## NaphthaCat   0.50381 0.5580 0.9393
## Polymer      0.34443 0.3458 0.6721
## Alkylat      0.84852 0.9850 0.9998
## NatEssence   0.67625 0.8370 0.9909
## Octane       0.92277 0.9762 0.9905
```

Note that the first PLS component has an associated R2 value of 0.9227. This means that the first PLS component could be enough for prediction purposes, explaining about 92% of the response variable.

Note also the table with the cumulative explained variance of X and y by T. Basically the first component summarizes the information of almost all of the variables except for `Reformed`. The second component is the one respondible for explaining the variable `Reformed`. In turn, the third component seems to be capturing the left-over information in `Polymer`.

4.6 Cross-validation results

The function `plsreg1()` offers the option to perform a cross-validation procedure. In this process, the data set is randomly split in 10 segments of approximately equal size. Then, the observations in one of the segments are left outside as a test set. The other nine segments are used as learning set to estimate a model and predict the observations in the test segment. This procedure is applied consecutively for each of the 10 segments. The cross-valition results can be checked in `$Q2`.

$$Q_h^2 = 1 - \frac{PRESS_h}{RSS_{h-1}}$$

where RSS_{h-1} is the squared sum of residuals using the t_{h-1} component, and $PRESS_h$ is the PRediction Error Sum of Squares using the t_h component. A component t_h is considered to be significant if Q_h^2 is greater than or equal to `LimQ2`=0.0975

```
# cross-validation indices
my_pls1$Q2
```

```
##    PRESS     RSS      Q2  LimQ2  Q2cum
## 1 1.1160 11.0000  0.8985 0.0975 0.8985
## 2 0.9518  0.8496 -0.1203 0.0975 0.8863
## 3 0.1903  0.2623  0.2746 0.0975 0.9175
```

4.7 Hotelling's T^2

Finally, `plsreg1()` provides results for the Hotelling's T^2. This is used to help us identify observations that are potential outliers. The table with the Hotelling's T^2 is in `$T2`.

```
# hotelling's T2
my_pls1$T2
```

```
##          H1      H2     H3
## T2 5.248030 9.77839 15.343
## 1  1.155064 2.19063  4.222
## 2  1.682005 2.33832  2.349
## 3  1.492826 2.82642  2.850
## 4  1.137928 5.05746  5.259
## 5  0.001274 0.07587  7.202
## 6  0.713273 3.85426  4.618
## 7  1.333694 1.38245  1.428
## 8  1.090494 1.10662  1.118
## 9  1.195953 1.23031  1.333
## 10 1.006866 1.16509  1.197
## 11 1.183383 1.50667  2.378
## 12 0.007241 1.26589  2.045
```

5 Plotting `"plsreg1"` objects

An accessory function is the `plot()` method that allows us to get some graphics of the basic results. Basically, we can plot either the variables and the observations on the specified components. When plotting variables, they are displayed inside a circle of correlations. Instead, when plotting the observations they are displayed using a scatter-plot.

5.1 Plotting variables

The default output when using `plot()` is a graphic showing the correlations of the variables with the first two axes (associated to the first two components). This plot can be regarded as a radar. The closer a variable appears on the perimiter of the circle, the better it is represented. In addition, if two variables are highly correlated, they will appear near each other. If two variables are negatively correlated, they will tend to appear in opposite extremes. If two variables are uncorrelated, they will be orthogonal to each other.

```
# default plot
plot(my_pls1)
```

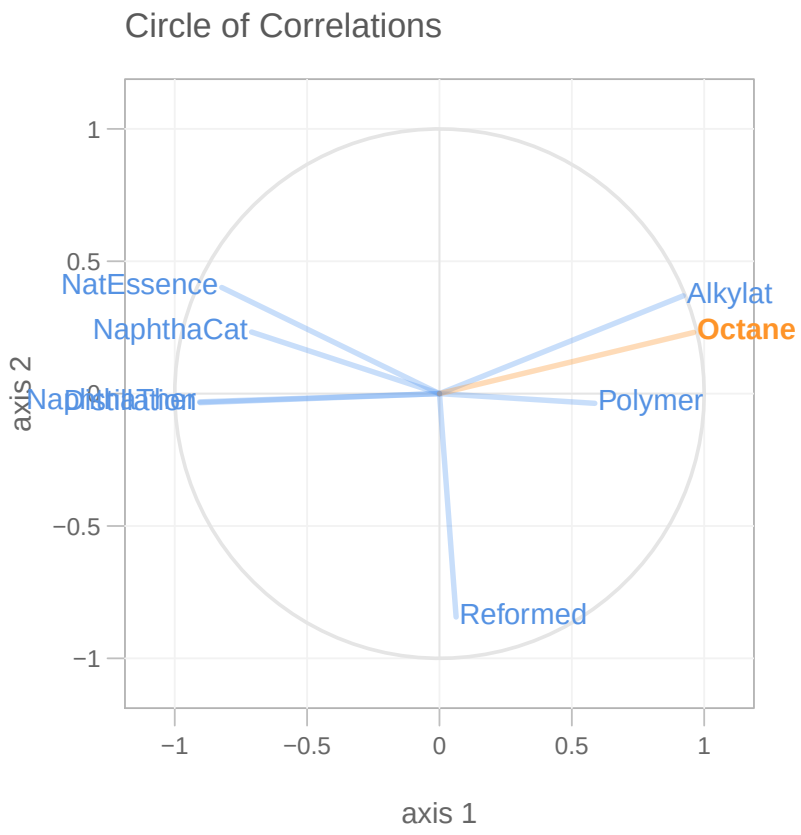

Figure 1: Circle of correlations

You can use the parameter `comps` to select a different pair of axes. For instance, if you wanted to see the correlations on the first and third axes, you would do it like this

```
# plot variables on axes 1,3
plot(my_pls1, comps = c(1, 3))
```

5.2 Plotting observations

The alternative output when using `plot()` is to show a scatter-plot of the observations on the specified components. Use the argument `what="observations"` to specify that you wnat to plot the score values.

```
# plot of observations
plot(my_pls1, what = "observations", show.names = TRUE)
```

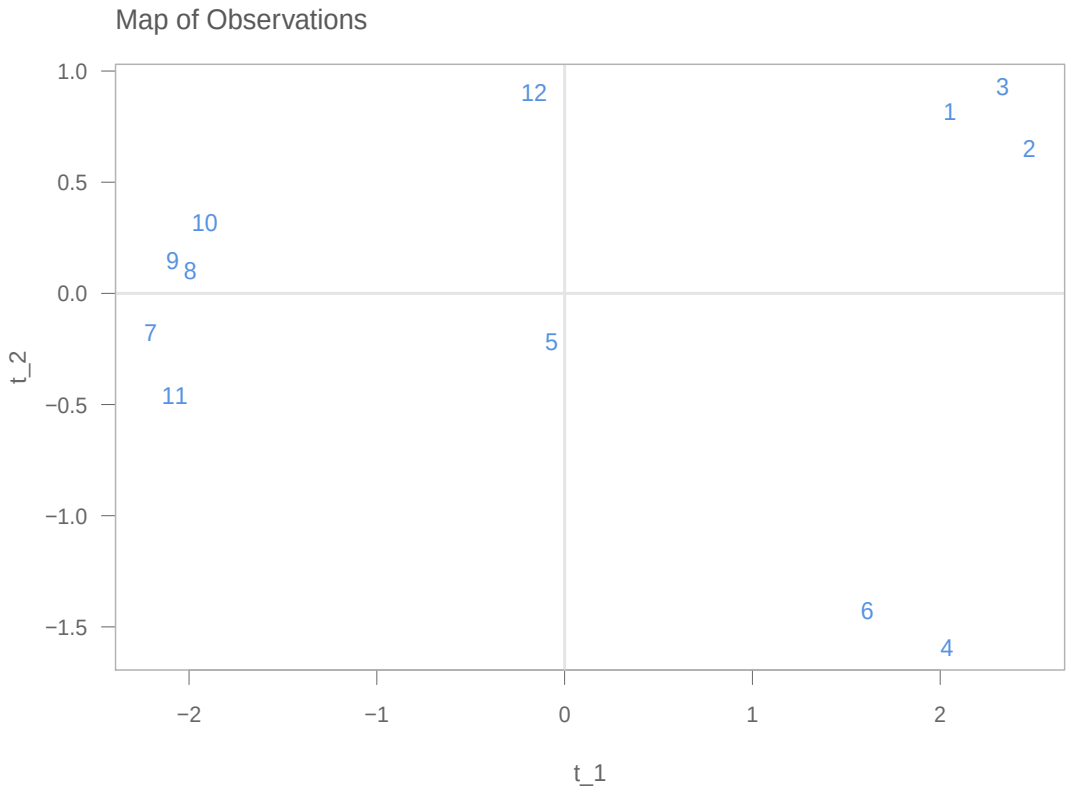

Figure 2: Plot of observations

5.3 Comparing responses

Finally, let's compare the predicted values from the PLS regression against the original response values.

```
# plot of observations
plot(cornell$Octane, my_pls1$y.pred, type = "n", xlab = "Original",
    ylab = "Predicted")
title("Comparison of responses", cex.main = 0.9)
abline(a = 0, b = 1, col = "gray85", lwd = 2)
text(cornell$Octane, my_pls1$y.pred, col = "#5592e3")
```

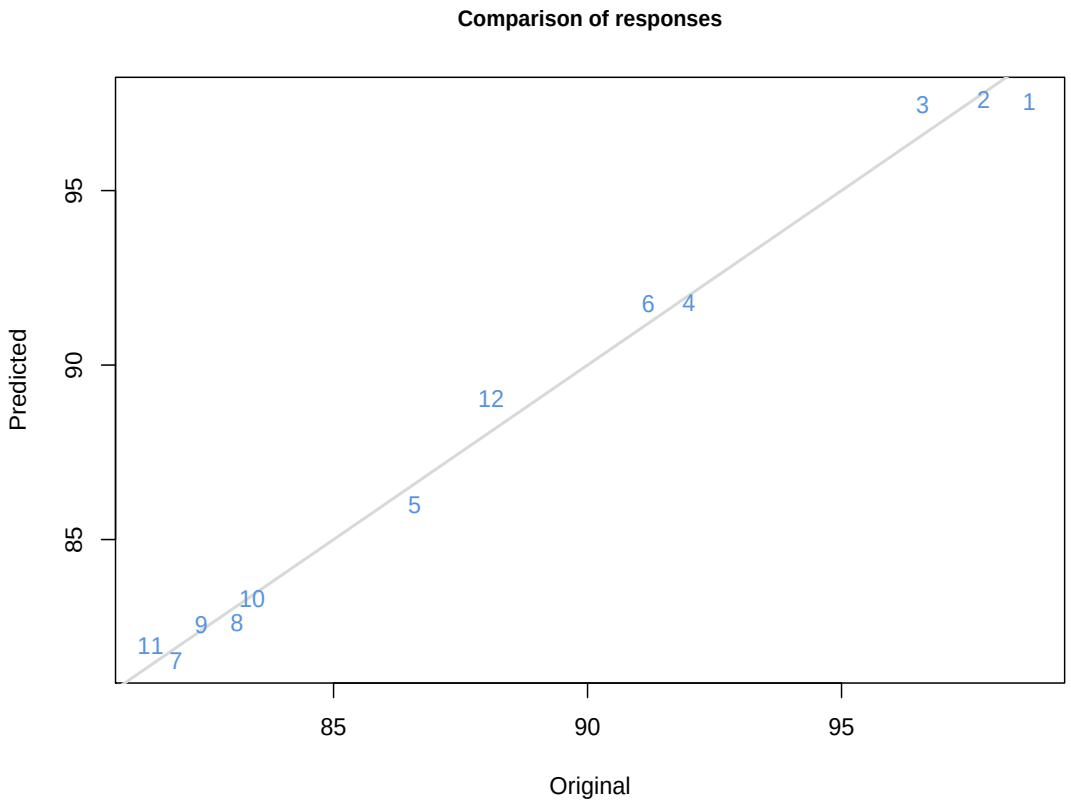

Figure 3: Original -vs- Predicted values