R package plsdepot
PLS Canonical Analysis

Gaston Sanchez
www.gastonsanchez.com/plsdepot

1 PLS-CA

Partial Least Squares Canonical Analysis (PLS-CA) is a mtehod to study the relationship between two blocks of variables, say X and Y, with the peculiar characteristic that there is a symmetric role between X and Y. In this case, there are neither predictors nor responses. Instead, both blocks of variables are simply two sets of descriptors. Hence the name *Canonical Analysis*.

As with many other multivariate techniques for analyzing the relationship between two data tables, PLS-CA aims to find a group of components t_h and u_h that summarize well enough their own block of variables, and at the same time share common information among each another.

PLS Canonical Analysis is closely related to PLS regression. In fact, PLS-CA can be seen as a variant of the PLS regression algorithm. Without entering in more details, instead of obtaining the residuals Y_h with t_h and Y_{h-1}, we use u_h and Y_{h-1}. Actually, the results are very similar to those obtained with SIMPLS-CA, which is no coincidence given the fact that both approaches have the same goals.

2 Data linnerud

In this demo we will consider the example discussed in chapter 11 of *La Regression PLS: Theorie et Pratique* (Tenenhaus, 1998). The analyzed data linnerud, consists of 6 variables measured on 20 individuals. The variables can be grouped in two blocks. One block X for three physical measurements, and another block Y for exercise outputs.

```
# load the package
library(plsdepot)
# load the data
data(linnerud)
# let's take a peek
head(linnerud)

##   Weight Waist Pulse Pulls Squats Jumps
## 1    191    36    50     5    162    60
## 2    189    37    52     2    110    60
## 3    193    38    58    12    101   101
## 4    162    35    62    12    105    37
## 5    189    35    46    13    155    58
## 6    182    36    56     4    101    42
```

3 Function plsca()

The package plsdepot provides the function plsca() for performing PLS Canonical Analysis. plsca() has 4 arguments: X, Y, comps and scaled. The first argument X is the data containing one block of variables. This can be either a matrix or a data frame. The second argment, Y, is the second data set, which can also be a matrix or a data frame. The third argument comps specifies the number of extracted components. And finally, scaled indicates whether to standardize the data (TRUE by default). Let's apply the PLS-CA algorithm on linnerud.

```
# apply plsca
my_plsca = plsca(linnerud[, 1:3], linnerud[, 4:6])

# what's in my_plsca?
my_plsca

##
## PLS Canonical Analysis
## ----------------------------------------------------
## $x.scores     X-scores (T-components)
## $x.wgs        X-weights
## $x.loads      X-loadings
## $y.scores     Y-scores (U-components)
## $y.wgs        Y-weights
## $y.loads      Y-loadings
## $cor.xt       X,T correlations
## $cor.yu       Y,U correlations
## $cor.tu       T,U correlations
## $cor.xu       X,U correlations
## $cor.yt       Y,T correlations
## $R2X          explained variance of X by T
## $R2Y          explained variance of Y by U
## $com.xu       communality of X with U
## $com.yt       communality of Y with T
## ----------------------------------------------------
##
```

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

3.1 PLS components

The first three elements in the list are $x.scores, $x.wgs, and $x.loads which contains the extracted PLS components, its associated weights (i.e. coefficients), and the loadings, respectively.

```r
# T components
head(my_plsca$x.scores)

##        t1       t2       t3
## 1 -0.6429  0.23026  0.60551
## 2 -0.7697 -0.04985  0.30740
## 3 -0.9074 -0.15769 -0.58762
## 4  0.6884 -0.39632 -0.53956
## 5 -0.4867  0.39520  1.21450
## 6 -0.2291 -0.03390 -0.06624

# T-weights
my_plsca$x.wgs

##             t1        t2      t3
## Weight -0.5899  0.716807 -0.3067
## Waist  -0.7713 -0.707923 -0.1978
## Pulse   0.2389 -0.003455 -0.9416

# X-loadings
my_plsca$x.loads

##             c1      c2      c3
## Weight -0.6659  0.7736 -0.1718
## Waist  -0.6760 -0.6287 -0.1692
## Pulse   0.3589 -0.1199 -0.9705
```

Similarly, elements fourth to sixth give the scores `$y.scores`, its associated weights `$y.wgs`, and the loadings `$y.loads`.

```r
# U components
head(my_plsca$y.scores)

##         u1       u2      u3
## 1 -0.37145 -0.08835 -0.7763
## 2 -1.34032 -0.58526 -0.4868
## 3 -0.08235 -0.55713  0.9958
## 4 -0.35497  0.57672  0.6453
## 5  0.46312  0.54078  0.2436
## 6 -1.30584 -0.15980 -0.2245

# U-weights
my_plsca$y.wgs

##            u1      u2      u3
## Pulls  0.6133  0.4159  0.6230
## Squats 0.7470  0.3139 -0.7737
## Jumps  0.2567 -0.8924  0.2412
```

```
# Y-loadings
my_plsca$y.loads

##            e1       e2      e3
## Pulls  0.6147  0.37878  0.7472
## Squats 0.6563  0.01197 -0.6541
## Jumps  0.5173 -0.93985  0.1182
```

3.2 Correlations between variables and components

In order to check how the PLS components are associated with their own group of variables, we use `$cor.xt` and `$cor.yu`.

```
# correlations between X and T
my_plsca$cor.xt

##             t1       t2      t3
## Weight -0.9476  0.28090 -0.1520
## Waist  -0.9620 -0.22831 -0.1497
## Pulse   0.5108 -0.04355 -0.8586

# correlations between Y and U
my_plsca$cor.yu

##            u1        u2       u3
## Pulls  0.8802  0.269599  0.39068
## Squats 0.9397  0.008517 -0.34201
## Jumps  0.7407 -0.668949  0.06179
```

3.3 Explained variance

`plsca()` provides two types of R-squared coefficients with values for the proportions of explained variance: `$R2X` are and `$R2Y`.

```
# explained variance of X by T
my_plsca$R2X

##            t1     t2 t3
## Weight 0.8980 0.9769  1
## Waist  0.9255 0.9776  1
## Pulse  0.2609 0.2628  1

# explained variance of Y by U
my_plsca$R2Y

##            u1     u2 u3
## Pulls  0.7747 0.8474  1
## Squats 0.8830 0.8830  1
## Jumps  0.5487 0.9962  1
```

3.4 Inter-group Communalities

Because the components t_h and u_h are extracted in such a way that they are as much correlated as possible, it is also interesting to check how well each group of components may explain the other block. This is done with the inter-group communalities:

```
# communality of X with U
my_plsca$com.xu

##             u1     u2      u3
## Weight 0.21597 0.2420 0.26792
## Waist  0.36926 0.5042 0.54784
## Pulse  0.03542 0.0560 0.07487
```

```
# communality of Y with T
my_plsca$com.yt

##            t1      t2     t3
## Pulls  0.2363 0.32992 0.3396
## Squats 0.3506 0.43075 0.4365
## Jumps  0.0414 0.04683 0.0539
```

4 Plotting "plsca" 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 a specified pair of components. The variables are plotted inside a circle of correlations. In turn, the observations are plotted using a scatter-plot.

4.1 Plotting variables

The default output when using `plot()` is a graphic showing the correlations of the variables with 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_plsca)
```

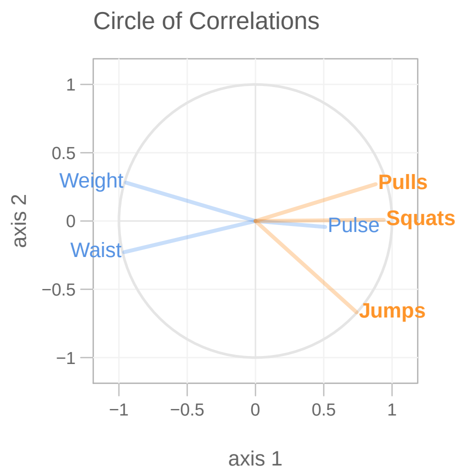

Figure 1: Circle of correlations (axes 1-2)

4.2 Plotting observations

The alternative output when using `plot()` is to show a scatter-plot of the observations on the specified components.

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

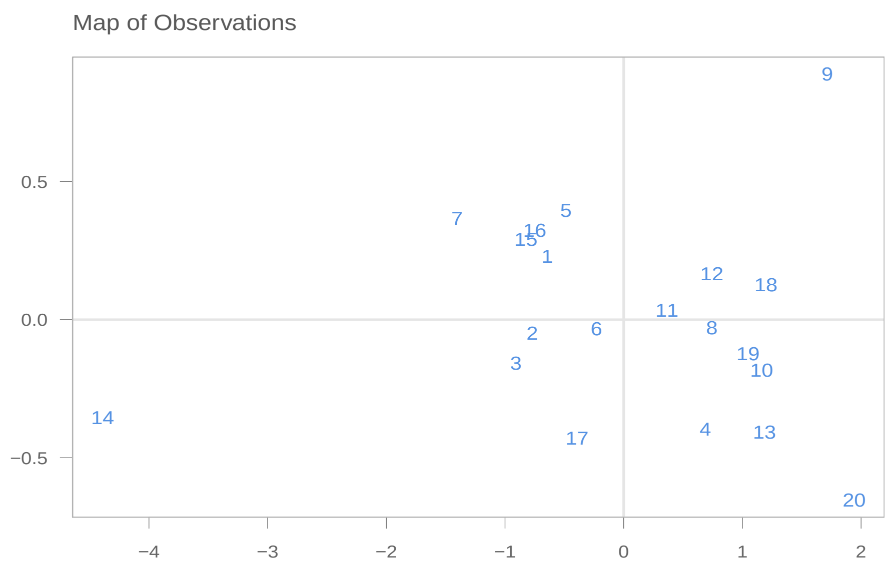

Figure 2: Plot of observations (comps 1-2)

References

Tenenhaus M. (1998) *La Regression PLS. Theorie et Pratique*. Paris: Editions TECHNIP