R package `plsdepot`
Principal Components with NIPALS

Gaston Sanchez
www.gastonsanchez.com/plsdepot

1 Introduction

NIPALS is the acronym for *Nonlinear Iterative Partial Least Squares* and it is the PLS technique for performing principal component analysis (PCA). Basically, NIPALS is just an iterative algorithm based on simple least squares regressions for calculating principal components. This algorithm comes implemented in the R package `plsdepot` with the function `nipals()`.

1.1 PCA reminder

Principal Component Analysis (PCA) is a multivariate technique that allows us to summarize the systematic patterns of variations in a data set (i.e. a matrix) X. From a data analysis standpoint, PCA is used for studying one table of observations and variables. As it is customary in many multivariate techniques, the rows of the matrix X represent the observations (e.g. individuals, objects, samples), and the columns represent the variables $x_1, x_2, \ldots, x_p$. In addition, the variables are assumed to be quantitative, ideally measured in a more or less continuous scale.

In PCA, we look for a decomposition of X given as

$$X = \sum_{h=1}^{q} t_h p'_h$$

where $\{t_1, t_2, \ldots, t_q\}$ are the principal components and $\{p_1, p_2, \ldots, p_q\}$ are the principal axes.

1.2 NIPALS algorithm

The algorithm behind NIPALS allows us to get the desired PCA decomposition by applying an iterative procedure based on simple least squares regressions. The pseudo-code with the main steps of NIPALS is given below:

1. $X_0 = X$
2. For $h = 1, 2, ..., q$
   (a) t_h = first column of X_{h-1}
   (b) repeat until convergence of p_h
      i. $p_h = X'_{h-1} t_h / t'_h t_h$ (*hint*: this is a LS regression)
      ii. Normalize p_h to 1
      iii. $t_h = X_{h-1} p_h / p'_h p_h$ (*hint*: this is a LS regression)
3. $X_h = X_{h-1} - t_h p'_h$

2 Data `carscomplete`

For this example we are going to use the data set `carscomplete` that already comes in `plsdepot`. This data appears in Michel Tenenhaus' French textbook *La Regression PLS: Theorie et Pratique*, and it contains six variables describing different characteristics of 24 cars:

Variable	Description
`Cylindree`	engine
`Puissance`	power
`Vitese`	speed
`Poids`	weight
`Longueur`	length
`Largeur`	height

```
# load the package
library(plsdepot)

# load the data
data(carscomplete)

# let's take a look at the data
head(carscomplete)

##                       Cylindree Puissance Vitese Poids Longueur Largeur
## Honda civic                1396        90    174   850      369     166
## Renault 19                 1721        92    180   965      415     169
## Fiat Tipo                  1580        83    170   970      395     170
## Peugeot 405                1769        90    180  1080      440     169
## Renault 21                 2068        88    180  1135      446     170
## Citroen BX                 1769        90    182  1060      424     168
```

2.1 Exploratory Analysis

Our initial exploratory analysis begins by getting some summary statistics of the variables

```
summary(carscomplete)

##    Cylindree      Puissance         Vitese         Poids          Longueur
##  Min.   :1116   Min.   : 50.0   Min.   :135   Min.   : 730   Min.   :350
##  1st Qu.:1550   1st Qu.: 89.5   1st Qu.:173   1st Qu.: 914   1st Qu.:371
##  Median :1928   Median :101.5   Median :182   Median :1128   Median :436
##  Mean   :1906   Mean   :113.7   Mean   :183   Mean   :1123   Mean   :422
##  3rd Qu.:2078   3rd Qu.:131.2   3rd Qu.:196   3rd Qu.:1326   3rd Qu.:452
##  Max.   :2986   Max.   :188.0   Max.   :226   Max.   :1600   Max.   :473
##     Largeur
##  Min.   :155
##  1st Qu.:164
```

```
##  Median :169
##  Mean   :169
##  3rd Qu.:175
##  Max.   :184
```

We can also calculate the correlations among the quantitative variables

```
# matrix of correlations
cor(carscomplete)
```

```
##            Cylindree Puissance Vitese  Poids Longueur Largeur
## Cylindree     1.0000    0.8610 0.6933 0.8973   0.8642  0.7091
## Puissance     0.8610    1.0000 0.8940 0.7692   0.6885  0.5523
## Vitese        0.6933    0.8940 1.0000 0.5074   0.5319  0.3632
## Poids         0.8973    0.7692 0.5074 1.0000   0.8634  0.7000
## Longueur      0.8642    0.6885 0.5319 0.8634   1.0000  0.8638
## Largeur       0.7091    0.5523 0.3632 0.7000   0.8638  1.0000
```

In this case there are only six variables which makes it relatively easy to visually inspect the correlations directly from the matrix of correlations. Another option to keep exploring the cars is by using some type of graphic display to visualize the similarities and differences between them. An interesting option is the function stars which plots observations using glyphs representations.

```
# star plot of vehicles
stars(carscomplete, cex = 0.7, ncol = 7)
```

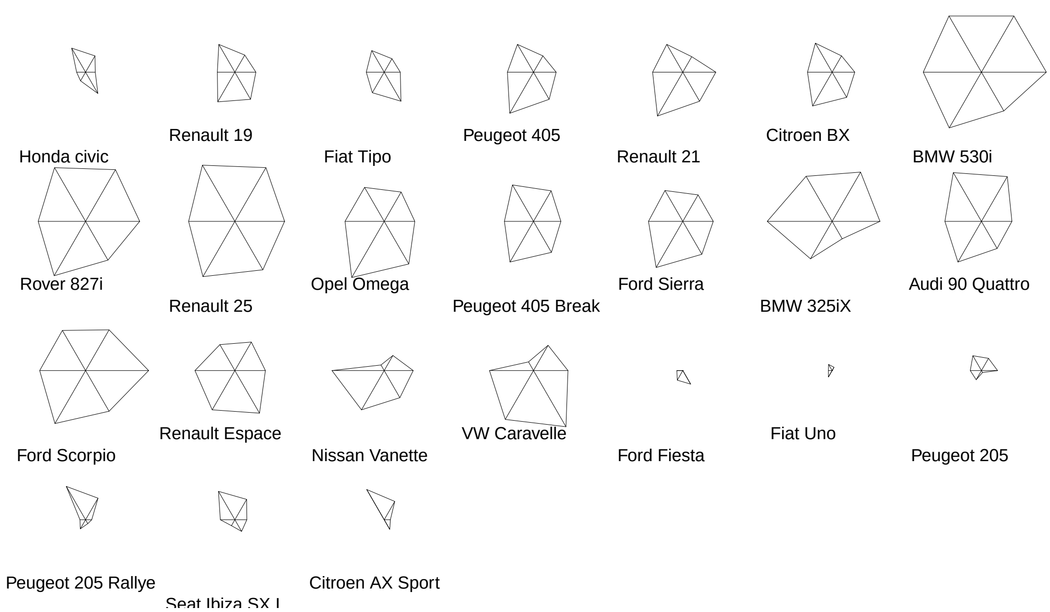

Figure 1: Star plot of cars

3 Function nipals()

To have a better understanding of the patterns in the data we can perform a PCA with NIPALS. The function `nipals()` has 3 arguments: `Data`, `comps`, and `scaled`. `Data`, as you may guess, is the data to be analyzed. This can be either a matrix or a data frame. `comps` is the number of components to be calculated. `scaled` specifies whether to standardize the data (`TRUE` by default). Let's perform a PCA on `carscomplete` with `nipals()`, asking for three compoenets.

```
# apply nipals with 3 components
my_nipals = nipals(carscomplete, comps = 3)

# what's in my_nipals?
my_nipals

##
## NIPALS algorithm
## ----------------------------------
## $values     eigenvalues
## $scores     scores (T-components)
## $loadings   loadings
## $cor.xt     X,T correlations
## $disto      distance to origin
## $contrib    contribution of rows
## $cos        squared cosinus
## $dmod       distance to the model
## ----------------------------------
##
```

3.1 Eigenvalues

The first element in the list of results is the table of eigenvalues. Note that the first two eigenvalues, associated to the first two components, explain up to 91% of the total variation in the data.

```
# check eigenvalues
my_nipals$values

##    values percentage cumulative
## v1 4.6173     76.956      76.96
## v2 0.8788     14.647      91.60
## v3 0.3035      5.058      96.66
```

3.2 Components

The second element in the list of results is `$scores` which is just another name for the components. We can inspect the scores of the first observations like this:

```
# T components
head(my_nipals$scores)

##                          t1      t2       t3
## Honda civic          -1.9630  0.3094 -0.48658
## Renault 19           -0.7649 -0.1522 -0.46533
## Fiat Tipo            -1.2707 -0.4251 -0.41889
## Peugeot 405          -0.2909 -0.4549 -0.21009
## Renault 21            0.1451 -0.6244 -0.01244
## Citroen BX           -0.5115 -0.1957 -0.17104
```

3.3 Correlations between variables and components

In order to check the association between the components and the variables, we use `$cor.xt` which is the matrix of correlations between the variables and the obtained scores.

```
# correlations between X-y and T
my_nipals$cor.xt

##               t1       t2       t3
## Cylindree 0.9612  0.01089  0.15044
## Puissance 0.9048  0.38218  0.01519
## Vitese    0.7509  0.61774 -0.20328
## Poids     0.9101 -0.17636  0.34699
## Longueur  0.9200 -0.30341 -0.05441
## Largeur   0.7977 -0.47737 -0.34053
```

Note that the first component is highly positively correlated with all the variables. This phenomenon is the typical *size effect* when all variables are positively correlated: since all the variables reflect -in one way or another- the size of a vehicle, the first component captures this information. Basically this component allows us to differentiate between smaller cars and larger cars. In contrast, the second component has positive correlations with `Puissance` (power) and `Vitese` (speed), and negative correlations with `Longeur` (length) and `Largeur` (height). This means that the second component is capturing information about the interaction speed-size. It distinguishes between smaller but faster cars, against bigger but slower cars.

3.4 Distance of each observation to the origin

`nipals()` also provides a couple of complementary results that will help us to assess the representation quality of the observations by the extracted components. The first of them is given by `$disto` which is the squared distance of each observation to the origin.

```
# distance to the origin
head(as.matrix(my_nipals$disto))

##                        [,1]
## Honda civic          4.4013
## Renault 19           0.8820
```

```
## Fiat Tipo           2.0937
## Peugeot 405         0.6841
## Renault 21          0.9212
## Citroen BX          0.5219
```

Besides the squared distances to the origin, we also have the squared cosinus (`$cos`) of the angle formed by each observation to the origin. Using both the distances and the squared cosinus, we can evaluate how well represented the observations are with the extracted components.

```
# cosinus
head(my_nipals$cos)

##                      cos2.1  cos2.2    cos2.3
## Honda civic         0.87555 0.02176 0.0537934
## Renault 19          0.66324 0.02627 0.2454909
## Fiat Tipo           0.77122 0.08629 0.0838069
## Peugeot 405         0.12372 0.30246 0.0645180
## Renault 21          0.02286 0.42322 0.0001679
## Citroen BX          0.50130 0.07338 0.0560489
```

A squared cosinus close to 1 indicates that the observation is near the given principal axis, thus being well represented by such axis. For instance, from the above table, Honda civic has `$cos2.1` of 0.875; this means that is well represented by the first principal axis. Conversely, Renault 21 has a very small squared cosinus (0.022), meaning that is not well represented on the first axis.

3.5 Contribution

Besides the distance to the origin, we can also evaluate the contribution that each observation has on the obtained components. The larger the value, the more contribution.

```
# contribution
head(my_nipals$contrib)

##                       ctr.1  ctr.2    ctr.3
## Honda civic         3.47742 0.4540 3.250377
## Renault 19          0.52790 0.1099 2.972641
## Fiat Tipo           1.45712 0.8566 2.408920
## Peugeot 405         0.07638 0.9810 0.605966
## Renault 21          0.01900 1.8484 0.002124
## Citroen BX          0.23611 0.1816 0.401613
```

3.6 Distance to the Model *DModX*

Another useful item to judge the representation quality of the observations, is given by the distance to the model (`$dmod`). These quantities represent a measure of how close the score value of an observation is to its real value. The larger the distance, the poorer the quality. Observations having a mediocre projection with the extracted components can be detected by checking their DModX. This is also used to identify outliers or atypical observations.

```
# distance to the model
head(my_nipals$dmod)

##                          t1     t2     t3
## Honda civic          0.3962 0.8971 1.0745
## Renault 19           0.2148 0.5436 0.2862
## Fiat Tipo            0.3464 0.5921 0.6133
## Peugeot 405          0.4336 0.7792 1.7394
## Renault 21           0.6510 1.0128 2.5465
## Citroen BX           0.1883 0.4406 0.9621
```

For instance, if we plot the distances-to-the-model of the first component, we can see that VW Caravelle, Nissan Vanette and BMW 325iX have higher distances. This is a clear indication that these observations are not well summarized by the first component.

```
# barplot
barplot(my_nipals$dmod[, 1], las = 2, border = NA)
```

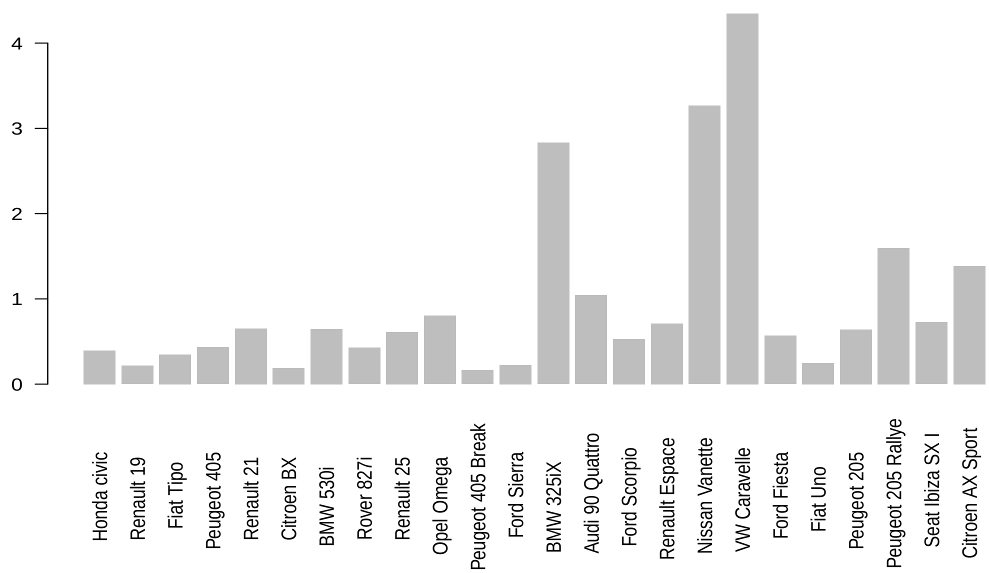

Figure 2: Distance to the Model: DModX

4 Plotting "nipals" 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. When plotting the variables, they are displayed 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 principal 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 orothogonal to each other.

```
# default plot
plot(my_nipals)
```

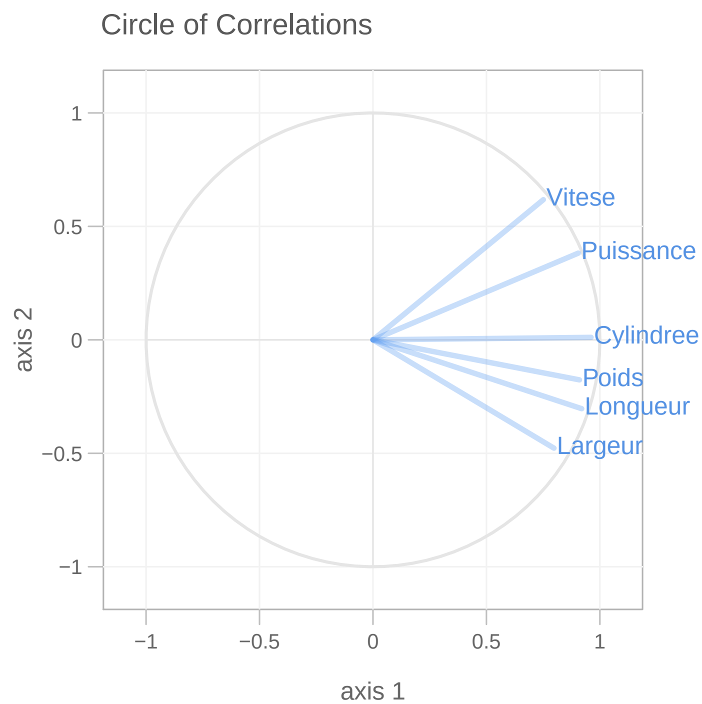

Figure 3: Circle of Correlations (axes 1-2)

We can also select a different pair of axes with the argument `comps`. For example, to see the correlations between the first and the third principal axes, we need to specify `comps= c(1,3)`

```r
# another plot
plot(my_nipals, comps = c(1, 3))
```

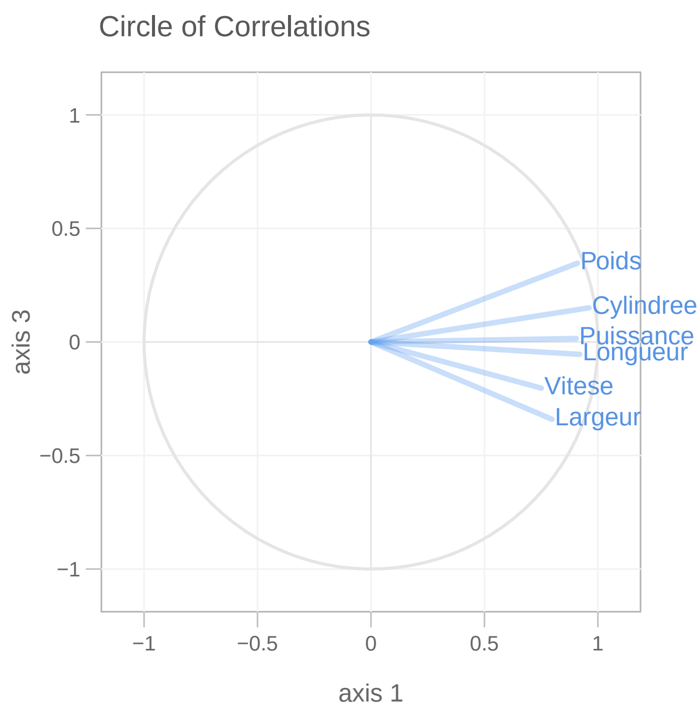

Figure 4: Circle of Correlations (axes 1-3)

4.2 Plotting observations

The alternative output of the function `plot()` is to show the observations with a scatter-plot on the specified components. To indicate that we want to plot the observations (the scores), we have to use the `what` argument: `what="observations"`. By default, the observations are plotted without showing their names. To show the labels, simply use the argument `show.names=TRUE`

```r
# default plot
plot(my_nipals, what = "observations", show.names = TRUE)
```

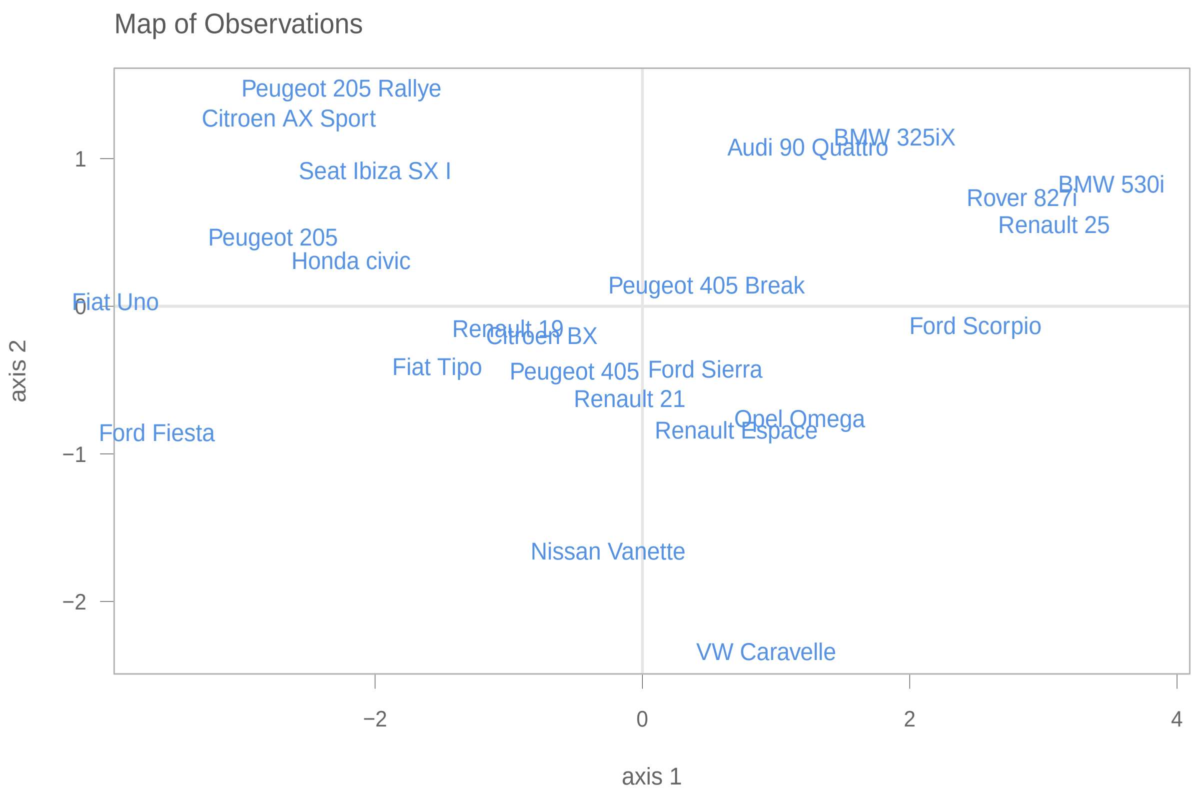

Figure 5: Plot of observations (comps 1-2)

References

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

Tenenhaus M. (2007) *Statistique: Methodes pour Decrire, Expliquer et Prevoir*. Paris: Dunod.