

# Zad1

June 12, 2019

```
[1]: # small dataset
set.seed(212)
Data = matrix(rnorm(30), 10, 3)
```

Data

-0.2391731	0.15034879	0.55113696
0.6769356	0.50967649	1.21557683
-2.4403360	-0.77331482	-0.60781873
1.2408845	1.87566596	1.35475360
-0.3265144	0.78830571	-1.33758388
0.1544909	-2.30671309	-0.79244866
1.0368712	-1.37584147	-1.02442868
-0.7796077	-1.02111184	-0.02768991
0.6212641	-1.37921485	-0.61268087
0.2994313	-0.03476825	0.16605702

```
[2]: # centering with 'scale()'
center_scale <- function(x) {
  scale(x, scale = FALSE)
}
```

# apply it

```
Data <- center_scale(Data)
```

```
[3]: pca.1 <- eigen(cov(Data))
pca.2 <- svd(Data)
print(pca.1)
print(pca.2)
```

eigen() decomposition

\$values

```
[1] 2.1502355 1.0035836 0.3943153
```

\$vectors

	[,1]	[,2]	[,3]
[1,]	-0.3733788	0.91006959	-0.1798933
[2,]	-0.7723187	-0.41237480	-0.4831882
[3,]	-0.5139183	-0.04147729	0.8568358

```
$d
[1] 4.399104 3.005371 1.883836
```

```
$u
      [,1]      [,2]      [,3]
[1,] 0.1440581 0.15853952 0.19651549
[2,] 0.3625205 -0.06039715 0.31907987
[3,] -0.3403220 0.68235028 0.11648889
[4,] 0.6664622 -0.04181722 -0.02183565
[5,] 0.0279996 0.24645712 -0.81783224
[6,] -0.4108599 -0.31635046 0.17802845
[7,] -0.1996413 -0.45902216 -0.25050631
[8,] -0.1750972 0.15346328 0.28532199
[9,] -0.1874068 -0.32795064 -0.02267588
[10,] 0.1122868 -0.03527257 0.01741539
```

```
$v
      [,1]      [,2]      [,3]
[1,] 0.3733788 -0.91006959 -0.1798933
[2,] 0.7723187 0.41237480 -0.4831882
[3,] 0.5139183 0.04147729 0.8568358
```

```
[4]: check_equal <- function(M, v, e) {
      n <- nrow(M) - 1
      v^2/n
    }

    check_equal(Data, pca.2$d, pca.1$values)

    1.2.15023551696937 2.1.00358360274837 3.0.394315339124668
```

```
[12]: prco <- prcomp(Data)
      print(prco$sdev^2)
      print(prco$rotation)
```

```
[1] 2.1502355 1.0035836 0.3943153
      PC1      PC2      PC3
[1,] 0.3733788 -0.91006959 -0.1798933
[2,] 0.7723187 0.41237480 -0.4831882
[3,] 0.5139183 0.04147729 0.8568358
```

```
[7]: library(FactoMineR)
```

```
Error in library(FactoMineR): there is no package called FactoMineR
Traceback:
```

```
1. library(FactoMineR)

2. stop(txt, domain = NA)
```

[6]: `PCA(Data)`

```
Error in PCA(Data): could not find function "PCA"
Traceback:
```

[17]: `?PCA()`

PCA Principal Component Analysis (PCA) PCA multivariate PCA Performs Principal Component Analysis (PCA) with supplementary individuals, supplementary quantitative variables and supplementary categorical variables.

Missing values are replaced by the column mean.

```
PCA(X, scale.unit = TRUE, ncp = 5, ind.sup = NULL,
     quanti.sup = NULL, quali.sup = NULL, row.w = NULL,
     col.w = NULL, graph = TRUE, axes = c(1,2))
```

a data frame with  $n$  rows (individuals) and  $p$  columns (numeric variables)

number of dimensions kept in the results (by default 5)

a boolean, if TRUE (value set by default) then data are scaled to unit variance

a vector indicating the indexes of the supplementary individuals

a vector indicating the indexes of the quantitative supplementary variables

a vector indicating the indexes of the categorical supplementary variables

an optional row weights (by default, a vector of 1 for uniform row weights); the weights are given only for the active individuals

an optional column weights (by default, uniform column weights); the weights are given only for the active variables

boolean, if TRUE a graph is displayed

a length 2 vector specifying the components to plot Returns a list including:

a matrix containing all the eigenvalues, the percentage of variance and the cumulative percentage of variance

a list of matrices containing all the results for the active variables (coordinates, correlation between variables and axes, square cosine, contributions)

a list of matrices containing all the results for the active individuals (coordinates, square cosine, contributions)

a list of matrices containing all the results for the supplementary individuals (coordinates, square cosine)

a list of matrices containing all the results for the supplementary quantitative variables (coordinates, correlation between variables and axes)

a list of matrices containing all the results for the supplementary categorical variables (coordinates of each categories of each variables, v.test which is a criterion with a Normal distribution, and eta2 which is the square correlation coefficient between a qualitative variable and a dimension)

Returns the individuals factor map and the variables factor map.

The plots may be improved using the argument `autolab`, modifying the size of the labels or selecting some elements thanks to the `plot.PCAplot.PCA` function.

Francois Husson  
Francois.Husson@agrocampus-ouest.fr, Jeremy Mazet  
Husson, F., Le, S. and Pages, J. (2010).  
Exploratory Multivariate Analysis by Example Using R, *Chapman and Hall*.  
`print.PCAprint.PCA`,  
`summary.PCAsummary.PCA`, `plot.PCAplot.PCA`, `dimdescdimdesc`,

[https://www.youtube.com/watch?v=CTSbxU6KLbMlist=PLnZgp6epRBbTsZEFXi\\_p6W48HhNyqwxIuindex=3VideoshowinghowtoperformPCAwithFactoMineR](https://www.youtube.com/watch?v=CTSbxU6KLbMlist=PLnZgp6epRBbTsZEFXi_p6W48HhNyqwxIuindex=3VideoshowinghowtoperformPCAwithFactoMineR)

`data(decathlon) res.pca <- PCA(decathlon, quanti.sup = 11:12, quali.sup=13)` plot of the eigenvalues  
`barplot(res.pcaeig[,1], main = "Eigenvalues", names.arg = 1 : nrow(res.pcaeig))`  
`summary(res.pca)` `plot(res.pca,choix="ind",habillage=13)` `dimdesc(res.pca, axes = 1:2)` To draw ellipses around the categories of the 13th variable (which is categorical) `plotellipses(res.pca,13)`

Example with missing data use package `missMDA` Not run: `require(missMDA)` `data(orange)`  
`nb <- estim_ncpPCA(orange,ncp.min = 0,ncp.max = 5,method.cv = "Kfold",nbsim = 50)`  
`imputed <- imputePCA(orange,ncp = nbncp)` `res.pca <- PCA(imputedcompleteObs)`

End(Not run)

[18]: `?prcomp`

```
prcompPrincipal Components Analysisprcomp plot.prcompprcompplot.prcomp
prcomp.defaultprcompprcomp.default prcomp.formulaprcompprcomp.formula
predict.prcompprcomppredict.prcomp print.prcompprcompprint.prcomp
print.summary.prcompprcompprint.summary.prcomp summary.prcompprcompsummary.prcomp
multivariateprcomp Performs a principal components analysis on the given data matrix and
returns the results as an object of class prcomp.
```

```
prcomp(x, ...)
```

```
## S3 method for class 'formula'
```

```
prcomp(formula, data = NULL, subset, na.action, ...)
```

```
## Default S3 method:
```

```
prcomp(x, retx = TRUE, center = TRUE, scale. = FALSE,
       tol = NULL, rank. = NULL, ...)
```

```
## S3 method for class 'prcomp'
```

```
predict(object, newdata, ...)
```

a formula with no response variable, referring only to numeric variables.

an optional data frame (or similar: see `model.frame`) containing the variables in the formula `formula`. By default the variables are taken from `environment(formula)`.

an optional vector used to select rows (observations) of the data matrix `x`.

a function which indicates what should happen when the data contain NAs. The default is set by the `na.action` setting of `options()`, and is `na.fail` if that is unset. The 'factory-fresh' default is `na.omit`.

arguments passed to or from other methods. If `x` is a formula one might specify `scale` or `tol`.

a numeric or complex matrix (or data frame) which provides the data for the principal components analysis.

a logical value indicating whether the rotated variables should be returned.

a logical value indicating whether the variables should be shifted to be zero centered. Alternately, a vector of length equal the number of columns of `x` can be supplied. The value is passed to `scale`.

a logical value indicating whether the variables should be scaled to have unit variance before the analysis takes place. The default is `FALSE` for consistency with `S`, but in general scaling is advisable. Alternatively, a vector of length equal the number of columns of `x` can be supplied. The value is passed to `scalescale`.

a value indicating the magnitude below which components should be omitted. (Components are omitted if their standard deviations are less than or equal to `tol` times the standard deviation of the first component.) With the default null setting, no components are omitted (unless `rank` is specified less than `min(dim(x))`). Other settings for `tol` could be `tol = 0` or `tol = sqrt(.Machine$double.eps)`, which would omit essentially constant components.

optionally, a number specifying the maximal rank, i.e., maximal number of principal components to be used. Can be set as alternative or in addition to `tol`, useful notably when the desired rank is considerably smaller than the dimensions of the matrix.

object of class inheriting from "prcomp"

An optional data frame or matrix in which to look for variables with which to predict. If omitted, the scores are used. If the original fit used a formula or a data frame or a matrix with column names, `newdata` must contain columns with the same names. Otherwise it must contain the same number of columns, to be used in the same order.

The calculation is done by a singular value decomposition of the (centered and possibly scaled) data matrix, not by using `eigen` on the covariance matrix. This is generally the preferred method for numerical accuracy. The `print` method for these objects prints the results in a nice format and the `plot` method produces a scree plot.

Unlike `princomp`, variances are computed with the usual divisor  $N - 1$ .

Note that `scale = TRUE` cannot be used if there are zero or constant (for `center = TRUE`) variables. `prcomp` returns a list with class "prcomp" containing the following components:

the standard deviations of the principal components (i.e., the square roots of the eigenvalues of the covariance/correlation matrix, though the calculation is actually done with the singular values of the data matrix).

the matrix of variable loadings (i.e., a matrix whose columns contain the eigenvectors). The function `princomp` returns this in the element `loadings`.

if `retx` is true the value of the rotated data (the centred (and scaled if requested) data multiplied by the rotation matrix) is returned. Hence, `cov(x)` is the diagonal matrix `diag(sdev^2)`. For the formula method, `na.predict` is applied to handle the treatment of values omitted by the `na.action`.

the centering and scaling used, or FALSE. The signs of the columns of the rotation matrix are arbitrary, and so may differ between different programs for PCA, and even between different builds of . Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole.

Mardia, K. V., J. T. Kent, and J. M. Bibby (1979) *Multivariate Analysis*, London: Academic Press.  
 Venables, W. N. and B. D. Ripley (2002) *Modern Applied Statistics with S*, Springer-Verlag.

```
biplot.prcompbiplot.prcomp,  screeplotscreeplot,  prin-
compprincomp,  corcor,  covcov,  svdsvd,  eigeneigen.      C <- chol(S <-
toeplitz(.9  (0      : 31)))Cov.matrixanditsrootall.equal(S,crossprod(C))set.seed(17)X <-
-matrix(rnorm(32000),1000,32)Z <- -Xall.equal(cov(Z),S,tol = 0.08)pZ <-
-prcomp(Z,tol = 0.1)summary(pZ)only 14PCs(outof32)orchooseonly3PCsmoredirectly : pz3 <-
-prcomp(Z,rank. = 3)summary(pz3)samenumbersasthefirst3abovestopifnot(ncol(pZrotation)
== 14, ncol(pz3rotation) == 3, all.equal(pz3sdev, pz3sdev, tol = 1e - 15))exactlyequaltypically
signs are random require(graphics) the variances of the variables in the USArrests
data vary by orders of magnitude, so scaling is appropriate prcomp(USArrests) inappropri-
ate prcomp(USArrests, scale = TRUE) prcomp( Murder + Assault + Rape, data = USAr-
rests, scale = TRUE) plot(prcomp(USArrests)) summary(prcomp(USArrests, scale = TRUE)) bi-
plot(prcomp(USArrests, scale = TRUE))
```

[19]: `?princomp`

princompPrincipal Components Analysisprincomp plot.princompprincompplot.princomp  
 predict.princompprincomppredict.princomp princomp.defaultprincompprincomp.default prin-  
 comp.formulaprincompprincomp.formula print.princompprincompprint.princomp multivari-  
 ateprincomp princomp performs a principal components analysis on the given numeric data ma-  
 trix and returns the results as an object of class princomp.

```
princomp(x, ...)
```

```
## S3 method for class 'formula'
```

```
princomp(formula, data = NULL, subset, na.action, ...)
```

```
## Default S3 method:
```

```
princomp(x, cor = FALSE, scores = TRUE, covmat = NULL,
         subset = rep_len(TRUE, nrow(as.matrix(x))), fix_sign = TRUE, ...)
```

```
## S3 method for class 'princomp'
```

```
predict(object, newdata, ...)
```

a formula with no response variable, referring only to numeric variables.

an optional data frame (or similar: see `model.frame`) containing the variables in the formula `formula`. By default the variables are taken from `environment(formula)`.

an optional vector used to select rows (observations) of the data matrix `x`.

a function which indicates what should happen when the data contain NAs. The default is set by the `na.action` setting of `options`, and is `na.fail` if that is unset. The ‘factory-fresh’ default is `na.omit`.

a numeric matrix or data frame which provides the data for the principal components analysis.

a logical value indicating whether the calculation should use the correlation matrix or the covariance matrix. (The correlation matrix can only be used if there are no constant variables.)

a logical value indicating whether the score on each principal component should be calculated.

a covariance matrix, or a covariance list as returned by `cov.wtcov.wt` (and `cov.mvecov.mve` or `cov.mcdcov.mcd` from package <https://CRAN.R-project.org/package=MASSMASS>). If supplied, this is used rather than the covariance matrix of `x`.

Should the signs of the loadings and scores be chosen so that the first element of each loading is non-negative?

arguments passed to or from other methods. If `x` is a formula one might specify `cor` or `scores`.

Object of class inheriting from "princomp".

An optional data frame or matrix in which to look for variables with which to predict. If omitted, the scores are used. If the original fit used a formula or a data frame or a matrix with column names, `newdata` must contain columns with the same names. Otherwise it must contain the same number of columns, to be used in the same order.

`princomp` is a generic function with "formula" and "default" methods.

The calculation is done using `eigen` on the correlation or covariance matrix, as determined by `corcor`. This is done for compatibility with the S-PLUS result. A preferred method of calculation is to use `svdsvd` on `x`, as is done in `prcomp`.

Note that the default calculation uses divisor `N` for the covariance matrix.

The `print` method for these objects prints the results in a nice format and the `plot` method produces a scree plot (`screeplot`). There is also a `biplot` method.

If `x` is a formula then the standard NA-handling is applied to the scores (if requested): see `na.predict`.

`princomp` only handles so-called R-mode PCA, that is feature extraction of variables. If a data matrix is supplied (possibly via a formula) it is required that there are at least as many units as variables. For Q-mode PCA use `prcomp`. `princomp` returns a list with class "princomp" containing the following components:

the standard deviations of the principal components.

the matrix of variable loadings (i.e., a matrix whose columns contain the eigenvectors). This is of class "loadings": see `loadings` for its print method.

the means that were subtracted.

the scalings applied to each variable.

the number of observations.

if `scores = TRUE`, the scores of the supplied data on the principal components. These are non-null only if `x` was supplied, and if `covmat` was also supplied if it was a covariance list. For the formula method, `na.predict` is applied to handle the treatment of values omitted by the `na.action`.

the matched call.

If relevant. The signs of the columns of the loadings and scores are arbitrary, and so may differ between different programs for PCA, and even between different builds of : `fix_sign = TRUE` alleviates that. Mardia, K. V., J. T. Kent and J. M. Bibby (1979). *Multivariate Analysis*, London: Academic Press.

Venables, W. N. and B. D. Ripley (2002). *Modern Applied Statistics with S*, Springer-Verlag.  
`summary.princomp`, `summary.princomp`, `screeplot`, `biplot.princomp`, `prcomp`, `corcor`, `covcov`, `eigen`. `require(graphics)`

The variances of the variables in the USArrests data vary by orders of magnitude, so scaling is appropriate (`pc.cr <- princomp(USArrests)`) inappropriate `princomp(USArrests, cor = TRUE)` *=* `prcomp(USArrests, scale = TRUE)` Similar, but different: The standard deviations differ by a factor of  $\sqrt{49/50}$

`summary(pc.cr <- princomp(USArrests, cor = TRUE))` `loadings(pc.cr)` note that blank entries are small but not zero The signs of the columns of the loadings are arbitrary `plot(pc.cr)` shows a screeplot. `biplot(pc.cr)`

Formula interface `princomp(., data = USArrests, cor = TRUE)`

NA-handling `USArrests[1, 2] <- NA` `pc.cr <- princomp( Murder + Assault + UrbanPop, data = USArrests, na.action = na.exclude, cor = TRUE)` `pc.cr$scores[1 : 5, ]`

(Simple) Robust PCA: Classical: (`pc.cl <- princomp(stackloss)`) Robust: (`pc.rob <- princomp(stackloss, covmat = MASS::cov.rob(stackloss))`)