

# Package ‘robustfa’

September 13, 2025

**Type** Package

**Title** Object Oriented Solution for Robust Factor Analysis

**Version** 1.2-0

**Date** 2025-09-10

**Author** Frederic Bertrand [cre] (ORCID:  
<<https://orcid.org/0000-0002-0837-8281>>),  
Ying-Ying Zhang (Robert) [aut]

**Maintainer** Frederic Bertrand <[frederic.bertrand@lecnam.net](mailto:frederic.bertrand@lecnam.net)>

**Description** Outliers virtually exist in any datasets of any application field.

To avoid the impact of outliers, we need to use robust estimators.

Classical estimators of multivariate mean and covariance matrix are the sample mean and the sample covariance matrix. Outliers will affect the sample mean and the sample covariance matrix, and thus they will affect the classical factor analysis which depends on the classical estimators (Pison, G., Rousseeuw, P.J., Filzmoser, P. and Croux, C. (2003) <[doi:10.1016/S0047-259X\(02\)00007-6](https://doi.org/10.1016/S0047-259X(02)00007-6)>).

So it is necessary to use the robust estimators of the sample mean and the sample covariance matrix. There are several robust estimators in the literature: Minimum Covariance Determinant estimator, Orthogonalized Gnanadesikan-Kettenring, Minimum Volume Ellipsoid, M, S, and Stahel-Donoho.

The most direct way to make multivariate analysis more robust is to replace the sample mean and the sample covariance matrix of the classical estimators to robust estimators (Maronna, R.A., Martin, D. and Yohai, V. (2006) <[doi:10.1002/0470010940](https://doi.org/10.1002/0470010940)>) (Todorov, V. and Filzmoser, P. (2009) <[doi:10.18637/jss.v032.i03](https://doi.org/10.18637/jss.v032.i03)>), which is our choice of robust factor analysis. We created an object oriented solution for robust factor analysis based on new S4 classes.

**License** GPL (>= 2)

**Depends** rrcov, R (>= 2.15.0)

**Imports** methods, stats4, stats

**Suggests** grid, lattice, cluster, mclust, MASS, ellipse, knitr,  
rmarkdown, testthat (>= 3.0.0)

**LazyLoad** yes**LazyData** yes**Encoding** UTF-8**RoxygenNote** 7.3.2**VignetteBuilder** knitr, rmarkdown**URL** <https://fbertran.github.io/robustfa/>,  
<https://github.com/fbertran/robustfa/>**BugReports** <https://github.com/fbertran/robustfa/issues/>**Config/testthat/edition** 3**NeedsCompilation** no**Repository** CRAN**Date/Publication** 2025-09-12 22:50:02 UTC

## Contents

robustfa-package . . . . .	3
computeScores . . . . .	4
compute_cov_cor . . . . .	5
detail . . . . .	6
Fa-class . . . . .	7
FaClassic . . . . .	10
FaClassic-class . . . . .	11
FaCov . . . . .	13
FaCov-class . . . . .	14
factorScorePca . . . . .	16
factorScorePfa . . . . .	18
FaRobust-class . . . . .	20
fsOrder . . . . .	22
getCenter-methods . . . . .	23
getEigenvalues-methods . . . . .	23
getFa-methods . . . . .	23
getLoadings-methods . . . . .	24
getQuan-methods . . . . .	24
getScores-methods . . . . .	24
getSdev-methods . . . . .	25
myFaPrint . . . . .	25
myplotDD . . . . .	26
plot-methods . . . . .	27
predict-methods . . . . .	29
print-methods . . . . .	30
stock611 . . . . .	31
summary-methods . . . . .	32
SummaryFa-class . . . . .	33
Ulogical-class . . . . .	34

Unnumeric-class . . . . .	34
---------------------------	----

<b>Index</b>	<b>36</b>
--------------	-----------

---

robustfa-package	<i>An Object Oriented Solution for Robust Factor Analysis</i>
------------------	---

---

## Description

Outliers virtually exist in any datasets of any application field. To avoid the impact of outliers, we need to use robust estimators. Classical estimators of multivariate mean and covariance matrix are the sample mean and the sample covariance matrix. Outliers will affect the sample mean and the sample covariance matrix, and thus they will affect the classical factor analysis which depends on the classical estimators (Pison, G., Rousseeuw, P.J., Filzmoser, P. and Croux, C. (2003) [doi:10.1016/S0047259X\(02\)000076](https://doi.org/10.1016/S0047259X(02)000076)). So it is necessary to use the robust estimators of the sample mean and the sample covariance matrix. There are several robust estimators in the literature: MCD, OGK, MVE, M, S, and Stahel-Donoho. The most direct way to robustify multivariate analysis is to replace the sample mean and the sample covariance matrix of the classical estimators to robust estimators (Maronna, R.A., Martin, D. and Yohai, V. (2006) [doi:10.1002/0470010940](https://doi.org/10.1002/0470010940)) (Todorov, V. and Filzmoser, P. (2009) [doi:10.18637/jss.v032.i03](https://doi.org/10.18637/jss.v032.i03)), which is our choice of robust factor analysis. robustfa is an object oriented solution for robust factor analysis. In the solution, new S4 classes "Fa", "FaClassic", "FaRobust", "FaCov", "SummaryFa" are created.

## Details

Package:	robustfa
Type:	Package
Version:	1.0-5
Date:	2013-11-09
License:	GPL (>= 2)
Depends:	methods

The most important functions are: [FaClassic](#), [FaCov](#), [factorScorePca](#), [factorScorePfa](#)

## Author(s)

Ying-Ying Zhang (Robert)

Maintainer: Ying-Ying Zhang (Robert) <[robertzhangying@qq.com](mailto:robertzhangying@qq.com)>

## References

Zhang, Y. Y. (2013), An Object Oriented Solution for Robust Factor Analysis.

## Examples

```
library("robustfa")
```

---

computeScores	<i>Compute Factor Scores</i>
---------------	------------------------------

---

### Description

Compute factor scores on the result of factor analysis method, the method is one of "mle", "pca", and "pfa".

### Usage

```
computeScores(out, x = data, covmat = covmat, cor = cor, scoresMethod = scoresMethod)
```

### Arguments

out	The result of factorScorePca(), factorScorePfa(), or factanal(). It is a list.
x	A numeric matrix.
covmat	A list with components: cov, center, and n.obs.
cor	A logical value indicating whether the calculation should use the covariance matrix (cor = FALSE) or the correlation matrix (cor = TRUE).
scoresMethod	Type of scores to produce, if any. The default is "none", "regression" gives Thompson's scores, "Bartlett" gives Bartlett's weighted least-squares scores.

### Value

The output is a list. Except for the components of out, it also has components:

scoringCoef	The scoring coefficients.
scores	The matrix of scores.
meanF	The sample mean of the scores.
corF	The sample correlation matrix of the scores.
eigenvalues	The eigenvalues of the running matrix.
covariance	The covariance matrix.
correlation	The correlation matrix.
usedMatrix	The used matrix (running matrix) to compute scoringCoef etc..
reducedCorrelation	NULL. The reduced correlation matrix, reducedCorrelation is calculated in factorScorePfa.R.

scoringCoef = F = meanF = corF = NULL if scoresMethod = "none".

### Author(s)

Ying-Ying Zhang (Robert) <robertzhangyiyang@qq.com>

## References

Zhang, Y. Y. (2013), An Object Oriented Solution for Robust Factor Analysis.

## Examples

```
data("stock611")
stock604 = stock611[-c(92,2,337,338,379,539,79), ]
data = as.matrix(stock604[, 3:12])

factors = 2
cor = TRUE
scoresMethod = "regression"

covx = rrcov::Cov(data)
covmat = list(cov = rrcov::getCov(covx), center = rrcov::getCenter(covx), n.obs = covx@n.obs)

out = stats::factanal(factors = factors, covmat = covmat)

out = computeScores(out, x = data, covmat = covmat, cor = cor, scoresMethod = scoresMethod)
out
```

---

compute_cov_cor	<i>Compute the Robust Covariance and Correlation Matrix of A Numeric Matrix</i>
-----------------	---

---

## Description

Compute the robust covariance and correlation matrix of a numeric matrix. The function is used to check whether  $S_r \neq S_{r\_tilda}$  and  $R_r == R_{r\_tilda}$ ?

## Usage

```
compute_cov_cor(x, control)
```

## Arguments

x	A numeric matrix or an object that can be coerced to a numeric matrix.
control	A control object (S4) for one of the available control classes, e.g. CovControlMcd-class, CovControlOgk-class, CovControlSest-class, etc., containing estimation options. The class of this object defines which estimator will be used. Alternatively a character string can be specified which names the estimator - one of auto, sde, mcd, ogk, m, mve, sfast, surreal, bisquare, rocke. If "auto" is specified or the argument is missing, the function will select the estimator.

**Value**

A list with the following components:

<code>S_r</code>	The robust covariance matrix of <code>cov_x</code> .
<code>S_r_tilda</code>	The robust covariance matrix of <code>cov_scale_x</code> .
<code>R_r</code>	The robust correlation matrix of <code>cov_x</code> .
<code>R_r_tilda</code>	The robust correlation matrix of <code>cov_scale_x</code> .

```
cov_x = rrcov::CovRobust(x = x, control = control) cov_scale_x = rrcov::CovRobust(x = scale(x), control = control)
```

**Author(s)**

Ying-Ying Zhang (Robert) <robertzhangying@qq.com>

**References**

Zhang, Y. Y. (2013), An Object Oriented Solution for Robust Factor Analysis.

**Examples**

```
data("hbk")
hbk.x = hbk[,1:3]

compute_cov_cor(x = hbk.x, control = "mcd")
```

---

detail

*Show Details of an Object*

---

**Description**

Show details of an object.

**Usage**

```
detail(x)
```

**Arguments**

`x` Any R object to be tested.

**Value**

A list with components:

<code>x</code>	The argument <code>x</code> .
<code>isS4</code>	Logical, indicates whether <code>x</code> is an S4 object.
<code>isObject</code>	Logical, indicates whether <code>x</code> is an object, i.e., with a class attribute.
<code>class</code>	The class of <code>x</code> .
<code>attributes</code>	The attributes of <code>x</code> . Usually <code>result\$attributes</code> is also a list.

**Author(s)**

Ying-Ying Zhang (Robert) <robertzhangying@qq.com>

**References**

Zhang, Y. Y. (2013), An Object Oriented Solution for Robust Factor Analysis.

**See Also**

[isS4](#), [is.object](#), [class](#), [attributes](#)

**Examples**

```
data(stock611)
detail(stock611)

facovRegOgk=FaCov(x=scale(stock611[,3:12]), factors=3, cov.control = rrcov::CovControlOgk(),
scoresMethod = "regression"); facovRegOgk
detail(facovRegOgk)
```

---

Fa-class

*Class "Fa"*

---

**Description**

Class "Fa" is a virtual base class for all classical and robust FA classes. "Fa" serves as a base class for deriving all other classes representing the results of the classical and robust Factor Analysis methods.

**Objects from the Class**

A virtual Class: No objects may be created from it.

**Slots**

**call:** Object of class "Language" an unevaluated function call.

**converged:** Object of class "Ulogical" a logical character indicates whether the iterations converged.

**loadings:** Object of class "matrix" the matrix of variable loadings.

**communality:** Object of class "Uvector" the communality.

**uniquenesses:** Object of class "vector" the uniquenesses computed.

**cor:** Object of class "Ulogical" A logical value indicating whether the calculation should use the covariance matrix (cor = FALSE) or the correlation matrix (cor = TRUE).

**covariance:** Object of class "matrix" The robust/classical covariance matrix.

**correlation:** Object of class "matrix" The robust/classical correlation matrix.

**usedMatrix:** Object of class "matrix" The used matrix (running matrix). It may be the covariance or correlation matrix according to the value of cor.

**reducedCorrelation:** Object of class "Umatrix" The last reduced correlation matrix. reduced-Correlation is only calculated in factorScorePfa.R.

**criteria:** Object of class "Unumeric". The results of the optimization: the value of the negative log-likelihood and information on the iterations used.

**factors:** Object of class "numeric" the number of factors.

**dof:** Object of class "Unumeric". The number of degrees of freedom of the factor analysis model.

**method:** Object of class "character". The method: one of "mle", "pca", and "pfa".

**scores:** Object of class "Umatrix". If requested, a matrix of scores.

**scoresMethod:** Object of class "character". The scores method: one of "none", "regression", and "Bartlett".

**scoringCoef:** Object of class "Umatrix" the matrix of scoring coefficients.

**meanF:** Object of class "Uvector" the column means of scores.

**corF:** Object of class "Umatrix" the correlation matrix of the scores.

**STATISTIC:** Object of class "Unumeric". The significance-test statistic, if it can be computed.

**PVAL:** Object of class "Unumeric". The significance-test P value, if it can be computed.

**n.obs:** Object of class "numeric". The number of observations.

**center:** Object of class "Uvector". The center of the data.

**eigenvalues:** Object of class "vector" the eigenvalues.

**cov.control:** Object of class "UCovControl". Record the cov control method.

**Methods**

**getCenter** signature(obj = "Fa"): center of the data

**getEigenvalues** signature(obj = "Fa"): the eigenvalues of the covariance/correlation matrix

**getFa** signature(obj = "Fa"): returns an S3 list of class fa for compatibility with the function factanal(). Thus the standard screeplot() can be used.

**getLoadings** signature(obj = "Fa"): returns the matrix loadings



- getQuan** signature(obj = "Fa"): returns the number of observations used in the computation, i.e., n.obs
- getScores** signature(obj = "Fa"): if requested, a matrix of scores.
- getSdev** signature(obj = "Fa"): returns the standard deviations of the factor analysis, i.e., the square roots of the eigenvalues of the covariance/correlation matrix
- plot** signature(x = "Fa", y = "missing"): produces a scatterplot of the factor scores (if which = "factorScore") or shows the eigenvalues plot (if which = "screeplot")
- predict** signature(object = "Fa"): calculates prediction using the results in object. The new-data argument is an optional data frame or matrix in which to look for variables with which to predict. If newdata is omitted, the scores are used.
- print** signature(x = "Fa"): prints the results. obj = print(obj)
- summary** signature(object = "Fa"): produce result summaries of an object of class "Fa".

### Author(s)

Ying-Ying Zhang (Robert) <robertzhangying@qq.com>

### References

- Bartlett, M. S. (1937) The statistical conception of mental factors. *British Journal of Psychology*, **28**, 97–104.
- Bartlett, M. S. (1938) Methods of estimating mental factors. *Nature*, **141**, 609–610.
- Joreskog, K. G. (1963) *Statistical Estimation in Factor Analysis*. Almqvist and Wicksell.
- Lawley, D. N. and Maxwell, A. E. (1971) *Factor Analysis as a Statistical Method*. Second edition. Butterworths.
- Thomson, G. H. (1951) *The Factorial Analysis of Human Ability*. London University Press.
- Zhang, Y. Y. (2013), An Object Oriented Solution for Robust Factor Analysis.
- Zhang, Y. Y. (2014), Robust Factor Analysis and Its Applications in the CSI 100 Index, *Open Journal of Social Sciences* 2(07):12-18, doi:10.4236/jss.2014.27003.

### See Also

[FaClassic-class](#), [FaCov-class](#), [FaRobust-class](#), [Fa-class](#)

### Examples

```
showClass("Fa")
```

FaClassic

*Classical Factor Analysis***Description**

Performs a classical factor analysis and returns the results as an object of class "FaClassic" (a.k.a. constructor).

**Usage**

```
FaClassic(x, ...)
## S3 method for class 'formula'
FaClassic(formula, data = NULL, factors = 2, cor = FALSE, method = "mle",
scoresMethod = "none", ...)
## Default S3 method:
FaClassic(x, factors = 2, cor = FALSE, method = c("mle", "pca", "pfa"),
scoresMethod = c("none", "regression", "Bartlett"),...)
```

**Arguments**

x	A formula or a numeric matrix or an object that can be coerced to a numeric matrix.
...	Arguments passed to or from other methods.
formula	A formula with no response variable, referring only to numeric variables.
data	An optional data frame (or similar: see <a href="#">model.frame</a> ) containing the variables in the formula.
factors	The number of factors to be fitted.
cor	A logical value indicating whether the calculation should use the covariance matrix (cor = FALSE) or the correlation matrix (cor = TRUE).
method	The method of factor analysis, one of "mle" (the default), "pca", and "pfa".
scoresMethod	Type of scores to produce, if any. The default is "none", "regression" gives Thompson's scores, "Bartlett" gives Bartlett's weighted least-squares scores.

**Value**

An S4 object of class [FaClassic-class](#) which is a subclass of the virtual class [Fa-class](#).

**Author(s)**

Ying-Ying Zhang (Robert) <[robertzhangyying@qq.com](mailto:robertzhangyying@qq.com)>

**References**

Zhang, Y. Y. (2013), An Object Oriented Solution for Robust Factor Analysis.

**See Also**

[FaClassic-class](#), [FaCov-class](#), [FaRobust-class](#), [Fa-class](#)

**Examples**

```
data("hbk")
hbk.x = hbk[,1:3]

## faClassicPcaReg uses the default method
faClassicPcaReg = FaClassic(x = hbk.x, factors = 2, method = "pca",
scoresMethod = "regression"); faClassicPcaReg
summary(faClassicPcaReg)

## faClassicForPcaReg uses the formula interface
## faClassicForPcaReg = faClassicPcaReg
faClassicForPcaReg = FaClassic(~., data=as.data.frame(hbk.x), factors = 2,
method = "pca", scoresMethod = "regression"); faClassicForPcaReg
summary(faClassicForPcaReg)
```

---

FaClassic-class	Class "FaClassic"
-----------------	-------------------

---

**Description**

Contains the results of a classical Factor Analysis

**Objects from the Class**

Objects can be created by calls of the form `new("FaClassic", ...)`. But the usual way of creating FaClassic objects is a call to the function `FaClassic` which serves as a constructor.

**Slots**

**call:** Object of class "language" an unevaluated function call

**converged:** Object of class "Ulogical" a logical character indicates whether the iterations converged

**loadings:** Object of class "matrix" the matrix of variable loadings

**uniquenesses:** Object of class "vector" the uniquenesses computed

**covariance:** Object of class "matrix" the covariance matrix

**correlation:** Object of class "matrix" the correlation matrix

**usedMatrix:** Object of class "matrix" the used matrix (running matrix)

**criteria:** Object of class "Unumeric". The results of the optimization: the value of the negative log-likelihood and information on the iterations used.

**factors:** Object of class "numeric" the number of factors

dof: Object of class "Unumeric". The number of degrees of freedom of the factor analysis model.

method: Object of class "character". The method: one of "mle", "pca", and "pfa".

scores: Object of class "Umatrix". If requested, a matrix of scores.

scoresMethod: Object of class "character". The scores method: one of "none", "regression", and "Bartlett".

STATISTIC: Object of class "Unumeric". The significance-test statistic, if it can be computed.

PVAL: Object of class "Unumeric". The significance-test P value, if it can be computed.

n.obs: Object of class "Unumeric". The number of observations if available.

center: Object of class "Uvector". The center of the data.

eigenvalues: Object of class "vector" the eigenvalues

cov.control: Object of class "UCovControl". Record the cov control method.

### Extends

Class ["Fa"](#), directly.

### Methods

No methods defined with class "FaClassic" in the signature.

### Author(s)

Ying-Ying Zhang (Robert) <[robertzhangying@qq.com](mailto:robertzhangying@qq.com)>

### References

Zhang, Y. Y. (2013), An Object Oriented Solution for Robust Factor Analysis.

### See Also

[FaClassic-class](#), [FaCov-class](#), [FaRobust-class](#), [Fa-class](#)

### Examples

```
showClass("FaClassic")
```

**Description**

Robust factor analysis are obtained by replacing the classical covariance matrix by a robust covariance estimator. This can be one of the available estimators in `rrcov`, i.e., MCD, OGK, M, S, SDE, or MVE estimator.

**Usage**

```
FaCov(x, ...)
## S3 method for class 'formula'
FaCov(formula, data = NULL, factors = 2, cor = FALSE, method = "mle",
scoresMethod = "none", ...)
## Default S3 method:
FaCov(x, factors = 2, cor = FALSE, cov.control = rrcov::CovControlMcd(),
method = c("mle", "pca", "pfa"),
scoresMethod = c("none", "regression", "Bartlett"), ...)
```

**Arguments**

<code>x</code>	A formula or a numeric matrix or an object that can be coerced to a numeric matrix.
<code>...</code>	Arguments passed to or from other methods.
<code>formula</code>	A formula with no response variable, referring only to numeric variables.
<code>data</code>	An optional data frame (or similar: see <a href="#">model.frame</a> ) containing the variables in the formula.
<code>factors</code>	The number of factors to be fitted.
<code>cor</code>	A logical value indicating whether the calculation should use the covariance matrix ( <code>cor = FALSE</code> ) or the correlation matrix ( <code>cor = TRUE</code> ).
<code>method</code>	The method of factor analysis, one of "mle" (the default), "pca", and "pfa".
<code>scoresMethod</code>	Type of scores to produce, if any. The default is "none", "regression" gives Thompson's scores, "Bartlett" gives Bartlett's weighted least-squares scores.
<code>cov.control</code>	Specifies which covariance estimator to use by providing a <a href="#">CovControl-class</a> object. The default is <a href="#">CovControlMcd-class</a> which will indirectly call <a href="#">CovMcd</a> . If <code>cov.control=NULL</code> is specified, the classical estimates will be used by calling <a href="#">CovClassic</a> .

**Details**

`FaCov`, serving as a constructor for objects of class [FaCov-class](#) is a generic function with "formula" and "default" methods.

**Value**

An S4 object of class [FaCov-class](#) which is a subclass of the virtual class [Fa-class](#).

**Author(s)**

Ying-Ying Zhang (Robert) <robertzhangying@qq.com>

**References**

Zhang, Y. Y. (2013), An Object Oriented Solution for Robust Factor Analysis.

**See Also**

[FaClassic-class](#), [FaCov-class](#), [FaRobust-class](#), [Fa-class](#)

**Examples**

```
data("hbk")
hbk.x = hbk[,1:3]

##
## faCovPcaRegMcd is obtained from FaCov.default
##
faCovPcaRegMcd = FaCov(x = hbk.x, factors = 2, method = "pca",
scoresMethod = "regression", cov.control = rrcov::CovControlMcd()); faCovPcaRegMcd

##
## In fact, it is equivalent to use FaCov.formula
## faCovForPcaRegMcd = faCovPcaRegMcd
##
faCovForPcaRegMcd = FaCov(~., data = as.data.frame(hbk.x),
factors = 2, method = "pca", scoresMethod = "regression",
cov.control = rrcov::CovControlMcd()); faCovForPcaRegMcd
```

---

FaCov-class

*Class "FaCov"*

---

**Description**

Robust FA based on a robust covariance matrix. Robust FA are obtained by replacing the classical covariance matrix by a robust covariance estimator. This can be one of the available in `rrcov` estimators, i.e., MCD, OGK, M, S, SDE, or MVE estimator.

**Objects from the Class**

Objects can be created by calls of the form `new("FaCov", ...)`. But the usual way of creating FaCov objects is a call to the function `FaCov` which serves as a constructor.

**Slots**

**call:** Object of class "Language" an unevaluated function call

**converged:** Object of class "Ulogical" a logical character indicates whether the iterations converged

**loadings:** Object of class "matrix" the matrix of variable loadings

**uniquenesses:** Object of class "vector" the uniquenesses computed

**covariance:** Object of class "matrix" the covariance matrix

**correlation:** Object of class "matrix" the correlation matrix

**usedMatrix:** Object of class "matrix" the used matrix (running matrix)

**criteria:** Object of class "Unumeric". The results of the optimization: the value of the negative log-likelihood and information on the iterations used.

**factors:** Object of class "numeric" the number of factors

**dof:** Object of class "Unumeric". The number of degrees of freedom of the factor analysis model.

**method:** Object of class "character". The method: one of "mle", "pca", and "pfa".

**scores:** Object of class "Umatrix". If requested, a matrix of scores.

**scoresMethod:** Object of class "character". The scores method: one of "none", "regression", and "Bartlett".

**STATISTIC:** Object of class "Unumeric". The significance-test statistic, if it can be computed.

**PVAL:** Object of class "Unumeric". The significance-test P value, if it can be computed.

**n.obs:** Object of class "Unumeric". The number of observations if available.

**center:** Object of class "Uvector". The center of the data.

**eigenvalues:** Object of class "vector" the eigenvalues

**cov.control:** Object of class "UCovControl". Record the cov control method.

**Extends**

Class "[FaRobust](#)", directly. Class "[Fa](#)", by class "[FaRobust](#)", distance 2.

**Methods**

No methods defined with class "FaCov" in the signature.

**Author(s)**

Ying-Ying Zhang (Robert) <[robertzhangying@qq.com](mailto:robertzhangying@qq.com)>

**References**

Zhang, Y. Y. (2013), An Object Oriented Solution for Robust Factor Analysis.

**See Also**

[FaClassic-class](#), [FaCov-class](#), [FaRobust-class](#), [Fa-class](#)

**Examples**

```
showClass("FaCov")
```

---

factorScorePca	<i>Factor Analysis by Principal Component Analysis (PCA)</i>
----------------	--

---

**Description**

Perform principal component factor analysis on a covariance matrix or data matrix.

**Usage**

```
factorScorePca(x, factors = 2, covmat = NULL, cor = FALSE,
rotation = c("varimax", "none"),
scoresMethod = c("none", "regression", "Bartlett"))
```

**Arguments**

x	A numeric matrix or an object that can be coerced to a numeric matrix.
factors	The number of factors to be fitted.
covmat	A covariance matrix, or a covariance list as returned by <code>cov.wt</code> . Of course, correlation matrices are covariance matrices.
cor	A logical value indicating whether the calculation should use the covariance matrix ( <code>cor = FALSE</code> ) or the correlation matrix ( <code>cor = TRUE</code> ).
rotation	character. "none" or "varimax": it will be called with first argument the loadings matrix, and should return a list with component loadings giving the rotated loadings, or just the rotated loadings.
scoresMethod	Type of scores to produce, if any. The default is "none", "regression" gives Thompson's scores, "Bartlett" gives Bartlett's weighted least-squares scores.

**Details**

Other feasible usages are:

```
factorScorePca(factors, covmat)
```

```
factorScorePca(x, factors, rotation, scoresMethod)
```

If `x` is missing, then the following components of the result will be `NULL`: `scores`, `ScoringCoef`, `meanF`, `corF`, and `n.obs`.



**Value**

An object of class "factorScorePca" with components:

call	The matched call.
loadings	A matrix of loadings, one column for each factor. This is of class "loadings" if rotation = "varimax": see <a href="#">loadings</a> for its print method; It is a plain matrix if rotation = "none".
communality	The common variance.
uniquenesses	The uniquenesses/specific variance computed.
covariance	The robust/classical covariance matrix.
correlation	The robust/classical correlation matrix.
usedMatrix	The used matrix (running matrix). It may be the covariance or correlation matrix according to the value of cor.
reducedCorrelation	NULL. The reduced correlation matrix, reducedCorrelation is calculated in factorScorePfa.R.
factors	The argument factors.
method	The method: always "pca".
scores	If requested, a matrix of scores. NULL if x is missing.
scoringCoef	The scoring coefficients. NULL if x is missing.
meanF	The sample mean of the scores. NULL if x is missing.
corF	The sample correlation matrix of the scores. NULL if x is missing.
scoresMethod	The argument scoresMethod.
n.obs	The number of observations if available. NULL if x is missing.
center	The center of the data.
eigenvalues	The eigenvalues of the usedMatrix.

**Author(s)**

Ying-Ying Zhang (Robert) <[robertzhangyying@qq.com](mailto:robertzhangyying@qq.com)>

**References**

Zhang, Y. Y. (2013), An Object Oriented Solution for Robust Factor Analysis.

**See Also**

[factorScorePfa](#), [factanal](#)

**Examples**

```

data(stock611)
R611=cor(stock611[,3:12]); R611

## covmat is a matrix
fsPca1=factorScorePca(factors = 3, covmat = R611); fsPca1

## covmat is a list
covx <- rrcov::Cov(stock611[,3:12])
covmat <- list(cov=rrcov::getCov(covx), center=rrcov::getCenter(covx), n.obs=covx@n.obs)
fsPca2=factorScorePca(factors = 3, covmat = covmat); fsPca2

## fsPca3 contains scores etc.
fsPca3=factorScorePca(x = stock611[,3:12], factors = 2, cor = TRUE, rotation = "varimax",
scoresMethod = "regression"); fsPca3

```

---

factorScorePfa

*Factor Analysis by Principal Factor Analysis (PFA)*


---

**Description**

Perform principal factor factor analysis on a covariance matrix or data matrix.

**Usage**

```

factorScorePfa(x, factors = 2, covmat = NULL, cor = FALSE,
rotation = c("varimax", "none"),
scoresMethod = c("none", "regression", "Bartlett"))

```

**Arguments**

x	A numeric matrix or an object that can be coerced to a numeric matrix.
factors	The number of factors to be fitted.
covmat	A covariance matrix, or a covariance list as returned by <code>cov.wt</code> . Of course, correlation matrices are covariance matrices.
cor	A logical value indicating whether the calculation should use the covariance matrix ( <code>cor = FALSE</code> ) or the correlation matrix ( <code>cor = TRUE</code> ).
rotation	character. "none" or "varimax": it will be called with first argument the loadings matrix, and should return a list with component loadings giving the rotated loadings, or just the rotated loadings.
scoresMethod	Type of scores to produce, if any. The default is "none", "regression" gives Thompson's scores, "Bartlett" gives Bartlett's weighted least-squares scores.

**Details**

Other feasible usages are:

```
factorScorePfa(factors, covmat)
```

```
factorScorePfa(x, factors, rotation, scoresMethod)
```

If `x` is missing, then the following components of the result will be NULL: `scores`, `ScoringCoef`, `meanF`, `corF`, and `n.obs`.

**Value**

An object of class "factorScorePfa" with components:

<code>call</code>	The matched call.
<code>loadings</code>	A matrix of loadings, one column for each factor. This is of class "loadings" if <code>rotation = "varimax"</code> : see <a href="#">loadings</a> for its print method; It is a plain matrix if <code>rotation = "none"</code> .
<code>communality</code>	The common variance.
<code>uniquenesses</code>	The uniquenesses/specific variance computed.
<code>covariance</code>	The robust/classical covariance matrix.
<code>correlation</code>	The robust/classical correlation matrix.
<code>usedMatrix</code>	The used matrix (running matrix). It may be the covariance or correlation matrix according to the value of <code>cor</code> .
<code>reducedCorrelation</code>	The last reduced correlation matrix.
<code>factors</code>	The argument factors.
<code>method</code>	The method: always "pfa".
<code>scores</code>	If requested, a matrix of scores. NULL if <code>x</code> is missing.
<code>scoringCoef</code>	The scoring coefficients. NULL if <code>x</code> is missing.
<code>meanF</code>	The sample mean of the scores. NULL if <code>x</code> is missing.
<code>corF</code>	The sample correlation matrix of the scores. NULL if <code>x</code> is missing.
<code>scoresMethod</code>	The argument <code>scoresMethod</code> .
<code>n.obs</code>	The number of observations if available. NULL if <code>x</code> is missing.
<code>center</code>	The center of the data.
<code>eigenvalues</code>	The eigenvalues of the <code>usedMatrix</code> .

**Author(s)**

Ying-Ying Zhang (Robert) <[robertzhangyiyi@qq.com](mailto:robertzhangyiyi@qq.com)>

**References**

Zhang, Y. Y. (2013), An Object Oriented Solution for Robust Factor Analysis.

**See Also**

[factorScorePca](#), [factanal](#)

**Examples**

```
data(stock611)
R611 = cor(stock611[,3:12]); R611

## covmat is a matrix
fsPfa1 = factorScorePfa(factors = 3, covmat = R611); fsPfa1

## covmat is a list
covx = rrcov::Cov(stock611[,3:12])
covmat = list(cov = rrcov::getCov(covx), center = rrcov::getCenter(covx), n.obs = covx@n.obs)
fsPfa2 = factorScorePfa(factors = 3, cor = TRUE, covmat = covmat); fsPfa2

## fsPfa3 contains scores etc.
fsPfa3 = factorScorePfa(x = stock611[,3:12], factors = 2,
cor = TRUE, rotation = "varimax", scoresMethod = "regression"); fsPfa3
```

---

FaRobust-class	<i>Class "FaRobust"</i>
----------------	-------------------------

---

**Description**

Class "FaRobust" is a virtual base class for all robust FA classes. Currently the only available robust FA class is "FaCov". The class "FaRobust" serves as a base class for deriving all other classes representing the results of the robust Factor Analysis methods.

**Objects from the Class**

A virtual Class: No objects may be created from it.

**Slots**

**call:** Object of class "language" an unevaluated function call

**converged:** Object of class "Ulogical" a logical character indicates whether the iterations converged

**loadings:** Object of class "matrix" the matrix of variable loadings

**uniquenesses:** Object of class "vector" the uniquenesses computed

**covariance:** Object of class "matrix" the covariance matrix

**correlation:** Object of class "matrix" the correlation matrix

**usedMatrix:** Object of class "matrix" the used matrix (running matrix)

**criteria:** Object of class "Unumeric". The results of the optimization: the value of the negative log-likelihood and information on the iterations used.

**factors:** Object of class "numeric" the number of factors

**dof:** Object of class "Unumeric". The number of degrees of freedom of the factor analysis model.

**method:** Object of class "character". The method: one of "mle", "pca", and "pfa".

**scores:** Object of class "Umatrix". If requested, a matrix of scores.

**scoresMethod:** Object of class "character". The scores method: one of "none", "regression", and "Bartlett".

**STATISTIC:** Object of class "Unumeric". The significance-test statistic, if it can be computed.

**PVAL:** Object of class "Unumeric". The significance-test P value, if it can be computed.

**n.obs:** Object of class "Unumeric". The number of observations if available.

**center:** Object of class "Uvector". The center of the data.

**eigenvalues:** Object of class "vector" the eigenvalues

**cov.control:** Object of class "UCovControl". Record the cov control method.

### Extends

Class "Fa", directly.

### Methods

No methods defined with class "FaRobust" in the signature.

### Author(s)

Ying-Ying Zhang (Robert) <robertzhangyying@qq.com>

### References

Zhang, Y. Y. (2013), An Object Oriented Solution for Robust Factor Analysis.

### See Also

[FaClassic-class](#), [FaCov-class](#), [FaRobust-class](#), [Fa-class](#)

### Examples

```
showClass("FaRobust")
```

---

`fsOrder`*Compute the Ordered Factor Scores*

---

**Description**

Compute the ordered factor scores according to the first/second/third... column of the original factor scores.

**Usage**

```
fsOrder(factorScores)
```

**Arguments**

`factorScores` The original factor scores.

**Value**

A list with `m` (the number of factors) components:

```
[[1]]      The ordered factor scores with a decreasing first column.  
[[2]]      The ordered factor scores with a decreasing second column.  
...  
[[m]]      The ordered factor scores with a decreasing m-th column.
```

**Author(s)**

Ying-Ying Zhang (Robert) <robertzhangying@qq.com>

**References**

Zhang, Y. Y. (2013), An Object Oriented Solution for Robust Factor Analysis.

**See Also**

[order](#)

**Examples**

```
data(stock611)  
R611=cor(stock611[,3:12]); R611  
  
## FS.pca contains scores etc.  
fsPca=factorScorePca(x = stock611[,3:12], factors = 2, cor = TRUE,  
rotation = "varimax", scoresMethod = "regression"); fsPca  
  
orderedFS=fsOrder(fsPca$scores); orderedFS
```

---

getCenter-methods      *Access Center slot*

---

**Description**

Accessor method to the Center slot of an object of class "Fa" and its subclasses.

**Methods**

signature(obj = "Fa") Accessor method to the Center slot of an object of class "Fa" and its subclasses

---

getEigenvalues-methods  
*Access Eigenvalues slot*

---

**Description**

Accessor method to the Eigenvalues slot of an object of class "Fa" and its subclasses.

**Methods**

signature(obj = "Fa") Accessor method to the Eigenvalues slot of an object of class "Fa" and its subclasses

---

getFa-methods      *Access slots of "Fa"*

---

**Description**

Accessor method to some slots of an object of class "Fa" and its subclasses. Return a list of class "fa".

**Methods**

signature(obj = "Fa") Accessor method to some slots of an object of class "Fa" and its subclasses. Return a list of class "fa".

---

*getLoadings-methods*      *Access Loadings slot*

---

**Description**

Accessor method to the Loadings slot of an object of class "Fa" and its subclasses.

**Methods**

signature(obj = "Fa") Accessor method to the Loadings slot of an object of class "Fa" and its subclasses

---

*getQuan-methods*      *Access n.obs slot*

---

**Description**

Accessor method to the n.obs slot of an object of class "Fa" and its subclasses.

**Methods**

signature(obj = "Fa") Accessor method to the n.obs slot of an object of class "Fa" and its subclasses

---

*getScores-methods*      *Access Scores slot*

---

**Description**

Accessor method to the Scores slot of an object of class "Fa" and its subclasses.

**Methods**

signature(obj = "Fa") Accessor method to the Scores slot of an object of class "Fa" and its subclasses



---

getSdev-methods	<i>Access Standard Deviation slot</i>
-----------------	---------------------------------------

---

**Description**

Accessor method to the Standard Deviation slot of an object of class "Fa" and its subclasses.

**Methods**

signature(obj = "Fa") Accessor method to the Standard Deviation slot of an object of class "Fa" and its subclasses

---

myFaPrint	<i>Show/Print/Display an Object</i>
-----------	-------------------------------------

---

**Description**

Show/print/display an object, including the Call, Standard deviations, Loadings, and Rotated variables (if print.x = TRUE).

**Usage**

```
myFaPrint(object, print.x=FALSE)
```

**Arguments**

object	an object of class "Fa" or of a class derived from "Fa".
print.x	Logical. If print.x = TRUE, then print the rotated variables (scores).

**Value**

An invisible argument object.

**Author(s)**

Ying-Ying Zhang (Robert) <robertzhangyying@qq.com>

**References**

Zhang, Y. Y. (2013), An Object Oriented Solution for Robust Factor Analysis.

**See Also**

[Fa-class](#)

**Examples**

```

data("hbk")
hbk.x = hbk[,1:3]

faCovPcaRegMcd = FaCov(x = hbk.x, factors = 2, method = "pca",
scoresMethod = "regression", cov.control = rrcov::CovControlMcd())

## you can use either object or print(object) or myFaPrint(object)
## since faCovPcaRegMcd is an object of class "Fa"

faCovPcaRegMcd
print(faCovPcaRegMcd)
myFaPrint(faCovPcaRegMcd)

```

---

myplotDD

*Distance-Distance Plot*


---

**Description**

"myplotDD" is a revised version of ".myddplot" in "plot-utils.R" in the package "rrcov". In "myplotDD", id.n and ind are printed out.

**Usage**

```
myplotDD(x, cutoff, id.n)
```

**Arguments**

x	An S4 object of class "CovRobust".
cutoff	The cutoff value used. If missing, <code>cutoff &lt;- sqrt(qchisq(0.975, p))</code> by default.
id.n	Number of observations to identify by a label. If not supplied, the number of observations with robust distance larger than cutoff is used.

**Details**

Distance-Distance Plot: Plot the vector  $y=rd$  (robust distances) against  $x=md$  (mahalanobis distances). Identify by a label the id.n observations with largest rd. If id.n is not supplied, calculate it as the number of observations larger than cutoff. Use cutoff to draw a horizontal and a vertical line. Draw also a dotted line with a slope 1.

"myplotDD(x)" is equivalent to "plot(x, which="dd")". which: indicate what kind of plot. If which = "dd", then a distance-distance Plot.

**Value**

A distance-distance plot is shown. Return a list with components:

cutoff	The cutoff value used. If missing, <code>cutoff &lt;- sqrt(qchisq(0.975, p))</code> by default.
id.n	Number of observations to identify by a label. If not supplied, the number of observations with robust distance larger than <code>cutoff</code> is used.
sort.y	A list containing the sorted values of <code>y</code> (the robust distance)
ind	The indices of the largest <code>id.n</code> observations whose robust distances are larger than <code>cutoff</code> .

**Author(s)**

Ying-Ying Zhang (Robert) <robertzhangyying@qq.com>

**References**

Zhang, Y. Y. (2013), An Object Oriented Solution for Robust Factor Analysis.

**See Also**

[plot](#), [qchisq](#), [CovClassic](#), [getDistance](#)

**Examples**

```
data(stock611)
covMcd=CovRobust(x=scale(stock611[,3:12]), control="mcd"); covMcd

## "myplotDD" shows id.n and ind.
## Note: id.n and ind change each time due to covMcd changes each time!
## However, the ind of largest robust distances do not change.
result = myplotDD(x=covMcd); result

## "myplotDD" is equivalent to "plot(x=covMcd, which="dd")".
plot(x=covMcd, which="dd")
```

**Description**

Plot an object of class "Fa". If `which = "factorScore"`, then a scatterplot of the factor scores is produced; if `which = "screeplot"`, shows the eigenvalues and is helpful to select the number of factors.

**Usage**

```
## S4 method for signature 'Fa'  
plot(x, which=c("factorScore", "screeplot"), choices=1:2)
```

**Arguments**

x	an object of class "Fa" or of a class derived from "Fa"
which	indicate what kind of plot. If which = "factorScore", then a scatterplot of the factor scores is produced; if which = "screeplot", shows the eigenvalues and is helpful to select the number of factors.
choices	an integer vector indicate which columns of the factor scores to plot

**Details**

The feasible usages are: `plot(x, which="factorScore", choices=1:2)` `plot(x, which="screeplot")`

**Methods**

`signature(x = "Fa", y = "missing")` generic functions - see `plot`

**Author(s)**

Ying-Ying Zhang (Robert) <robertzhangyying@qq.com>

**References**

Zhang, Y. Y. (2013), An Object Oriented Solution for Robust Factor Analysis.

**Examples**

```
data("hbk")  
hbk.x = hbk[,1:3]  
  
faClassicPcaReg = FaClassic(x = hbk.x, factors = 2, method = "pca",  
scoresMethod = "regression"); faClassicPcaReg  
summary(faClassicPcaReg)  
  
plot(faClassicPcaReg, which = "factorScore", choices = 1:2)  
plot(faClassicPcaReg, which = "screeplot")
```

---

predict-methods	<i>Calculates prediction</i>
-----------------	------------------------------

---

**Description**

Calculates prediction using the results in object. The newdata argument is an optional data frame or matrix in which to look for variables with which to predict. If newdata is omitted, the scores are used.

**Usage**

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

**Arguments**

object	an object of class "Fa" or of a class derived from "Fa"
...	additional arguments, e.g., newdata: an optional data frame or matrix in which to look for variables with which to predict. If newdata is not missing, newdata should be scaled before "predict".

**Methods**

signature(object = "Fa") generic functions - see print, summary, predict, plot, getCenter, getEigenvalues, getFa, getLoadings, getQuan, getScores, getSdev

**Author(s)**

Ying-Ying Zhang (Robert) <robertzhangying@qq.com>

**References**

Zhang, Y. Y. (2013), An Object Oriented Solution for Robust Factor Analysis.

**Examples**

```
data("hbk")
hbk.x = hbk[,1:3]

faCovPcaRegMcd = FaCov(x = hbk.x, factors = 2, method = "pca",
scoresMethod = "regression", cov.control = rrcov::CovControlMcd()); faCovPcaRegMcd

## If missing newdata, the scores are used
predict(faCovPcaRegMcd)

##
## If not missing newdata, newdata should be scaled first.
##
newdata = hbk.x[1, ]
cor = FALSE # the default
```

```
newdata = {
  if (cor == TRUE)
    # standardized transformation
    scale(newdata, center = faCovPcaRegMcd@center,
          scale = sqrt(diag(faCovPcaRegMcd@covariance)))
  else # cor == FALSE
    # centralized transformation
    scale(newdata, center = faCovPcaRegMcd@center, scale = FALSE)
}

##
## Now, prediction = predict(faCovPcaRegMcd)[1,] = faCovPcaRegMcd@scores[1,]
##
prediction = predict(faCovPcaRegMcd, newdata = newdata)
prediction
```

---

print-methods

*Print/Display an Object*

---

## Description

Print/display an object, including the Call, Standard deviations, Loadings.

## Usage

```
print(x, ...)
```

## Arguments

**x** an object of class "Fa" or "SummaryFa" or of a class derived from "Fa" or of class or "SummaryFa".

**...** additional arguments, e.g., print.x=TRUE

## Value

An invisible argument x.

## Methods

**x = "Fa"** generic functions - see print, summary, predict, plot, getCenter, getEigenvalues, getFa, getLoadings, getQuan, getScores, getSdev

**x = "SummaryFa"** generic functions - see print, summary, predict, plot, getCenter, getEigenvalues, getFa, getLoadings, getQuan, getScores, getSdev

## Author(s)

Ying-Ying Zhang (Robert) <robertzhangyying@qq.com>

**References**

Zhang, Y. Y. (2013), An Object Oriented Solution for Robust Factor Analysis.

**See Also**

[Fa-class](#), [SummaryFa-class](#)

**Examples**

```
data("hbk")
hbk.x = hbk[,1:3]

faCovPcaRegMcd = FaCov(x = hbk.x, factors = 2, method = "pca",
scoresMethod = "regression", cov.control = rrcov::CovControlMcd())

## you can use either object or print(object) or myFaPrint(object)
## since faCovPcaRegMcd is an object of class "Fa"

faCovPcaRegMcd
print(faCovPcaRegMcd)
myFaPrint(faCovPcaRegMcd)
```

---

stock611

*The Stocks Data - Year 2001*


---

**Description**

This data set consists of 611 observations with 12 variables.

**Usage**

```
data(stock611)
```

**Format**

A data frame with 611 observations on the following 12 variables.

code a numeric vector

name a numeric vector: the Chinese stocks name is replaced by integer, it can be found by its code.

x1 a numeric vector: main business income (China Yuan)

x2 a numeric vector: main business profit (China Yuan)

x3 a numeric vector: total profit (China Yuan)

x4 a numeric vector: net profit (China Yuan)

x5 a numeric vector: earnings per share (EPS) (China Yuan)

x6 a numeric vector: net assets per share (China Yuan)

- x7 a numeric vector: net return on assets (%)
- x8 a numeric vector: total return on assets (%)
- x9 a numeric vector: total assets (China Yuan)
- x10 a numeric vector: equity

### Details

The data set is from Chinese stock market in the year 2001. It was used in Wang X. M. (2009) to illustrate the factor analysis methods.

### Source

Wang X. M. (2009) *Applied Multivariate Analysis*. Third edition. ShangHai University of Finance & Economics Press. (This is a Chinese book)

Note: In Wang X. M.'s homepage, he provided a link to download materials related to his book (including the data set stock611): <http://bb.shufe.edu.cn/bbcswebdav/institution/>

### Examples

```
data(stock611)
str(stock611)
plot(stock611)
```

---

summary-methods

*Summary an Object*

---

### Description

Produce result summaries of an object of class "Fa".

### Usage

```
summary(object, ...)
```

### Arguments

object            an object of class "Fa" or of a class derived from "Fa".  
...                additional arguments, e.g., print.x=TRUE.

### Methods

signature(object = "Fa") Summary an object of class "Fa".

### Author(s)

Ying-Ying Zhang (Robert) <[robertzhangyying@qq.com](mailto:robertzhangyying@qq.com)>



**References**

Zhang, Y. Y. (2013), An Object Oriented Solution for Robust Factor Analysis.

**Examples**

```
data("hbk")
hbk.x = hbk[,1:3]

faCovPcaRegMcd = FaCov(x = hbk.x, factors = 2, method = "pca",
scoresMethod = "regression", cov.control = rrcov::CovControlMcd()); faCovPcaRegMcd

faCovPcaRegMcd
summary(faCovPcaRegMcd)
```

---

SummaryFa-class	Class "SummaryFa"
-----------------	-------------------

---

**Description**

Summary of "Fa" objects. The "Fa" object plus some additional summary information.

**Objects from the Class**

Objects can be created by calls of the form `new("SummaryFa", ...)`. But most often by invoking 'summary' on an "Fa" object. They contain values meant for printing by 'show'.

**Slots**

**faobj:** Object of class "Fa"  
**importance:** Object of class "matrix". Matrix with additional information: importance of components.

**Methods**

**show** signature(object = "SummaryFa"): display the object

**Author(s)**

Ying-Ying Zhang (Robert) <robertzhangying@qq.com>

**References**

Zhang, Y. Y. (2013), An Object Oriented Solution for Robust Factor Analysis.

**See Also**

[Fa-class](#)

**Examples**

```
showClass("SummaryFa")
```

---

Ulogical-class	<i>Class "Ulogical"</i>
----------------	-------------------------

---

**Description**

Define class unions for optional slots, e.g., for definition of slots which will be computed on demand.

**Objects from the Class**

A virtual Class: No objects may be created from it.

**Methods**

No methods defined with class "Ulogical" in the signature.

**Author(s)**

Ying-Ying Zhang (Robert) <robertzhangyying@qq.com>

**References**

Zhang, Y. Y. (2013), An Object Oriented Solution for Robust Factor Analysis.

---

Unumeric-class	<i>Class "Unumeric"</i>
----------------	-------------------------

---

**Description**

Define class unions for optional slots, e.g., for definition of slots which will be computed on demand.

**Objects from the Class**

A virtual Class: No objects may be created from it.

**Methods**

No methods defined with class "Unumeric" in the signature.

**Author(s)**

Ying-Ying Zhang (Robert) <robertzhangyying@qq.com>

**References**

Zhang, Y. Y. (2013), An Object Oriented Solution for Robust Factor Analysis.

# Index

## \* classes

- Fa-class, [7](#)
- FaClassic-class, [11](#)
- FaCov-class, [14](#)
- FaRobust-class, [20](#)
- SummaryFa-class, [33](#)
- Ulogical-class, [34](#)
- Unumeric-class, [34](#)

## \* datasets

- stock611, [31](#)

## \* methods

- getCenter-methods, [23](#)
- getEigenvalues-methods, [23](#)
- getFa-methods, [23](#)
- getLoadings-methods, [24](#)
- getQuan-methods, [24](#)
- getScores-methods, [24](#)
- getSdev-methods, [25](#)
- plot-methods, [27](#)
- predict-methods, [29](#)
- print-methods, [30](#)
- summary-methods, [32](#)

## \* package

- robustfa-package, [3](#)

## \* robust

- compute\_cov\_cor, [5](#)
- computeScores, [4](#)
- detail, [6](#)
- FaClassic, [10](#)
- FaCov, [13](#)
- factorScorePca, [16](#)
- factorScorePfa, [18](#)
- fsOrder, [22](#)
- myFaPrint, [25](#)
- myplotDD, [26](#)

attributes, [7](#)

class, [7](#)

compute\_cov\_cor, [5](#)

computeScores, [4](#)

cov.wt, [16](#), [18](#)

CovClassic, [13](#), [27](#)

CovMcd, [13](#)

detail, [6](#)

Fa, [12](#), [15](#), [21](#)

Fa-class, [7](#)

FaClassic, [3](#), [10](#)

FaClassic-class, [11](#)

FaCov, [3](#), [13](#)

FaCov-class, [14](#)

factanal, [17](#), [20](#)

factorScorePca, [3](#), [16](#), [20](#)

factorScorePfa, [3](#), [17](#), [18](#)

FaRobust, [15](#)

FaRobust-class, [20](#)

fsOrder, [22](#)

getCenter, Fa-method  
(getCenter-methods), [23](#)

getCenter-methods, [23](#)

getDistance, [27](#)

getEigenvalues, Fa-method  
(getEigenvalues-methods), [23](#)

getEigenvalues-methods, [23](#)

getFa (getFa-methods), [23](#)

getFa, Fa-method (getFa-methods), [23](#)

getFa-methods, [23](#)

getLoadings, Fa-method  
(getLoadings-methods), [24](#)

getLoadings-methods, [24](#)

getQuan, Fa-method (getQuan-methods), [24](#)

getQuan-methods, [24](#)

getScores, Fa-method  
(getScores-methods), [24](#)

getScores-methods, [24](#)

getSdev, Fa-method (getSdev-methods), [25](#)

getSdev-methods, [25](#)

is.object, [7](#)  
isS4, [7](#)

loadings, [17](#), [19](#)

model.frame, [10](#), [13](#)  
myFaPrint, [25](#)  
myplotDD, [26](#)

order, [22](#)

plot, [27](#)  
plot(myplotDD), [26](#)  
plot, Fa-method (plot-methods),  
[27](#)  
plot, Fa-method (plot-methods), [27](#)  
plot-methods, [27](#)  
predict (predict-methods), [29](#)  
predict, Fa-method (predict-methods), [29](#)  
predict-methods, [29](#)  
print (print-methods), [30](#)  
print, Fa-method (print-methods), [30](#)  
print, SummaryFa-method (print-methods),  
[30](#)  
print-methods, [30](#)

qchisq, [27](#)

robustfa (robustfa-package), [3](#)  
robustfa-package, [3](#)

stock611, [31](#)  
summary (summary-methods), [32](#)  
summary, Fa-method (summary-methods), [32](#)  
summary-methods, [32](#)  
SummaryFa-class, [33](#)

Ulogical-class, [34](#)  
Unumeric-class, [34](#)