

# Package ‘pcaPP’

March 9, 2010

**Version** 1.8

**Date** 2009-07-02

**Title** Robust PCA by Projection Pursuit

**Author** Peter Filzmoser Heinrich Fritz Klaudius Kalcher

**Maintainer** Peter Filzmoser <P.Filzmoser@tuwien.ac.at>

**Depends** mvtnorm

**Description** Robust PCA by Projection Pursuit

**License** GPL (>= 3)

**URL** <http://www.statistik.tuwien.ac.at/public/filz/>

**Repository** CRAN

**Date/Publication** 2010-03-09 11:37:53

## R topics documented:

covPC . . . . .	2
covPCA . . . . .	3
l1median . . . . .	4
l1median_NLM . . . . .	5
PCAGrid . . . . .	6
PCAprj . . . . .	8
PCdiagplot . . . . .	11
plotcov . . . . .	12
qn . . . . .	14
ScaleAdv . . . . .	15
<b>Index</b>	<b>17</b>

---

`covPC`*Covariance Matrix Estimation from princomp Object*

---

**Description**

computes the covariance matrix from a princomp object. The number of components k can be given as input.

**Usage**

```
covPC(x, k, method)
```

**Arguments**

x	an object of class princomp.
k	number of PCs to use for covariance estimation (optional).
method	method how the PCs have been estimated (optional).

**Details**

There are several possibilities to estimate the principal components (PCs) from an input data matrix, including the functions [PCAproj](#) and [PCAgrid](#). This function uses the estimated PCs to reconstruct the covariance matrix. Not all PCs have to be used, the number k of PCs (first k PCs) can be given as input to the function.

**Value**

cov	the estimated covariance matrix
center	the center of the data, as provided from the princomp object.
method	a string describing the method that was used to calculate the PCs.

**Author(s)**

Heinrich Fritz, Peter Filzmoser <<P.Filzmoser@tuwien.ac.at>>

**References**

C. Croux, P. Filzmoser, M. Oliveira, (2007). Algorithms for Projection-Pursuit Robust Principal Component Analysis, *Chemometrics and Intelligent Laboratory Systems*, Vol. 87, pp. 218-225.

**See Also**

[PCAgrid](#), [PCAproj](#), [princomp](#)

**Examples**

```
# multivariate data with outliers
library(mvtnorm)
x <- rbind(rmvnorm(200, rep(0, 6), diag(c(5, rep(1,5)))),
           rmvnorm( 15, c(0, rep(20, 5)), diag(rep(1, 6))))
pc <- princomp(x)
covPC(pc, k=2)
```

covPCA

*Robust Covariance Matrix Estimation***Description**

computes the robust covariance matrix using the `PCAgrid` and `PCAproj` functions.

**Usage**

```
covPCAproj(x, control)
covPCAgrid(x, control)
```

**Arguments**

`x` a numeric matrix or data frame which provides the data.

`control` a list whose elements must be the same as (or a subset of) the parameters of the appropriate PCA function (`PCAgrid` or `PCAproj`).

**Details**

The functions `covPCAproj` and `covPCAgrid` use the functions `PCAproj` and `PCAgrid` respectively to estimate the covariance matrix of the data matrix `x`.

**Value**

`cov` the actual covariance matrix estimated from `x`

`center` the center of the data `x` that was subtracted from them before the PCA algorithms were run.

`method` a string describing the method that was used to calculate the covariance matrix estimation

**Author(s)**

Heinrich Fritz, Peter Filzmoser <<P.Filzmoser@tuwien.ac.at>>

**References**

C. Croux, P. Filzmoser, M. Oliveira, (2007). Algorithms for Projection-Pursuit Robust Principal Component Analysis, *Chemometrics and Intelligent Laboratory Systems*, Vol. 87, pp. 218-225.

**See Also**

[PCAGrid](#), [ScaleAdv](#), [princomp](#)

**Examples**

```
# multivariate data with outliers
library(mvtnorm)
x <- rbind(rmvnorm(200, rep(0, 6), diag(c(5, rep(1,5)))),
          rmvnorm( 15, c(0, rep(20, 5)), diag(rep(1, 6))))
covPCAproj(x)
# compare with classical covariance matrix:
cov(x)
```

---

l1median

*Multivariate L1 Median*

---

**Description**

Computes the multivariate L1 median (also called spatial median) of a data matrix.

**Usage**

```
l1median(X, MaxStep = 200, ItTol = 10^-8)
```

**Arguments**

X	a matrix containing the values whose multivariate L1 median is to be computed.
MaxStep	Maximal number of step-halvings.
ItTol	Tolerance for convergence of the algorithm.

**Value**

returns the vector of the coordinates of the L1 median.

**Author(s)**

Heinrich Fritz, Peter Filzmoser <<P.Filzmoser@tuwien.ac.at>>

**References**

C. Croux, P. Filzmoser, M. Oliveira, (2007). Algorithms for Projection-Pursuit Robust Principal Component Analysis, *Chemometrics and Intelligent Laboratory Systems*, Vol. 87, pp. 218-225.

**See Also**

[median](#)

**Examples**

```

l1median(rnorm(100)) # this returns the median of the sample

# multivariate data with outliers
library(mvtnorm)
x <- rbind(rmvnorm(200, rep(0, 4), diag(c(1, 1, 2, 2))),
           rmvnorm( 50, rep(3, 4), diag(rep(2, 4))))
l1median(x)
# compare with coordinate-wise median:
apply(x,2,median)

```

---

l1median_NLM	<i>Multivariate L1 Median</i>
--------------	-------------------------------

---

**Description**

Computes the multivariate L1 median (also called spatial median) of a data matrix  $X$ .

**Usage**

```

l1median_NM (X, maxit = 200, tol = 10^-8, trace = 0, m.init = apply(X, 2, median),
l1median_CG (X, maxit = 200, tol = 10^-8, trace = 0, m.init = apply(X, 2, median),
l1median_BFGS (X, maxit = 200, tol = 10^-8, trace = 0, m.init = apply(X, 2, median),

l1median_NLM (X, maxit = 200, tol = 10^-8, trace = 0, m.init = apply(X, 2, median),

```

**Arguments**

<code>X</code>	a matrix of dimension $n \times p$ .
<code>maxit</code>	The maximum number of iterations to be performed.
<code>tol</code>	The convergence tolerance.
<code>trace</code>	The tracing level. Set <code>trace &gt; 0</code> to retrieve additional information on the single iterations.
<code>m.init</code>	A vector of length $p$ containing the initial value of the L1-median.
<code>pscale</code>	A vector of length $p$ containing the variables scale to be used.
<code>REPORT</code>	A parameter passed to <code>optim</code> .

**Details**

The L1-median is computed using the built-in functions `optim` and `nlm`. These functions are a transcript of the `L1median` method of package `robustX` (available at <ftp://stat.ethz.ch/U/maechler/R/>), porting as much code as possible into C++.

**Value**

par	A vector of length $p$ containing the L1-median.
value	The value of the objective function $\ X - \text{llmedian}\ $ which is minimized for finding the L1-median.
code	The return code of the optimization algorithm. See <a href="#">optim</a> and <a href="#">nlm</a> for further information.
iterations	The number of iterations performed.
iterations_gr	When using a gradient function this value holds the number of times the gradient had to be computed.

**Author(s)**

Heinrich Fritz, Peter Filzmoser <<P.Filzmoser@tuwien.ac.at>>

**See Also**

[median](#)

**Examples**

```
# multivariate data with outliers
library(mvtnorm)
x <- rbind(rmvnorm(200, rep(0, 4), diag(c(1, 1, 2, 2))),
          rmvnorm( 50, rep(3, 4), diag(rep(2, 4))))

llmedian_NM (x)$par
llmedian_CG (x)$par
llmedian_BFGS (x)$par
llmedian_NLM (x)$par

# compare with coordinate-wise median:
apply(x,2,median)
```

---

PCAgrid

*Robust Principal Components using the Grid search algorithm*

---

**Description**

Computes a desired number of (robust) principal components using the grid search algorithm in the plane. The global optimum of the objective function is searched in planes, not in the  $p$ -dimensional space, using regular grids in these planes.

**Usage**

```
PCAgrid(x, k = 2, method = c("mad", "sd", "qn"), maxiter = 10, splitcircle = 10,
scores = TRUE, anglehalving = TRUE, fact2dim = 10, scale = NULL, center = llmedian,
```

**Arguments**

<code>x</code>	a numeric matrix or data frame which provides the data for the principal components analysis.
<code>k</code>	desired number of components to compute
<code>method</code>	scale estimator used to detect the direction with the largest variance. Possible values are "sd", "mad" and "qn", the latter can be called "Qn" too. "mad" is the default value.
<code>maxiter</code>	maximum number of iterations.
<code>splitcircle</code>	the number of directions in which the algorithm should search for the largest variance. The direction with the largest variance is searched for in the directions defined by a number of equally spaced points on the unit circle. This argument determines, how many such points are used to split the unit circle.
<code>scores</code>	a logical value indicating whether the scores of the principal component should be calculated.
<code>anglehalving</code>	boolean stating whether angle halving is to be used or not. Angle halving will usually improve the solution quite a lot.
<code>fact2dim</code>	an integer that is multiplied to <code>splitcircle</code> if <code>x</code> is only two-dimensional. In higher dimensions, fewer search directions are needed to allow for faster computation. In two dimensions, more search directions are required to grant higher precision. <code>fact2dim</code> is used to take account of this.
<code>scale</code>	this argument indicates how the data is to be rescaled. It can be a function like <code>sd</code> or <code>mad</code> or a vector of length <code>ncol(x)</code> containing the scale value of each column.
<code>center</code>	this argument indicates how the data is to be centered. It can be a function like <code>mean</code> or <code>median</code> or a vector of length <code>ncol(x)</code> containing the center value of each column.
<code>control</code>	a list whose elements must be the same as (or a subset of) the parameters above. If the control object is supplied, the parameters from it will be used and any other given parameters are overridden.

**Details**

Angle halving is an extension of the original algorithm. In the original algorithm, the search directions are determined by a number of points on the unit circle in the interval  $[-\pi/2 ; \pi/2)$ . Angle halving means this angle is halved in each iteration, eg. for the first approximation, the above mentioned angle is used, for the second approximation, the angle is halved to  $[-\pi/4 ; \pi/4)$  and so on. This usually gives better results with less iterations needed.

Similar to the function `princomp`, there is a `print` method for these objects that prints the results in a nice format and the `plot` method produces a scree plot (`screeplot`). There is also a `biplot` method.

**Value**

The function returns an object of class "princomp", i.e. a list similar to the output of the function `princomp`.

sdev	the (robust) standard deviations of the principal components.
loadings	the matrix of variable loadings (i.e., a matrix whose columns contain the eigenvectors). This is of class "loadings": see <a href="#">loadings</a> for its <code>print</code> method.
center	the means that were subtracted.
scale	the scalings applied to each variable.
n.obs	the number of observations.
scores	if <code>scores = TRUE</code> , the scores of the supplied data on the principal components.
call	the matched call.

**Author(s)**

Heinrich Fritz, Peter Filzmoser <<P.Filzmoser@tuwien.ac.at>>

**References**

C. Croux, P. Filzmoser, M. Oliveira, (2007). Algorithms for Projection-Pursuit Robust Principal Component Analysis, *Chemometrics and Intelligent Laboratory Systems*, Vol. 87, pp. 218-225.

**See Also**

[PCAproj](#), [ScaleAdv](#), [princomp](#)

**Examples**

```
# multivariate data with outliers
library(mvtnorm)
x <- rbind(rmvnorm(200, rep(0, 6), diag(c(5, rep(1,5)))),
           rmvnorm( 15, c(0, rep(20, 5)), diag(rep(1, 6))))
# Here we calculate the principal components with PCAgrid
pc <- PCAgrid(x)
# we could draw a biplot too:
biplot(pc)
# now we want to compare the results with the non-robust principal components
pc <- princomp(x)
# again, a biplot for comparison:
biplot(pc)
```

---

PCAproj

*Robust Principal Components using the algorithm of Croux and Ruiz-Gazen (2005)*

---

**Description**

Computes a desired number of (robust) principal components using the algorithm of Croux and Ruiz-Gazen (JMVA, 2005).

**Usage**

```
PCAproj(x, k = 2, method = c("mad", "sd", "qn"), CalcMethod = c("eachobs",
"lincomb", "sphere"), nmax = 1000, update = TRUE, scores = TRUE, maxit = 5,
maxhalf = 5, scale = NULL, center = llmedian, control)
```

**Arguments**

<code>x</code>	a numeric matrix or data frame which provides the data for the principal components analysis.
<code>k</code>	desired number of components to compute
<code>method</code>	scale estimator used to detect the direction with the largest variance. Possible values are "sd", "mad" and "qn", the latter can be called "Qn" too. "mad" is the default value.
<code>CalcMethod</code>	the variant of the algorithm to be used. Possible values are "eachobs", "lincomb" and "sphere", with "eachobs" being the default.
<code>nmax</code>	maximum number of directions to search in each step (only when using "sphere" or "lincomb" as the <code>CalcMethod</code> ).
<code>update</code>	a logical value indicating whether an update algorithm should be used.
<code>scores</code>	a logical value indicating whether the scores of the principal component should be calculated.
<code>maxit</code>	maximum number of iterations.
<code>maxhalf</code>	maximum number of steps for angle halving.
<code>scale</code>	this argument indicates how the data is to be rescaled. It can be a function like <code>sd</code> or <code>mad</code> or a vector of length <code>ncol(x)</code> containing the scale value of each column.
<code>center</code>	this argument indicates how the data is to be centered. It can be a function like <code>mean</code> or <code>median</code> or a vector of length <code>ncol(x)</code> containing the center value of each column.
<code>control</code>	a list whose elements must be the same as (or a subset of) the parameters above. If the control object is supplied, the parameters from it will be used and any other given parameters are overridden.

**Details**

Basically, this algorithm considers the directions of each observation through the origin of the centered data as possible projection directions. As this algorithm has some drawbacks, especially if `ncol(x) > nrow(x)` in the data matrix, there are several improvements that can be used with this algorithm.

- `update`An updating step basing on the algorithm for finding the eigenvectors is added to the algorithm. This can be used with any `CalcMethod`
- `sphere`Additional search directions are added using random directions. The random directions are determined using random data points generated from a p-dimensional multivariate standard normal distribution. These new data points are projected to the unit sphere, giving the new search directions.

- `lincomb` Additional search directions are added using linear combinations of the observations. It is similar to the "sphere"-algorithm, but the new data points are generated using linear combinations of the original data  $b_1 \cdot x_1 + \dots + b_n \cdot x_n$  where the coefficients  $b_i$  come from a uniform distribution in the interval  $[0, 1]$ .

Similar to the function `princomp`, there is a `print` method for these objects that prints the results in a nice format and the `plot` method produces a scree plot (`screepLOT`). There is also a `biplot` method.

### Value

The function returns a list of class "princomp", i.e. a list similar to the output of the function `princomp`.

<code>sdev</code>	the (robust) standard deviations of the principal components.
<code>loadings</code>	the matrix of variable loadings (i.e., a matrix whose columns contain the eigenvectors). This is of class "loadings": see <code>loadings</code> for its <code>print</code> method.
<code>center</code>	the means that were subtracted.
<code>scale</code>	the scalings applied to each variable.
<code>n.obs</code>	the number of observations.
<code>scores</code>	if <code>scores = TRUE</code> , the scores of the supplied data on the principal components.
<code>call</code>	the matched call.

### Author(s)

Heinrich Fritz, Peter Filzmoser <<P.Filzmoser@tuwien.ac.at>>

### References

C. Croux, P. Filzmoser, M. Oliveira, (2007). Algorithms for Projection-Pursuit Robust Principal Component Analysis, *Chemometrics and Intelligent Laboratory Systems*, Vol. 87, pp. 218-225.

### See Also

`PCAgrid`, `ScaleAdv`, `princomp`

### Examples

```
# multivariate data with outliers
library(mvtnorm)
x <- rbind(rmvnorm(200, rep(0, 6), diag(c(5, rep(1,5)))),
           rmvnorm( 15, c(0, rep(20, 5)), diag(rep(1, 6))))
# Here we calculate the principal components with PCAgrid
pc <- PCAproj(x, 6)
# we could draw a biplot too:
biplot(pc)

# we could use another calculation method and another objective function, and
```

```

# maybe only calculate the first three principal components:
pc <- PCAproj(x, 3, "qn", "sphere")
biplot(pc)

# now we want to compare the results with the non-robust principal components
pc <- princomp(x)
# again, a biplot for comparision:
biplot(pc)

```

---

PCdiagplot

*Diagnostic plot for principal components*


---

### Description

Computes Orthogonal Distances (OD) and Score Distances (SD) for already computed principal components using the projection pursuit technique.

### Usage

```
PCdiagplot(x, PCobj, crit = c(0.975, 0.99, 0.999), ksel = NULL, plot = TRUE, plotbw
```

### Arguments

<code>x</code>	a numeric matrix or data frame which provides the data for the principal components analysis.
<code>PCobj</code>	a PCA object resulting from <a href="#">PCAproj</a> or <a href="#">PCAgrid</a>
<code>crit</code>	quantile(s) used for the critical value(s) for OD and SD
<code>ksel</code>	range for the number of PCs to be used in the plot; if NULL all PCs provided are used
<code>plot</code>	if TRUE a plot is generated, otherwise only the values are returned
<code>plotbw</code>	if TRUE the plot uses gray, otherwise color representation
<code>raw</code>	if FALSE, the distribution of the SD will be transformed to approach chisquare distribution, otherwise the raw values are reported and used for plotting
<code>colgrid</code>	the color used for the grid lines in the plot
<code>...</code>	additional graphics parameters as used in <a href="#">par</a>

### Details

Based on (robust) principal components, a diagnostics plot is made using Orthogonal Distance (OD) and Score Distance (SD). This plot can provide important information about the multivariate data structure.

**Value**

ODist	matrix with OD for each observation (rows) and each selected PC (cols)
SDist	matrix with SD for each observation (rows) and each selected PC (cols)
critOD	matrix with critical values for OD for each selected PC (rows) and each critical value (cols)
critSD	matrix with critical values for SD for each selected PC (rows) and each critical value (cols)

**Author(s)**

Peter Filzmoser <<P.Filzmoser@tuwien.ac.at>>

**References**

P. Filzmoser and H. Fritz (2007). Exploring high-dimensional data with robust principal components. In S. Aivazian, P. Filzmoser, and Yu. Kharin, editors, Proceedings of the Eighth International Conference on Computer Data Analysis and Modeling, volume 1, pp. 43-50, Belarusian State University, Minsk.

M. Hubert, P.J. Rousseeuw, K. Vanden Branden (2005). ROBPCA: a new approach to robust principal component analysis *Technometrics* 47, pp. 64-79.

**See Also**

[PCAproj](#), [PCAgrid](#)

**Examples**

```
# multivariate data with outliers
library(mvtnorm)
x <- rbind(rmvnorm(85, rep(0, 6), diag(c(5, rep(1,5))))),
          rmvnorm( 15, c(0, rep(20, 5)), diag(rep(1, 6))))
# Here we calculate the principal components with PCAgrid
pcprob <- PCAgrid(x, k=6)
resrob <- PCdiagplot(x,pcprob,plotbw=FALSE)

# compare with classical method:
pcclass <- PCAgrid(x, k=6, method="sd")
resclass <- PCdiagplot(x,pcclass,plotbw=FALSE)
```

---

plotcov

*Compare two Covariance Matrices in Plots*

---

**Description**

allows a direct comparison of two estimations of the covariance matrix (e.g. resulting from covPC) in a plot.

**Usage**

```
plotcov(cov1, cov2, method1, labels1, method2, labels2, ndigits, ...)
```

**Arguments**

<code>cov1</code>	a covariance matrix (from <code>cov</code> , <code>covMcd</code> , <code>covPC</code> , <code>covPCAgrid</code> , <code>covPCAproj</code> , etc.
<code>cov2</code>	a covariance matrix (from <code>cov</code> , <code>covMcd</code> , <code>covPC</code> , <code>covPCAgrid</code> , <code>covPCAproj</code> , etc.
<code>method1</code>	legend for ellipses of estimation method1
<code>method2</code>	legend for ellipses of estimation method2
<code>labels1</code>	legend for numbers of estimation method1
<code>labels2</code>	legend for numbers of estimation method2
<code>ndigits</code>	number of digits to use for printing covariances, by default <code>ndigits=4</code>
<code>...</code>	additional arguments for text or plot

**Details**

Since (robust) PCA can be used to re-compute the (robust) covariance matrix, one might be interested to compare two different methods of covariance estimation visually. This routine takes as input objects for the covariances to compare the output of `cov`, but also the return objects from `covPCAgrid`, `covPCAproj`, `covPC`, and `covMcd`. The comparison of the two covariance matrices is done by numbers (the covariances) and by ellipses.

**Value**

only the plot is generated

**Author(s)**

Heinrich Fritz, Peter Filzmoser <<P.Filzmoser@tuwien.ac.at>>

**References**

C. Croux, P. Filzmoser, M. Oliveira, (2007). Algorithms for Projection-Pursuit Robust Principal Component Analysis, *Chemometrics and Intelligent Laboratory Systems*, Vol. 87, pp. 218-225.

**See Also**

[PCAgrid](#), [PCAproj](#), [princomp](#)

**Examples**

```
# multivariate data with outliers
library(mvtnorm)
x <- rbind(rmvnorm(200, rep(0, 6), diag(c(5, rep(1,5)))),
           rmvnorm( 15, c(0, rep(20, 5)), diag(rep(1, 6))))
plotcov(covPCAproj(x), covPCAgrid(x))
```

---

`qn`*scale estimation using the robust Qn estimator*

---

**Description**

Returns a scale estimation as calculated by the (robust) Qn estimator.

**Usage**

```
qn(x)
```

**Arguments**

`x` a vector of data

**Details**

The Qn estimator computes the first quartile of the pairwise absolute differences of all data values.

**Value**

The estimated scale of the data.

**Author(s)**

Heinrich Fritz, Peter Filzmoser <<P.Filzmoser@tuwien.ac.at>>

**References**

P.J. Rousseeuw, C. Croux (1993) Alternatives to the Median Absolute Deviation, *JASA*, **88**, 1273-1283.

**See Also**

[mad](#)

**Examples**

```
# data with outliers
x <- c(rnorm(100), rnorm(10, 10))
qn(x)
```

---

`ScaleAdv`*centers and rescales data*

---

**Description**

Data is centered and rescaled (to have mean 0 and a standard deviation of 1).

**Usage**

```
ScaleAdv(x, center = mean, scale = sd)
```

**Arguments**

<code>x</code>	matrix containing the observations. If this is not a matrix, but a data frame, it is automatically converted into a matrix using the function <code>as.matrix</code> . In any other case, (eg. a vector) it is converted into a matrix with one single column.
<code>center</code>	this argument indicates how the data is to be centered. It can be a function like <code>mean</code> or <code>median</code> or a vector of length <code>ncol(x)</code> containing the center value of each column.
<code>scale</code>	this argument indicates how the data is to be rescaled. It can be a function like <code>sd</code> or <code>mad</code> or a vector of length <code>ncol(x)</code> containing the scale value of each column.

**Details**

The default `scale` being `NULL` means that no rescaling is done.

**Value**

The function returns a list containing

<code>x</code>	centered and rescaled data matrix.
<code>center</code>	a vector of the centers of each column <code>x</code> . If you add to each column of <code>x</code> the appropriate value from <code>center</code> , you will obtain the data with the original location of the observations.
<code>scale</code>	a vector of the scale factors of each column <code>x</code> . If you multiply each column of <code>x</code> by the appropriate value from <code>scale</code> , you will obtain the data with the original scales.

**Author(s)**

Heinrich Fritz, Peter Filzmoser <<P.Filzmoser@tuwien.ac.at>>

**References**

C. Croux, P. Filzmoser, M. Oliveira, (2007). Algorithms for Projection-Pursuit Robust Principal Component Analysis, *Chemometrics and Intelligent Laboratory Systems*, Vol. 87, pp. 218-225.

**Examples**

```
x <- rnorm(100, 10, 5)
x <- ScaleAdv(x)$x
c(mean(x), sd(x))

# can be used with multivariate data too
library(mvtnorm)
x <- rmvnorm(100, 3:7, diag((7:3)^2))
res <- ScaleAdv(x, center = l1median, scale = mad)
res

# instead of using an estimator, you could specify the center and scale yourself too
x <- rmvnorm(100, 3:7, diag((7:3)^2))
res <- ScaleAdv(x, 3:7, 7:3)
res
```

# Index

## \*Topic **multivariate**

- covPC, 1
- covPCA, 3
- llmedian, 4
- llmedian\_NLM, 5
- PCAGrid, 6
- PCAprj, 8
- plotcov, 12
- qn, 14
- ScaleAdv, 15

## \*Topic **robust**

- covPCA, 3
- llmedian, 4
- llmedian\_NLM, 5
- PCAGrid, 6
- PCAprj, 8
- PCdiagplot, 11
- qn, 14

as.matrix, 15

biplot, 10

cov, 13

- covMcd, 13
- covPC, 1, 13
- covPCA, 3
- covPCAGrid, 13
- covPCAGrid(covPCA), 3
- covPCAprj, 13
- covPCAprj(covPCA), 3

llmedian, 4

- llmedian\_BFGS(llmedian\_NLM), 5
- llmedian\_CG(llmedian\_NLM), 5
- llmedian\_NLM, 5
- llmedian\_NM(llmedian\_NLM), 5

loadings, 8, 10

mad, 7, 9, 14, 15

mean, 7, 9, 15

median, 4, 6, 7, 9, 15

nlm, 5, 6

optim, 5, 6

par, 11

- PCAGrid, 2, 3, 6, 10–13
- PCAprj, 2, 3, 8, 8, 11–13
- PCdiagplot, 11
- plotcov, 12
- princomp, 2, 3, 7, 8, 10, 13
- print, 8, 10

qn, 14

- ScaleAdv, 3, 8, 10, 15
- screplot, 7, 10
- sd, 7, 9, 15