

# Package ‘munfold’

January 2, 2012

**Type** Package

**Title** Metric Unfolding

**Version** 0.2

**Date** 2009-08-19

**Author** Martin Elff

**Maintainer** Martin Elff <me1ff@essex.ac.uk>

**Description** This package provides Schoenemans algorithm for metric multidimensional unfolding and Procrustes rotation of unfolding results.

**License** GPL-2

**LazyLoad** Yes

**Depends** memisc, MASS

**Repository** CRAN

**Date/Publication** 2010-04-08 06:08:23

## R topics documented:

procrustes . . . . .	2
unfold . . . . .	2

**Index** [6](#)

---

procrustes	<i>Procrustes Rotation</i>
------------	----------------------------

---

### Description

procrustes performs procrustes rotation, at the moment only of unfold solutions.

### Usage

```
procrustes(x, ...)

## S3 method for class 'unfolding'
procrustes(x, use=attr(x,"procrustes_use"), target, ...)
```

### Arguments

x	an object the components of which to rotate.,
use	which of the components of x should be used as criterion for rotation.
target	a matrix to which the rotation criterion should be brought as close as possible.
...	further arguments for future methods, currently ignored.

### Value

a copy of x with components appropriately rotated.

---

unfold	<i>Metric Unfolding</i>
--------	-------------------------

---

### Description

unfold computes a metric unfolding solution based on a rectangular matrix, that is, reconstructs two sets of points from the distances between points of the first set and the points of the second set. uapply applies a function the two point sets that are reconstructed by unfold.

### Usage

```
unfold(x,...)

## S3 method for class 'matrix'
unfold(x, ndims=NULL, squared=FALSE, tol=1e-7,
       method=c("Schoenemann", "CG"), ...)

## S3 method for class 'formula'
unfold(x,data=parent.frame(), ...)
```

```
## S3 method for class 'unfolding'
biplot(x, dimen=c(1,2), type=attr(x,"biplot_type"),
       xlim, ylim, tpos=c(4,2), tposdim=1,
       asp=1, lty=c(1,2), lwd=c(1,1), pch=c(1,3),
       contour.col="black", contour.lty=1,
       xlab=paste("Dimension ",dimen[1]),
       ylab=paste("Dimension ",dimen[2]),
       ...)

## S3 method for class 'unfolding'
plot(x, y=NULL, dimen=1, discrete=attr(x,"plot_discrete"),
     use.rownames=discrete, xlab=paste("Dimension ",dimen), ...)

uapply(x,FUN)
```

### Arguments

x	for <code>unfold.matrix</code> : a rectangular matrix that contains distances or squared distances (if argument <code>squared</code> is <code>TRUE</code> ). For <code>unfold.formula</code> : a formula which specifies the variables that form the columns of the matrix of distances. For <code>biplot.unfolding</code> and <code>plot.unfolding</code> : an object that contains an unfolding solution.
data	a data frame or an environment that contains variables specified in the formula given as first argument.
ndims	an optional integer value that specifies the dimensionality of the solution. If <code>NULL</code> the dimensionality is selected automatically based on a singular value decomposition of the matrix of squared distances.
squared	a logical value; does the matrix <code>D</code> contain squared distances?
tol	a tolerance value for the convergence of the conjugate gradients method.
method	a method for the iterative computation of the unfolding solution.
y	a dummy argument for compatibility with default methods, ignored.
dimen	for <code>biplot</code> : a two-element integer vector, for <code>plot</code> : a single integer value, that specifies the dimension(s) of the unfolding solution to be plotted.
type	a character vector of length less than or equal to 2. Determines how each of the two point sets of the unfolding solutions are represented in the biplot. Valid choices are <ul style="list-style-type: none"> <li>"points" the respective set of points are plotted as points in the biplot.</li> <li>"lines" the points of the respective set are connected by lines.</li> <li>"both" the points of the respective set are plotted as points and connected by lines.</li> <li>"text" the points of the respective set are represented by the corresponding row names and, if argument <code>tpos</code> is present, by points.</li> <li>"density" contour lines are drawn of two-dimensional kernel density estimate for the respective set of points. This biplot type uses the function <code>kde2d</code> of library <code>MASS</code>.</li> </ul>

<code>tpos</code>	a two-element integer vector; specifies the position of text labels relative to the points. For the meaning of these integer values see <a href="#">text</a>
<code>tposdim</code>	an integer value; specifies which how elements of <code>tpos</code> are used. Labels of points with negative positions along coordinate axis <code>dimen[tposdim]</code> are positioned according to <code>tpos[1]</code> , labels of other points are positioned according to <code>tpos[1]</code> .
<code>xlab, ylab, xlim, ylim, asp, lty, lwd, pch</code>	arguments passed to <a href="#">plot.default</a> .
<code>contour.col, contour.lty</code>	colour and line type for contour lines, see <a href="#">contour</a> .
<code>discrete</code>	a logical vector of length 2; if TRUE, the respective set of points are represented by spikes in theplot, otherwise the set is represented by a graph of a kernel density estimate.
<code>use.rownames</code>	logical; should row names used for annotation?
<code>...</code>	further arguments passed to <a href="#">optim</a> in case of <code>unfold</code> or <a href="#">plot.default</a> in case of the plotting methods.
<code>FUN</code>	a function applied to the two sets of points that result from the unfolding.

### Details

`unfold` first computes an unfolding solution according to Schoenemanns metric unfolding algorithm that uses only linear algebra operations. This preliminary solution is then refined by minimizing the stress using a conjugate-gradients method.

`uapply` applies a given function to the two sets of points recovered by an unfolding solution. It applies the function to the components A and B of an object of class "unfolding".

### Value

`unfold` returns an object of class "unfolding" with components

<code>A</code>	a numeric matrix representing the first set of points. Each row contains the coordinate of one point of the first set.
<code>B</code>	a numeric matrix representing the second set of points. Each row contains the coordinate of one point of the second set.
<code>fitted</code>	a numeric matrix that contains the fitted squared distances.
<code>stress</code>	A stress value, denotes the "badness of fit".

### Examples

```
r <- seq(from=0, to=2*pi, length=24)
a1 <- cos(r)*4 + 0.00001*rnorm(r)
a2 <- sin(r)*4 + 0.00001*rnorm(r)
b1 <- c(.5, -.5, -.5, .5)*3 + 5
b2 <- c(.5, .5, -.5, -.5)*3 + 1

D1 <- outer(b1, a1, "-")
D2 <- outer(b2, a2, "-")
```

```
Dsq <- D1^2+D2^2

Dsq.uf<-unfold(sqrt(Dsq),squared=FALSE)

oldpar <- par(mfrow=c(1,2))
A <- cbind(a1,a2)
B <- cbind(b1,b2)

ltype <- c(rep(1,NROW(A)),rep(2,NROW(A)))

orig <- rbind(A,B)
unfolded <- rbind(Dsq.uf$A,Dsq.uf$B)

xlim <- ylim <- range(orig)#*1.5

plot(A,type="b",pch=1,
      xlim=xlim,ylim=ylim,
      xlab="Dimension 1",ylab="Dimension 2",main=expression("Original data"),asp=1)
lines(B,type="b",pch=3,lty=2)
abline(h=0,v=0,lty=3)

biplot(Dsq.uf,type="b",
        xlim=xlim,ylim=ylim,
        main=expression(paste(italic(unfold)," solution")),asp=1)

par(oldpar)
```

# Index

\*Topic **hplot**

unfold, [2](#)

\*Topic **multivariate**

unfold, [2](#)

biplot.unfolding (unfold), [2](#)

contour, [4](#)

kde2d, [3](#)

optim, [4](#)

plot.default, [4](#)

plot.unfolding (unfold), [2](#)

procrustes, [2](#)

text, [4](#)

uapply (unfold), [2](#)

unfold, [2](#)