Package ‘Momocs’

April 30, 2012

Type Package

Title Shape Analysis of Outlines

Version 0.1

Date 2012-05-01

Author Vincent Bonhomme, Sandrine Picq, Julien Claude

Maintainer Vincent Bonhomme <bonhomme.vincent@gmail.com>

Description Momocs is intended to ease and popularize shape analysis of outlines (especially using elliptical Fourier analysis). It mostly hinges on the functions developped in Morphometrics with R (Claude, 2008). From outline extraction of images and elliptical Fourier calculation to multivariate analysis and the visualization of transformations within the morphological space, Momocs provides a complete and convenient toolkit to specialists within every field that are, or may be, interested in morphological comparisons of outlines.

License GPL(>=2)

Depends methods, sp, ReadImages

Collate global.R Coo.R Nef.R

R topics documented:

Momocs-package 2
bottles dataset 3
closed.outline 3
col.sel 4
cont.sample 5
cont.smooth 5
Coo-class 6
dev.qual 7
dev.quant 7
draw.Fell 8
eFa 9
efourier 9
get.cont 10
get.Nef 11
harm.pow 11

iefourier 12
manova.nef 12
morph.PC 13
morph.sp 14
Nef-class 15
panel.lm 15
pca 16
pca2shp 18
plot 19
show-methods 19
tps 19
tps.grid 20
tps2d 21
traj 22
Index **23**

Momocs-package *Outline Analysis using Elliptical Fourier Analysis.*

Description

Momocs is intended to ease and popularize shape analysis of outlines (especially using elliptical Fourier analysis). It mostly hinges on the functions developped in Morphometrics with R (Claude, 2008). From outline extraction of images and elliptical Fourier calculation to multivariate analysis and the visualization of transformations within the morphological space, Momocs provides a complete and convenient toolkit to specialists within every field that are, or may be, interested in morphological comparisons of outlines.

It comes with its vignette that details step by step how to perform Elliptical Fourier Analysis on a set of shapes, whether starting from images or coordinates in `.txt` files.

Author(s)

1. Vincent Bonhomme, French Institute of Pondicherry, India.
2. Sandrine Picq, UMR CBAE, Montpellier, France.
3. Julien Claude UMR, ISEM, Universite de Montpellier II, France.

References

Claude, J. (2008) *Morphometrics with R*, Use R! series, Springer 316 pp.

Examples

```
## Not run:
data(bottles.cont)
plot(bottles.cont)
dev.qual(bottles.cont)
dev.quant(bottles.cont)
harm.pow(bottles.cont)
nef <- get.Nef(bottles.cont)
fac <- factor(rep(c("beer","whisky"), each=20))
pca(nef, fac)
```

```
pca3(nef, fac)
pca(tps(nef, fac)
from <- c(-0.5, 0.25)
to <- -from
tps.iso(nef, fr=from, to=to)
tps.vf(nef, fr=from, to=to)
tps.grid(nef, fr=from, to=to)

## End(Not run)
```

bottles dataset *Two "bottles" datasets of outlines and harmonic coefficients.*

Description

Two datasets are provided. First, `bottles.cont` contains 20 whisky and 20 beer bottles full outlines coordinates. Then, `bottles.nef` consists of a `Nef` object obtained with default parameters (32 harmonics, no smoothing) on `bottles.cont`.

Usage

```
data(bottles.cont)
data(bottles.nef)
```

Format

A `Coo`- class object that contains in the slot `@coo`, the lists of $(x; y)$ coordinates. A `Nef`-class object that contains in the slot `@coeff`, the matrix of harmonic coefficients.

Source

Images grabbed on the internet and prepared by the package's authors.

Examples

```
data(bottles.cont)
bottles.cont

data(bottles.nef)
bottles.nef
```

closed.outline *Closes outlines.*

Description

Closes lists of outline coordinates.

Usage

```
closed.outline(cont)
```

Arguments

`cont`	`list` or `matrix` of $(x; y)$ coordinates

Value

The `list` of $(x; y)$ coordinates provided with the first coordinates added at the end of the list.

Examples

```
fake <- list(x=1:4, y=5:8)
closed.outline(fake)
```

col.sel *Helps to select the columns indices of an harmonic coefficient matrix.*

Description

Returns the columns' indices of a matrix which colums are in the format:

$$(A_1, ..., A_n, B_1, ..., B_n, C_1, ..., C_n, D_1, ..., D_n)$$

Usage

```
col.sel(h.fr = 1, h.to = 8, h.max = 32, drop = FALSE)
```

Arguments

`h.fr`	`integer` indicating the first harmonic to retain
`h.to`	`integer` indicating the last harmonic to retain
`h.max`	`integer` indicating the total number of harmonics (usually number of columns/4)
`drop`	`logical` indicating whether to drop or not the first A_1 harmonic

Value

Returns a vector of `integer` indicating the selected columns indices.

Examples

```
col.sel(1, 8, 32)
col.sel(1, 4, 8, TRUE)
```

cont.sample *Samples points along a list of $(x; y)$ coordinates.*

Description

Given a list of $(x; y)$ coordinates, samples equidistant points along this outline.

Usage

```
cont.sample(coo, n)
```

Arguments

`coo`	the `Coo` object
`n`	`integer` indicating the number of points to sample

Value

Returns a, usually shortened, `list` of eqn(x; y) coordinates.

Examples

```
fake <- list(x=1:100, y=101:200)
cont.sample(fake, 10)
```

cont.smooth *Smoothes a list or a matrix of $(x; y)$ coordinates.*

Description

Applies a simple algorithm to smooth outlines, particularly to remove, if needed, digitalization artefacts.

Usage

```
cont.smooth(M, n)
```

Arguments

`M`	`list` or a matrix of $(x; y)$ coordinates to smooth
`n`	`integer` indicating how many smoothing iterations to perform

Details

The algorithm used is simplistic: the new $(x; y)_n$ coordinates are calculated as:

$$\frac{1}{4} \times (x; y)_{n-1} + \frac{1}{2} \times (x; y)_n + \frac{1}{4} \times (x; y)_{n+1}$$

Value

A `list` of smoothed coordinates.

Examples

```
fake <- list(x=1:10, y=20:11)
cont.smooth(fake, 1)
cont.smooth(fake, 10)
```

Coo-class *Class* "Coo".

Description

This class contains, so far, a single slot `@coo` that contains list of $(x; y)$ coordinates. Calibration methods and Elliptical Fourier Analysis can be applied to `Coo`-objects. Additional slots will be implemented and store the dataset structure. A `Coo`-object builder, `Coo()` is available to coerce a matrix and create a `Coo`-object.

Slots

`coo`**:** a list of `(x; y)` coordinates.

Methods

dev.qual Calculates and plots inverse reconstruction of outlines

dev.quant Calculates and plots deviations between original and reconstructed outlines

get.Nef Calculates Elliptical Fourier Analysis with specified parameters

harm.pow Calculates and plots the Fourier power spectrum

plot Plots a single or a range of outlines

Examples

```
data(bottles.cont)
## Not run:
bottles.cont
plot(bottles.cont)
dev.qual(bottles.cont)
dev.quant(bottles.cont)
(get.Nef(bottles.cont))
bottles.cont@coo # to access the coordinates list

## End(Not run)
```

dev.qual *Calculates and plots reconstructed outlines.*

Description

Calculates and plots inverse reconstruction of outlines based on the list of coordinates in a Coo-object and a given number of harmonics and/or smoothing iterations. This methods is the visual way to calibrate Elliptical Fourier Analysis parameters.

Usage

```
dev.qual(Coo, id = 1:length(Coo@coo),
 nb.h = 32, smooth.it = 0, range = seq(1, nb.h, len=4))
```

Arguments

`Coo`	the Coo-object
`id`	`integer` indicating the single or the range of outlines indices to consider
`nb.h`	`integer` indicating how many harmonics to calculate
`smooth.it`	`integer` indicating how many smoothing iterations to perform
`range`	`integer` indicating the range of harmonics orders to explore

Examples

```
data(bottles.cont)
## Not run:
dev.qual(bottles.cont)
dev.qual(bottles.cont, id=24)
dev.qual(bottles.cont, id=24, nb.h=64)
dev.qual(bottles.cont, range=seq(1,16))
dev.qual(bottles.cont, smooth.it = 50)

## End(Not run)
```

dev.quant *Calculates and plots sum of euclidean deviations between original and reconstructed outlines.*

Description

Calculates and plots sum of euclidean deviations between one or a range of original and reconstructed shapes, normalized by the calliper length, *i.e.* the longest length measured between two outlines points.

Usage

```
dev.quant(Coo, id = 1:length(Coo@coo), nb.h = 32, smooth.it = 0, plot=TRUE)
```

Arguments

Coo	the Coo object
id	integer indicating the single or the range of outlines indices to consider
nb.h	integer indicating how many harmonics to calculate
smooth.it	integer indicating how many smoothing iterations to perform
plot	a logical indicating whether to plot or not the results

Examples

```
## Not run:
data(bottles.cont)
dev.quant(bottles.cont, id=4)
dev.quant(bottles.cont, id=4, nb.h=12)

## End(Not run)
```

draw.Fell *Draws "Fourier Ellipses".*

Description

Calculates and draws a "Fourier Ellipse" corresponding to the harmonic coefficients provided

Usage

```
draw.Fell(an = pi, bn = -pi, cn = pi, dn = pi,
n = 200, cols = topo.colors, title = FALSE)
```

Arguments

an	a numeric corresponding to the a_n harmonic coefficient
bn	a numeric corresponding to the b_n harmonic coefficient
cn	a numeric corresponding to the c_n harmonic coefficient
dn	a numeric corresponding to the d_n harmonic coefficient
n	integer indicating how many points to retrieve from outline reconstruction
cols	a color palette such as topo.colors, or those produced by colorRampPalette
title	integer indicating whether to add a title to the plot

Examples

```
draw.Fell()
draw.Fell(2*pi, -pi, pi, 3*pi, title=TRUE)
```

eFa *Elliptical Fourier Analysis on Coo objects.*

Description

Once the number of harmonics to calculate and the number of smoothing iterations to perform have been determined, calculates elliptic Fourier transforms on the list of $(x; y)$ outline coordinates included in Coo-objects.

Usage

```
eFa(coo, nb.h = 32, smooth.it = 0, fromrt = FALSE)
```

Arguments

`coo`	the Coo-object
`nb.h`	`integer` indicating how many harmonics to calculate
`smooth.it`	`integer` indicating how many smoothing iterations to perform
`fromrt`	`logical` indicating whether the position of the starting point has to be preserved or not

Examples

```
data(bottles.cont)
eFa(bottles.cont@coo[[1]])
```

efourier *Computes the Fourier coefficients on a list of coordinates.*

Description

Computes the Fourier coefficients a_o, a_n, b_n, c_o, c_n, d_n from a list of $(x; y)$ coordinates of the sampled points.

Usage

```
efourier(coo, nb.h = 32, smooth.it = NULL)
```

Arguments

`coo`	the Coo-object
`nb.h`	codeinteger indicating how many harmonics to calculate
`smooth.it`	codeinteger indicating how many smoothing iterations to perform

Value

ao	numeric: the a_o harmonic coefficient
co	numeric: the c_o harmonic coefficient
an	a vector of numeric indicating the $a_(1 \rightarrow n)$ harmonic coefficients
bn	a vector of numeric indicating the $b_(1 \rightarrow n)$ harmonic coefficients
cn	a vector of numeric indicating the $c_(1 \rightarrow n)$ harmonic coefficients
dn	a vector of numeric indicating the $d_(1 \rightarrow n)$ harmonic coefficients

Author(s)

Originally written by Julien Claude. Claude, J. (2008) *Morphometrics Using R*, Use R! series, Springer 330 pp.

Examples

```
data(bottles.cont)
efourier(bottles.cont@coo[[1]])
```

get.cont *Extract $(x; y)$ coordinates and create a* Coo*-object.*

Description

Extracts from a set of black and white images or a list of coordinates written in a set of 2-columns ("x" and "y") `.txt` files.

Usage

```
get.cont(path)
```

Arguments

path	a path to indicate where are the images or the `.txt` files to use

Details

If no `path` is provided, the user is interactively asked to choose a folder.

`get.cont` uses the `Conte` algorithm that starts on the center of every outline in the `Coo`-object provided (or the `imagematrix` provided if `Conte()` is directly used). If this point does not correspond to a black pixel, *i.e.* not contained within the shape, the user is interactively asked to select interactively a point within the shape.

Value

a `Coo`-object is returned.

Author(s)

`Conte` was originally written by Julien Claude. Claude, J. (2008) *Morphometrics Using R*, Use R! series, Springer 330 pp.

Examples

```
## Not run:
data(bottles.cont)
get.cont()

## End(Not run)
```

get.Nef *Calculates Elliptical Fourier Analysis.*

Description

Calculates Elliptical Fourier Analysis with specified parameters.

Usage

```
get.Nef(Coo, nb.h=32, smooth.it=0, fromrt=FALSE)
```

Arguments

`Coo`	the `Coo` object
`nb.h`	`integer` indicating how many harmonics to calculate
`smooth.it`	`integer` indicating how many smoothing iterations to perform
`fromrt`	`logical` whether the position of the first point has to be preserved

Value

a `Coo`-object is returned.

Examples

```
data(bottles.cont)
nef <- get.Nef(bottles.cont)
pca3(nef)
```

harm.pow *Calculates and plots Fourier harmonic spectra.*

Description

Calculates and plots Fourier power spectra calculated as: $Power_n = \frac{A_n^2+B_n^2+C_n^2+D_n^2}{2}$.

Arguments

`coo`	the `Coo` object
`nb.h`	`integer` indicating how many harmonics to calculate
`smooth.it`	`integer` indicating how many smoothing iterations to perform
`plot`	`logical` indicating whether to plot the results
`max.h`	`integer` specifying the total number of harmonics to include
`first`	`logical` indicating whether to include the first harmonic

iefourier *Calculates inverse Fourier Elliptical.*

Description

Calculates inverse Fourier Elliptical if passed with harmonic coefficients.

Usage

```
iefourier(an, bn, cn, dn, k, n, ao = 0, co = 0)
```

Arguments

an	a vector of `numeric` indicating the $a_{1\rightarrow n}$ harmonic coefficient
bn	a vector of `numeric` indicating the $b_{1\rightarrow n}$ harmonic coefficient
cn	a vector of `numeric` indicating the $c_{1\rightarrow n}$ harmonic coefficient
dn	a vector of `numeric` indicating the $d_{1\rightarrow n}$harmonic coefficient
k	`integer` indicating the number of harmonics to claculate
n	`integer` indicating the number of points to sample on the calculated outline
ao	`numeric`: the a_0 harmonic coefficient
co	`numeric`: c_0 harmonic coefficient

Value

a `list` of `(x; y)` coordinates.

Author(s)

Entirely written by Julien Claude. Claude, J. (2008) *Morphometrics Using R*, Use R! series, Springer 330 pp.

manova.nef *Calculates MANOVA on a harmonic coefficient matrix.*

Description

Calculates Multivariate Analysis of Variance (MANOVA) on the harmonic coefficient matrix contained in `Nef`-objects.

Usage

```
manova.nef(Nef, fac, harmonics.retained, drop=FALSE)
```

Arguments

`Nef`	the `Nef`-object
`fac`	a factor indicating the grouping desing
`harmonics.retained`	codeinteger indicating how many harmonics to include
`drop`	codelogical indicating whether to drop or retain the first harmonic

Details

This function is a wrapper to calculate MANOVAs *i.e.* test the significance of *between* vs. *within* geometric differences between sets of shapes. If not specified, the number of harmonics retained is calculated so that it is lower than the number of individuals minus two.

Examples

```
data(bottles.nef)
fac <- factor(rep(c("beer", "whisky"), each=20))
manova.nef(bottles.nef, fac=fac)
```

morph.PC *Plots the morphological space.*

Description

Given a matrix of harmonic coefficients, a `Nef`-object, calculates and plots morphological space *i.e.* reconstructed shapes using and distributed on the orthonormal set defined by Principal Component axes.

Usage

```
morph.PC(Nef, sd.nb=1, pca.ax=seq(1, 3))
```

Arguments

`Nef`	the `Nef` object
`sd.nb`	a `numeric` given the number of standard deviation to represent shape deviation along each PC axis
`pca.ax`	a numeric or a vector of numeric indicating on which Principal Component to display variation

Examples

```
data(bottles.nef)
fac <- factor(rep(c("beer", "whisky"), each=20))

morph.sp(bottles.nef)
morph.PC(bottles.nef, 1, 1:5)
morph.PC(bottles.nef, 2, 1:3)
```

`morph.sp` *Plots the morphological space.*

Description

Given a matrix of harmonic coefficients, a `Nef`-object, calculates and plots morphological space *i.e.* reconstructed shapes using and distributed on the orthonormal set defined by Principal Component axes.

Usage

```
morph.sp(Nef,
           PCa = 1, PCb = 2, nb.PCa = 5, nb.PCb = 6, fac = NA,
           morph.sp.extend = 1, zoom.extend = 1.2, asp,
           pch = 20, shp.col = NA, shp.lwd = 1, shp.size, col = "grey40",
   ell = FALSE, r = 1, lwd = 1, title = "Morphological space")
```

Arguments

`Nef` the `Nef` object

`PCa` a `numeric` indicating the first Principal Component axis on which to reconstruct shape

`PCb` a `numeric` indicating the second Pricipal Component axis on which to reconstruct shape

`nb.PCa` a `numeric` indicating how many shape to reconstruct on the first PC axis considered

`nb.PCb` a `numeric` indicating how many shape to reconstruct on the second PC axis considered

`fac` a factor indicating the grouping desing

`morph.sp.extend`
`integer` how much to extend morphological space reconstruction beyond range on the first PC considered

`zoom.extend` `integer` indicating how much to extend the graphical window

`asp` `numeric` and `optionnal` indicating the `asp` of the plotting window

`pch` `integer` or a character indicating the `pch` for each groups to plot

`shp.col` `integer` or a character indicating the `col` of these shapes

`shp.lwd` `numeric` indicating the `lwd` of these shapes borders

`shp.size` `numeric` for fine-tuning of shapes size

`col` `integer` or a character indicating the `col` for each confidence ellipse to plot

`ell` `logical` indicating whether to draw confidence ellipses for every group

`r` `numeric` indicating the number of standard deviation for confidence ellipses computation

`lwd` `numeric` indicating the `lwd` for the confidence ellipses

`title` `character` to change the title of the plot

Examples

```
data(bottles.nef)
fac <- factor(rep(c("beer", "whisky"), each=20))

morph.sp(bottles.nef)
morph.sp(bottles.nef, fac=fac, ell=TRUE)
morph.sp(bottles.nef, fac=fac, nb.PCa=10, nb.PCb=10, ell=TRUE)
morph.sp(bottles.nef, PCa=2, PCb=3)
```

`Nef-class` *Class* `"Nef"`.

Description

A class that contains all the information to visualize and performe multivariate analysis. Contains so far a single slot `@coeff` containing the harmonic coefficient matrix after an Elliptical Fourier Analysis. A `Nef`-object builder, `Nef()` is available.

Slots

`coeff`**:** a matrix of harmonic coefficients.

Methods

manova.nef Calculates MANOVA on a harmonic coefficient matrix

morph.sp Plots the morphological space

pca.tps Plots a single PCA with deformation grids

pca Plots a single PCA

pca3 Plots all the first three PCA axes

show A simple object description

tps.grid Thin Plate Splines deformation grids between two shapes

tps.iso TPS and iso-deformation lines between two shapes

tps.vf TPS and "vector field" of deformation between two shapes

traj Calculates shape intermediates

`panel.lm` *Calculates and plots confidence ellipses.*

Description

Given a set of $(x; y)$ coordinates, calculates confidence ellipses and plots them.

Usage

```
panel.lm(x, y, r = 1, col = "black", lwd = 1, lty = 1)
```

Arguments

x	a vector of numeric x coordinates
y	a vector of numeric y coordinates
r	a numeric indicating the number of standard deviations to calculates confidence ellipses
col	codeinteger or a character indicating the color of the ellipses to draw
lwd	a numeric indicating the lwd to use when drawing ellipses
lty	codeinteger indicating the lty to use when drawing ellipses

Examples

```
plot(x <- rnorm(50), y <- rnorm(50))
panel.lm(x,y)
```

pca *Calculates and plots Principal Component Analysis.*

Description

Calculates and plots Principal Component Analysis usng `prcomp()`. Methods for plotting a single PCA, a triple PCA and deformation grids are detailed below.

Usage

```
pca(Nef, fac = NA,PCa = 1, PCb = 2,
col = "black", pch = 1, lty=1, shp.nb=NA, shp.size,
shp.col="#00000022", shp.border="black", title = "Principal Component Analysis",
legend = TRUE, lab = FALSE, lab.txt = rownames(Nef@coeff), lab.cex = 1, lab.box = TRUE,
ell = TRUE, r = 1, lwd = 1, zoom.x = 0.25, zoom.y = 0.3)

pca3(Nef,  fac = NA,
col = 1:nlevels(fac), pch = 1:nlevels(fac), lty = rep(1,nlevels(fac)),
lab = FALSE, lab.txt = rownames(Nef@coeff), lab.cex = 1, lab.box = TRUE,
ell = 1, r = 1, lwd = 1, zoom = 1.4, legend = FALSE)

pca.tps(Nef, fac = NA, PCa = 1, PCb = 2,
col = "black", pch = 1, ell = TRUE, zoom = 1.4, ncells = 20,
title = "Deformations alongs PC axes")
```

Arguments

Nef	the `Nef` object
fac	the grouping `factor`
PCa	`integer` corresponding to the `a^th` PCA axis to plot
PCb	`integer` corresponding to the `a^th` PCA axis to plot
col	`integer` or `character` indicating the `col` for each group to plot
pch	`integer` or `character` indicating the pch for each groups to plot
lty	`integer` indicating the `lty` for each confidence ellipse to plot

`shp.nb` `integer` indicating how many, if any, shapes to plot

`shp.size` `numeric` indicating the size of these shapes

`shp.col` `integer` or a character indicating the color of these shapes

`shp.border` `integer` or a character indicating the border color of these shapes

`title` `character` to add a better title to the plot

`lab` `logical` indicating whether to plot labels for every point

`lab.txt` `character` vector containing labels names

`lab.cex` `numeric` indicating the `cex` size of these labels

`lab.box` `logical` indicating whether to draw a border for these labs

`ell` `logical` indicating whether to draw confidence ellipses for every group

`r` `numeric` indicating the number of standard deviations for confidence ellipses computation

`lwd` `numeric` indicating the `lwd` for the confidence ellipses

`zoom` `numeric` used to adjust the range of the plot

`zoom.x` `numeric` used to adjust the x-range of the plot

`zoom.y` `numeric` used to adjust the y-range of the plot

`legend` `logical` indicating whether to add a legend on the plot

`ncells` `integer` indicating the number of cells for deformation grids

Examples

```
## Not run:
data(bottles.nef)
fac <- factor(rep(c("beer", "whisky"), each=20))

### pca
pca(bottles.nef)
pca(bottles.nef, fac=fac)
pca(bottles.nef, fac=fac, pch=c(4,5))
pca(bottles.nef, fac=fac, pch=c(4,5), ell=FALSE)
pca(bottles.nef, fac=fac,pch=c(4,5), lty=c(2,3))
pca(bottles.nef, pch=c(4,5), fac=fac, lty=c(2,3), col=c("dodgerblue","firebrick"))
pca(bottles.nef, fac=fac, lab=T)
pca(bottles.nef, fac=fac, lab=T, lab.cex=0.8)
pca(bottles.nef, fac=fac, lab=T, lab.box=FALSE)
pca(bottles.nef, fac=fac, lab=T, lab.txt=c(letters[1:20],LETTERS[1:20]))
pca(bottles.nef, fac=fac, r=0.5)
pca(bottles.nef, fac=fac, r=0.5, zoom.x=0.1, zoom.y=0.1)
pca(bottles.nef, PCa=2, PCb=3)
pca(bottles.nef, shp.nb=5)
pca(bottles.nef, shp.nb=5, shp.col="#FF660033", shp.border="#FF6600")

### pca3
pca3(bottles.nef)
pca3(bottles.nef, fac=fac)
pca3(bottles.nef, fac=fac, pch=c(4,5))
pca3(bottles.nef, fac=fac, pch=c(4,5), lty=c(2,3))
pca3(bottles.nef, fac=fac, pch=c(4,5), lty=c(2,3), col=c("dodgerblue","firebrick"))
pca3(bottles.nef, fac=fac, pch=c(4,5), lty=c(2,3), col=c("dodgerblue","firebrick"), legend=T)
```

```
### pca.tps
pca.tps(bottles.nef)
pca.tps(bottles.nef, fac=fac)

## End(Not run)
```

pca2shp *Reconstructs a shape given using PCA.*

Description

Provided with a harmonic matrix coefficient on which to perform PCA, and given the $(PC_1; PC_2)$ coordinates, it reconstructs the corresponding shape.

Usage

```
pca2shp(pc1 = 0, pc2 = 0, data, nb.h = ncol(data)/4, nb.pts = 500,
amp = 1, col = "black", lwd = 2, plot = TRUE)
```

Arguments

`pc1`	`numeric` indicating the position on the first PC axis
`pc2`	`numeric` indicating the position on the second PC axis
`data`	the harmonic coefficient `matrix`
`nb.h`	`integer` indication how many harmonics to use
`nb.pts`	`integer` indicating the number of points sampled from the reconstructed outlines
`amp`	`numeric` indicating the magnifying factor
`col`	`integer` or `character` indicating the color to ue for drawing the shape
`lwd`	`numeric` indicating the `lwd` for the shape
`plot`	codelogical indicating whether to plot the shape

Examples

```
data(bottles.nef)
pca2shp(0.5, 0.5, bottles.nef@coeff, amp=2)
pca2shp(0, 0, bottles.nef@coeff) # "average" shape
```

plot *Plots coordinate outlines.*

Description

Plots one or a range of the outline(s) contained in Coo-object.

Examples

```
data(bottles.cont)
## Not run:
plot(bottles.cont)
plot(bottles.cont, range=21:40)

## End(Not run)
```

show-methods *show methods for Momocs' objects*

Description

Momocs objects have show methods. They will be expanded in further versions.

Methods

`signature(object = "Coo")` show a Coo-object

`signature(object = "Nef")` show a Coo-object

Examples

```
data(bottles.cont)
bottles.cont

data(bottles.nef)
bottles.nef
```

tps *Produces deformation grids.*

Description

Produces deformation grids using Thin Plate Splines

Usage

```
tps(matr, matt, n, plot = TRUE, col = "black")
```

Arguments

`matr` the reference configuration `matrix`

`matt` the target configuration `matrix`

`n` `integer` indicating the number of displayed column cells

`plot` `logical` indicating whether to plot the grid

`col` `integer` or a character indicating the grid color

Value

Deformation grid obtained by Thin Plate Splines interpolation.

Author(s)

Entirely written by Julien Claude. Claude, J. (2008) *Morphometrics Using R*, Use R! series, Springer 330 pp.

`tps.grid` *Thin Plate Splines deformation grids between two shapes.*

Description

Passed with a `Nef`-object, and two positions on the set defined by PC_1 and PC_2, calculates and plots deformation grids, "vector field" or iso-deformation lines between these two shapes.

Usage

```
tps.grid(Nef, fr, to, nb.pts = 50, amp = 1, grid.size = 50,
 grid.col = "grey40", cont = TRUE,
 cont.col = c("dodgerblue3", "firebrick3"), cont.lwd = rep(3,2))

tps.iso(Nef, fr, to, nb.pts = 200, amp = 1, iso.pts = 1000,
col.pal = topo.colors, col.lev = 500,
cont.to = TRUE, cont.fr = TRUE,  cont.lev = 10,
cont.col = c("dodgerblue3", "firebrick3"),  cont.lwd = rep(3,2))

tps.vf(Nef, fr, to, nb.pts = 100, amp = 1, arr.nb = 300,
arr.len = 0.05, arr.ang = 30, arr.col = "grey40",
arr.pal = FALSE, arr.palette = topo.colors, arr.pal.lev = 20,
arr.lwd = 1, cont.col = c("dodgerblue3", "firebrick3"),
cont.lwd = rep(3, 2))
```

Arguments

`Nef` the `Nef` object

`fr` a vector with two `numerics` indicating the $(x; y)$ coordinates of the starting point. If not provided, `locator(1)` is called

`to` a vector with two `numerics` indicating the $(x; y)$ coordinates of the ending point. If not provided, `locator(1)` is called

`nb.pts` `integer` indicating the number of points used to calculate deformation grids `amp` a `numeric` indicating magnifying factor for deformations

`cont` `logical` indicating whether to plot original and deformed shapes

`cont.to` `logical` indicating whether to plot the original shape

`cont.fr` `logical` indicating whether to plot the deformed shape

`cont.col` `integer` or `character` indicating the `col` of the two outlines compared

`cont.lwd` a `numeric` indicating a vector containing the `lwd` of the two outlines compared

`grid.size` a `numeric` indicating deformation grid size

`grid.col` codeinteger or a character indicatinf the deformation grid color

`iso.pts` a `integer` indicating the number of iso points to use for isolines calculation

`col.pal` a color palette such as `rainbow`, `heat.colors` or such as build by `colorRampPalette` for isolines drawing

`col.lev` `integer` indicating how many color levels to use

`cont.lev` `integer` indicating how many isolines to calculate

`arr.nb` `integer` indicating how many arrows to display

`arr.len` a `numeric` indicating the arrows length

`arr.ang` a `numeric` indicating the arrows angle

`arr.col` `integer` or a character indicating the arrows color

`arr.pal` `logical` indicating whether to use or not a color palette for drawing arrows

`arr.palette` a color palette such as `rainbow`, `heat.colors` or such as build by `colorRampPalette`

`arr.pal.lev` `integer` specifying how many levels to use for the color palette

`arr.lwd` a `numeric` indicating the arrows `lwd`

Examples

```
data(bottles.nef)

## Not run:
tps.grid(bottles.nef, fr=c(-0.05, -0.05), to=c(0.15, 0.05))
tps.vf(bottles.nef, fr=c(-0.05, -0.05), to=c(0.15, 0.05))
tps.iso(bottles.nef, fr=c(-0.05, -0.05), to=c(0.15, 0.05))

## End(Not run)
```

`tps2d` *Returns the position of interpolated coordinates.*

Description

Returns the position of interpolated coordinates using Thin Plate Splines.

Usage

```
tps2d(M, matr, matt)
```

Arguments

`M` original coordinates to be mapped by TPS

`matr` Reference configuration matrix

`matt` Target configuration matrix

Value

Interpolated coordinates arranged in a matrix object.

Author(s)

Entirely written by Julien Claude. Claude, J. (2008) *Morphometrics Using R*, Use R! series, Springer 330 pp.

`traj` *Calculates shape intermediates.*

Description

Given a `Nef` object, and two positions on the set defined by PC_1 and PC_2, calculates and plots intermediate shapes along the euclidean distance between these two shapes.

Usage

```
traj(Nef, fr = c(0, 0), to = c(1, 1), nb.int = 50, nb.pts= 500, save = FALSE, prog = TRUE, pause
```

Arguments

`Nef` the `Nef` object

`fr` a vector with two `numerics` indicating the $(x;y)$ coordinates of the starting point. If not provided, `locator(1) is called`

`to` a vector with two `numerics` indicating the $(x;y)$ coordinates of the ending point. If not provided, `locator(1) is called`

`nb.int` codeinteger giving the number of shape intermediates to calculate

`nb.pts` codeinteger giving the number of points of the reconstructed outlines

`save` `logical` indicating whether to save the images in a dedicated folder

`prog` `logical` indicating whether to plot or not a progression bar

`pause` `logical` indicating whether to ask the user to display successive intermediate shapes

Examples

```
## Not run:
data(bottles.nef)
traj(bottles.nef)
traj(bottles.nef, fr=c(-0.05, -0.05), to=c(0.15, 0.05))

## End(Not run)
```

Index

∗Topic **Coo methods**
- Coo-class, 6
- dev.qual, 7
- dev.quant, 7
- get.Nef, 11
- harm.pow, 11
- plot, 19

∗Topic **Misc functions**
- closed.outline, 3
- col.sel, 4
- cont.sample, 5
- cont.smooth, 5
- draw.Fell, 8
- eFa, 9
- efourier, 9
- get.cont, 10
- iefourier, 12
- panel.lm, 15
- pca2shp, 18
- tps, 19
- tps2d, 21

∗Topic **Nef methods**
- manova.nef, 12
- morph.PC, 13
- morph.sp, 14
- Nef-class, 15
- pca, 16
- tps.grid, 20
- traj, 22

∗Topic **Package**
- bottles dataset, 3
- Momocs-package, 2

∗Topic **methods**
- show-methods, 19

bottles dataset, 3
bottles.cont *(bottles dataset)*, 3
bottles.nef *(bottles dataset)*, 3

closed.outline, 3
col.sel, 4
cont.sample, 5
cont.smooth, 5
Conte *(get.cont)*, 10
Coo *(Coo-class)*, 6
Coo-class, 6

dev.qual, 7
dev.qual,Coo-method *(dev.qual)*, 7
dev.qual-methods *(dev.qual)*, 7
dev.quant, 7
dev.quant,Coo-method *(dev.quant)*, 7
dev.quant-methods *(dev.quant)*, 7
draw.Fell, 8

eFa, 9
efourier, 9

get.cont, 10
get.Nef, 11
get.Nef,Coo-method *(get.Nef)*, 11
get.Nef-methods *(get.Nef)*, 11

harm.pow, 11
harm.pow,Coo-method *(harm.pow)*, 11
harm.pow-methods *(harm.pow)*, 11

iefourier, 12

manova.nef, 12
manova.nef,Nef-method *(manova.nef)*, 12
manova.nef-methods *(manova.nef)*, 12
Momocs *(Momocs-package)*, 2
Momocs-package, 2
morph.PC, 13
morph.PC,Nef-method *(morph.PC)*, 13
morph.PC-methods *(morph.PC)*, 13
morph.sp, 14
morph.sp,Nef-method *(morph.sp)*, 14
morph.sp-methods *(morph.sp)*, 14

Nef *(Nef-class)*, 15
Nef-class, 15

panel.lm, 15
pca, 16
pca,Nef-method *(pca)*, 16
pca-methods *(pca)*, 16
pca.tps *(pca)*, 16

pca.tps,Nef-method *(pca)*, 16
pca.tps-methods *(pca)*, 16
pca2shp, 18
pca3 *(pca)*, 16
pca3,Nef-method *(pca)*, 16
pca3-methods *(pca)*, 16
plot, 19
plot,ANY,ANY-method *(plot)*, 19
plot,Coo,ANY-method *(plot)*, 19
plot-methods *(plot)*, 19
prcomp(), *16*

show,Coo-method *(show-methods)*, 19
show,Nef-method *(show-methods)*, 19
show-methods, 19

topo.colors, *8*
tps, 19
tps.grid, 20
tps.grid,Nef-method *(tps.grid)*, 20
tps.grid-methods *(tps.grid)*, 20
tps.iso *(tps.grid)*, 20
tps.iso,Nef-method *(tps.grid)*, 20
tps.iso-methods *(tps.grid)*, 20
tps.vf *(tps.grid)*, 20
tps.vf,Nef-method *(tps.grid)*, 20
tps.vf-methods *(tps.grid)*, 20
tps2d, 21
traj, 22
traj,Nef-method *(traj)*, 22
traj-methods *(traj)*, 22