Package 'Arothron'

December 5, 2019

Type Package

Title Geometric Morphometrics Analysis and Virtual Anthropology

Version 1.1.1

Author Antonio Profico, Costantino Buzi, Marina Melchionna, Paolo Piras, Pasquale Raia, Alessio Veneziano

Maintainer Antonio Profico <antonio.profico@york.ac.uk>

Description Tools for geometric morphometric analysis. The package includes tools of virtual anthropology to align two not articulated parts belonging to the same specimen, to build virtual cavities as endocast (Profico et al, 2018 <doi:10.1002/ajpa.23493>).

Depends R (>= 3.4.0)

Imports abind (>= 1.4), alphashape3d (>= 1.3), compositions (>= 1.40-1), doParallel (>= 1.0.11), foreach (>= 1.4.4), geometry (>= 0.3-6), graphics(>= 3.4.0), grDevices(>= 3.4.0), methods (>= 3.5), Morpho (>= 2.5.0), parallel (>= 1.0), rgl (>= 0.93.0), Rvcg (>= 0.17), stats (>= 3.4.0), stringr (>= 1.3.0), utils (>= 3.4.0), vegan (>= 2.4)

License GPL-2

Encoding UTF-8

LazyLoad yes

RoxygenNote 6.1.1

NeedsCompilation no

Repository CRAN

Date/Publication 2019-12-05 17:10:03 UTC

R topics documented:

Arothron-package *Geometric Morphometrics Analyses*

Description

Tools for geometric morphometric analysis. The package includes tools of virtual anthropology to align two not articulated parts belonging to the same specimen, to build virtual cavities as endocast, and functions to import and export the coordinates of landmarks and 3D paths into the 'landmarkAscii' and 'am' format files.

Author(s)

Antonio Profico, Costantino Buzi, Marina Melchionna, Paolo Piras, Pasquale Raia, Alessio Veneziano

aro.clo.points *aro.clo.points*

Description

Find the closest matches between a reference (2D or 3D matrix) and a target matrix (2D/3D) or mesh returning row indices and distances

Usage

```
aro.clo.points(target, reference)
```

Arguments

target	kxm matrix or object of class "mesh3d"
reference	numeric: a kxm matrix (coordinates)

Value

position numeric: a vector of the row indices

distances numeric: a vector of the coordinates distances

Author(s)

Antonio Profico, Costantino Buzi, Marina Melchionna, Paolo Piras, Pasquale Raia, Alessio Veneziano

Examples

```
#load an example: mesh, and L set
data(yoda_sur)
data(yoda_set)
sur<-yoda_sur
set<-yoda_set
ver_pos<-aro.clo.points(target=sur,reference=set)
```

arraytolist *arraytolist*

Description

converts an array in a list storing each element of the third dimension of the array (specimen) as element of the list

Usage

```
arraytolist(array)
```

Arguments

array	a kx3xn array with landmark coordinates

Value

a list containing the landmark configurations stored as separated elements

Author(s)

Antonio Profico, Costantino Buzi, Marina Melchionna, Paolo Piras, Pasquale Raia, Alessio Veneziano

bary.mesh *bary.mesh*

Description

This function calculates the barycenter of a matrix or a 3D mesh

Usage

```
bary.mesh(mesh)
```

Arguments

mesh	matrix mesh vertex

Value

barycenter numeric: x,y,z coordinates of the barycenter of the mesh

Author(s)

Antonio Profico, Costantino Buzi, Marina Melchionna, Paolo Piras, Pasquale Raia, Alessio Veneziano

Examples

```
#load an example: mesh, and L set
data(SCP1.mesh)
sur<-SCP1.mesh
bary<-bary.mesh(mesh=sur)
```

`compare_check.set` *compare_check.set*

Description

This function applyes the Digital Alignment Tool (DTA) on a disarticulated model using a reference landmark configuration

Usage

```
compare_check.set(RM_set_1, RM_set_2, DM_set_1, DM_set_2, DM_mesh_1,
  DM_mesh_2)
```

Arguments

`RM_set_1`	matrix: 3D landmark set of the first module acquired on the reference model
`RM_set_2`	matrix: 3D landmark set of the second module acquired on the reference model
`DM_set_1`	matrix: 3D landmark set of the first module acquired on the disarticulated model
`DM_set_2`	matrix: 3D landmark set of the second module acquired on the disarticulated model
`DM_mesh_1`	mesh3d: mesh of the disarticulated model (first module)
`DM_mesh_2`	mesh3d: mesh of the disarticulated model (second module)

Value

SF1 numeric: scale factor used to scale the reference set (first module)

SF2 numeric: scale factor used to scale the reference set (second module)

RM_set_1_sc matrix: scaled 3D reference set (first module)

RM_set_2_sc matrix: scaled 3D reference set (second module)

AM_model list: output of the Morpho::rotmesh.onto function

dist_from_mesh numeric: mesh distance between the aligned model and the scaled reference set

eucl_dist_1 numeric: euclidean distance between the landmark configuration of the disarticulated and reference model (first module)

eucl_dist_2 numeric: euclidean distance between the landmark configuration of the disarticulated and reference model (second module)

procr_dist numeric: procrustes distance between the landmark configuration of the aligned and reference model

procr_dist_1 numeric: procrustes distance between the landmark configuration of the disarticulated and reference model (first module)

procr_dist_2 numeric: procrustes distance between the landmark configuration of the disarticulated and reference model (second module)

eucl_dist numeric: euclidean distance between the landmark configuration of the aligned and reference model

single_l_1 numeric: euclidean distance between the landmark configuration of the disarticulated and reference model (first module)

single_l_2 numeric: euclidean distance between the landmark configuration of the disarticulated and reference model (second module)

Author(s)

Antonio Profico, Costantino Buzi, Marina Melchionna, Paolo Piras, Pasquale Raia, Alessio Veneziano

---

`dec.curve` *dec.curve*

Description

This function computes the order of points on a open 3D curve and finds intermediate points

Usage

```
dec.curve(mat_input, mag, plot = TRUE)
```

Arguments

`mat_input`	numeric: a kx3 matrix
`mag`	numeric: how many times will be divided by the number of initial points
`plot`	logical: if TRUE will be plotted the starting and final point matrices

Value

matt numeric: a kx3 matrix with points coordinates

Author(s)

Antonio Profico, Costantino Buzi, Marina Melchionna, Paolo Piras, Pasquale Raia, Alessio Veneziano

Examples

```
## Create and plot a 3D curve
require(compositions)
require(rgl)
curve_3D<-cbind(1:10,seq(1,5,length=10),rnorm(10,sd = 0.2))
plot3D(curve_3D,bbox=FALSE)
rgl.close()
## Create and plot the new 3D curve (with intermediate points)
dec_curve_3D<-dec.curve(curve_3D, 2, plot = TRUE)
```

DM_base_sur *example dataset*

Description

3D mesh of the first part of the Homo sapiens disarticulated model

Usage

```
data(DM_base_sur)
```

Author(s)

Antonio Profico, Costantino Buzi, Marina Melchionna, Paolo Piras, Pasquale Raia, Alessio Veneziano

---

DM_face_sur *example dataset*

Description

3D mesh of the second part of the Homo sapiens disarticulated model

Usage

```
data(DM_face_sur)
```

Author(s)

Antonio Profico, Costantino Buzi, Marina Melchionna, Paolo Piras, Pasquale Raia, Alessio Veneziano

DM_set *example dataset*

Description

Landmark configurations of the two part of the disarticulated model

Usage

```
data(DM_set)
```

Author(s)

Antonio Profico, Costantino Buzi, Marina Melchionna, Paolo Piras, Pasquale Raia, Alessio Veneziano

dta *dta*

Description

This function applyes the Digital Alignment Tool (DTA) on a disarticulated model using a reference sample

Usage

```
dta(RM_sample, mod_1, mod_2, pairs_1, pairs_2, DM_mesh_1, DM_mesh_2,
  DM_set_1, DM_set_2, method = c("euclidean"))
```

Arguments

RM_sample	3D array: 3D landmark configurations of the reference sample
mod_1	numeric vector: vector containing the position of which landmarks belong to the first module
mod_2	numeric vector: vector containing the position of which landmarks belong to the second module
pairs_1	matrix: a X x 2 matrix containing the indices of right and left landmarks of the first module
pairs_2	matrix: a X x 2 matrix containing the indices of right and left landmarks of the second module
DM_mesh_1	mesh3d: mesh of the disarticulated model (first module)
DM_mesh_2	mesh3d: mesh of the disarticulated model (second module)
DM_set_1	matrix: 3D landmark set of the first module acquired on the disarticulated model
DM_set_2	matrix: 3D landmark set of the second module acquired on the disarticulated model
method	character: specify method to be used to individuate the best DTA ("euclidean" or "procrustes")

Value

AM_mesh mesh3d: mesh of the aligned model

AM_set matrix: landmark configuration of the aligned model

AM_id character: name of the item of the reference sample resulted as best DTA

AM_SF_1 numeric: scale factor used to scale the reference set (first module)

AM_SF_2 numeric: scale factor used to scale the reference set (second module)

distance numeric: distance between the landmark configuration of the aligned and the reference model

tot_proc numeric vector: procrustes distances between aligned and reference models (all DTAs)

tot_eucl numeric vector: euclidean distances between aligned and reference models (all DTAs)

setarray 3D array: landmark configurations of the disarticulated model aligned on each item of the reference sample

Author(s)

Antonio Profico, Costantino Buzi, Marina Melchionna, Paolo Piras, Pasquale Raia, Alessio Veneziano

Examples

```
## Load and plot the disarticulated model of the Homo sapiens case study
library(compositions)
library(rgl)
data(DM_base_sur)
data(DM_face_sur)
open3d()
wire3d(DM_base_sur,col="white")
wire3d(DM_face_sur,col="white")
## Load the landmark configurations associated to the DM
data(DM_set)
## Load the reference sample
data(RMs_sets)
## Define the landmarks belonging to the first and second module
mod_1<-c(1:17) #cranial base
mod_2<-c(18:32) #facial complex
## Define the paired landmarks for each module (optional symmetrization process)
pairs_1<-cbind(c(4,6,8,10,12,14,16),c(5,7,9,11,13,15,17))
pairs_2<-cbind(c(23,25,27,29,31),c(24,26,28,30,32))
## Run DTA
ex.dta<-dta(RM_sample=RMs_sets, mod_1=mod_1, mod_2=mod_2, pairs_1=pairs_1, pairs_2=pairs_2,
DM_mesh_1=DM_base_sur,DM_mesh_2=DM_face_sur, DM_set_1= DM_set[mod_1,], DM_set_2=DM_set[mod_2,])
## Print the name of the best RM
ex.dta$AM_id
## Save the mesh and the landmark set of the AM
AM_mesh<-ex.dta$AM_mesh
AM_set<-ex.dta$AM_set
## Plot the aligned 3D model
library(compositions)
library(rgl)
```

```
open3d()
wire3d(AM_mesh,col="white")
plot3D(AM_set,bbox=FALSE,add=TRUE)
```

---

endomaker *endomaker*

Description

Build endocast from a skull 3D mesh

Usage

```
endomaker(mesh = NULL, path_in = NULL, param1_endo = 1, npovs = 50,
  volume = TRUE, alpha_vol = 100, nVoxels = 1e+05, decmesh = 20000,
  alpha_ext = 30, ncells = 50000, npovs_calse = 50,
  param1_calse = 2, param1_ast = 1.3, decendo = 20000,
  scalendo = 0.5, alpha_end = 100, mpovdist = 10, plot = FALSE,
  colmesh = "orange", save = FALSE, outpath = tempdir(),
  num.cores = NULL)
```

Arguments

mesh	mesh3d: 3D model of the skull
path_in	character: path of the skull where is stored
param1_endo	numeric: parameter for spherical flipping
npovs	numeric: number of Points of View used in the endocast construction
volume	logical: if TRUE the calculation of the volume (expressed in cc) through concave is returned
alpha_vol	numeric: alpha shape for volume calculation
nVoxels	numeric: number of voxels for estimation endocranial volume
decmesh	numeric: decmesh
alpha_ext	numeric: alpha shape for construction external cranial mesh
ncells	numeric: approximative number of cell for 3D grid construction
npovs_calse	numeric: number of Points of View for construction of skull shell
param1_calse	numeric: parameter for calse (construction shell)
param1_ast	numeric: parameter for ast3d (construction row endocast)
decendo	numeric: desired number of triangles (row endocast)
scalendo	numeric: scale factor row endocast (for definition of POVs)
alpha_end	numeric: alpha shape value for concave hull (row endocast)
mpovdist	numeric: mean value between POVs and mesh
plot	logical: if TRUE the endocast is plotted

colmesh	character: color of the mesh to be plotted
save	logical: if TRUE the mesh of the endocast is saved
outpath	character: path where save the endocast
num.cores	numeric: numbers of cores to be used in parallel elaboration

Value

endocast mesh3d: mesh of the endocast

volume numeric: volume of the endocast expressed in cc

Author(s)

Antonio Profico, Costantino Buzi, Marina Melchionna, Paolo Piras, Pasquale Raia, Alessio Veneziano

Examples

```
## Not run:
library(rgl)
data(human_skull)
sapendo<-endomaker(human_skull,param1_endo = 1.0,decmesh = 20000, num.cores=NULL)
open3d()
wire3d(sapendo$endocast,col="violet")
ecv<-sapendo$volume

## End(Not run)
```

---

endomaker_dir *endomaker_dir*

Description

Build library of endocasts from skull 3D meshes

Usage

```
endomaker_dir(dir_path, param1_endo = 1.5, npovs = 50, volume = TRUE,
  alpha_vol = 50, nVoxels = 1e+05, decmesh = 20000, alpha_ext = 30,
  ncells = 50000, npovs_calse = 50, param1_calse = 3,
  param1_ast = 1.3, decendo = 20000, scalendo = 0.5,
  alpha_end = 100, mpovdist = 10, plotall = FALSE,
  colmesh = "orange", save = FALSE, outpath = tempdir(),
  num.cores = NULL)
```

Arguments

dir_path	character: path of the folder where the skull meshes are stored
param1_endo	numeric vector: parameter for spherical flipping
npovs	numeric: number of Points of View used in the endocast construction
volume	logical: if TRUE the volume of the endocast (ECV) is estimated
alpha_vol	numeric: alpha shape for volume calculation
nVoxels	numeric: number of voxels for estimation endocranial volume
decmesh	numeric: decmesh
alpha_ext	numeric: alpha shape for construction external cranial mesh
ncells	numeric: approximative number of cell for 3D grid construction
npovs_calse	numeric: number of Points of View for construction of skull shell
param1_calse	numeric: parameter for calse (construction shell)
param1_ast	numeric: parameter for ast3d (construction row endocast)
decendo	numeric: desired number of triangles (row endocast)
scalendo	numeric: scale factor row endocast (for definition of POVs)
alpha_end	numeric: alpha shape value for concave hull (row endocast)
mpovdist	numeric vector: mean value between POVs and mesh
plotall	logical: if TRUE the endocasts are plotted
colmesh	character: color of the mesh to be plotted
save	logical: if TRUE the mesh of the endocast is saved
outpath	character: path where save the endocast
num.cores	numeric: number of cores to be used in parallel elaboration

Value

endocasts mesh3d: list of meshes of the extracted endocasts

volumes numeric: volumes of the endocasts expressed in cc

Author(s)

Antonio Profico, Costantino Buzi, Marina Melchionna, Paolo Piras, Pasquale Raia, Alessio Veneziano

endo_set *example dataset*

Description

POVs defined inside the endocranial cavity

Usage

```
data(endo_set)
```

Author(s)

Antonio Profico, Costantino Buzi, Marina Melchionna, Paolo Piras, Pasquale Raia, Alessio Veneziano

export_amira *export_amira*

Description

This function exports a list of 3D landmark set in separate files (format landmarkAscii)

Usage

```
export_amira(lista, path)
```

Arguments

lista	list containing 3D landmark sets
path	character: path of the folder where saving the Amira landmark sets

Author(s)

Antonio Profico, Costantino Buzi, Marina Melchionna, Paolo Piras, Pasquale Raia, Alessio Veneziano

Examples

```
x<-c(1:20)
y<-seq(1,3,length=20)
z<-rnorm(20,0.01)
vertices<-cbind(x,y,z)
set<-list(vertices)
example<-export_amira(set,path=tempdir())
```

export_amira.path *export_amira.path*

Description

Convert and save a 3D matrix into a AmiraMesh ASCII Lineset (.am) object

Usage

```
export_amira.path(vertices, filename, Lines = c(1:(dim(vertices)[1] - 1)
  - 1, -1), path)
```

Arguments

vertices	numeric: a kx3 matrix
filename	character: name of the requested output
Lines	numeric: sequence of the vertices that defines the line
path	character: folder path

Author(s)

Antonio Profico, Costantino Buzi, Marina Melchionna, Paolo Piras, Pasquale Raia, Alessio Veneziano

Examples

```
x<-c(1:20)
y<-seq(1,3,length=20)
z<-rnorm(20,0.01)
vertices<-cbind(x,y,z)
export_amira.path(vertices=vertices,filename="example_line",path=tempdir())
```

ext.int.mesh *ext.int.mesh*

Description

This function finds the vertices visible from a set of points of view

Usage

```
ext.int.mesh(mesh, views = 20, dist.sphere = 3, param1 = 2.5,
  param2 = 10, default = TRUE, import_pov, matrix_pov, expand = 1,
  scale.factor, method = "ast3d", start.points = 250,
  num.cores = NULL)
```

Arguments

mesh	object of class mesh3d
views	numeric: number of points of view
dist.sphere	numeric: scale factor. This parameter the distance betweem the barycenter of the mesh and the radius of the sphere used to define set of points of view
param1	numeric: first parameter for spherical flipping (usually ranged from 0.5 to 5, try!)
param2	numeric second paramter for spherical flipping (don't change it!)
default	logical: if TRUE the points of views are defined automatically, if FALSE define the matrix_pov
import_pov	logical: if NULL an interactive 3D plot for the definition of the points of view is returned
matrix_pov	matrix: external set of points of view
expand	numeric: scale factor for the grid for the interactive 3D plot
scale.factor	numeric: scale factor for sphere inscribed into the mesh
method	character: select "a" or "c"
start.points	numeric: number of POVs available
num.cores	numeric: number of cores

Value

position numeric: a vector with vertex number nearest the landmark set

Author(s)

Antonio Profico, Costantino Buzi, Marina Melchionna, Paolo Piras, Pasquale Raia, Alessio Veneziano

---

ext.mesh.rai *ext.mesh.rai*

Description

This function returns a 3D mesh with colours based on the vertices visibile from each point of view

Usage

```
ext.mesh.rai(scans, mesh)
```

Arguments

scans	an ext.int.mesh
mesh	matrix mesh vertex (the same of the ext.int.mesh object)

Author(s)

Antonio Profico, Costantino Buzi, Marina Melchionna, Paolo Piras, Pasquale Raia, Alessio Veneziano

grid_pov *grid_pov*

Description

This function creates a grid for an interactive way to define the set of the points of view

Usage

```
grid_pov(mesh, expand = 1)
```

Arguments

mesh	object of class mesh3d
expand	numeric: scale factor for the grid for the interactive 3D plot

Value

matrice matrix: matrix with the x,y,z coordinates of the points of view

Author(s)

Antonio Profico, Costantino Buzi, Marina Melchionna, Paolo Piras, Pasquale Raia, Alessio Veneziano

human_skull *example dataset*

Description

3D mesh of a human skull

Usage

```
data(human_skull)
```

Author(s)

Antonio Profico, Costantino Buzi, Marina Melchionna, Paolo Piras, Pasquale Raia, Alessio Veneziano

krd1_tooth *example dataset*

Description

3D mesh of a decidous Neanderthal tooth

Usage

```
data(krd1_tooth)
```

Author(s)

Antonio Profico, Costantino Buzi, Marina Melchionna, Paolo Piras, Pasquale Raia, Alessio Veneziano

landmark_frm2amira *landmark_frm2amira*

Description

This function converts the .frm files, from Evan Toolbox, stored in a folder into the format landmarkAscii

Usage

```
landmark_frm2amira(path_folder_frm, path_amira_folder)
```

Arguments

`path_folder_frm`
: character: path of the folder where the .frm files are stored

`path_amira_folder`
: character: path folder to store the landmarkAscii configurations

Author(s)

Antonio Profico, Costantino Buzi, Marina Melchionna, Paolo Piras, Pasquale Raia, Alessio Veneziano

listtoarray *listtoarray convert a list into an array*

Description

listtoarray convert a list into an array

Usage

```
listtoarray(mylist)
```

Arguments

mylist	a list

Value

a kx3xn array with landmark coordinates

Author(s)

Antonio Profico, Costantino Buzi, Marina Melchionna, Paolo Piras, Pasquale Raia, Alessio Veneziano

Lset2D_list *example dataset*

Description

List containing five 2D-landmark configurations acquired along five different anatomical views

Usage

```
data(Lset2D_list)
```

Author(s)

Antonio Profico, Costantino Buzi, Marina Melchionna, Paolo Piras, Pasquale Raia, Alessio Veneziano

Lset3D_array *example dataset*

Description

Array containing a cranial 3D-landmark configuration acquired on a Primate sample

Usage

```
data(Lset3D_array)
```

Author(s)

Antonio Profico, Costantino Buzi, Marina Melchionna, Paolo Piras, Pasquale Raia, Alessio Veneziano

malleus_bone *example dataset*

Description

3D mesh of a human malleus

Usage

```
data(malleus_bone)
```

Author(s)

Antonio Profico, Costantino Buzi, Marina Melchionna, Paolo Piras, Pasquale Raia, Alessio Veneziano

MAs_sets *example dataset*

Description

Landmark configurations of the manual alignments

Usage

```
data(MAs_sets)
```

Author(s)

Antonio Profico, Costantino Buzi, Marina Melchionna, Paolo Piras, Pasquale Raia, Alessio Veneziano

noise.mesh *noise.mesh*

Description

This function adds noise to a mesh

Usage

```
noise.mesh(mesh, noise = 0.025, seed = 123)
```

Arguments

mesh	triangular mesh stored as object of class "mesh3d"
noise	sd deviation to define vertex noise
seed	seed for random number generator

Value

mesh_n a 3D model of class "mesh3d" with noise

Author(s)

Antonio Profico, Costantino Buzi, Marina Melchionna, Paolo Piras, Pasquale Raia, Alessio Veneziano

Examples

```
#load mesh
library(compositions)
library(rgl)
data("SCP1.mesh")
mesh<-SCP1.mesh
#add noise
noised<-noise.mesh(mesh,noise=0.05)
#plot original and mesh with noise added
open3d()
shade3d(mesh,col=3)
shade3d(noised,col=2,add=TRUE)
```

out.inn.mesh *out.inn.mesh*

Description

This function separates a 3D mesh subjected to the ext.int.mesh into two 3D models: the visible mesh and the not visible one

Usage

```
out.inn.mesh(scans, mesh, plot = TRUE)
```

Arguments

scans	an ext.int.mesh
mesh	matrix mesh vertex (the same of the ext.int.mesh object)
plot	logical: if TRUE the wireframe of the mesh with the visible vertices is plotted

Author(s)

Antonio Profico, Costantino Buzi, Marina Melchionna, Paolo Piras, Pasquale Raia, Alessio Veneziano

Examples

```
## Not run:
#CA-LSE tool on Neanderthal tooth
#load a mesh
data(krd1_tooth)
library(rgl)
library(Rvcg)
library(compositions)
ca_lse_krd1<-ext.int.mesh(mesh= krd1_tooth, views=50, param1=3, default=TRUE,
import_pov = NULL,expand=1, scale.factor=1,num.cores = NULL)
vis_inv_krd1<-out.inn.mesh(ca_lse_krd1, krd1_tooth, plot=TRUE)
inv_mesh<-vcgIsolated(vis_inv_krd1$invisible)
open3d()
shade3d(inv_mesh,col=2)
open3d()
shade3d(vis_inv_krd1$visible, col=3)
#CA-LSE tool on human malleus
#load a mesh
data(malleus_bone)
ca_lse_malleus<-ext.int.mesh(mesh= malleus_bone, views=50, param1=3,
default=TRUE, import_pov = NULL, expand=1, scale.factor=1)
vis_inv_malleus<-out.inn.mesh(ca_lse_malleus, malleus_bone, plot=TRUE)
inv_mesh<- vis_inv_malleus$invisible
inv_mesh<-ca_lse_malleus$invisible

#AST-3D tool
```

```
#load a mesh
data(human_skull)
data(endo_set)
ast3d_endocast<-ext.int.mesh(mesh=human_skull, views=50, param1=0.6, default=FALSE,
import_pov = TRUE,expand=1, matrix_pov =endo_set, scale.factor=1,num.cores = NULL)
vis_inv_endo<-out.inn.mesh(ast3d_endocast,human_skull,plot=TRUE)
vis_mesh<-vcgIsolated(vis_inv_endo$visible)
open3d()
shade3d(vis_mesh,col=3)
open3d()
shade3d(vis_inv_endo$invisible, col=2)

## End(Not run)
```

`patches_frm2amira` *patches_frm2amira*

Description

This function converts the .frm files, from Evan Toolbox, stored in a folder into the format landmarkAscii (semilandmark patches)

Usage

```
patches_frm2amira(path_folder_frm, path_amira_folder)
```

Arguments

`path_folder_frm`
: character: path of the folder where the .frm files are stored

`path_amira_folder`
: character: path folder to store the landmarkAscii configurations

Author(s)

Antonio Profico, Costantino Buzi, Marina Melchionna, Paolo Piras, Pasquale Raia, Alessio Veneziano

---

`pov_selecter` *pov_selecter*

Description

Internal function to define the points of view

Usage

```
pov_selecter(mesh, grid, start.points = 250, method = "ast3d")
```

Arguments

mesh	object of class mesh3d
grid	matrix: a 3D grid
start.points	numeric: number of center to be found
method	character: select "a" or "c" for respectively AST-3D and CA-LSE method

Value

selection numeric: positioning vector of the selected points of the grid

Author(s)

Antonio Profico, Costantino Buzi, Marina Melchionna, Paolo Piras, Pasquale Raia, Alessio Veneziano

---

read.amira.dir *read.amira.dir*

Description

This function reads and stores in an array the coordinated allocated in a folder in separate files (format landmarkAscii)

Usage

```
read.amira.dir(path.dir, nland)
```

Arguments

path.dir	character: path of the folder
nland	numeric: number of landmark sampled in Amira

Value

array.set numeric: a kx3xn array with landmark coordinates

Author(s)

Antonio Profico, Costantino Buzi, Marina Melchionna, Paolo Piras, Pasquale Raia, Alessio Veneziano

`read.amira.set` *read.amira.set*

Description

This function converts a landmarkAscii file set in a kx3x1 array

Usage

```
read.amira.set(name.file, nland)
```

Arguments

`name.file`	character: path of a landmarkAscii file
`nland`	numeric: number of landmark sampled in Amira, if is set on "auto" it will be automatically recognized

Value

array.set numeric: a kx3x1 array with landmark coordinates

Author(s)

Antonio Profico, Costantino Buzi, Marina Melchionna, Paolo Piras, Pasquale Raia, Alessio Veneziano

`read.path.amira` *read.path.amira*

Description

This function extracts and orders the coordinate matrix from a surface path file from Amira

Usage

```
read.path.amira(path.name)
```

Arguments

`path.name`	character: path of surface path .ascii extension file

Value

data numeric: a kxd matrix with xyz coordinates

Author(s)

Antonio Profico, Costantino Buzi, Marina Melchionna, Paolo Piras, Pasquale Raia, Alessio Veneziano

repmat *repmat*

Description

This function repeats copies of a matrix

Usage

```
repmat(X, m, n)
```

Arguments

X	numeric: a matrix
m	numeric: number of times to repeat the X matrix in row and column dimension
n	numeric: repetition factor for each dimesion

Value

matrice: repeated matrix

Author(s)

Antonio Profico, Costantino Buzi, Marina Melchionna, Paolo Piras, Pasquale Raia, Alessio Veneziano

RMs_sets *example dataset*

Description

Array containing the landmark coordinates of the reference sample for Digital Alignment Tool example

Usage

```
data(RMs_sets)
```

Author(s)

Antonio Profico, Costantino Buzi, Marina Melchionna, Paolo Piras, Pasquale Raia, Alessio Veneziano

SCP1.mesh *example dataset*

Description

Mesh of the Saccopastore 1 Neanderthal skull

Usage

```
data(SCP1.mesh)
```

Author(s)

Antonio Profico, Costantino Buzi, Marina Melchionna, Paolo Piras, Pasquale Raia, Alessio Veneziano

sinus_set *example dataset*

Description

POVs sampled inside the maxillary sinus cavity

Usage

```
data(sinus_set)
```

Author(s)

Antonio Profico, Costantino Buzi, Marina Melchionna, Paolo Piras, Pasquale Raia, Alessio Veneziano

SM_set *example dataset*

Description

Landmark configuration associated to the starting model

Usage

```
data(SM_set)
```

Author(s)

Antonio Profico, Costantino Buzi, Marina Melchionna, Paolo Piras, Pasquale Raia, Alessio Veneziano

`spherical.flipping` *spherical.flipping*

Description

Internal spherical flippping function

Usage

```
spherical.flipping(C, mesh, param1, param2)
```

Arguments

`C`	numeric: coordinates of the point of view
`mesh`	object of class mesh3d
`param1`	numeric: first parameter for spherical flipping (usually ranged from 0.1 to 3, try!)
`param2`	numeric second paramter for spherical flipping (don't change it!)

Author(s)

Antonio Profico, Costantino Buzi, Marina Melchionna, Paolo Piras, Pasquale Raia, Alessio Veneziano

---

`trasf.mesh` *trasf.mesh*

Description

This function centers a mesh on the barycenter coordinates

Usage

```
trasf.mesh(mesh, barycenter)
```

Arguments

`mesh`	a 3D mesh of class "mesh3d"
`barycenter`	numeric: coordinates of the center

Value

mesh a 3D mesh of class "mesh3d"

Author(s)

Antonio Profico, Costantino Buzi, Marina Melchionna, Paolo Piras, Pasquale Raia, Alessio Veneziano

twodvarshape *twodvarshape Calculates the shape variation associated to a value of PC scores associated to a specific 2D view*

Description

twodvarshape Calculates the shape variation associated to a value of PC scores associated to a specific 2D view

Usage

```
twodvarshape(twodviews_ob, scores, PC, view)
```

Arguments

twodviews_ob	object from twodviews()
scores	numeric: the values of the PC scores for which the visualization is called
PC	PC chosen
view	numeric: 2D set to be used for shape variation visualization

Value

mat matrix of 2D coordinates associated to the called shape variation

Author(s)

Antonio Profico, Costantino Buzi, Marina Melchionna, Paolo Piras, Pasquale Raia, Alessio Veneziano

Examples

```
#load the 2D primate dataset
data("Lset2D_list")
#combine the 2D datasets and PCA
combin2D<-twodviews(Lset2D_list,scale=TRUE,vector=c(1:5))
#calculate the shape variation associated to the negative extreme value of PC1
min_PC1<-twodvarshape(combin2D,min(combin2D$PCscores[,1]),1,5)
plot(min_PC1,asp=1)
#calculate the shape variation associated to the positive extreme value of PC1
max_PC1<-twodvarshape(combin2D,max(combin2D$PCscores[,1]),1,5)
plot(max_PC1,asp=1)
```

twodviews *twodviews Calculates the PCscores of the 2Dcomp matrix. an array in a list storing each element of the third dimension of the array (specimen) as element of the list*

Description

twodviews Calculates the PCscores of the 2Dcomp matrix. an array in a list storing each element of the third dimension of the array (specimen) as element of the list

Usage

```
twodviews(twodlist, scale = TRUE, vector = c(1:5))
```

Arguments

twodlist	a list containing the landmark configurations of each anatomical view stored as separated lists
scale	logical: TRUE for shape-space, FALSE for form-space
vector	numeric vector: defines which views are to be used

Value

PCscores PC scores

PCs Pricipal Components (eigenvector matrix)

Variance table of the explained variance by the PCs

size vector containing the Centroid Size of each configuration

mshapes a list containing the mean shape of each 2D configuration

dims number of landmarks of each 2D configuration

twodlist the list used as input

Author(s)

Antonio Profico, Costantino Buzi, Marina Melchionna, Paolo Piras, Pasquale Raia, Alessio Veneziano

Examples

```
library(Morpho)
#load the 2D primate dataset
data("Lset2D_list")
#combine the 2D datasets and PCA
combin2D<-twodviews(Lset2D_list,scale=TRUE,vector=c(1:5))
#plot of the first two Principal Components
plot(combin2D$PCscores)
text(combin2D$PCscores,labels=rownames(combin2D$PCscores))
#load the 3D primate dataset
```

```
data("Lset3D_array")
#GPA and PCA
GPA_3D<-procSym(Lset3D_array)
#plot of the first two Principal Components
plot(GPA_3D$PCscores)
text(GPA_3D$PCscores,labels=rownames(GPA_3D$PCscores))
```

---

volendo *volendo*

Description

Calculate the volume of a mesh by using a voxel-based method

Usage

```
volendo(mesh, alpha_vol = 100, ncells = 1e+05)
```

Arguments

mesh	object of class mesh3d
alpha_vol	numeric: alpha shape for construction external concave hull
ncells	numeric: approximative number of cell for 3D grid construction

Value

vol numeric: volume of the mesh expressed in cc

Author(s)

Antonio Profico, Costantino Buzi, Marina Melchionna, Paolo Piras, Pasquale Raia, Alessio Veneziano

Examples

```
## Not run:
#load the human skull
library(rgl)
data(human_skull)
sapendo<-endomaker(human_skull,param1_endo = 1.0,vol=FALSE, num.cores=NULL)
volsap<-volendo(sapendo$endocast)

## End(Not run)
```

yoda_set *example dataset*

Description

Landmark set on Yoda

Usage

```
data(yoda_set)
```

Author(s)

Antonio Profico, Costantino Buzi, Marina Melchionna, Paolo Piras, Pasquale Raia, Alessio Veneziano

yoda_sur *example dataset*

Description

Mesh of Yoda

Usage

```
data(yoda_sur)
```

Author(s)

Antonio Profico, Costantino Buzi, Marina Melchionna, Paolo Piras, Pasquale Raia, Alessio Veneziano

Index