Package 'molaR'

January 27, 2023

Title Dental Surface Complexity Measurement Tools

Version 5.3

Description Surface topography calculations of Dirichlet's normal energy, relief index, surface slope, and orientation patch count for teeth using scans of enamel caps.
Importantly, for the relief index and orientation patch count calculations to work, the scanned tooth files must be oriented with the occlusal plane parallel to the x and y axes, and perpendicular to the z axis. The files should also be simplified, and smoothed in some other software prior to uploading into R.

Depends R (>= 2.10), alphahull, rgl, Rvcg, pracma

License GPL

LazyData true

Encoding UTF-8

Suggests knitr, rmarkdown, rglwidget

VignetteBuilder knitr

NeedsCompilation no

Author James D. Pampush [aut, cre, cph],
Paul E. Morse [aut, cph],
Alexander Q. Vining [aut, cph],
Edward Fuselier [aut, cph]

Maintainer James D. Pampush <jdpampush@gmail.com>

RoxygenNote 7.2.3

Repository CRAN

Date/Publication 2023-01-27 00:30:02 UTC

R topics documented:

`ARC` *Calculate several measures of Area Relative Curvature*

Description

A function that calculates the average slope over a tooth or some other 3D surface

Usage

```
ARC(plyFile, BoundaryDiscard = "Vertex", Range = c(0.01, 0.99))
```

Arguments

`plyFile`	An object of classes 'mesh3d' and 'shape3d' with calculated normals
`BoundaryDiscard`	String indicating how to handle the exclusion of boundary faces. Default of Vertex excludes faces which have at least 1 vertex on the boundary
`Range`	A pair of values which set lower and upper outlier exclusions.

Details

The function requires an object created by reading in a ply file.

This function calculates Area Relative Curvature, as described by Guy et al. (2013)

Examples

```
arc_output <- ARC(Tooth)
summary(arc_output)
```

ARC3d *Plot Area Relative Curvature on Tooth Mesh*

Description

a molaR surface plotting function

Usage

```
ARC3d(
  ARC_object,
  main = "",
  cex.main = 2,
  cex = 1,
  colors = c("darkblue", "blue", "powderblue", "gray", "gray", "tan", "orange",
    "darkorange1"),
  fieldofview = 0,
  legend = T
)
```

Arguments

ARC_object	An object that stores the output of the ARC() function
main	string indicating plot title. Defaults to empty
cex.main	numeric value setting the relative size of the plot title,
cex	numeric value setting the relative size of the legend, default=1
colors	a concatenated string of colors for plotting different values of positive and negative curvature.
fieldofview	Passes an argument to par3d() changing the field of view (in degrees) of the resulting 3D plot
legend	Logical indicating whether or not a legend shold be displayed. Default=T

Details

This function creates a surface map of the discarded DNE faces. DNE calculations typically discard the top 1 tenth of one percent of faces, associated with extreme pockets and broken parts of surfaces. DNE calculations also typically discard the boundary faces from the calculation, either on the basis of 2 vertices on the boundary, or at least one vertext on the boundary. concaveCol defaults to gray and therefore is turned off. When an alternative color is provided, the function will identify the the areas of the tooth that are concave vs convex.

Details of the other function arguments can be found in the DNE3d() description and identical terms are organized to function the same way.

Examples

```
ARC_output <- ARC(Tooth)
ARC3d(ARC_output)
```

`Check2D` *Plot 2D footprint and footprint triangle points to check for erros in 2D calculation*

Description

This function will plot the points used for the 2D footprint area calculation. This is meant to be a visual checking mechanism to ensure that there are no 'extra' triangles within the footprint erroneously adding to the total 2D area of the footprint. If a user finds extra points within the boundaries of the footprint, they should assume that the alpha value used for the RFI calculation was too small, and they are getting a 2D footprint calculation which was too large.

Usage

```
Check2D(RFI_Output, FootColor = "red", TriPointsColor = "black")
```

Arguments

`RFI_Output`	An object that stores the output of the RFI function
`FootColor`	changes color of the 2D surface footprint
`TriPointsColor`	color for the points of the footprint triangles

Details

This function will plot the points used for the 2D footprint area calculation. This is meant to be a visual checking mechanism to ensure that there are no 'extra' triangles within the footprint erroneously adding to the total 2D area of the footprint. If a user finds extra points within the boundaries of the footprint, they should assume that the alpha value used for the RFI calculation was too small, and they are getting a 2D footprint calculation which was too large.

Examples

```
RFI_output <- RFI(Tooth, alpha=0.5)
Check2D(RFI_output)
```

DNE *Calculate Dirichlet normal energy of a surface*

Description

A function that calculates Dirichlet normal energy following the method of Bunn et al. (2011) Comparing Dirichlet normal surface energy of tooth crowns, a new technique of molar shape quantification for dietary inference, with previous methods in isolation and in combination. Am J Phys Anthropol 145:247-261 doi: 10.1002 ajpa.21489

Usage

```
DNE(plyFile, outliers = 0.1, kappa = 0, BoundaryDiscard = "Vertex", oex = "c")
```

Arguments

plyFile	An object of classes 'mesh3d' and 'shape3d' with calculated normals
outliers	The percentile of Dirichlet energy density values to be excluded defaults to top 0.1 percent
kappa	An integer value of mean curvature to define concave vs convex faces
BoundaryDiscard	String indicating how to handle the exclusion of boundary faces. Default of Vertex excludes faces which have at least 1 vertex on the boundary
oex	String indicating outlier exclusion principle. Defaults to 'c', which combines all convex and concave faces and removes the percentage of outliers defined by outliers. See details.

Details

The function requires an object created by reading in a ply file.

Dirichlet normal energy is calculated on meshes that represent specimen surfaces and have already been simplified and pre-smoothed in a 3D data editing program.

In the default settings, the function seeks to discard boundary faces. This can be changed by adjusting the BoundaryDiscard argument to 'None' which will not discard any faces on the boundary. Further, there are two ways of excluding boundary faces. Either if they have a leg on the boundary by setting BoundaryDiscard='Leg' or by excluding any face which has a vertex on the boundary with BoundaryDiscard='Vertex'. The function defaults to remove the top 0.1 percent of calculated energy densities as outliers. Mesh orientation does not affect this calculation.

Faces are labeled as concave or convex on the basis of the kappa value, which defaults to 0. Each face is assigned a kappa value, which describes the the localized degree of convergence or divergence among the three vertex normals on each face. Faces with positive kappa values have vertex normals that are divergent. Faces with negative kappa values possess vertex normals that are convergent. Users can adjust the kappa value to redefine areas of the tooth assigned to the convex or concave bin.

The mode of Outlier exclusion can be modified with the `oex` argument. The default, `oex='c'` considers the Dirichlet energy density values of all faces on the surface and removes those in the top percentile defined by `outliers`, regardless of the convexity or concavity bins. The alternative, `oex='s'`, divides the surface into concave and convex portions, then removes the percentile defined by `outliers` from each of these subsets before calculating total surface DNE.

Examples

```
DNE_output <- DNE(Tooth)
summary(DNE_output)
```

`DNE3d` *Plot results of a DNE analysis of a surface*

Description

a molaR surface plotting function

Usage

```
DNE3d(
  DNE_File,
  setMax = 0,
  logColors = TRUE,
  signColor = TRUE,
  main = "",
  cex = 1,
  cex.main = 2,
  legend = TRUE,
  leftOffset = 1,
  fieldofview = 0,
  fileName = NA,
  binary = FALSE
)
```

Arguments

`DNE_File`	An object that stores the output of the `DNE()` function
`setMax`	User-defined upper range for plotting color scheme, see Details
`logColors`	Logical that log transforms the color scheme
`signColor`	Logical indicating whether or not to plot by concavity vs convexity. Plotting by curve orientation is the default.
`main`	String indicating plot title
`cex`	Numeric setting the relative size of the legend
`cex.main`	Numeric setting the size of the title

legend	Logical indicating whether or not a legend should be displayed
leftOffset	Numeric between -1 and 1 setting the amount of offset for the plotted surface to the left. Larger values push surface farther to right.
fieldofview	Passes an argument to par3d() changing the field of view (in degrees) of the resulting 3D plot
fileName	String indicating a name to save the plotted surface to as a *.ply file; default of 'NA' will not save a file
binary	Logical indicating whether or not the saved surface plot should be binary, passed to vcgPlyWrite()

Details

This function creates a heat map on the mesh surface corresponding to the Dirichlet energy density of each face calculated by the DNE() function. Hottest colors represent highest normal energy values.

Dirichlet energy densities for the faces of a mesh surface tend to be positively skewed, with a small proportion of the faces contributing most of the total energy for the surface. When logColors is enabled, the function colorizes based on the log-transformed Dirichlet energy densities, allowing for finer resolution between faces near the mode of the energy per face distribution. Disabling logColors will display the un-transformed Dirichlet energy densities.

The legend will update to reflect the other arguments chosen by the user. By default, the function sets the lowest Dirichlet energy density calculated among all faces to a cool color and the absolute highest normal energy calculated among all faces to a hot color, and then colors the remaining faces on a continuous color spectrum between these two end points using either absolute or log transformed Dirichlet energy density values (depending on the status of logColors). Since the scale is relative to the energies of the input surface, visual comparisons cannot directly be made between multiple plots of different surfaces.

The setMax argument allows users to define the maximum of the plotting color scheme for use across multiple plots. This enables the direct comparison of different surfaces to one another with red equal to the user-defined maximum and a cool color equal to the minimum. The user should choose a reasonable upper bound for the maximum. setMax will not accept negative values. If there are faces with Dirichlet normal energy values higher than the setMax value, these faces are marked with the highest possible color.

The logical signColor colors the surface with two separate gradients, one for the convex and one for the concave faces (curvature sign). By default, the plot now makes this distinction.

A title can be added to the plot by supplying a character string to the main argument. Title and legend size are controlled with the cex argument, analogous to that in the default R graphics device.

The leftOffset value sets how far to the left the surface will plot, intended to help avoid overlap with the legend. Value of 0 will center the surface and should be invoked if the legend argument is disabled. Higher values will push the surface farther left and negative values will push it to the right. It is recommended that these values be restricted between -1 and 1 to avoid plotting the surface outside of the rgl window.

fieldofview is set to a default of 0, which is an isometric projection. Increasing it alters the degree of parallax in the perspective view, up to a maximum of 179 degrees (see rgl::par3d()).

The plotted, colorized surface can be saved as a *.ply to the working directory by changing the `fileName` argument from `NA` to a string (e.g., "DNEPlot"). The resultant ply file can be opened and manipulated in other 3D visualizing programs, such as MeshLab, but will **NOT** retain its legend (a background of the plotting window). To retain the legend, the user is encouraged to utilize the function 'snapshot3d()' in the rgl package. (see `rgl::rgl.snapshot()`) The `binary` argument saves a file in ascii format by default, which is supported by more 3D visualization software than is binary. However, binary files will be considerably smaller.

Examples

```
DNE_output <- DNE(Tooth)
DNE3d(DNE_output)
```

`DNE3dDiscard` *Plot advanced results of a DNE surface analysis*

Description

a molaR surface plotting function

Usage

```
DNE3dDiscard(
  DNE_File,
  baseCol = "gray",
  boundCol = "red",
  outlierCol = "lawngreen",
  concaveCol = baseCol,
  main = "",
  cex = 1,
  cex.main = 2.5,
  legend = T,
  leftOffset = 1,
  fieldofview = 0,
  fileName = NA,
  binary = FALSE
)
```

Arguments

`DNE_File`	An object that stores the output of the `DNE()` function
`baseCol`	Base color for typical face on surface. Default is gray
`boundCol`	Color for the boundary faces discarded from the DNE calculation. Default is red.
`outlierCol`	Color for the faces discarded as outliers from the DNE calculation. Default is lawngreen

concaveCol	Color of the Concave faces on the surface. When left in default concave faces remain undistinguished on the plotted surface and are colored the same as baseCol.
main	string indicating plot title. Defaults to empty
cex	numeric value setting the relative size of the legend, default=1
cex.main	numeric value setting the relative size of the plot title,
legend	Logical indicating whether or not a legend shold be displayed. Default=T
leftOffset	numeric value between -1 and 1 setting the degree of offset for the plotted surface to the left. Larger values set further to right. Default=1
fieldofview	Passes an argument to par3d() changing the field of view (in degrees) of the resulting 3D plot
fileName	String indicating a name to save the plotted surface to as a *.ply file; default of 'NA' will not save a file
binary	Logical indicating whether or not the saved surface plot should be binary, passed to vcgPlyWrite()

Details

This function creates a surface map of the discarded DNE faces. DNE calculations typically discard the top 1 tenth of one percent of faces, associated with extreme pockets and broken parts of surfaces. DNE calculations also typically discard the boundary faces from the calculation, either on the basis of 2 vertices on the boundary, or at least one vertext on the boundary. concaveCol defaults to gray and therefore is turned off. When an alternative color is provided, the function will identify the the areas of the tooth that are concave vs convex.

Details of the other function arguments can be found in the DNE3d() description and identical terms are organized to function the same way.

Examples

```
DNE_output <- DNE(Tooth)
DNE3dDiscard(DNE_output)
```

DNEbar *Plot advanced results of a DNE surface analysis*

Description

a molaR plotting function

Usage

```
DNEbar(
  DNE_File,
  main = "",
  convexCol = "hotpink",
  concaveCol = "deepskyblue",
```

```
  type = "both",
  legendPos = "topright",
  legendInset = 0,
  las = 1,
  names.arg = "",
  cex.names = 1
)
```

Arguments

DNE_File	An object that stores the output of the DNE() function
main	User's title for plot.
convexCol	Color for the convex DNE total. Default='hotpink'
concaveCol	Color for the concave DNE total. Default='deepskyblue'
type	string to determine what parameters to plot. Default=both and both concave and convex DNE totals will be plotted in stacked bar plot. See details
legendPos	string to determine location of the legend. Default='topright' see details.
legendInset	numeric value determining how far to inset the legend from plot boarder. Default=0
las	logical indicating orientation of the x-axis labels for each bar plot. Enter either 1 or 2.
names.arg	concatenated string of surface names for labels. If none supplied function will pull names from the object itself.
cex.names	Font size for the bar labels. Default is 1.

Details

This function creates a stacked barplot of DNE values. It colors them according to curve orientation, which is defined by the kappa parameter in DNE() function. If multiple DNE objects are grouped together the barplot will return a set. When employed on a single DNE object this will return a single stacked bar.

The argument type accepts either 'Concave' or 'Convex' to plot only concave or convex DNE totals respectively. Default=NA and results in both totals being plotted in stacked barplot.

The argument legendPos is a string that determines the position of the legend. Default='topright' but will accept any of the following keywords: 'bottomright', 'bottom', 'bottomleft', 'left', 'topleft', 'top', 'topright', 'right', or 'center'.

Examples

```
DNEs <- list()
DNEs$Tooth <- DNE(Tooth)
DNEs$Hills <- DNE(Hills)
DNEbar(DNEs)
```

DNEDensities *Plot advanced results of a DNE surface analysis*

Description

Plot advanced results of a DNE surface analysis

Usage

```
DNEDensities(
  DNE_File,
  main = "",
  type = "DNE",
  legendPos = "topright",
  convexCol = "hotpink",
  concaveCol = "deepskyblue"
)
```

Arguments

DNE_File	An object that stores the output of the DNE function
main	User's title for plot. Default is blank
type	string determining which density plots to make. Default
legendPos	string to determine location of the legend. Default='topright' see details. is to plot DNE face densities. Alternatively can plot face areas with 'area'
convexCol	Color for the convex density polygon, Default='hotpink'
concaveCol	Color for the concave density polygon, Default='deepskyblue'

Details

This function creates a set of overlapping density plots of two potential types. The user can plot overlapping density plots that sort the surface into concave and convex portions for plotting. The function will default to plotting DNE density values, however density of face surface areas sorted into concave and convex portions of the surface can be plotted by calling `type='area'`. Colors can be customized by altering the `convexCol` and `concaveCol` arguments.

Examples

```
DNE_output <- DNE(Tooth)
DNEDensities(DNE_output)
```

DNEpie *Plot advanced results of a DNE surface analysis*

Description

Plot advanced results of a DNE surface analysis

Usage

```
DNEpie(
  DNE_File,
  main = "",
  type = "area",
  convexCol = "hotpink",
  concaveCol = "deepskyblue"
)
```

Arguments

DNE_File	An object that stores the output of the DNE() function
main	User's title for the plot
type	string determine which parameters to plot. Default='DNE' also accepts 'area' to plot pie charts of the area.
convexCol	Color for the portion of the pie chart representing convex contribution. Default='hotpink'. Accepts any color keyword.
concaveCol	Color for the portion of the pie chart representing concave contribution. Default='deepskyblue'. Accepts any color keyword.

Details

This function creates a pie chart of the total area or DNE of the surface originating from the concave or convex portions of the surface. The function defaults to plotting surface area, however, relative proportion of total DNE from the concave and convex portions of the surface can be plotted by calling type='DNE'. Colors can be customized by altering the convexCol and concaveCol arguments.

Examples

```
DNE_output <- DNE(Tooth)
DNEpie(DNE_output)
```

Hills *Hills surface mesh*

Description

Sample mesh created with the formula: 'z=3cos(x/2)+3sin(y/2)

A triangular mesh representing a sine-cosine plane - called by data(Hills)

Usage

```
data(Hills)
```

Format

An object of class "mesh3d"

Hills: triangular mesh representing a sine-cosine plane.

Examples

```
data(Hills)
DNE_Output <- DNE(Hills)
DNE3d(DNE_Output)
```

molaR_Batch *Run molaR analyses on a batch of specimens*

Description

A function that automates molaR analyses on multiple specimens. Several different analyses can be performed on each surface, with specifications for analysis parameters.

Usage

```
molaR_Batch(
  pathName = getwd(),
  fileName = "molaR_Batch.csv",
  DNE = TRUE,
  RFI = TRUE,
  OPCr = TRUE,
  OPC = FALSE,
  Slope = TRUE,
  details = TRUE,
  parameters = TRUE,
  ...
)
```

Arguments

pathName	The path to the folder containing all ply surfaces to be analyzed. Defaults to the working directory.
fileName	Name for the output .csv file containing results and parameters
DNE	Logical indicating whether or not to perform the DNE calculation
RFI	Logical indicating whether or not to perform the RFI calculation
OPCr	Logical indicating whether or not to perform the OPCr calculation
OPC	Logical indicating whether or not to perform the OPC calculation
Slope	Logical indicating whether or not to perform the Slope calculation
details	Logical indicating whether or not to save additional output from some of the topographic analyses
parameters	Logical indicating whether or not to save the list of analysis parameters used in the batch run
...	Additional arguments passed to the topographic analysis functions. See Details.

Details

This function allows a user to set the analyses from molaR they want to run on a batch of ply files. Output is saved to a csv file. By default, the batch function will perform specified analyses on all ply files in the working directory. A different folder can be specified with pathName. Output saves as .csv to the folder that contains the analyzed ply files.

Any of the default arguments of the various topographic analysis functions can be modified for the batch by specifying them when calling molaR_Batch, e.g., the DNE kappa value can be changed to 'X' by specifying kappa = X. Users are **strongly** encouraged to review the documentation for DNE(), RFI(), OPCr(), OPC(), and Slope() and to understand the effects of alterations before making changes. A recommended practice for analyzing RFI in a batch of specimens is to enable findAlpha = TRUE given that the ideal alpha value is likely to vary among different specimens. However, this will increase calculation time (see documentation for RFI).

By default, the batch output will retain some additional details of the analysis. These include, in the case of DNE: convex and concave DNE values, convex and concave surface areas; in the case of RFI: 3D and 2D surface areas and analysis alpha values; in the case of OPCr: the surface OPC value calculated at each rotation; and in the case of OPC: the patch count for each bin. These results will be discarded and only the final result of each topographic analysis will be retained if details = FALSE.

The function will save a list of all parameters used in all batch analyses to the output .csv file, below the results. This can be suppressed with parameters = FALSE, but is recommended as a check on how analyses were performed when returning to results in the future. If the function is assigned to an object in R, the parameters are not included in the resultant data.frame, but will still be included in the .csv file by default.

Note that batch processing updates will not display by default if using RGui for Windows. Disable Misc -> Buffered output (Ctrl+W) if you wish to view batch processing progress in RGui for Windows.

molaR_Clean *Clean up problem ply files*

Description

Function will remove floating verticies, and faces with zero area. These can cause issues when using molaR's primary functions of DNE, RFI, and OPC

Usage

```
molaR_Clean(plyFile, cleanType = "Both", verbose = TRUE)
```

Arguments

plyFile	An object of classes 'mesh3d' and 'shape3d'
cleanType	String with three arguments defining what to clean: Vertices, Faces, or Both. Defaults to Both.
verbose	Logical indicating if the function should report changes to ply

Details

This function cleans up problematic ply files. Some smoothed files will have faces of zero area, or floating vertices. DNE and OPC cannot be calculated on these files. Running the plys through this function will allow those calculations to be made.

Examples

```
Tooth <- molaR_Clean(Tooth)
```

OPC *Calculate orientation patch count of a surface*

Description

A function that bins patches of a mesh surface that share general orientation and sums the number of unique patches given certain parameters Modified into 3D from the original 2.5D method described by Evans et al. (2007) High-level similarity of dentitions in carnivorans and rodents. Nature 445:78-81 doi: 10.1038/nature05433

Usage

```
OPC(plyFile, rotation = 0, minimum_faces = 3, minimum_area = 0)
```

Arguments

plyFile	An object of classes "mesh3d" and "shape3d" with calculated vertex normals
rotation	Rotates the file in degrees about the center vertical axis
minimum_faces	Minimum number of ply faces required for a patch to be counted towards the total patch count
minimum_area	Minimum proportion (100%=1.0) of total surface area a patch must occupy to be counted towards the total patch count

Details

The function requires a mesh object created by reading in a ply file utilizing either the, vcgPlyRead function.

Orientation patch count is calculated on meshes that represent specimen surfaces and have already been downsampled to 10,000 faces and pre-smoothed in a 3D data editing program. Alignment of the surface will have a large effect on patch orientation and must be performed in a 3D data editing program such as Avizo. The occlusal surface of the specimen must be made parallel to the X- and Y-axes and perpendicular to the Z-axis.

The default for minimum_faces is to ignore patches consisting of two or fewer faces on the mesh. Changing the minimum_area value will disable minimum_faces.

Examples

```
OPC_output <- OPC(Tooth)
summary(OPC_output)
```

OPC3d *Plot results of OPC analysis of a surface*

Description

A function that produces a three-dimensional rendering of face orientation on a surface. The OPC function will identify the orientations of mesh faces and assign them to patches. It must be performed prior to using the OPC3d function.

Usage

```
OPC3d(
  OPC_File,
  binColors = hsv(h = (seq(10, 290, 40)/360), s = 0.9, v = 0.85),
  patchOutline = FALSE,
  outlineColor = "black",
  maskDiscard = FALSE,
  legend = TRUE,
  main = "",
  cex = 1,
```

```
  scaleLegend = FALSE,
  legendTextCol = "black",
  legendLineCol = "black",
  leftOffset = 1,
  fieldofview = 0,
  fileName = NA,
  binary = FALSE
)
```

Arguments

OPC_File	An object that stores the output of the OPC() function
binColors	Allows the user to define the fill colors for each directional bin
patchOutline	Logical whether or not to outline the patches
outlineColor	Parameter defining the patch outline color
maskDiscard	Logical indicating whether or not to mask (in black) the patches excluded from the OPC value
legend	Logical indicating whether or not a legend should be displayed
main	String indicating plot title
cex	Numeric setting the relative size of the legend and title
scaleLegend	Logical indicating if legend bins should scale to patch counts
legendTextCol	Parameter defining color for the legend text
legendLineCol	Parameter defining the color for the legend lines
leftOffset	Numeric between -1 and 1 setting the amount of offset for the plotted surface to the left. Larger values push surface farther to right.
fieldofview	Passes an argument to par3d() changing the field of view in degrees of the resulting surface plot
fileName	String indicating a name to save the plotted surface to as a *.ply file; default of 'NA' will not save a file
binary	Logical indicating whether or not the saved surface plot should be binary, passed to vcgPlyWrite()

Details

This function will assign a uniform color to all faces on the mesh surface that share one of the orientation bins identified by the OPC function. The function returns a colored mesh so that patches can be visually inspected.

binColors will support any vector of colors, in any coloration scheme. Default draws from the HSV color space to evenly space color information, however the user can supply a list of RGB values or character strings in place. If there are fewer colors than directional bins, remaining bins will default to white.

A title can be added to the plot by supplying a character string to the main argument. Title and legend size are controlled with the cex argument, analogous to that in the default R graphics device.

Several legend plotting options are available. The default legend shape is a circular pie with sectors indicating the orientation of directional bins, shaded according to the color scheme in `binColors`. By setting `scaleLegend = TRUE`, the legend sectors will scale proportionally to the number of patches in each directional bin. The legend text and line colors can be customized with `legendTextCol` and `legendLineCol`, which both default to black.

The `leftOffset` value sets how far to the left the surface will plot, intended to help avoid overlap with the legend. Value of 0 will center the surface and should be invoked if the `legend` argument is disabled. Higher values will push the surface farther left and negative values will push it to the right. It is recommended that these values be restricted between -1 and 1 to avoid plotting the surface outside of the rgl window.

`fieldofview` is set to a default of 0, which is an isometric projection. Increasing it alters the degree of parallax in the perspective view, up to a maximum of 179 degrees (see `rgl::par3d()`).

The plotted, colorized surface can be saved as a *.ply to the working directory by changing the `fileName` argument from `NA` to a string (e.g., "OPCPlot"). The resultant ply file can be opened and manipulated in other 3D visualizing programs, such as MeshLab, but will **NOT** retain its legend (a background of the plotting window). To retain the legend, the user is encouraged to utilize the `rgl::snapshot3d()` function. The `binary` argument saves a file in ascii format by default, which is supported by more 3D visualization software than is binary. However, binary files will be considerably smaller.

Examples

```
OPC_output <- OPC(Tooth)
OPC3d(OPC_output)
```

`OPCbinareas` *Visualize surface area distribution into separate OPC orientation bins.*

Description

This function will make either a bar plot or pie chart showing the surface area assigned to each OPC orientation bin.

Usage

```
OPCbinareas(
  OPC_File,
  main = "",
  binColors = hsv(h = (seq(10, 290, 40)/360), s = 0.9, v = 0.85),
  type = "bar"
)
```

Arguments

`OPC_File`	An object that stores the output of an OPC analysis using `OPC()`.
`main`	Title for plot.

binColors	Allows the user to define the fill colors for each directional bin. see details
type	String argument to determine type of plot, either bar' or 'pie'. Default is set to 'bar'

Details

This function will create either bar or pie chats visualising the distribution of surface area into each of the OPC orientation bins. Colors can be customized but are meant to match the default settings in the OPC3d() function.

Examples

```
OPC_Object <- OPC(Tooth)
OPCbinareas(OPC_Object)
```

OPCr *Calculate average orientation patch count after several rotations*

Description

A function that calls OPC iteratively after rotating mesh a selected number of degrees around the Z-axis following Evans and Jernvall (2009) Patterns and constraints in carnivoran and rodent dental complexity and tooth size. J Vert Paleo 29:24A

Usage

```
OPCr(plyFile, steps = 8, stepSize = 5.625, minimum_faces = 3, minimum_area = 0)
```

Arguments

plyFile	An object of classes 'mesh3d' and 'shape3d' with calculated normals
steps	Number of iterations to run the OPC() function on the mesh
stepSize	Amount of rotation (in degrees) about the Z-axis to adjust mesh surface by between each iteration of OPC()
minimum_faces	Argument to pass to the OPC() function
minimum_area	Argument to pass to the OPC() function

Details

The function requires a mesh object created by reading in a ply file utilizing either the, vcgPlyRead function.

Default number of steps is 8, with a stepSize of 5.625 degrees, following the original published definition of OPCR.

See the details for the OPC function for more information about preparing mesh surfaces and the effects of minimum_faces and minimum_area.

OPCr_Example1 — *OPCr_Example1 - object created by OPCr function used as an example.*

Description

This object is needed to pass the CRAN upload requirements and still keep the vignette.

Format

`OPCr_Example1`: molaR produced object.

OPCr_Example2 — *OPCr_Example2 - object created by OPCr function used as an example.*

Description

This object is needed to pass the CRAN upload requirements and still keep the vignette.

Format

`OPCr_Example2`: molaR produced object.

plyPlaneCut — *Cut a PLY Mesh Along a Specified Plane*

Description

`plyPlaneCut` permits several different approaches for specifying a cutting plane and returns either a portion of the original mesh from one side of the plane, or both portions from each side of the plane stored as separate list elements.

Usage

```
plyPlaneCut(
  plyFile,
  axis = "Z",
  vertIndex = NA,
  keepBoth = FALSE,
  plane = NA,
  col = "rainbow",
  flipAxis = FALSE,
  displayNew = TRUE
)
```

Arguments

plyFile	An object of class 'mesh3d'.
axis	String indicating the axis plane on which to cut the mesh. May be 'X', 'Y', or 'Z', Defaults to 'Z'. Ignored if plane is specified, see details.
vertIndex	Numeric index of a mesh vertex to define clipping plane. Ignored if plane is specified, see details.
keepBoth	Logical indicating if both sides of the cut mesh should be returned, defaults to FALSE. If TRUE (and the cutting plane intersects the mesh), the function output is a list containing meshA and meshB representing the two portions.
plane	Requires four numeric values specifying the coordinates of the plane normal (*a, b, c*) and the "offset" (*d*). Overrides input for axis and vertIndex, see details.
col	Vector indicating the color for vertex drawing when interactively choosing cutting plane. Defaults to "rainbow", a magenta-to-red color ramp along the specified axis.
flipAxis	Logical indicating whether or not to reverse the output about the normal of the plane, defaults to FALSE.
displayNew	Logical indicating whether or not to display the function results when a value is supplied to either vertIndex or plane, defaults to TRUE.

Details

plyPlaneCut draws a cutting plane using the parametrization $ax + by + cz + d = 0$ (Hesse normal form), wherein $<a, b, c>$ constitute the normal to the plane, and d is the "offset" value. See planes3d for further information. Users can supply any parameters for a, b, c, and d in the plane argument to produce an arbitrary cutting plane (see Examples), however the function is designed to aid users in choosing a cutting plane without foreknowledge of the desired parameters.

When plane is NA, the function will cut the mesh along a plane orthogonal to one of the primary axes (X, Y, or Z, as indicated by axis) at the location of a focal vertex. The focal vertex can be defined by its index value, supplied to vertIndex. If no value is given for either plane or vertIndex, then an interactive 3D window allows the user to select the focal vertex. A 3D window will open displaying all mesh vertices, colored according to col, with a semi-transparent mesh surface. The display can be rotated with the left mouse button and zoomed with the mouse wheel. The right mouse button allows the user to define a rectangular region in which to identify the focal vertex. The focal vertex is the vertex in the user-selected region with the *minimum value* in the dimension indicated by the axis argument. A preview of the resulting cutting will be supplied, and for the function to finish users must supply a "Y" or "y" confirmation to the Cut mesh?: prompt in the terminal. Any other response will restart the selection process.

The col argument is only invoked when choosing a focal vertex in an interactive 3D window (i.e., vertIndex and plane are set to NA). This argument will apply any acceptable color vector to the displayed vertices. Alternatively, users can specify a color ramp by supplying a string, including: "rainbow", "heat.colors", "terrain.colors", "topo.colors", "cm.colors", or "gray.colors"; see hcl.colors and gray.colors for further details. Color ramps will plot along the axis specified by axis and reverse if flipAxis = TRUE.

If users prefer that the function is inverted with respect to mesh geometry (i.e., that it identifies the focal vertex as the *maximum value* with respect to axis, or that the resulting mesh be that along

the *negative* normal to the plane), then they should set `flipAxis = TRUE`. If `keepBoth` is enabled, the function will return a list of two 'mesh3d' objects: `meshA`, and `meshB`. Enabling `keepBoth` but providing a plane that does not intersect the mesh will result in a list with one of the objects set to `NULL` (see Examples).

This function can be used to cut meshes representing tooth surfaces so as to retain only the area of the tooth crown above the lowest point of the occlusal basin. This cropping procedure is consistent with the one used to prepare surfaces for measurement of occlusal relief (OR) by Ungar & M'Kirera (2003) "A solution to the worn tooth conundrum in primate functional anatomy" PNAS 100(7):3874-3877 Unreferenced vertices can cause errors, so users are encouraged to clean their mesh with `molaR_Clean` prior to using this function.

Value

An object of class 'mesh3d' corresponding to the portion of the mesh on one side of the cutting plane. If `keepBoth` is enabled, a list of two such objects corresponding to the portions from both sides of the plane.

Examples

```
# Result from providing plane parameters and keeping meshes from both sides of plane

cutMesh <- plyPlaneCut(Tooth, plane = c(0.5, 0.5, 0.5, -4), keepBoth = TRUE)
open3d()
shade3d(cutMesh$meshA, col = "gray")
wire3d(cutMesh$meshB)
planes3d(0.5, 0.5, 0.5, -4, col = "red", alpha = 0.66)


# Result from providing parameters for a plane that does not intersect the mesh

cutMesh <- plyPlaneCut(Tooth, plane = c(1, 0.75, 0.5, -11))
identical(Tooth, cutMesh)

cutMesh <- plyPlaneCut(Tooth, plane = c(1, 0.75, 0.5, -11), keepBoth = TRUE)
identical(Tooth, cutMesh)
```

RFI *Calculate relief index for a surface*

Description

A function that calculates relief index following Boyer (2008) Relief index of second mandibular molars is a correlate of diet among prosimian primates and other mammals. J Hum Evol 55:1118-1137 doi: 10.1016/j.jhevol.2008.08.002

Usage

```
RFI(plyFile, alpha = 0.075, findAlpha = FALSE)
```

Arguments

plyFile	An object of classes 'mesh3d' and 'shape3d'
alpha	Step size for calculating the outline. See details.
findAlpha	Logical indicating that alpha will be auto-calculated. See details.

Details

The function requires an object created by reading in a ply file utilizing either the vcgPlyRead function.

Relief index is calculated by the ratio of three-dimensional surface area to two dimensional area on meshes that represent specimen surfaces and have already been pre-smoothed in a 3D data editing program. Surface alignment will have a large effect on 2D area calculation and must be performed prior to creating and reading in the ply file. The mesh must be oriented such that the occlusal plane is parallel to the X- and Y-axes and perpendicular to the Z-axis (i.e., tooth cusps pointing towards +Z).

The alpha parameter traces the outline of the 2D footprint. An alpha that is too low will result in a tracing error (returning an "Alpha adjustment required" message), while an alpha value that is too high may result in an overestimate of the 2D footprint area by failing to take into account infoldings. The user is encouraged to carefully review results using the RFI3d or Check2D functions.

Alternatively, the findAlpha argument can be used to compute an ideal alpha value for a particular PLY file for use in the RFI calculation. This is defined as the lowest value (to the nearest thousandth) returning no error or warning messages. This feature ensures accuracy, but may increase computing time significantly, depending on the number of alpha values tested. Unfortunately, there is no way to guess an appropriate alpha value a priori. After 100 unsuccessful attempts to find an appropriate alpha, the function will terminate.

The alpha value used in the calculation (whether chosen by the user or auto- computed with findAlpha) is returned in the analysis results.

Examples

```
RFI_output <- RFI(Tooth, alpha=0.5, findAlpha = FALSE)
summary(RFI_output)
```

RFI3d *Plot 3D and 2D areas of a mesh used to calculate relief index*

Description

A function that plots a three-dimensional model of the mesh surface and includes a footprint of the two-dimensional area for visual comparison.

Usage

```
RFI3d(
  RFI_File,
  displacement = -1.9,
  SurfaceColor = "gray",
  FootColor = "red",
  FootPts = FALSE,
  FootPtsColor = "black",
  Opacity = 1,
  legend = F,
  main = "",
  cex = 1,
  leftOffset = 0,
  fieldofview = 0,
  fileName = NA,
  binary = FALSE
)
```

Arguments

RFI_File	An object that stores the output of the RFI function
displacement	Numeric that moves the surface footprint some proportion of the height of the mesh. 0 is in the vertical center of the surface, negative values displace the footprint downward.
SurfaceColor	String that controls the color of the 3D surface mesh
FootColor	String that controls the color of the 2D surface footprint
FootPts	Logical indicating whether to plot the flattened points of the footprint from the original ply file
FootPtsColor	Color of the plotted footprint points if FootPts = TRUE
Opacity	Numeric that adjusts the opacity of the 3D mesh surface
legend	Logical indicating whether or not to include a legend of the colors chosen to represent the 3D surface and footprint
main	String indicating plot title
cex	Numeric setting the relative size of the legend and title
leftOffset	Numeric between -1 and 1 setting the amount of offset for the plotted surface to the left. Larger values push surface farther to right.
fieldofview	Passes an argument to par3d() changing the field of view in degrees of the resulting surface plot
fileName	String indicating a name to save the plotted surface to as a *.ply file; default of 'NA' will not save a file
binary	Logical indicating whether or not the saved surface plot should be binary, passed to vcgPlyWrite()

Details

This function can help to visualize the three-dimensional and two dimensional areas that are used in calculating the relief index of a surface by displaying both at the same time. The RFI function must be performed first.

`Opacity` can be adjusted in a range from fully opaque (`1`) to fully transparent (`0`) in order to help visualize the footprint. The vertical placement of the footprint along the Z axis can be altered with `displacement`, depending on how the user wishes to view the surface, or on the original mesh orientation.

A title can be added to the plot by supplying a character string to the `main` argument. Title and legend size are controlled with the `cex` argument, analogous to that in the default R graphics device.

The `leftOffset` value sets how far to the left the surface will plot, intended to help avoid overlap with the legend. Value of 0 will center the surface and should be invoked if the `legend` argument is disabled. Higher values will push the surface farther left and negative values will push it to the right. It is recommended that these values be restricted between -1 and 1 to avoid plotting the surface outside of the rgl window.

`fieldofview` is set to a default of 0, which is an isometric projection. Increasing it alters the degree of parallax in the perspective view, up to a maximum of 179 degrees (see `rgl::par3d()`).

The plotted, colorized surface can be saved as a *.ply to the working directory by changing the `fileName` argument from `NA` to a string (e.g., "RFIPlot"). The resultant ply file can be opened and manipulated in other 3D visualizing programs, such as MeshLab, but will **NOT** retain its legend (a background of the plotting window). To retain the legend, the user is encouraged to utilize the function 'snapshot3d()' in the rgl package. (see `rgl::rgl.snapshot()`) The `binary` argument saves a file in ascii format by default, which is supported by more 3D visualization software than is binary. However, binary files will be considerably smaller.

Examples

```
RFI_File <- RFI(Tooth, alpha=0.5)
RFI3d(RFI_File)
```

Slope *Function to calculate the average slope of a surface*

Description

A function that calculates the average slope over a tooth or some other 3D surface

Usage

```
Slope(plyFile, Guess = FALSE)
```

Arguments

`plyFile`	An object of classes 'mesh3d' and 'shape3d' with calculated normals
`Guess`	Logical indicating whether the function should 'guess' as to the 'up' direction for the surface and to remove negative slopes from the calculation

Details

This function requires a ply file. It will calculate the slope on each face of the surface and will average the slope across the surface. This is functionally equivalent to the slope calculation used by Ungar and M'Kirera "A solution to the worn tooth conundrum in primate functional anatomy" PNAS (2003) 100(7):3874-3877

In the case of applying this function to teeth (its intended purpose), the function expects a surface with the occlusal plane normal to the Z-axis.

The `Guess` parameter is a logical asking whether or not you want the function to both guess as to the right side up of the surface, and to then discard all of the 'negative' slopes, i.e. surfaces which are over-hangs, as is frequently found on the sidewalls of teeth. If `Guess` is not engaged the mean slope will include the negative values of the overhang and will likely underestimate the average slope of the surface.

Regardless of if the `Guess` parameter is engaged, the function will also return a vector containing all of the face slope values ("Face_Slopes")

Examples

```
Slope_output <- Slope(Tooth)
summary(Slope_output)
```

Slope3d *Plot results of a Slope analysis of a surface*

Description

A function that produces a three-dimensional rendering of surface slope. The Slope function will identify the slope of each mesh face. It must be performed prior to using the Slope3d function.

Usage

```
Slope3d(
  Slope_File,
 colors = c("blue", "cornflowerblue", "green", "yellowgreen", "yellow", "orangered",
    "red"),
  maskNegatives = TRUE,
  legend = TRUE,
  main = "",
  cex = 1,
  leftOffset = 1,
  fieldofview = 0,
  fileName = NA,
  binary = FALSE
)
```

Arguments

Slope_File	An object that stores the output of the Slope function
colors	String of colors to build the color gradient
maskNegatives	Logical indicating whether or not to mask (in black) negative slopes, or to reflect them into positive slopes
legend	Logical indicating whether or not a legend should be displayed
main	String indicating plot title
cex	Numeric setting the relative size of the legend and title
leftOffset	Numeric between -1 and 1 setting the amount of offset for the plotted surface to the left. Larger values push surface farther to right.
fieldofview	Passes an argument to par3d() changing the field of view in degrees of the resulting surface plot
fileName	String indicating a name to save the plotted surface to as a *.ply file; default of 'NA' will not save a file
binary	Logical indicating whether or not the saved surface plot should be binary, passed to vcgPlyWrite()

Details

This function creates a heat map on the mesh surface corresponding to the slope of each face calculated by the Slope function.

Colors are taken as a series inputs to define a color ramp and can be customized indefinitely in value or order. The default is suggested as an intuitive display of increasing color heat corresponding with steeper face slope.

A title can be added to the plot by supplying a character string to the main argument. Title and legend size are controlled with the cex argument, analogous to that in the default R graphics device.

The leftOffset value sets how far to the left the surface will plot, intended to help avoid overlap with the legend. Value of 0 will center the surface and should be invoked if the legend argument is disabled. Higher values will push the surface farther left and negative values will push it to the right. It is recommended that these values be restricted between -1 and 1 to avoid plotting the surface outside of the rgl window.

fieldofview is set to a default of 0, which is an isometric projection. Increasing it alters the degree of parallax in the perspective view, up to a maximum of 179 degrees (see rgl::par3d()).

The plotted, colorized surface can be saved as a *.ply to the working directory by changing the fileName argument from NA to a string (e.g., "SlopePlot"). The resultant ply file can be opened and manipulated in other 3D visualizing programs, such as MeshLab, but will **NOT** retain its legend (a background of the plotting window). To retain the legend, the user is encouraged to utilize the function 'snapshot3d()' in the rgl package. (see rgl::rgl.snapshot()) The binary argument saves a file in ascii format by default, which is supported by more 3D visualization software than is binary. However, binary files will be considerably smaller.

Examples

```
Slope_output <- Slope(Tooth)
Slope3d(Slope_output)
```

Tooth *Tooth a surface mesh of a tooth.*

Description

Tooth scan of USNM_112176 lower M~1~ from *Chlorocebus sp.*

A triangular mesh representing a lower M1 Chlorocebus spp. tooth - called by data(Tooth)

Usage

```
data(Tooth)
```

Format

An object of class `"mesh3d"`

`Tooth`: triangular mesh representing a sine-cosine plane.

Examples

```
data(Tooth)
DNE_Output <- DNE(Tooth)
DNE3d(DNE_Output)
```

Index