Package ‘molaR’

March 4, 2020

Title Dental Surface Complexity Measurement Tools

Version 4.5

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

License ACM

LazyData true

Encoding UTF-8

Suggests knitr, rmarkdown, rglwidget

VignetteBuilder knitr

NeedsCompilation no

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

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

RoxygenNote 7.0.2

Repository CRAN

Date/Publication 2020-03-04 00:20:02 UTC

R topics documented:

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)
```

clustered_patches — *A clustering function*

Description

This function gathers linked faces into patches

Usage

```
clustered_patches(Directional_Bin_Face_Pairs)
```

Arguments

`Directional_Bin_Face_Pairs`
the bins of face directions clustered_patches()

`compute_energy_per_face` *Function will compute the DNE per face.*

Description

This will generate each Dirichlet's normal energy for each triangular face on the surface.

Usage

```
compute_energy_per_face(plyFile)
```

Arguments

`plyFile`	a stanford PLY file compute_energy_per_face()

`cross` *Cross product function*

Description

Get the cross product

Usage

```
cross(x, y)
```

Arguments

`x`	vector
`y`	vector cross

cSize *Centroid Size Function*

Description

Get the centroid size

Usage

```
cSize(x)
```

Arguments

x	point cloud cSize

Directional_Bins *This bins the faces into directional categories*

Description

bins into 8 directional categories on the basis of their orientations

Usage

```
Directional_Bins(plyFile, rotation = 0)
```

Arguments

plyFile	a stanford PLY file
rotation	the amount to rotate the specimen by Directional_Bins()

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, BoundaryDiscard = "Vertex")
```

Arguments

- `plyFile` An object of class 'mesh3d' and 'shape3d' with calculated normals
- `outliers` The percentile of Dirichlet energy density values to be excluded defaults to top 0.1 percent
- `BoundaryDiscard` Logical indicating how to handle the exclusion of boundary faces. Defaults to Leg which exlcudes faces which have a leg on the boundary

Details

The function requires an object created by reading in a ply file utilizing either the read.ply or the read.AVIZO.ply function, with calculated normals.

Dirichlet normal energy is calculated on meshes that represent specimen surfaces and have already been simplified to 10,000 faces 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 argumen to 'None' which will not discard any faces on 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 for this calculation.

Examples

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

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

Description

A function that produces a three-dimensional rendering of surface DNE. The DNE function will identify amount of change in mesh face normal orientation between adjacent faces, and associate these values (adjusted for face size) to each face on the surface. It must be performed prior to using the DNE3d function.

Usage

```
DNE3d(
  DNE_File,
  setRange = c(0, 0),
  logColors = TRUE,
  edgeMask = TRUE,
  outlierMask = TRUE,
  legend = TRUE,
  legendScale = 1,
```

```
  leftOffset = 1,
  fieldofview = 0,
  fileName = NA,
  binary = FALSE
)
```

Arguments

`DNE_File`	An object that stores the output of the DNE function
`setRange`	User-defined range for plotting color scheme, see Details
`logColors`	Logical that log transforms the color scheme
`edgeMask`	Logical that colors edge faces black to indicate their lack of contribution to the total Dirichlet normal energy
`outlierMask`	Logical that colors outlier faces dark gray to indicate their lack of contribution to the Dirichlet normal energy
`legend`	Logical indicating whether or not a legend should be displayed
`legendScale`	numeric value setting the relative size of the legend, similar in function to cex
`leftOffset`	numeric value between -1 and 1 setting the degree of offset for the plotted surface to the left; larger values set further to left while 0 is centered
`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 Dirichlet normal energy of each face calculated by the DNE function. Hottest colors represent highest normal energy values.

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

The legend reflects the other arguments chosen by the user, including log transformation, and whether or not an outlier or edge mask is enabled.

By default, the function sets the lowest Dirichlet normal energy calculated among all faces to a cool color and the highest normal energy calculated among all faces to red, and then colors the remaining faces on a continuous color spectrum between these two end points using either absolute or log-transformed Dirichlet normal energy values (depending on the value 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 setRange argument allows users to define the minimum and maximum of the plotting color scheme and use it in 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 user-defined minimum. The user should choose reasonable bounds for the maximum and minimum that are near the maximum and minimum Dirichlet normal energies calculated for their surfaces. setRange will not accept negative values.

The leftOffset value sets how far to the left the surface will appear, intended to help avoid overlap with the legend. A value of 0 for this argument will center the surface in the plotting window and negative values will shift it to the right.

legendScale sets the relative size of the legend, analogous to the cex argument of par graphics.

fieldofview is set to a default of 0, which is an isometric parallel projection. Raising it corresondingly increases the amount of obliquity used to render the surface in the plotting window, up to a maximum of 179 degrees.

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 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

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

DNE_Legend *Make legend for DNE3d plot*

Description

plotting subfunction

Usage

```
DNE_Legend(
  DNELabels,
  scaled = F,
  edgeMask = F,
  outlierMask = F,
  logColors = F,
  size = 1
)
```

Arguments

DNELabels	values for the labels
scaled	logical indicating whether the values are scaled
edgeMask	logical indicating whether of not edges are being masked and that information to be included in the legend

`outlierMask`	logical indicating whether outliers are masked
`logColors`	logical indicating colors are on log scale
`size`	legend scaling factor

Details

This is an internal function which builds a better DNE plot legend

The legend will reflect the elements used in the plot. This is an internal function. Users will have little need or call to interact with it. DNE_Legend()

`edge_vertices` *Function for finding the edge vertices*

Description

Function will sort through all the vertices of the surface and find the ones which are on the edge. This will be needed for identifying which should be masked and not inlcuded in the calculation of the final DNE value.

Usage

```
edge_vertices(plyFile)
```

Arguments

`plyFile`	a stanford PLY file edge_vertices()

`Equal_Vertex_Normals` *Important function for re-doing the vertex normals for the DNE calculation.*

Description

The geomorph import function does not generate the correct vertex normals.

Usage

```
Equal_Vertex_Normals(plyFile)
```

Arguments

`plyFile`	a stanford PLY file Equal_Vertex_Normals()

`face_areas` *Function to calculate face areas.*

Description

This function calculates the area of each face on a ply file

Usage

```
face_areas(plyFile)
```

Arguments

`plyFile`	a stanford PLY file face_areas()

`Face_Normals` *Function to find Face Normals*

Description

This function re-computes the face normals in a way consistent with MorphoTester.

Usage

```
Face_Normals(plyFile)
```

Arguments

`plyFile`	a stanford PLY file Face_Normals()

`Hills` *Hills - sine-cosine plane*

Description

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

Format

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

`index_paired_directed_faces` *Index of paired faces with directions*

Description

This does some heavy lifting to pull together faces which are paired together. This is needed for many later functions for compiling OPC

Usage

```
index_paired_directed_faces(plyFile)
```

Arguments

`plyFile`	a stanford PLY file index_paired_directed_faces()

`molaR_Batch` *Run a batch of molaR analyses*

Description

A function which automates molaR analyses. User simply sets up the functions they want run and can leave the computer to do the rest.

Usage

```
molaR_Batch(
  pathname = getwd(),
  filename = "molaR_Batch.csv",
  DNE = TRUE,
  RFI = TRUE,
  OPCr = TRUE,
  OPC = FALSE,
  Slope = TRUE,
  Details = FALSE,
  DNE_outliers = 0.1,
  DNE_BoundaryDiscard = "Leg",
  RFI_alpha = 0.01,
  OPCr_steps = 8,
  OPCr_stepSize = 5.625,
  OPCr_minimum_faces = 3,
  OPCr_minimum_area = 0,
  OPC_rotation = 0,
```

```
    OPC_minimum_faces = 3,
    OPC_minimum_area = 0,
    Slope_Guess = FALSE,
    Parameters = FALSE
  )
```

Arguments

pathname	The path to the file containing all the PLY surfaces to be analyzed. Defaults to the working directory
filename	Name for the output csv file.
DNE	logical indicating whether or not to perform DNE calculation Defaults to true
RFI	logical indicating whether or not to perform RFI calculation Defaults to true
OPCr	logical indicating whether or not to perform OPCr calculation Defaults to true
OPC	logical indicating whether or not to perform OPC calculation Defaults to false
Slope	logical indicating whether or not to perform Slope calculation, Defaults to true
Details	logical indicating whether or not to save the details of the RFI and OPCr calculations
DNE_outliers	the percentile at which outliers will be excluded is passed to the DNE function, defaults to 0.1
DNE_BoundaryDiscard	is a logical indicating how to handle the exclusion of the faces on the edge of the surface, defaults to excluding faces which have a leg on the boundary.
RFI_alpha	the size of the alpha passed to RFI function, defaults to 0.01
OPCr_steps	the number of steps the OPCr function should take, is passed to the OPCr function. Defaults to 8
OPCr_stepSize	the size of each rotation. Passed to the OPCr function. Defaults to 5.626 degrees
OPCr_minimum_faces	sets the lower boundary for number of faces a patch must have for inclusion in total count. Defaults to 3 or more.
OPCr_minimum_area	sets the lower boundary for percentage of the surface area a patch must make up for inclusion in the total patch count. Cannot be used with minimum_faces on. Defaults to zero
OPC_rotation	amount of rotation to apply during OPC calculation. Defaults to zero
OPC_minimum_faces	minimum number of faces a patch must contain to be counted in the OPC function. Defaults to 3.
OPC_minimum_area	minimum percentage of the surface area a patch must make up to be counted in the OPC function. Defaults to off
Slope_Guess	logical indicating whether or not to Guess as to the orientation of the surface during the Slope calculation (see Slope function for details)
Parameters	defaults to off. When engaged a list of all the parameters used during molaR analysis will be appended to the output file.

Details

This function allows a user to set the analyses from molaR they want to run, along with the specific parameters for each function and have a whole batch of PLY files analyzed and saved to a csv file. Function will perform analyses on all PLY files in the working directory or user can specify a pathname to a folder containing PLY files. Output saves to the folder that contains the analyzed PLY files.

This function will accept a vector of parameters for any of the function arguments if the user wishes to vary the settings over the course of the batch run. It is recommended that when making use of this feature the Parameters argument is set to TRUE for a record of how analyses were performed.

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

`molaR_bgplot` *Internal function for making molaR plot legends*

Description

Function properly scales legends and prints them to the background of rgl devices

Usage

```
molaR_bgplot(expression)
```

Arguments

`expression`	it knows what to do... molaR_bgplot()

`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")
```

Arguments

`plyFile`	An object of classes 'mesh3d' and 'shape3d'
`cleanType`	logical asking what to clean, Verticies, Faces or Both. Defaults to Both.

Details

This function cleans up problematic ply files. Some smoothed files will have faces of zero area, or floating verticies. 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 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 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 read.ply, vcgPlyread, or read.AVIZO.ply 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 point cloud will have a large effect on patch orientation and must be done in a 3D data editing program such as Avizo, or using the R package auto3dgm prior to creating and reading in the ply file. 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 only a single face 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_Output_Object,
  binColors = hsv(h = (seq(10, 290, 40)/360), s = 0.9, v = 0.85),
  patchOutline = FALSE,
  outlineColor = "black",
  maskDiscard = FALSE,
  legend = TRUE,
  legendScale = 1,
  legendTextCol = "black",
  legendLineCol = "black",
  leftOffset = 1,
  fieldofview = 0,
  fileName = NA,
  binary = FALSE
)
```

Arguments

`OPC_Output_Object`	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
`legendScale`	numeric value setting the relative size of the legend, similar in function to cex
`legendTextCol`	Parameter defining color for the legend text
`legendLineCol`	Parameter defining the color for the legend lines

`leftOffset` numeric value between -1 and 1 setting the degree of offset for the plotted surface to the left; larger values set further to left while 0 is centered

`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.

Several legend plotting options are availble, including customizing the line and text colors with legendTextCol and legendLineCol, which both default to black.

The leftOffset value sets how far to the left the surface will appear, intended to help avoid overlap with the legend. A value of 0 for this argument will center the surface in the plotting window and negative values will shift it to the right.

legendScale sets the relative size of the legend, analogous to the cex argument of par graphics.

fieldofview is set to a default of 0, which is an isometric parallel projection. Raising it corresondingly increases the amount of obliquity used to render the surface in the plotting window, up to a maximum of 179 degrees.

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 snapshot3d function. Patch outlines are currently not retained with surface saving. 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)
```

`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 an object created by reading in a ply file utilizing either the read.ply or the read.AVIZO.ply function, with calculated normals.

Default number of Steps is 8, with a stepSize of 5.625 degrees, following the original 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.

`OPC_Legend` *function for building a legend in OPC plots*

Description

crucial graphics subfunction

Usage

```
OPC_Legend(
  binColors = c(1:8),
  binNumber = 8,
  maskDiscard = F,
  textCol = "black",
  size = 1,
  lineCol = "black"
)
```

Arguments

`binColors`	number sequence for bins and their colors
`binNumber`	numeric number of different bins
`maskDiscard`	logical determining whether faces will be blacked out because they are discarded
`textCol`	color for the text in the circle legend
`size`	scaling factor for the legend size
`lineCol`	color for the lines in the legend OPC_Legend()

`patches_for_each_direction` *Function for gathering the patches for each direction*

Description

This function will gather the patches in each of the 8 bins and ready it for patches_for_each_direction()

Usage

```
patches_for_each_direction(indexed_pairs)
```

Arguments

`indexed_pairs`	Pairs of touching faces

`patches_per` *A function for patches within each face*

Description

this gets some important information out of each patch

Usage

```
patches_per(patch_details, plyFile, minimum_faces = 3, minimum_area = 0)
```

Arguments

`patch_details`	information on each patch
`plyFile`	a stanford PLY file
`minimum_faces`	minimum number of faces in each counted patch
`minimum_area`	minimum area for a patch to be counted patches_per()

patch_details *Function for gathering patch details for each Orientation patch*

Description

This function does some simple math to lets us know about the patches

Usage

```
patch_details(clusterlist, plyFile)
```

Arguments

`clusterlist`	a list of faces in the cluster patch_details()
`plyFile`	a stanford PLY file

plyClip *Clip a ply file*

Description

Function will clip a ply file along either the X, Y, or Z plane. The location for clipping can be indicated by the user through an interactive 3D window, or can be the index number of any vertex in the ply file.

Usage

```
plyClip(
  plyFile,
  axis = "Z",
  vertIndex = NA,
  meshInvert = FALSE,
  button = "right",
  displayNew = TRUE,
  keepBoth = FALSE,
  edgeRefine = FALSE
)
```

Arguments

`plyFile`	An object of classes 'mesh3d' and 'shape3d'
`axis`	Logical indicating the axis plane on which to clip the mesh. May be "X", "Y", or "Z". Defaults to "Z".
`vertIndex`	Numeric index of a ply vertex to define clipping plane. See Details.

meshInvert	Logical indicating whether or not to invert the mesh about the user-indicated axis.
button	Logical indicating which button on the mouse will select a region of the ply file. Must be one of 'right' (default), 'middle', or 'left'.
displayNew	Logical indicating whether or not to display the ply file after clipping.
keepBoth	Logical indicating whether or not to save both sides of the clipped ply.
edgeRefine	Logical indicating whether or not to create a new, smooth edge along the indicated clipping plane.

Details

This function returns a ply file that is clipped along a plane parallel to one of the three primary axes: X, Y, or Z. The location of the clipping plane is defined by one of the vertices in the ply file, and this 'focal vertex' is treated as a *minimum*. This means that regions of the ply file extending in a positive direction from the focal vertex along the user-defined axis (X, Y, or Z) will be retained as the user's selection, while regions of the negative to the focal vertex along the user-defined axis will be clipped out. The function offers two ways for the user to define the focal vertex: either by supplying an index number for a specific vertex in the ply file, or by allowing the user to interact with their ply in 3D and define a region of the ply in which to capture the focal vertex. It is assumed that most users will want to choose the region of their ply with the focal vertex, and the function therefore defaults to the interactive method unless a value is supplied for vertIndex.

When choosing the region with the focal vertex, a 3D interactive window displaying all ply vertices will appear. Users can manipulate this display with the left mouse button and zoom with the mouse wheel. The mouse button indicated by the button argument (defaults to 'right') allows the user to define a rectangular region of space in which to identify the focal vertex. The focal vertex is calculated as that vertex in the user-selected region with the minimum value along the axis indicated by the user in the axis argument (defaults to 'Z' axis). The function will identify the focal vertex in the user-defined region and allow users an opportunity to re-select their region before clipping. If users find that the function is retaining the opposite region of interest from the one they were interested in (i.e., if they wish to select the region of the ply file *negative* to the focal vertex along their specified axis), then the meshInvert argument should be altered to TRUE.

The keepBoth parameter allows users to retain both the positive and negative sides of their original ply file, now separated into two distinct plys. If this option is enabled, the function will return a list containing two objects: a 'Positive' ply file representing the region of the surface the function would have returned by default, and a 'Negative' ply file representing the remainder portion of the surface. This argument cannot currently be enabled with the edgeRefine argument is enabled.

The edgeRefine parameter is intended to produce a smooth edge along the plane identified by the user. By default the function will only retain those vertices and faces that were positive to the focal vertex along the axis specified by the user. This often produces an irregular, jagged surface edge along the boundaries of the retained faces. Enabling the edgeRefine argument smooths this edge out by adding new vertices and faces to fill in the jagged areas, creating a surface boundary even with the focal vertex. Implementing this step for a mesh with a large number of faces (>50,000) may be time-intensive. This argument cannot currently be enabled when the keepBoth argument is enabled. NOTE: ENABLING edgeRefine WILL MAKE THE OUTPUT UNSUITABLE FOR CALCULATING RFI.

This function can be used to clip ply files representing mammal tooth surfaces in such a way 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 RFI by Ungar and M'Kirera (2003).

It is recommended that users clean their ply files with the molaR_Clean() function prior to using this function, as unreferenced vertices can cause errors when recreating ply files. Future implementations of this function will allow the keepBoth and edgeRefine arguments to be enabled simultaneously and will allow users to define arbitrary planes (rather than those parallel to the primary axes) for clipping ply files.

`read.AVIZO.ply` *Read mesh data from ply files saved by AVIZO*

Description

A function that reads Stanford ply files as saved by the 3D data visualization software Avizo

Usage

```
read.AVIZO.ply(file, ShowSpecimen = TRUE, addNormals = TRUE)
```

Arguments

`file`	An ASCII PLY file generated by Avizo
`ShowSpecimen`	Logical indicating whether or not the mesh should be displayed
`addNormals`	Logical indicating whether or not normals of mesh vertices shold be calculated and appended to object

Details

If ShowSpecimen is True, a gray shade3d of the mesh is generated in a new rgl window for previewing the specimen. When saving to the ply file type, Avizo inserts additional property parameters into the file heading that sometimes describe various components of the mesh. These additional properties cause the read.ply function native to the geomorph package to fail. This function properly reads ply files generated by Avizo (like read.ply) and can be stored as an object accepted as input in the other molaR functions. Ply files generated through other software (such as MeshLab) can be read using read.ply.

remove_boundary_faces *Remove boundary faces*

Description

Important function for masking the edge faces

Usage

```
remove_boundary_faces(Energy_Per_Face_Values, plyFile)
```

Arguments

`Energy_Per_Face_Values`
: information on E per face remove_boundary_faces()

`plyFile`
: a stanford PLY file

remove_outliers *Mask outliers on some faces*

Description

This function will block out the top 0.1 percent of the faces

Usage

```
remove_outliers(Energy_values, X)
```

Arguments

`Energy_values`
: energy density values on faces

`X`
: percentile above which to remove
remove_outliers()

`repmat` *Replicate a matrix*

Description

This function replicates a matrix

Usage

```
repmat(a, n, m = n)
```

Arguments

`a`	vector or matrix to replicate
`n`	number of replicates in n-direction
`m`	number of replicates in m-direction repmat

`RFI` *Calculate Boyer's (2008) 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 = 0.06)
```

Arguments

`plyFile`	An object of classes 'mesh3d' and 'shape3d'
`alpha`	Step size for calculating the outline. See details

Details

The function requires an object created by reading in a ply file utilizing either the read.ply or the read.AVIZO.ply function, with calculated normals.

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. Alignment of the point cloud will have a large effect on patch orientation and must be done in a 3D data editing program or auto3dgm 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.

Some files may fail with pancake[TempF,] : subscript out of bounds. In these files it may be necessary to increase the alpha value which is default set to 0.06. Increasing the alpha value can cause the RFI function to over-estimate the size of the footprint. Caution should be exercised when troubleshooting by adjusting alpha

Examples

```
RFI_output <- RFI(Tooth, alpha=0.5)
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_Output,
  displacement = -1.9,
  SurfaceColor = "gray",
  FootColor = "red",
  FootPts = FALSE,
  FootPtsColor = "black",
  Opacity = 1,
  legend = F,
  legendScale = 1,
  leftOffset = 0,
  fieldofview = 0
)
```

Arguments

`RFI_Output`	An object that stores the output of the RFI function
`displacement`	Moves the surface footprint some proportion of the height of the mesh. 0 is no displacement. Expects a value, negative values displace the footprint downward.
`SurfaceColor`	changes the color of the 3D surface mesh
`FootColor`	changes 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 for the plotted footprint points

`Opacity`	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
`legendScale`	cex style numeric relative scaling factor for the legend
`leftOffset`	how numeric between -1 and 1 for which to offset the surface relative to the legend.
`fieldofview`	Passes an argument to par3d changing the field of view in degrees of the resulting rgl window.

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 displace depending on how the user wishes to view the surface, or on the original mesh orientation.

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.

Examples

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

RFI_Legend — *function for building a legend for RFI*

Description

crucial plotting subfunction for RFI3d

Usage

```
RFI_Legend(surfCol = "gray", footCol = "red", size = 1, opac = 1)
```

Arguments

`surfCol`	color for the 3D surface defaults to gray
`footCol`	color for the 2D footprint defualts to red
`size`	cex style scaling parameter
`opac`	sets the value for the opacity of the tooth surface when that is engaged RFI_Legend()

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

Description

A function that calcualate the average slope over tooth or some other surface

Usage

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

Arguments

`plyFile`	An oject of classes 'mesh3d' and 'shape3d'
`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 (2003).

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,
  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
`leftOffset`	numeric value between -1 and 1 setting the degree of offset for the plotted surface to the left; larger values set further to left while 0 is centered
`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.

The leftOffset value sets how far to the left the surface will appear, intended to help avoid overlap with the legend. A value of 0 for this argument will center the surface in the plotting window and negative values will shift it to the right.

fieldofview is set to a default of 0, which is an isometric parallel projection. Raising it corresondingly increases the amount of obliquity used to render the surface in the plotting window, up to a maximum of 179 degrees.

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 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

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

`Slope_Legend` *Plotting subfunction for making slope plot legend*

Description

Plotting subfunction for making slope plot legend

Usage

```
Slope_Legend(colors = colors, maskNegatives = T)
```

Arguments

`colors`	provided from Slope3d
`maskNegatives`	logical whether to mask

`Tooth` *Tooth - scan of USNM_112176 lower M1 from Chlorocebus spp.*

Description

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

Format

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

tr — *Trace function*

Description

Matrix algebra

Usage

```
tr(m)
```

Arguments

`m`	a square matrix tr()

vertex_to_face_list — *function for making a list of faces on each vertex*

Description

crucial function for getting a list of faces which will gather the faces per vertex.

Usage

```
vertex_to_face_list(plyFile)
```

Arguments

`plyFile`	a stanford PLY file vertex_to_face_list()

Index