Package ‘habtools’

November 15, 2024

Title Tools and Metrics for 3D Surfaces and Objects

Version 1.0.7

Description A collection of functions for sampling and simulating 3D surfaces and objects and estimating metrics like rugosity, fractal dimension, convexity, sphericity, circularity, second moments of area and volume, and more.

License MIT + file LICENSE

URL https://jmadinlab.github.io/habtools/

BugReports https://github.com/jmadinlab/habtools/issues

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

Depends R (>= 2.10)

Imports raster (>= 3.5), terra, methods, Rvcg, sp, geometry, concaveman, magrittr, purrr, dplyr

Suggests knitr, rmarkdown, rgl, parallel, ggplot2, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author Joshua Madin [aut] (<https://orcid.org/0000-0002-5005-6227>), Nina Schiettekatte [aut, cre] (<https://orcid.org/0000-0002-1925-3484>)

Maintainer Nina Schiettekatte <nina.schiettekatte@gmail.com>

Repository CRAN

Date/Publication 2024-11-15 21:10:02 UTC

Contents

cell_count_1d — *Count filled cells in 1D*

Description

A helper function for segment, box and cube counting fractal methods. The function divide the array into n pieces and counts how many are occupied.

Usage

```
cell_count_1d(pts, xmin, xmax, n)
```

Arguments

pts	Data frame with x coordinates
xmin	Minimum x-value
xmax	Maximum x-value
n	Multiplier

Value

Number of filled cells

Examples

```
pts <- data.frame(x = rnorm(200, 0, 5))
cell_count_1d(pts, xmin = min(pts$x), xmax = max(pts$x), n = 5)
```

cell_count_2d — *Count filled cells in 2D*

Description

A helper function for segment, box and cube counting fractal methods. The function divide the array into n pieces and counts how many are occupied.

Usage

```
cell_count_2d(pts, xmin, xmax, ymin, ymax, n)
```

Arguments

`pts`	Data frame with x and y coordinates
`xmin`	Minimum x-value
`xmax`	Maximum x-value
`ymin`	Minimum y-value
`ymax`	Maximum y-value
`n`	Multiplier

Value

Number of filled cells

`cell_count_3d` *Count filled cells 3D*

Description

A helper function for segment, box and cube counting fractal methods. The function divide the array into `n` pieces and counts how many are occupied.

Usage

```
cell_count_3d(pts, xmin, xmax, ymin, ymax, zmin, zmax, n)
```

Arguments

`pts`	Data frame with x, y, and z coordinates
`xmin`	Minimum x-value
`xmax`	Maximum x-value
`ymin`	Minimum y-value
`ymax`	Maximum y-value
`zmin`	Minimum z-value
`zmax`	Maximum z-value
`n`	Multiplier

Value

Number of filled cells

centroid — *Calculate the centroid of 3D points*

Description

Calculates the centroid for a given set of XYZ coordinates.

Usage

```
centroid(data)
```

Arguments

`data`	A data frame with x, y, and z coordinates.

Value

The coordinates of the centroid.

Examples

```
data <- mesh_to_points(mcap)
centroid(data)
```

circularity — *Calculate circularity of a 2D shape*

Description

The perimeter of the 2D shape is divided by the perimeter of a circle with the same area as the shape. The more irregular the shape is, the closer the output value is to zero. The closer the shape is to a circle, the closer the output value is to 1.

Usage

```
circularity(data)
```

Arguments

`data`	A data frame with the first two columns x and y coordinates, respectively.

Value

A value between 0 (infinitely irregular) and 1 (a perfect circle).

See Also

sphericity()

Examples

```
mcap_2d <- mesh_to_2d(mcap)
plot(mcap_2d, asp=1)
circularity(mcap_2d)

circ <- sim_circle() # simulate xy coordinates for a circle
plot(circ, asp=1)
circularity(circ)
```

---

convexity — *Calculate convexity of a 3D mesh*

Description

The ratio of the volume of the object and the volume of the convex hull around the object. Objects with fewer concavities will be closer to 1.

Usage

```
convexity(mesh)
```

Arguments

mesh	A triangular mesh of class mesh3d. #'

Value

The convexity value.

Examples

```
convexity(mcap)
```

`csf` *Calculate mechanical shape factor*

Description

Calculates mechanical vulnerability of rigid, cantilever-type structural elements.

Usage

```
csf(mesh, z_min, res, keep_data = FALSE)
```

Arguments

`mesh`	A triangular mesh of class mesh3d.
`z_min`	The z plane about which csf should be calculated. Defaults to min(z).
`res`	The resolution to be used for the calculation. Defaults to the resolution of the mesh.
`keep_data`	Logical. Return list with supplemental info? Defaults to FALSE.

Details

This function calculates the mechanical vulnerability of a structural element, like a hard coral colony, to fluid flow. While developed for corals, and originally called the Colony Shape Factor (CSF), the function is applicable to any attached, rigid cantilever type structure. CSF is dimensionless and can be used to compare the vulnerability among structures. Mechanistically, if the CSF of a structure becomes greater than the dislodgement mechanical threshold, breakage occurs. This threshold is a function of material tensile strength and inversely related to fluid velocity and density (Madin & Connolly 2006).

Value

A value for csf or if keep_data = TRUE, a list containing the colony shape factor (csf), the parallel to flow (dy) and perpendicular (dx) diameters of the cantilever base, and the bending moment (mom).

Note

The orientation of the 3D mesh is important for this function. The function assumes the fluid flow is parallel with the y-axis. The function also assumes the base of the cantilever over which the bending moment acts can be approximated as an ellipse with the diameter on the y-axis parallel with flow (dy). You can set a z_min if the base of your mesh is not flat at the base (i.e., shift the plane upon which the cantilever is attached upwards). The function output includes dy and dx for monitoring anticipated values.

References

Madin JS & Connolly SR (2006) Ecological consequences of major hydrodynamic disturbances on coral reefs. Nature. 444:477-480.

Examples

```
csf(mcap, z_min = -3.65)
csf(mcap, z_min = -3.65, keep_data = TRUE)
```

---

dem_crop *Crop DEM around points*

Description

A function for sampling a DEM by cropping squares of a given size around xy coordinates.

Usage

```
dem_crop(data, x0, y0, L, plot = FALSE)
```

Arguments

data	A DEM in RasterLayer format.
x0	A value or vector of central x coordinate(s).
y0	A value or vector of central y coordinate(s).
L	Size of squares to cropped from the DEM.
plot	Logical. Plot the DEM and the cropped sections?

Value

A cropped RasterLayer or list of RasterLayers.

Examples

```
# around one point
dem_cropped <- dem_crop(horseshoe, -468, 1266, L = 2)
raster::plot(dem_cropped)
points(-468, 1266)

# around multiple points
points <- data.frame(x = c(-467, -465, -466), y = c(1270, 1265, 1268))
dem_list <- dem_crop(horseshoe, points$x, points$y, L = 1, plot = TRUE)

# plot the first element
raster::plot(dem_list[[1]])
```

dem_sample — *Sample a random DEM with specified size from a larger DEM*

Description

Sample a random DEM with specified size from a larger DEM

Usage

```
dem_sample(data, L, allow_NA = 0, plot = FALSE, max_iter = 100)
```

Arguments

`data`	Digital elevation model of class RasterLayer.
`L`	Size of square to cut out of DEM.
`allow_NA`	Proportion of NA values allowed in the sample. Useful when DEM is not regular.
`plot`	Logical. Plot the DEM and the cropped section?
`max_iter`	Maximum number of random crops to try when allow_NA = FALSE before failing.

Value

Digital elevation model of class RasterLayer.

Note

Not allowing NAs may increase sampling time for irregular DEMs that contain a lot of NAs; e.g., structure from motion transects.

Examples

```
dem <- dem_sample(horseshoe, L = 2, plot=TRUE)
```

dem_split — *Split DEM into smaller tiles*

Description

Split DEM into smaller tiles

Usage

```
dem_split(data, size, parallel = FALSE, ncores = (parallel::detectCores() - 1))
```

Arguments

`data`	Digital elevation model of class RasterLayer.
`size`	Size of tiles, in the same unit as the RasterLayer.
`parallel`	Logical. Use parallel processing? Note: parallel must be installed.
`ncores`	Number of cores to use when parallel = TRUE.

Value

List of RasterLayers.

Examples

```
L <- habtools::extent(horseshoe) # size of horseshoe = 8m
size <- 2 # size of target tiles
(L / size)^2 # number of target tiles = 16
dem_list <- dem_split(horseshoe, 2)
length(dem_list)
```

`dem_to_points` *Transform DEM to 3D pointcloud of raster corners*

Description

Transform DEM to 3D pointcloud of raster corners

Usage

```
dem_to_points(dem, bh = NULL, parallel = FALSE)
```

Arguments

`dem`	Digital elevation model of class RasterLayer.
`bh`	Border height from lowest point.
`parallel`	Logical. Use parallel computation?

Value

A 3D point cloud for raster cell corners.

Examples

```
dem <- sim_dem(20, 0.5)
raster::plot(dem)
pts <- dem_to_points(dem)
rgl::plot3d(pts)
```

`detect_drop` *Detect a sudden drop, edge, or overhang in a DEM*

Description

Detect a sudden drop, edge, or overhang in a DEM

Usage

```
detect_drop(data, d = 0.1)
```

Arguments

`data`	DEM of class RasterLayer.
`d`	The threshold height difference to define a drop.

Value

A RasterLayer marking edges. Values indicate maximum height difference of surrounding cells.

Examples

```
edges <- detect_drop(horseshoe, d = 0.2)

raster::plot(horseshoe)
raster::plot(edges)
```

`extent` *Calculate extent of a 3D object*

Description

This function calculates the extent or largest length of the bounding box of a mesh or a DEM.

Usage

```
extent(data)
```

Arguments

`data`	Digital elevation model of class RasterLayer or a triangular mesh of class mesh3d.

Value

A value, the extent of the mesh or DEM.

Note

There are several extent function is other packages, including the raster package. Therefore it is recommended to use the package namespace, see examples below.

Examples

```
habtools::extent(mcap)
habtools::extent(horseshoe)
```

---

fd *Calculate fractal dimension*

Description

Calculate fractal dimension

Usage

```
fd(data, method, lvec, keep_data = FALSE, diagnose = FALSE, ...)
```

Arguments

`data`	Digital elevation model of class RasterLayer or a triangular mesh of class mesh3d.
`method`	If data is a RasterLayer, possible methods are:"hvar", "area", "sd", and "cubes" (defaults to "hvar"). If data is a mesh3d, possible methods are "cubes" and "area" (defaults to "cubes").
`lvec`	Vector of scales to use for calculation.
`keep_data`	Logical. Keep data? Default is FALSE.
`diagnose`	Logical. Show diagnostic plot and metrics?
`...`	Arguments from method-specific fd_ functions.

Details

Calculates fractal dimension using the specified method. Note that methods are distinctly different and should not be mixed when comparing values for multiple objects. See `fd_hvar()`, `fd_area()`, `fd_cubes()`, `fd_sd()` for details about each method. If `lvec` is not specified, a default based on resolution, extent, and method will be used. The `cubes` method is not recommended if the height range is much smaller than the extent of a 3d object or DEM, which is typically the case for DEMs. Most objects and surfaces are not perfectly fractal. It is recommended to investigate scale transitions by setting diagnose to TRUE.

Value

A value for fractal dimension, typically between 2 and 3 or a list if keep_data = TRUE.

See Also

fd_hvar()

fd_area()

fd_sd()

fd_cubes()

fd_diagnose()

Examples

```
dem <- dem_crop(horseshoe, x0 = -469, y0 = 1267, L = 2, plot = TRUE)
fd(dem, method = "hvar", lvec = c(0.125, 0.25, 0.5, 1, 2))
fd(dem, method = "area", diagnose = TRUE)
fd(dem, method = "sd")
fd(mcap2, method = "cubes",  plot = TRUE)
fd(mcap2, method = "area",  diagnose = TRUE)
```

---

fd_area — *Calculate fractal dimension using the surface area method*

Description

Calculate fractal dimension using the surface area method

Usage

```
fd_area(data, lvec = NULL, keep_data = FALSE, plot = FALSE, scale = FALSE)
```

Arguments

data	DEM of class "RasterLayer" or mesh of class "mesh3d".
lvec	Vector of scales to use for calculation.
keep_data	Logical. Keep data? Default is FALSE.
plot	Logical. Plot surface with area resolutions superimposed? Defaults to FALSE.
scale	Logical. Rescale height values to fit the extent? Only relevant for DEMs. Defaults to FALSE.

Details

This function calculates fractal dimension using the area method. Based on values in `lvec`, the DEM or mesh is reprojected to varying scales. Fractal dimension is defined as `2 - s` with s being the slope of the regression between the log-transformed surface areas across scales and the log-transformed scales. Considerate bias is introduced if scales approach the extent of the object due to an edge effect. Therefore, this approach is only appropriate when the object is large relative to the scales of interest to be used as `lvec`. Diagnostic plots may help visualize whether bias is present for the scales chosen (i.e. points do not fall on a straight line).

Value

A value for fractal dimension, typically between 2 and 3 or a list if keep_data = TRUE.

Examples

```
fd_area(horseshoe, lvec = c(0.125, 0.25, 0.5))

# Look at diagnostic plot
fdata <- fd_area(horseshoe, lvec = c(0.05, 0.1, 0.2, 0.4), keep_data = TRUE)
fd_diagnose(fdata)
# points fall on straight line

fdata <- fd_area(horseshoe, lvec = c(0.5, 1, 2, 4), keep_data = TRUE)
fd_diagnose(fdata)
# points fall on hollow curve, indicating that lvec includes values that
# are too high.
```

---

fd_boxes — *Calculate fractal dimension using the box counting method*

Description

Calculate fractal dimension using the box counting method

Usage

```
fd_boxes(data, lvec, keep_data = FALSE, plot = FALSE)
```

Arguments

`data`	A data frame in which the first two columns are x and y coordinates, respectively.
`lvec`	The scales to use for calculation (i.e. box sizes).
`keep_data`	Logical. Keep calculation data? Default = TRUE.
`plot`	Logical. Plot the shape with box sizes superimposed? Defaults to FALSE.

Details

This function calculates fractal dimension using the box counting method. If `lvec` is not specified, a default based on resolution and extent will be used. Based on lvec, boxes of different sizes are defined and the function counts boxes that capture the outline of the shape. It is recommended to specify the maximum value of `lvec` so that the largest box encapsulates the entire object. The smallest scale included in `lvec` should not be smaller than the resolution of your object.

Value

A value for fractal dimension, typically between 1 and 2 or a list if keep_data = TRUE.

Examples

```
mcap_2d <- mesh_to_2d(mcap)

fd_boxes(mcap_2d, plot = TRUE, keep_data = TRUE)
fd_boxes(mcap_2d, lvec = c(0.05, 0.1, 0.2, 0.4), plot = TRUE)
```

---

fd_cubes — *Calculate fractal dimension using the cube counting method*

Description

Calculate fractal dimension using the cube counting method

Usage

```
fd_cubes(data, lvec = NULL, plot = FALSE, keep_data = FALSE, scale = FALSE)
```

Arguments

`data`	An object of class RasterLayer or mesh3d.
`lvec`	Vector of scales to use for calculation (i.e. cube sizes).
`plot`	Planar representation of cubes superimposed on 3D mesh or DEM for visualizing `lvec`. Default = FALSE.
`keep_data`	Logical. Keep calculation data? Default = TRUE.
`scale`	Logical. Rescale height values to the extent? Only relevant for RasterLayer objects. (Defaults to FALSE).

Details

This function calculates fractal dimension using the cube counting method. If `lvec` is not specified, a default based on resolution and extent will be used. Based on lvec, cubes of different sizes are defined and the function counts mesh points that fall within each cube. It is recommended to specify the maximum value of `lvec` so that the largest box encapsulates the entire object. The smallest scale included in `lvec` should not be smaller than the resolution of your object.

Value

A value for fractal dimension, typically between 2 and 3 or a list if keep_data = TRUE.

See Also

`fd()`

Examples

```
fd_cubes(mcap, keep_data = TRUE, plot = TRUE)
fd_cubes(mcap, lvec = c(0.05, 0.1, 0.25, 0.5), plot = TRUE)

dem <- dem_crop(horseshoe, x0 = -469, y0 = 1267, L = 2, plot = TRUE)
fd_cubes(dem, plot = TRUE, keep_data = TRUE)
fd_cubes(dem, plot = TRUE, keep_data = TRUE, scale = TRUE)
```

---

fd_diagnose *Diagnose fractal dimension*

Description

Diagnoses fractal dimension variation across neighboring scales.

Usage

```
fd_diagnose(data, keep_data = TRUE)
```

Arguments

data	Output of fd() with option keep_data = TRUE.
keep_data	Logical. Keep diagnostics data?

Value

A list with fractal dimension across scales, mean fractal dimension, and sd of fractal dimensions across scales.

Examples

```
fd_data <- fd(horseshoe, lvec = c(0.05, 0.1, 0.2, 0.4), method = "area", keep_data = TRUE)
fd_diagnose(fd_data)
fd_diagnose(fd_data, keep_data = FALSE)
```

fd_hvar — *Calculate fractal Dimension using the height variation method*

Description

Calculate fractal Dimension using the height variation method

Usage

```
fd_hvar(
  data,
  lvec,
  regmethod = "mean",
  keep_data = FALSE,
  plot = FALSE,
  parallel = FALSE,
  ncores = (parallel::detectCores() - 1)
)
```

Arguments

`data`	Digital elevation model of class RasterLayer or dataframe (output of hvar function)
`lvec`	Vector of scales to use for calculation.
`regmethod`	Method to use for linear regression between scale (lvec) and height range. One of `raw` (all data), `mean` (default) `median` or `ends` (minimum and maximum scale only)
`keep_data`	Keep the data used for fd calculation? defaults to FALSE
`plot`	Logical. Show plot of scales relative to data?
`parallel`	Logical. Use parallel processing? Note: parallel must be installed.
`ncores`	Number of cores to use when parallel = TRUE.

Details

Calculates fractal dimension using the height variation regression. If `lvec` is not specified, a default based on resolution and extent will be used. `data` can be a DEM or a `data.frame` with columns labeled `l` and `h` for grid cell length and height range of that cell, respectively (output of `hvar()`). A rule of thumb is that `l` should range an order of magnitude. However, large ranges also average-out fractal dimension of a surface that might have phase transitions, and therefore a thorough exploration of height ranges is suggested using the `plot`. `regmethod` specifies whether data is summarized by taking the mean or median of height ranges across scales or all data is used. `regmethod` "raw" is not recommended because the regression will give much more weight to the lower scales that include more points and likely underestimate D.

Value

A value for fractal dimension, typically between 2 and 3 or a list if keep_data = TRUE.

Examples

```
dem <- habtools::dem_crop(horseshoe, x0 = -469, y0 = 1267, L = 2, plot = TRUE)
fd_hvar(dem, lvec = c(0.125, 0.25, 0.5, 1, 2))
fd_hvar(dem, regmethod = "mean", plot = TRUE, keep_data = TRUE)
fd_hvar(dem, regmethod = "median", plot = TRUE, keep_data = TRUE)
fd_hvar(dem)
```

---

fd_sd — *Calculate fractal Dimension using the standard deviation method*

Description

Calculate fractal Dimension using the standard deviation method

Usage

```
fd_sd(
  data,
  lvec,
  regmethod = "mean",
  keep_data = FALSE,
  plot = FALSE,
  parallel = FALSE,
  ncores = (parallel::detectCores() - 1)
)
```

Arguments

`data`	Digital elevation model of class RasterLayer.
`lvec`	Vector of scales to use for calculation.
`regmethod`	Method to use for linear regression between scale (lvec) and height range. One of `raw` (all data), `mean` (default) `median` or `ends` (minimum and maximum scale only)
`keep_data`	Logical. Keep the data used for fd calculation? Defaults to FALSE.
`plot`	Logical. Show plot of scales relative to data?
`parallel`	Logical. Use parallel processing? Note: parallel must be installed.
`ncores`	Number of cores to use when parallel = TRUE.

Details

Calculates fractal dimension using the standard deviation method, an analogue of the variation method, but using the standard deviation in height per grid cell instead of the full height range. If `lvec` is not specified, a default based on resolution and extent will be used. A rule of thumb is that `lvec` should range at least an order of magnitude. However, large ranges also average-out fractal dimension of a surface that might have phase transitions, and therefore a thorough exploration of height ranges is suggested using the `plot`. `regmethod` specifies whether data is summarized by taking the mean or median of height ranges across scales or all data is used. `regmethod` "raw" is not recommended because the regression will give much more weight to the lower scales that include more points and likely underestimate D.

Value

A value for fractal dimension, typically between 2 and 3 or a list if keep_data = TRUE.

Examples

```
dem <- habtools::dem_crop(horseshoe, x0 = -469, y0 = 1267, L = 2, plot = TRUE)
fd_sd(dem, lvec = c(0.125, 0.25, 0.5, 1, 2))
```

`horseshoe` *Horseshoe reef*

Description

A digital elevation model (DEM) of a reef patch in the Great Barrier Reef.

Usage

```
horseshoe
```

Format

A 800 by 800 digital elevation model (of class RasterLayer).

Values depth

Resolution 0.01 m

Extent 8 m ...

Examples

```
raster::plot(habtools::horseshoe)
```

hr — *Calculate height range*

Description

Calculates the distance between the lowest and highest point in a 3D object.

Usage

```
hr(data)
```

Arguments

`data`	A RasterLayer or mesh3d object.

Value

Value of height range.

Examples

```
# for a DEM
hr(horseshoe)

# for a 3D mesh
hr(mcap)
```

hvar — *Calculate height variation in cells at different scales*

Description

This is a helper function used for calculating fractal dimension using the height variation and standard deviation methods.

Usage

```
hvar(
  data,
  lvec = NULL,
  parallel = FALSE,
  ncores = (parallel::detectCores() - 1)
)
```

Arguments

`data`	Digital elevation model of class RasterLayer.
`lvec`	Scales to use for calculation.
`parallel`	Logical. Use parallel processing? Note: parallel must be installed.
`ncores`	Number of cores to use when parallel = TRUE.

Value

A `data.frame` containing height ranges of cells at different scales.

Examples

```
hvar(horseshoe, lvec = c(1, 2, 4, 8))
```

`mcap` *Montipora capitata*

Description

A laser scan of a coral colony.

Usage

```
mcap
```

Format

mesh3d object with 5568 vertices, 10939 triangles.

Examples

```
library(rgl)
plot3d(mcap)
```

mcap2 *Montipora capitata 2*

Description

A remeshed version of mcap with resolution = 0.005.

Usage

```
mcap2
```

Format

mesh3d object.

Examples

```
library(rgl)
plot3d(mcap2)
```

mesh_to_2d *Transform 3D mesh into 2D outline*

Description

Turns a 3D Mesh file into an xy data frame.

Usage

```
mesh_to_2d(mesh, L0 = NULL, plot = FALSE, silent = TRUE)
```

Arguments

mesh	A mesh3d object.
L0	(Optional) The desired DEM resolution in same units at the 3D mesh.
plot	logical. Plot the output?
silent	logical. Defaults to not showing warnings.

Details

The function uses the vertices of the mesh object and projects them on the XY plane. Then, only points that define the perimeter of the shape are maintained.

Value

A data frame.

Examples

```
mcap_2d <- mesh_to_2d(mcap, plot = TRUE)

geometry::polyarea(mcap_2d$x, mcap_2d$y) # area
planar(mcap)

perimeter(mcap_2d) # perimeter
circularity(mcap_2d) # circularity
fd_boxes(mcap_2d) # fractal dimension
```

`mesh_to_dem` *Transform 3D mesh to DEM*

Description

Turns a 3D mesh file into a Digital Elevation Model (DEM) of class RasterLayer format.

Usage

```
mesh_to_dem(mesh, res, fill = TRUE)
```

Arguments

`mesh`	A mesh3d object.
`res`	(Optional) The desired DEM resolution in same units at the 3D mesh.
`fill`	Logical. Fill `NA` values in raster with minimum value?

Details

The function rasterizes uses the vertices of the mesh file. If resolution is not given, it is calculated by finding the maximum nearest neighbor of vertices projected on the `xy` plane. `fill` is used when irregular 3D meshes result in `NA` values in raster cells. The default is to fill these cells with the minimum, non-`NA` raster value.

Value

A dem of class RasterLayer.

Examples

```
dem <- mesh_to_dem(mcap)
raster::plot(dem)

dem <- mesh_to_dem(mcap, res = 0.05)
raster::plot(dem)

# Don't fill empty raster cells
```

```
dem <- mesh_to_dem(mcap, res = 0.02, fill = FALSE)
raster::plot(dem)
```

`mesh_to_points` *Transform mesh to 3D point cloud*

Description

Transform mesh to 3D point cloud

Usage

```
mesh_to_points(mesh)
```

Arguments

`mesh`	A triangular mesh of class mesh3d.

Value

A data frame with XYZ coordinates.

`mid_find` *Find midpoint of a DEM*

Description

Find midpoint of a DEM

Usage

```
mid_find(data)
```

Arguments

`data`	A DEM in RasterLayer format.

Value

A data frame with x and y midpoints.

Examples

```
mid_find(horseshoe)
```

`packing` *Calculate packing of 3D object*

Description

The ratio of the surface area of the object and the surface area of the convex hull around the object.

Usage

```
packing(mesh)
```

Arguments

mesh	A triangular mesh of class mesh3d.

Value

Value of packing.

Examples

```
packing(mcap)
```

`perimeter` *Calculate perimeter of a 2D shape*

Description

Calculates the perimeter of a 2D shape.

Usage

```
perimeter(data, keep_data = FALSE)
```

Arguments

data	A data frame with the first two columns ordered x and y coordinates.
keep_data	Logical. Keep lengths of all segments of the perimeter? Defaults to FALSE.

Value

The perimeter.

Examples

```
mcap_2d <- mesh_to_2d(mcap)
plot(mcap_2d)

perimeter(mcap_2d)

r <- 1 # radius
circ <- sim_circle(r=r) # simulate xy coordinates for a circle of radius 1
plot(circ, asp=1)
perimeter(circ)

2 * pi * r # Note xy resolution affects output
```

`planar` *Calculates planar area of a mesh*

Description

Calculates planar area of a mesh

Usage

```
planar(mesh, L0, silent = FALSE)
```

Arguments

`mesh`	A triangular mesh of class mesh3d.
`L0`	Resolution of the planar area. Is set to the resolution of the mesh when left empty.
`silent`	Logical. Suppress messages and warnings?

Value

A value for planar area.

Examples

```
planar(mcap)
```

rdh — *Calculate rugosity, fractal dimension, and height for a DEM*

Description

Calculate rugosity, fractal dimension, and height for a DEM

Usage

```
rdh(
  data,
  lvec,
  method_fd = "hvar",
  method_rg = "area",
  parallel = FALSE,
  ncores = (parallel::detectCores() - 1),
  ...
)
```

Arguments

`data`	A dem of class RasterLayer.
`lvec`	Scales to use for calculation.
`method_fd`	method for the calculation of rugosity and fractal dimension. Can be "hvar", "sd", "cubes", or "area". Defaults to "hvar".
`method_rg`	Method to be used for the rugosity calculation. Defaults to "area".
`parallel`	Logical. Use parallel processing? Defaults to FALSE.
`ncores`	Number of cores to use if parallel = TRUE.
`...`	Additional arguments see `fd()`.

Details

Uses area method for rugosity and hvar method for fractal dimension calculations as default.

Value

A dataframe with the three complexity metrics.

See Also

`fd()`

`rg()`

`hr()`

Examples

```
dem <- dem_sample(horseshoe, L = 1)
rdh(dem, lvec = c(0.125, 0.25, 0.5, 1))
```

---

rdh_theory *Calculate metric based on geometric plane equation*

Description

Calculates either rugosity, fractal dimension or height range based on the other two variables.

Usage

```
rdh_theory(R, D, H, L, L0)
```

Arguments

R, D, H	Two of the three variables to calculate the third.
L	Extent.
L0	Resolution.

Details

This function uses the geometric plane equation from Torres-Pulliza et al. (2020) to calculate one of rugosity, fractal dimension or height range based on the other two variables.

Value

A value corresponding one of the three variables not given to the function.

References

Torres-Pulliza D, Dornelas M, Pizarro O, Bewley M, Blowes SA, Boutros N, Brambilla V, Chase TJ, Frank G, Friedman A, Hoogenboom MO, Williams S, Zawada KJA, Madin JS (2020) A geometric basis for surface habitat complexity and biodiversity. *Nature Ecology & Evolution* 4:1495-1501. doi:10.1038/s4155902012818

Examples

```
rdh_theory(R=4, H=1, L=1, L0=0.01)
rdh_theory(D=2.36928, H=1, L=1, L0=0.01)
rdh_theory(D=2.36928, R=4, L=1, L0=0.01)
```

rg *Calculate rugosity*

Description

Rugosity is defined as the surface area divided by the planar area. For digital elevation models, there are two methods: "hvar" and "area". The "hvar" method for calculating rugosity is described in Torres-Pulliza et al. (2004) and is based on height variations. The "area" method uses the `sp::surfaceArea()` function and is detailed in Jenness (2004). `method` is ignored if `data` is a mesh3D object. In that case the function uses `Rvcg::vcgArea()` to calculate surface area of a triangular mesh of class mesh3d.

Usage

```
rg(
  data,
  L0,
  method = "area",
  parallel = FALSE,
  ncores = (parallel::detectCores() - 1)
)
```

Arguments

`data`	Digital elevation model of class RasterLayer or a triangular mesh of class mesh3d.
`L0`	Grain or resolution of calculation.
`method`	If data is a RasterLayer methods "hvar" or "area" are allowed. Defaults to "hvar".
`parallel`	Logical. Use parallel processing? Defaults to FALSE.
`ncores`	Number of cores to use if parallel = TRUE. (Defaults to umber of available cores - 1)

Value

Rugosity value

References

Jenness, J.S. Calculating Landscape Surface Area from Digital Elevation Models. Wildlife Society Bulletin, Vol. 32, No. 3 (Autumn, 2004), pp. 829-839n

Torres-Pulliza, D., Dornelas, M.A., Pizarro, O. et al. A geometric basis for surface habitat complexity and biodiversity. Nat Ecol Evol 4, 1495–1501 (2020). https://doi.org/10.1038/s41559-020-1281-8

Examples

```
rg(horseshoe, L0 = 0.1)
rg(mcap, L0 = 0.01)
```

---

sa_triangle *Calculate surface area of triangle*

Description

Calculates the surface area of a triangle based on a set of XYZCoords.

Usage

```
sa_triangle(XYZcoords)
```

Arguments

XYZcoords	A data frame with XYZ coordinates of three points in following order: X1,X2,X3,Y1,Y2,Y3,Z1,Z2,Z3.

Value

The surface area of the triangle.

Examples

```
sa_triangle(c(X1 = 1, X2 = 2, X3 = 3 , Y1 = 1, Y2 = 2, Y3 = 1, Z1 = 1, Z2 = 1, Z3 = 1))
```

---

scale_area *Re-scale mesh based on a fixed area*

Description

Re-scale mesh based on a fixed area

Usage

```
scale_area(mesh, target_area = 1)
```

Arguments

mesh	A triangular mesh of class mesh3d.
target_area	The target area of the scaled 3D mesh. Defaults to 1.

Value

A mesh with area = target_area (1 as default).

Examples

```
Rvcg::vcgArea(mcap)
mcap_scaled <- scale_area(mcap)
Rvcg::vcgArea(mcap_scaled)
```

`scale_volume` *Re-scale mesh based on a fixed volume of 1*

Description

Re-scale mesh based on a fixed volume of 1

Usage

```
scale_volume(mesh)
```

Arguments

`mesh`	A triangular mesh of class mesh3d.

Value

A mesh with volume = 1.

Examples

```
Rvcg::vcgVolume(mcap)
mcap_scaled <- scale_volume(mcap)
Rvcg::vcgVolume(mcap_scaled)
```

`set_origin` *Set the origin of a mesh*

Description

Transforms XYZ coordinates relative to a chosen origin

Transforms coordinates so that the origin lies at the reference vertex (defaults to the minimum of x, y, and z coordinates).

Usage

```
set_origin(mesh, reference = NULL)
```

Arguments

`mesh`	A triangular mesh of class mesh3d.
`reference`	Vector containing coordinates of the reference vertex. If left empty, this will default to the minimum of x, y, and z.

Value

mesh3d object

Examples

```
mesh <- set_origin(mcap)
```

`sim_circle` *Simulate a circle*

Description

Simulates xy coordinates for a circle of given radius. Created for package testing purposes, but might be useful for others.

Usage

```
sim_circle(r = 1, n = 100, mid = c(0, 0))
```

Arguments

`r`	Radius of the circle (default 1).
`n`	Number of xy coordinates defining the circle (default 100).
`mid`	Mid point of the circle (default 0, 0).

Value

A data frame of n xy-coordinates.

Examples

```
circ <- sim_circle()
plot(circ)

circularity(circ)
perimeter(circ)
```

sim_dem *Simulates a fractal DEM*

Description

Simulates z-values based on the Diamond-square algorithm.

Usage

```
sim_dem(
  L,
  smoothness,
  H,
  R,
  plot = FALSE,
  prop = 0.1,
  n = 100,
  method = "area",
  parallel = FALSE
)
```

Arguments

`L`	The extent.
`smoothness`	A value between 0.0 and 1.0 (lower values produce rougher DEM).
`H`	Desired height range (optional).
`R`	Desired rugosity value (optional).
`plot`	Logical. Plot the simulated DEM during simulation? Only relevant if R is provided.
`prop`	Proportion of cells that undergo smoothing at each iteration when R is provided.
`n`	Number of iterations to try and reach desired R. Recommended to adapt R and H instead of increasing n if simulation fails.
`method`	The method to be used for rugosity calculation in case R is given. Can be "hvar" or "area"
`parallel`	Logical. Use parallel processing? Defaults to FALSE. Only relevant if method = "hvar".

Details

Warning: this function gets slow for n > 128. If H is provided, the simulated DEM is rescaled based on the value for H. If R is provided, a DEM is simulated using the same algorithm based on R, H, and the predicted D based on `rdh_theory()`, while smoothness is ignored. From that first simulated DEM, R is calculated and the DEM undergoes smoothing at each iteration until the rugosity approximates the inputted R. Argument prop defined the proportion of random cells of the DEM that are smoothed by averaging the z values of cell and neighboring cells at each iteration.

Caution: When R is provided, the DEM may become increasingly less fractal as it is modified at each iteration.

Value

Digital elevation model of class RasterLayer.

Examples

```
library(raster)
dem <- sim_dem(L = 32, smoothness = 0.5)
plot(dem)
dem <- sim_dem(L = 32, smoothness = 0.2, H = 20)
plot(dem)
```

---

sma *Calculate second moment of area*

Description

Calculates the 2nd moment of surface area about the origin by multiplying the surface area of each triangle in the mesh by its distance from the origin (should be set to the attachment point of the mesh). The sum of these values is the 2nd moment of area.#' This metric is size-dependent so to compare moments in terms of shape only, set scale = TRUE.

Usage

```
sma(mesh, axis = "z", scale = FALSE, origin = TRUE)
```

Arguments

`mesh`	A triangular mesh of class mesh3d.
`axis`	The axis along which to calculate the second moment of area. z is the default.
`scale`	Logical. Scale the object to have a volume = 1? Default = FALSE
`origin`	Logical. Set the origin to the bottom left corner of bounding box? Default = TRUE.

Value

SMA value.

Examples

```
sma(mcap)
sma(mcap, scale = TRUE)
```

smv — *Calculate second moment of volume*

Description

Calculates the 2nd moment of volume (SMV) by multiplying the volume of each triangle in the mesh by its centroids' distance from the origin (should be set to the attachment point of the mesh). The sum of these values is the 2nd moment of volume. Axis is z by default, meaning it will calculate the vertical second moment, but this can be changed if needed. This metric is size-dependent so to compare moments in terms of shape only, set scale = TRUE.

Usage

```
smv(mesh, axis = "z", scale = FALSE, origin = TRUE)
```

Arguments

`mesh`	A triangular mesh of class mesh3d.
`axis`	The axis along which to calculate the second moment of volume z is the default.
`scale`	Logical. Scale the object to have a volume = 1? Default = TRUE.
`origin`	Logical. Set the origin to the bottom left corner of bounding box? Default = FALSE

Value

SMV value.

Examples

```
smv(mcap)
```

sphericity — *Calculate sphericity of a 3D object*

Description

Calculates the ratio of the surface area of a sphere with the same volume as the object and the surface area of the object.

Usage

```
sphericity(mesh)
```

Arguments

`mesh`	A triangular mesh of class mesh3d.

Value

Sphericity value.

See Also

`circularity()`

Examples

```
sphericity(mcap)
```

`surface_area` *Calculate surface area*

Description

Calculates surface area of a 3D or 2D object.

Usage

```
surface_area(data)
```

Arguments

`data`	DEM in RasterLayer format, mesh3d object or data frame with xy coordinates.

Value

Surface area value.

References

Jenness, J.S. Calculating Landscape Surface Area from Digital Elevation Models. Wildlife Society Bulletin, Vol. 32, No. 3 (Autumn, 2004), pp. 829-839n

Examples

```
surface_area(mcap)
surface_area(horseshoe)
surface_area(mesh_to_2d(mcap))
```

svol_triangle *Calculate signed volume of triangle*

Description

Calculates the signed volume of a triangle based on a set of XYZCoords. Signed volume means that volumes can take on a negative value depending on whether the surface normal of the triangle is facing towards or away from the origin. When all positive and negative volumes are integrated across the entire mesh, these values cancel out so that the final volume is an approximation of the total volume of the mesh.

Usage

```
svol_triangle(XYZCoords)
```

Arguments

`XYZCoords`	A dataframe with XYZ coordinates of three points in following order: X1,X2,X3,Y1,Y2,Y3,Z1,Z2,Z3

Value

Value for the signed volume of a triangle.

Examples

```
svol_triangle(c(X1 = 1, X2 = 2, X3 = 3 , Y1 = 1, Y2 = 2, Y3 = 1, Z1 = 1, Z2 = 1, Z3 = 1))
```

z *Extract mean depth or elevation of a DEM*

Description

Extract mean depth or elevation of a DEM

Usage

```
z(data)
```

Arguments

`data`	A DEM in RasterLayer format.

Value

Value: mean depth or elevation of DEM.

Examples

```
z(horseshoe)
```

Index