Package: guerrilla (via r-universe)

September 6, 2024

Version 0.2.0.9001

Title Illustrate Various Methods of Interpolation for Irregular Data

Description Examples of interpolating irregular data, to illustrate the mechanics of various methods and some easy tools to run them.

License GPL-3

Encoding UTF-8

LazyData true

ByteCompile true

URL https://github.com/hypertidy/guerrilla, https://hypertidy.github.io/guerrilla/

BugReports https://github.com/hypertidy/guerrilla/issues

Depends R (>= 3.3.0)

Imports raster, sp

Suggests anglr, Rvcg, akima, knitr, spatstat.geom, geometry (>= 0.4.0), viridis, rmarkdown, fields, gstat, palr, readxl, reproj, covr, testthat, mgcv, dplyr, maps, rgl

VignetteBuilder knitr

Roxygen list(markdown = TRUE)

Remotes hypertidy/anglr

RoxygenNote 7.1.1

Repository https://hypertidy.r-universe.dev

RemoteUrl https://github.com/hypertidy/guerrilla

RemoteRef HEAD

RemoteSha 8e95d43481b652dc3f93d0f3415efd1585fcc884

Contents

`bathy` *Bathymetry data.*

Description

`bathy` is a simple polygon region layer to sit over the SST data.

Format

A raster

`defaultgrid` *Title*

Description

Title

Usage

```
defaultgrid(
  xy,
  ncols = 60,
  nrows = 50,
  prj = "+proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs +towgs84=0,0,0"
)
```

Arguments

`xy`	coordinates
`ncols`	number of columns
`nrows`	number of rows
`prj`	projection metadata

Value

raster

facets *Title*

Description

Title

Usage

```
facets(
  X,
  nx,
  ny,
  x = NULL,
  y = NULL,
  na.v = 0,
  method = c("dirichlet", "delaunay")
)
```

Arguments

X	spatstat object
nx	number of x coords
ny	number of y coords
x	option input x values
y	optional input y values
na.v	na value
method	dirichlet or delaunay

Value

ppp object

mesh_raster *Mesh raster*

Description

Create a raster by interpolating across triangles

Usage

```
mesh_raster(x, grid = NULL, n = 128)

## S3 method for class 'mesh3d'
mesh_raster(x, grid = NULL, n = 128)

## S3 method for class 'matrix'
mesh_raster(x, grid = NULL, n = 128)

## S3 method for class 'data.frame'
mesh_raster(x, grid = NULL, n = 128)
```

Arguments

x	matrix of points, or a mesh3d
grid	raster to populate
n	grid size of raster if 'grid' not supplied

Details

At the moment, mesh_raster is identical to tri_fun for the matrix x-y-z case, but adds capability for a mesh3d object (of triangles). Barycentric interpolation is used to efficiently obtain a within-triangle estimate of a field of values

Value

Raster

Examples

```
data("humface", package = "Rvcg")
x <- humface
grid <- mesh_raster(x, n = 256)
raster::plot(grid, col = grey.colors(21),
    breaks = quantile(grid, seq(0, 1, length = 22), na.rm = TRUE))
anglr::plot3d(grid)

## interpolate from raw points
xyz <- quakes[c("long", "lat", "depth")]
xyz$depth <- -xyz$depth
gx <- mesh_raster(xyz)
rat <- 1/cos(mean(xyz[["lat"]]) * pi/180)
raster::image(gx, asp = rat,
  col = hcl.colors(12, "YlOrRd"))
maps::map(add = TRUE)
points(xyz, pch = "+", cex = 0.3)
## add some dummy points (we aren't modelling the world)

xex <- cbind(expand.grid(long = range(xyz$long),
                         lat = range(xyz$lat)), depth = 0)
```

```
g2 <- mesh_raster(rbind(xex, xyz))
raster::image(g2, asp = rat)
maps::map(add = TRUE)
points(xyz, pch = "+", cex = 0.3)
anglr::plot3d(g2); rgl::aspect3d(1, rat, 0.1)
rgl::points3d(xyz$long, xyz$lat, xyz$depth + 30)
```

`tri_fun` *Interpolation to a regular grid via triangulation*

Description

Interpolation to a regular grid via triangulation

Usage

```
tri_fun(xy, value, grid = NULL, ...)
```

Arguments

`xy`	coordinates
`value`	value to interpolate
`grid`	grid to use
`...`	ignored

Value

raster

Examples

```
zero_extent <- raster::extent(0, ncol(volcano), 0, nrow(volcano))
r <- raster::setExtent(raster::raster(volcano), zero_extent)
xy <- raster::sampleRandom(r, size = 150, xy = TRUE)[, 1:2, drop = FALSE]
tri_est <- tri_fun(xy, raster::extract(r, xy))

grd <- raster::raster(raster::extent(xy) ,res = 0.1)
tri_est2 <- tri_fun(xy, raster::extract(r, xy), grid = grd)
```

tri_pip *Identify point-in-triangle by conversion to polygons*

Description

This functio was used by an early version of `tri_fun`.

Usage

```
tri_pip(tri, pts)
```

Arguments

`tri`	list P n*2 coordinates and T matrix of n*3 indices defining triangles
`pts`	input points

Index