Package: potential (via r-universe)

August 20, 2024

Title Implementation of the Potential Model

Version 0.3.0.0

Description Provides functions to compute the potential model as defined by Stewart (1941) <doi:10.1126/science.93.2404.89>. Several options are available to customize the model, such as the possibility to fine-tune the distance friction functions or to use custom distance matrices. Some computations are parallelized to improve their efficiency.

Depends R (>= 3.5.0)

License GPL-3

LazyData true

Imports sf, graphics, mapiso, parallel, doParallel, foreach

Suggests covr, lwgeom, eurostat, giscoR, mapsf, knitr, tinytest, rmarkdown

URL https://github.com/riatelab/potential

BugReports https://github.com/riatelab/potential/issues

Encoding UTF-8

VignetteBuilder knitr

RoxygenNote 7.2.3

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

RemoteUrl https://github.com/riatelab/potential

RemoteRef HEAD

RemoteSha 3dc649b6661a2595754d3b848198305a5f3c3fd6

Contents

`potential-package` *Implementation of the Potential Model*

Description

This package provides functions to compute the potential model as defined by Stewart (1941) <doi:10.1126/science.93.2404.89>. Several options are available to customize the model, such as the possibility to fine-tune the distance friction functions or to use custom distance matrices. Some computations are parallelized to improve their efficiency.

`create_grid` *Create a Regularly Spaced Points Grid*

Description

This function creates a regular grid of points from the extent of a given spatial object and a given resolution.

Usage

```
create_grid(x, res)
```

Arguments

`x`	an sf or sfc object, the spatial extent of this object is used to create the regular grid.
`res`	resolution of the grid (in map units).

Value

The output of the function is an sf object of regularly spaced points with the extent of x.

Examples

```
library(sf)
g <- create_grid(x = n3_poly, res = 200000)
plot(st_geometry(g))
plot(st_geometry(n3_poly), border = "red", add = TRUE)
```

`create_matrix` *Create a Distance Matrix Between Two Spatial Objects*

Description

This function creates a distance matrix between two spatial objects.

Usage

```
create_matrix(x, y, checksize = TRUE, longlat = FALSE)
```

Arguments

`x`	an sf object (POINT), rows of the distance matrix, row names are used as row names of the matrix.
`y`	an sf object (POINT), columns of the distance matrix, row names are used as column names of the matrix.
`checksize`	if FALSE, bypass the distance matrix size control (see Details).
`longlat`	if FALSE, the Euclidean distance is used, if TRUE Great Circle (WGS84 ellipsoid) distance is used.

Details

The function returns a full matrix of distances in meters. If the matrix to compute is too large (more than 100,000,000 cells, more than 10,000,000 origins or more than 10,000,000 destinations) the function may sends a message to warn users about the amount of RAM mobilized.

Value

A distance matrix, row names are `x` row names, column names are `y` row names.

Examples

```
g <- create_grid(x = n3_poly, res = 200000)
mat <- create_matrix(x = n3_pt, y = g)
mat[1:5, 1:5]
```

`equipotential` *Create Polygons of Equipotential*

Description

This function creates polygons of equipotential from a regular grid of potential points.

Usage

```
equipotential(x, var, nclass = 8, breaks, mask, buffer, xcoords, ycoords)
```

Arguments

`x`	an sf object of regularly spaced points.
`var`	name of the variable to use in `x`.
`nclass`	a number of class.
`breaks`	a vector of break values.
`mask`	an sf object of polygons or multipolygons. `mask` is used to clip polygons of contours equipotential.
`buffer`	if set, a buffer is added to the mask in order to reach more precisely the number of breaks. The buffer is defined in `x` units.
`xcoords`	not used.
`ycoords`	not used.

Value

The output is an sf object (POLYGONS). The data frame contains four fields: id (id of each polygon), min and max (minimum and maximum breaks of the polygon) and center (central values of classes).

Examples

```
library(sf)
y <- create_grid(x = n3_poly, res = 200000)
d <- create_matrix(n3_pt, y)
pot <- potential(
  x = n3_pt, y = y, d = d, var = "POP19",
  fun = "e", span = 200000, beta = 2
)
y$OUTPUT <- pot
equipot <- equipotential(y, var = "OUTPUT", mask = n3_poly)
plot(equipot["center"], pal = hcl.colors(nrow(equipot), "cividis"))
```

`mcpotential` *Compute the Potential Model using Parallelization*

Description

This function computes the potential model with a cutoff distance and parallel computation.

Usage

```
mcpotential(x, y, var, fun, span, beta, limit = 3 * span, ncl, size = 500)
```

Arguments

`x`	an sf object (POINT), the set of known observations to estimate the potentials from.
`y`	an sf object (POINT), the set of unknown units for which the function computes the estimates.
`var`	names of the variables in `x` from which potentials are computed. Quantitative variables with no negative values.
`fun`	spatial interaction function. Options are "p" (pareto, power law) or "e" (exponential). For pareto the interaction is defined as: (1 + alpha * mDistance) ^ (-beta). For "exponential" the interaction is defined as: exp(- alpha * mDistance ^ beta). The alpha parameter is computed from parameters given by the user (`beta` and `span`).
`span`	distance where the density of probability of the spatial interaction function equals 0.5.
`beta`	impedance factor for the spatial interaction function.
`limit`	maximum distance used to retrieve `x` points, in map units.
`ncl`	number of clusters. `ncl` is set to `parallel::detectCores() - 1` by default.
`size`	`mcpotential` splits `y` in smaller chunks and dispatches the computation in `ncl` cores, `size` indicates the size of each chunks.

Value

If only one variable is computed a vector is returned, if more than one variable is computed a matrix is returned.

Examples

```
library(sf)
g <- create_grid(x = n3_poly, res = 20000)
pot <- mcpotential(
  x = n3_pt, y = g, var = "POP19",
  fun = "e", span = 75000, beta = 2,
  limit = 300000,
```

```
  ncl = 2
)
g$OUTPUT <- pot
equipot <- equipotential(g, var = "OUTPUT", mask = n3_poly)
plot(equipot["center"], pal = hcl.colors(nrow(equipot), "cividis"))
```

`n3_poly` *Points and Polygons Layers of European Statistical Units (NUTS3)*

Description

n3_pt (POINTS) and n3_poly (MULTIPOLYGONS) are sf objects of 1506 NUTS3 statistical units of continental Europe.

Population dataset (2019 an 2018 total population) downloaded on the Eurostat website (05/10/2020) from the "demo_r_pjanaggr3" dataset (last update: 16/06/2020).

Geometries are downloaded from the GISCO website (NUTS3 - 2016 - 1:60 Million)

When data from this packgage is used in any printed or electronic publication, in addition to any other provisions applicable to the whole Eurostat website, data source will have to be acknowledged in the legend of the map and in the introductory page of the publication with the following copyright notice: "© EuroGeographics for the administrative boundaries and © Eurostat for data".

Usage

```
data(nuts3)
```

Format

An object of class `sf` (inherits from `data.frame`) with 1506 rows and 4 columns.

`n3_pt` *Points and Polygons Layers of European Statistical Units (NUTS3)*

Description

n3_pt (POINTS) and n3_poly (MULTIPOLYGONS) are sf objects of 1506 NUTS3 statistical units of continental Europe.

Population dataset (2019 an 2018 total population) downloaded on the Eurostat website (05/10/2020) from the "demo_r_pjanaggr3" dataset (last update: 16/06/2020).

Geometries are downloaded from the GISCO website (NUTS3 - 2016 - 1:60 Million)

When data from this package is used in any printed or electronic publication, in addition to any other provisions applicable to the whole Eurostat website, data source will have to be acknowledged in the legend of the map and in the introductory page of the publication with the following copyright notice: "© EuroGeographics for the administrative boundaries and © Eurostat for data".

Usage

```
data(nuts3)
```

Format

An object of class `sf` (inherits from `data.frame`) with 1506 rows and 4 columns.

---

plot_inter *Display a Spatial Interaction Function*

Description

Display a spatial interaction function.

Usage

```
plot_inter(fun = "e", span, beta, limit = span * 5)
```

Arguments

`fun`	spatial interaction function. Options are "p" (pareto, power law) or "e" (exponential). For pareto the interaction is defined as: (1 + alpha * mDistance) ^ (-beta). For "exponential" the interaction is defined as: exp(- alpha * mDistance ^ beta). The alpha parameter is computed from parameters given by the user (`beta` and `span`).
`span`	distance where the density of probability of the spatial interaction function equals 0.5.
`beta`	impedance factor for the spatial interaction function.
`limit`	maximum distance used to retrieved `x` points, in map units.

Value

a plot

Examples

```
plot_inter(fun = "e", span = 2000, beta = 2, limit = 4000)
plot_inter(fun = "p", span = 2000, beta = 2, limit = 20000)
```

potential *Compute the Potential Model*

Description

This function computes the potential model as defined by J.Q. Stewart (1941).

Usage

```
potential(x, y, d, var, fun, span, beta)
```

Arguments

x	an sf object (POINT), the set of known observations to estimate the potentials from.
y	an sf object (POINT), the set of unknown units for which the function computes the estimates.
d	a distance matrix between known observations and unknown units for which the function computes the estimates. Row names match the row names of x and column names match the row names of y. d can contain any distance metric (time distance or euclidean distance for example).
var	names of the variables in x from which potentials are computed. Quantitative variables with no negative values.
fun	spatial interaction function. Options are "p" (pareto, power law) or "e" (exponential). For pareto the interaction is defined as: (1 + alpha * mDistance) ^ (-beta). For "exponential" the interaction is defined as: exp(- alpha * mDistance ^ beta). The alpha parameter is computed from parameters given by the user (beta and span).
span	distance where the density of probability of the spatial interaction function equals 0.5.
beta	impedance factor for the spatial interaction function.

Value

If only one variable is computed a vector is returned, if more than one variable is computed a matrix is returned.

References

STEWART, JOHN Q. 1941. "An Inverse Distance Variation for Certain Social Influences." *Science* 93 (2404): 89–90. doi:10.1126/science.93.2404.89.

Examples

```
library(sf)
y <- create_grid(x = n3_poly, res = 200000)
d <- create_matrix(n3_pt, y)
pot <- potential(
  x = n3_pt, y = y, d = d, var = "POP19",
  fun = "e", span = 200000, beta = 2
)
y$OUTPUT <- pot
equipot <- equipotential(y, var = "OUTPUT", mask = n3_poly)
plot(equipot["center"], pal = hcl.colors(nrow(equipot), "cividis"))
```

Index