Package: geocomplexity (via r-universe)

September 17, 2024

Title Geocomplexity Mitigates Spatial Bias

Version 0.0.0.9000

Description The geographical complexity of individual variables can be characterized using the differences in local attribute variables, while the common geographical complexity of multiple variables can be represented by the fluctuations in similarity of vectors composed of multiple variables. In spatial regression tasks, the goodness of fit can be improved by incorporating a geographical complexity representation vector during modeling, using a geographical complexity-weighted spatial weight matrix, or employing local geographical complexity kernel density. Similarly, in spatial sampling tasks, samples can be selected more effectively using a method that weights based on geographical complexity. By optimizing performance in spatial regression and spatial sampling tasks, the spatial bias of the model can be effectively reduced.

License GPL-3

Encoding UTF-8

URL https://ausgis.github.io/geocomplexity/, https://github.com/ausgis/geocomplexity

BugReports https://github.com/ausgis/geocomplexity/issues

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Depends R (>= 4.1.0)

Imports dplyr, magrittr, pander, purrr, sdsfun, sf, terra, tibble

Suggests ggplot2, knitr, Rcpp, RcppArmadillo, rmarkdown, tidyverse, viridis

LinkingTo Rcpp, RcppArmadillo

VignetteBuilder knitr

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

RemoteUrl https://github.com/ausgis/geocomplexity

RemoteRef HEAD

RemoteSha 4ba2305af9dd255a2dd61b1887600083095ff929

Contents

`geocd_raster` *geocomplexity for raster data based on spatial dependence*

Description

This function calculates geocomplexity for raster data based on spatial dependence.

Usage

```
geocd_raster(r, order = 1, normalize = TRUE, method = "moran")
```

Arguments

`r`	Raster object that can be converted to `SpatRaster` by `terra::rast()`.
`order`	(optional) The order of the adjacency object. Default is `1`.
`normalize`	(optional) Whether to further normalizes the calculated geocomplexity. Default is `TRUE`.
`method`	(optional) In instances where the method is `moran`, geocomplexity is determined using local moran measure method. Conversely, when the method is `spvar`, the spatial variance of attribute data serves to characterize geocomplexity. For all other methods, the shannon information entropy of attribute data is employed to represent geocomplexity. The selection of the method can be made from any one of the three options: `moran`, `spvar` or `entropy`. Default is `moran`.

Value

A SpatRaster object

Note

In contrast to the `geocd_vector()` function, the `geocd_raster()` performs operations internally on raster data based on neighborhood operations(focal) without providing additional wt object.

References

Zehua Zhang, Yongze Song, Peng Luo & Peng Wu (2023) Geocomplexity explains spatial errors, International Journal of Geographical Information Science, 37:7, 1449-1469, DOI: 10.1080/13658816.2023.2203212

Anselin, L. (2019). A local indicator of multivariate spatial association: Extending geary's c. Geographical Analysis, 51(2), 133–150. https://doi.org/10.1111/gean.12164

Examples

```
library(terra)
m = matrix(c(3,3,3,3,1,3,
             3,3,3,2,1,2,
             3,3,3,1,2,1,
             1,3,2,2,2,2,
             2,2,2,1,1,2,
             1,2,1,1,1,1),
           nrow = 6,
           byrow = TRUE)
m = rast(m)
names(m) = 'sim'
plot(m, col = c("#d2eaac", "#a3dae1", "#8cc1e1"))
gc1 = geocd_raster(m,1)
gc2 = geocd_raster(m,2)
gc1
plot(gc1)
gc2
plot(gc2)
```

`geocd_vector` *geocomplexity for vector data based on spatial dependence*

Description

This function calculates geocomplexity for vector data based on spatial dependence.

Usage

```
geocd_vector(
  sfj,
  wt = NULL,
  method = "moran",
  normalize = TRUE,
  returnsf = TRUE
)
```

Arguments

sfj	An sf object or vector object that can be converted to sf by sf::st_as_sf().
wt	(optional) Spatial weight matrix. Must be a matrix class.
method	(optional) In instances where the method is moran, geocomplexity is determined using local moran measure method. Conversely, when the method is spvar, the spatial variance of attribute data serves to characterize geocomplexity. For all other methods, the shannon information entropy of attribute data is employed to represent geocomplexity. The selection of the method can be made from any one of the three options: moran, spvar or entropy. Default is moran.
normalize	(optional) Whether to further normalizes the calculated geocomplexity. Default is TRUE.
returnsf	(optional) When returnsf is TRUE, return an sf object, otherwise a tibble. Default is TRUE.

Details

The formula for geocomplexity which uses local moran measure method is

$$\rho_i = -\frac{1}{m} Z_i \sum_{j=1}^{m} W_{ij} Z_j - \frac{1}{m} \sum_{j=1}^{m} W_{ij} Z_j \frac{1}{V_k} \sum_{k=1}^{n} W_{jk} W_{ik} Z_k$$

Value

A tibble (returnsf is FALSE) or an sf object (returnsf is TRUE)

Note

If wt is not provided, for polygon vector data, geocomplexity will use a first-order queen adjacency binary matrix; for point vector data, the six nearest points are used as adjacency objects to generate an adjacency binary matrix.

References

Zehua Zhang, Yongze Song, Peng Luo & Peng Wu (2023) Geocomplexity explains spatial errors, International Journal of Geographical Information Science, 37:7, 1449-1469, DOI: 10.1080/13658816.2023.2203212

Anselin, L. (2019). A local indicator of multivariate spatial association: Extending geary's c. Geographical Analysis, 51(2), 133–150. https://doi.org/10.1111/gean.12164

Examples

```
econineq = sf::read_sf(system.file('extdata/econineq.gpkg',package = 'geocomplexity'))
gc = geocd_vector(econineq)
gc

library(ggplot2)
library(viridis)
ggplot(gc) +
   geom_sf(aes(fill = GC_Gini)) +
   scale_fill_viridis(option="mako", direction = -1) +
```

```
theme_bw()
```

`geocs_raster` *geocomplexity for raster data based on geographical similarity*

Description

This function calculates geocomplexity for raster data based on geographical similarity.

Usage

```
geocs_raster(r, order = 1, normalize = TRUE, similarity = 1, method = "spvar")
```

Arguments

`r`	Raster object that can be converted to `SpatRaster` by `terra::rast()`.
`order`	(optional) The order of the adjacency object. Default is `1`.
`normalize`	(optional) Whether to further normalizes the calculated geocomplexity. Default is `TRUE`.
`similarity`	(optional) When `similarity` is `1`, the similarity is calculated using geographical configuration similarity, otherwise the cosine similarity is calculated. Default is `1`.
`method`	(optional) When `method` is `spvar`, variation of the similarity vector is represented using spatial variance, otherwise shannon information entropy is used. Default is `spvar`.

Value

A SpatRaster object

Note

In contrast to the `geocs_vector()` function, the `geocs_raster()` performs operations internally on raster data without providing additional wt object.

Examples

```
library(terra)
m1 = matrix(c(3,3,3,3,1,3,
              3,3,3,2,1,2,
              3,3,3,1,2,1,
              1,3,2,2,2,2,
              2,2,2,1,1,2,
              1,2,1,1,1,1),
            nrow = 6,
            byrow = TRUE)
```

```
m1 = rast(m1)
names(m1) = 'sim1'
m2 = m1
set.seed(123456789)
values(m2) = values(m1) + runif(ncell(m1),-1,1)
names(m2) = 'sim2'
m = c(m1,m2)
gc1 = geocs_raster(m,1)
gc2 = geocs_raster(m,2)
gc1
plot(gc1)
gc2
plot(gc2)
```

`geocs_vector` *geocomplexity for vector data based on geographical similarity*

Description

This function calculates geocomplexity for in vector data based on geographical similarity.

Usage

```
geocs_vector(
  sfj,
  wt = NULL,
  method = "spvar",
  similarity = 1,
  normalize = TRUE,
  returnsf = TRUE
)
```

Arguments

`sfj`	An `sf` object or vector object that can be converted to `sf` by `sf::st_as_sf()`.
`wt`	(optional) Spatial weight matrix. Must be a `matrix` class. If `wt` is not provided, `geocomplexity` will use a first-order inverse distance weight matrix via `sdsfun::inverse_distance_swm()` function.
`method`	(optional) When `method` is `spvar`, variation of the similarity vector is represented using spatial variance, otherwise shannon information entropy is used. Default is `spvar`.
`similarity`	(optional) When `similarity` is `1`, the similarity is calculated using geographical configuration similarity, otherwise the cosine similarity is calculated. Default is `1`.
`normalize`	(optional) Whether to further normalizes the calculated geocomplexity. Default is `TRUE`.
`returnsf`	(optional) When `returnsf` is `TRUE`, return an sf object, otherwise a tibble. Default is `TRUE`.

Details

The geographical configuration similarity is calculated as:

$S(\mathbf{u}_\alpha, \mathbf{v}_\beta) = min\{E_i(e_i(\mathbf{u}_\alpha), e_i(\mathbf{v}_\beta))\}$

$E_i(\mathbf{u}_\alpha, \mathbf{v}_\beta) = \exp\left(-\frac{(e_i(\mathbf{u}_\alpha)-e_i(\mathbf{v}_\beta))^2}{2(\sigma^2/\delta(\mathbf{v}_\beta))^2}\right)$

$\delta(\mathbf{u}_\alpha, \mathbf{v}) = \sqrt{\frac{\sum_{\beta=1}^{n}(e(\mathbf{u}_\alpha)-e(\mathbf{v}_\beta))^2}{n}}$

The spatial variance is calculated as:

$\Gamma = \frac{\sum_i \sum_{j\neq i} \omega_{ij} \frac{(y_i-y_j)^2}{2}}{\sum_i \sum_{j\neq i} \omega_{ij}}$

Value

A tibble (`returnsf` is `FALSE`) or an sf object (`returnsf` is `TRUE`)

Examples

```
econineq = sf::read_sf(system.file('extdata/econineq.gpkg',package = 'geocomplexity'))
gc = geocs_vector(dplyr::select(econineq,-Gini))
gc

library(ggplot2)
library(viridis)
ggplot(gc) +
   geom_sf(aes(fill = GC)) +
   scale_fill_viridis(option="mako", direction = -1) +
   theme_bw()
```

`geoc_swm` *spatial weight matrix based on geographical complexity*

Description

spatial weight matrix based on geographical complexity

Usage

```
geoc_swm(gc, wt, style = "W")
```

Arguments

`gc`	Geographical complexity vector.
`wt`	Spatial weight matrix based on spatial adjacency or spatial distance relationships.
`style`	(optional) A character that can be `B`,`W`,`C`. More to see `spdep::nb2mat()`. Default is `W`.

Value

A matrix

Examples

```
econineq = sf::read_sf(system.file('extdata/econineq.gpkg',package = 'geocomplexity'))
gc = econineq %>%
  dplyr::select(Gini) %>%
  geocd_vector(returnsf = FALSE) %>%
  dplyr::pull(1)
wt = sdsfun::spdep_contiguity_swm(econineq,style = 'B')
wt_gc = geoc_swm(gc,wt)
wt_gc[1:5,1:5]
```

---

moran_test *global spatial autocorrelation test*

Description

Spatial autocorrelation test based on global Moran'I Index.

Usage

```
moran_test(sfj, wt = NULL, alternative = "greater", symmetrize = FALSE)
```

Arguments

sfj	An sf object or vector object that can be converted to sf by sf::st_as_sf().
wt	(optional) Spatial weight matrix. Must be a matrix class. If wt is not provided, geocomplexity will use a first-order queen adjacency binary matrix via sdsfun package.
alternative	(optional) Specification of alternative hypothesis as greater (default), lower, or two.sided.
symmetrize	(optional) Whether or not to symmetrize the asymmetrical spatial weight matrix **wt** by: 1/2 * (**wt** + **wt**'). Default is FALSE.

Value

A list with moran_test class and result stored on the result tibble. Which contains the following information for each variable:

MoranI observed value of the Moran coefficient

EI expected value of Moran's I

VarI variance of Moran's I (under normality)

ZI standardized Moran coefficient

PI *p*-value of the test statistic

Note

This is a C++ implementation of the MI.vec function in spfilteR package, and embellishes the console output.

The return result of this function is actually a list, please access the result tibble using $result.

The non-numeric columns of the attribute columns in sfj are ignored.

Examples

```
econineq = sf::read_sf(system.file('extdata/econineq.gpkg',package = 'geocomplexity'))
moran_test(econineq)
```

Index