Package ‘GWpcor’

May 4, 2020

Type       Package
Title      Geographically Weighted Partial Correlation Coefficient
Version    0.1.2
Description Implements a geographically weighted partial correlation which is an extension from gwss() function in the 'GWmodel' package (Percival and Tsutsumida (2017) <doi:10.1553/giscience2017_01_s36>).
License    GPL (>= 2)
Encoding   UTF-8
LazyData   true
Language   en-US
Depends    R (>= 3.5.0)
Imports    methods, dplyr, sp, sf, GWmodel, corpcor, foreach, parallel, doParallel
SystemRequirements C++11, GDAL (>= 2.0.1), GEOS (>= 3.4.0), PROJ (>= 4.8.0)
NeedsCompilation no
Author      Narumasa Tsutsumida [aut, cre]
            (<https://orcid.org/0000-0002-6333-0301>),
            Joseph Percival [aut]
Maintainer  Narumasa Tsutsumida <rsnaru.jp@gmail.com>
Repository  CRAN
Date/Publication 2020-05-04 14:00:04 UTC

R topics documented:

         gwpcor .................................................. 2

Index       4
**Description**

This function calculates the geographically weighted correlation and partial correlation between two variables given others. The function is designed by the gwss function from the GWmodel package, and the cor2pcor function from the corpcor package.

**Usage**

```r
gwpcor(sdata, summary.locat, vars, method = c("pearson", "spearman"),
kernel = "bisquare", adaptive = FALSE, bw,
p = 2, theta = 0, longlat = FALSE, dMat, foreach = FALSE)
```

**Arguments**

- `sdata`: a Spatial*DataFrame (i.e. SpatialPointsDataFrame or SpatialPolygonsDataFrame as defined in package sp), or a sf object.
- `summary.locat`: A Spatial*DataFrame object for providing summary locations, i.e. SpatialPointsDataFrame or SpatialPolygonsDataFrame as defined in package sp, or a sf object.
- `vars`: A vector of variable names to be used for the analysis.
- `method`: A character string indicating which correlation and partial correlation coefficients to compute. "pearson" or "spearman" are accepted.
- `kernel`: function chosen as follows: gaussian: \( wgt = \exp(-0.5 \times (vdist / bw)^2) \); exponential: \( wgt = \exp(-vdist / bw) \); bisquare: \( wgt = (1 - (vdist / bw)^2)^2 \) if \( vdist < bw \), \( wgt = 0 \) otherwise; tricube: \( wgt = (1 - (vdist / bw)^3)^3 \) if \( vdist < bw \), \( wgt = 0 \) otherwise; boxcar: \( wgt = 1 \) if \( dist < bw \), \( wgt = 0 \) otherwise
- `adaptive`: if TRUE, an adaptive kernel where the bandwidth (bw) corresponds to the number of nearest neighbours (i.e. adaptive distance) is employed. The default is FALSE, where a fixed kernel is employed (bandwidth is a fixed distance).
- `bw`: Bandwidth size. If adaptive kernel, bw should be the number of nearest neighbours. For fixed kernel, the Euclid distance.
- `p`: The power of the Minkowski distance, default is 2, i.e. the Euclidean distance.
- `theta`: An angle in radians to rotate the coordinate system, default is 0.
- `longlat`: If TRUE, the coordinate of sdata is longlat.
- `dMat`: A pre-specified distance matrix, it can be calculated by the function gw.dist().
- `foreach`: Whether parallel computation is implemented or not.
Value

SDF
A SpatialPointsDataFrame (may be gridded) or SpatialPolygonsDataFrame object (see package ‘sp’) when the input is Spatial*DataFrame or a sf class object when input is sf, with local covariances, local correlations (Pearson’s), local correlations (Spearman’s), p-values of local correlations (Pearson’s), p-values of local correlations (Spearman’s), local partial correlations (Pearson’s), local partial correlations (Spearman’s), p-values of local partial correlations (Pearson’s), and p-values of local partial correlations (Spearman’s).

vars
Names of variables used for the calculation.

kernel
The name of kernel used for the calculation.

adaptive
Whether adaptive kernel is employed or not (TRUE/FALSE).

bw
The bandwidth size used for the calculation.

Author(s)
Tsutsumida N. and Percival J.

References


Examples

#NOTE: This example only shows how to implement gwpcor using sample data (meuse) in sp package. #Results do not suggest any meanings.

#Import data from sp package
library(sp)
library(sf)
data(meuse, package = "sp")
meuse_sf <- st_as_sf(meuse, coords = c("x", "y"), crs = 28992)

#implement gwpcor as an example
#the bandwidth is arbitrary.
res <- gwpcor(sdata = meuse_sf, vars = c("cadmium", "copper", "zinc"),
method = "pearson", kernel = "bisquare", adaptive = TRUE,
bw = 50, longlat = FALSE, foreach = FALSE)
Index

gwpcor, 2