Package 'seg'

December 18, 2019

Version 0.5-7

Date 2019-12-18

Title Measuring Spatial Segregation

Author Seong-Yun Hong <syhong@khu.ac.kr> and
David O'Sullivan <david.osullivan@vuw.ac.nz>, with code contributions by
Benjamin Jarvis and Changlock Choi

Maintainer Seong-Yun Hong <syhong@khu.ac.kr>

Depends R (>= 3.4.0), methods, stats, sp

Imports splancs

Suggests spdep, spgrass6, rgdal

Description Measuring spatial segregation. The methods implemented in this
package include White's P index (1983) <doi:10.1086/227768>,
Morrill's D(adj) (1991), Wong's D(w) and D(s) (1993)
<doi:10.1080/00420989320080551>, and Reardon and O'Sullivan's set of
spatial segregation measures (2004)
<doi:10.1111/j.0081-1750.2004.00150.x>.

License GPL (>= 3)

URL https://github.com/syunhong/seg

NeedsCompilation yes

Repository CRAN

Date/Publication 2019-12-18 15:20:02 UTC

R topics documented:

`conprof` *Concentration Profile*

Description

Draws a graph that shows the pattern of residential concentration for a population group and calculates its summary statistic as suggested by Hong and Sadahiro (2013).

Usage

```
conprof(data, grpID = 1, n = 999, graph = TRUE, add = FALSE, ...)
```

Arguments

`data`	an object of class `matrix`, or one that can be coerced to that class. Each column represents a population group. The number of columns should be greater than one (i.e., at least two population groups are required).
`grpID`	a numeric value specifying the population group (i.e., column in 'data') to be analysed. Multiple values are not allowed.
`n`	a numeric value indicating the number of thresholds to be used. A large value of 'n' creates a smoother-looking graph but slows down the calculation speed.
`graph`	logical. If TRUE, draw the concentration profile for the specified population group.
`add`	logical. If TRUE, add the graph to the current plot.
`...`	optional arguments to be passed to `plot` when 'add' is FALSE, or to `lines` otherwise. Ignored when `graph` is FALSE.

Details

For 'n' equally-spaced thresholds between 0 and 1, `conprof` identifies the areas where the selected population group comprises at least the given threshold proportions; computes how many of the group members live in these areas; and plots them on a 2D plane with the threshold values in the horizontal axis and the proportions of the people in the vertical axis.

The summary statistic is calculated by estimating the area between the concentration profile and a hypothetical line that represents a uniform distribution (see the examples).

Value

A `list` object with the following three elements:

`x`	the threshold values.
`y`	the proportions of the people who live in the areas where they comprise at least the corresponding threshold percentages in the local population composition.
`d`	the summary statistic for the concentration profile.

Author(s)

Seong-Yun Hong

References

Poulsen, M., Johnston, R., and Forrest J. (2002) Plural cities and ethnic enclaves: Introducing a measurement procedure for comparative study. *International Journal of Urban and Regional Research*, **26**, 229-243.

Hong, S.-Y. and Sadahiro, Y. (2013) Measuring geographic segregation: A graph-based approach. *Journal of Geographical Systems*, **na**, na-na.

Examples

```
xx <- runif(100) # random distribution
xx <- xx * (4000 / sum(xx))
yy <- rep(c(40, 60), 100) # no segregation
zz <- rep(c(100, 0), c(40, 60)) # complete segregation

set1 <- cbind(xx, 100 - xx)
set2 <- matrix(yy, ncol = 2, byrow = TRUE)
set3 <- cbind(zz, 100 - zz)

par(mar = c(5.1, 4.1, 2.1, 2.1))
out1 <- conprof(set1, grpID = 1,
  xlab = "Threshold level (%)",
  ylab = "Population proportion (%)",
  cex.lab = 0.9, cex.axis = 0.9, lty = "dotted")
out2 <- conprof(set2, grpID = 1, add = TRUE,
  lty = "longdash")
out3 <- conprof(set3, grpID = 1, add = TRUE)
title(main = paste("R =", round(out1$d, 2)))

# shaded areas represent the summary statistic value
if (require(graphics)) {
  polygon(c(out1$x[1:400], 0.4, 0),
          c(out1$y[1:400], 1, 1),
          density = 10, angle = 60,
          border = "transparent")
  polygon(c(out1$x[401:999], 1, 0.4),
          c(out1$y[401:999], 0, 0),
          density = 10, angle = 60,
```

```
          border = "transparent")
}
```

---

deseg *Decomposable Segregation Measure*

Description

Calculates the decomposable segregation measure developed by Sadahiro and Hong (2013).

Usage

```
deseg(x, data, smoothing = "kernel", nrow = 100, ncol = 100,
      window, sigma, verbose = FALSE)
```

Arguments

x	a numeric matrix or data frame with coordinates (each row is a point), or an object of class Spatial or ppp.
data	an object of class matrix, or one that can be coerced to that class. The number of rows in 'data' should equal the number of points in 'x', and the number of columns should be greater than one (i.e., at least two population groups are required). This can be missing if 'x' has a data frame attached to it.
smoothing	a character string indicating how to perform spatial smoothing of the population data. Currently only "kernel" is supported.
nrow	a numeric value indicating the number of row cells in the rasterised data surface.
ncol	a numeric value indicating the number of column cells.
window	an optional object of class matrix to be passed to kernel2d. See 'Details'.
sigma	an optional numeric value specifying the kernel bandwidth to be passed to kernel2d. See also 'Details'.
verbose	logical. If TRUE, print the current stage of the computation and time spent on each job to the screen.

Details

The decomposable segregation measure is a surface-based, non-spatial method. The index calculation does not take into account the spatial arrangement of the population. It is the spatial smoothing process that deals with the spatial aspect of segregation. If the spatial smoothing is not performed properly, this measure may suffer from the same checkerboard problem as the traditional index of dissimilarity.

Currently deseg uses the function kernel2d in **splancs**, which employs a quartic kernel estimator. The points outside 'window' are not considered when computing the kernel estimates. The argument 'window' must be a polygon represented in a matrix form such that each row corresponds to a vertex. This is passed to kernel2d as 'poly'. If 'window' is missing, a square that covers all points in 'x' will be used.

If 'sigma' is not given, bw.nrd in **stats** is used to find the optimal bandwidth.

Value

An object of SegDecomp-class.

Author(s)

Seong-Yun Hong

References

Sadahiro, Y. and Hong, S.-Y. (2013) Decomposition approach to the measurement of spatial segregation. *CSIS Discussion Paper Series*, **119**, 1-33.

See Also

SegDecomp-class, kernel2d

Examples

```
# uses the idealised landscapes in 'segdata'
data(segdata)
grd <- GridTopology(cellcentre.offset=c(0.5,0.5),
                    cellsize=c(1,1), cells.dim=c(10,10))
grd.sp <- as.SpatialPolygons.GridTopology(grd)

# displays the test data
plot(grd.sp)
plot(grd.sp[segdata[,9] == 100,], col = "Black", add = TRUE)
plot(grd.sp[segdata[,9] == 50,], col = "Grey", add = TRUE)

# tries different bandwidths for the same data
bw1 <- deseg(grd.sp, segdata[,9:10], sigma = 1, nrow = 20, ncol = 20)
print(bw1, digits = 3)
bw1.val <- sum(as.vector(bw1))
spplot(bw1, col.regions=rev(heat.colors(20)),
       main = paste("Bandwidth = 1, S =", round(bw1.val, 2)))

bw2 <- deseg(grd.sp, segdata[,9:10], sigma = 2, nrow = 20, ncol = 20)
print(bw2, digits = 3)
bw2.val <- sum(as(bw2, "vector"))
spplot(bw2, col.regions=rev(heat.colors(20)),
       main = paste("Bandwidth = 2, S =", round(bw2.val, 2)))

## Not run:
# let's see how the index value changes with different bandwidths
h0 <- seq(1, 5, length.out = 10); vals <- numeric()
for (i in 1:10) {
  d <- deseg(grd.sp, segdata[,9:10], sigma = h0[i], verbose = TRUE)
  vals <- append(vals, sum(as(d, "vector")))
}
plot(h0, vals, type = "b", xlab = "Bandwidth", ylab = "S")
title(main = "segdata[,9:10]")
```

```
# calculates the index for all data sets in 'segdata'
d.segdata <- matrix(NA, nrow = 3, ncol = 8)
for (i in 1:8) {
  idx <- 2 * i
  tmp <- deseg(grd.sp, segdata[,(idx-1):idx])
  d.segdata[,i] <- as(tmp, "vector")
}

# presents the results as a bar chart
barplot(d.segdata, names.arg = LETTERS[1:8], main = "segdata",
        legend.text = c(expression(italic(paste("S"[L]))),
                        expression(italic(paste("S"[C]))),
                        expression(italic(paste("S"[Q])))),
        args.legend = list(x = "topright", horiz = TRUE))
## End(Not run)
```

`dissim` *Index of Dissimilarity*

Description

Calculates the index of dissimilarity proposed by Duncan and Duncan (1955). If 'x' or 'nb' is given, the index is adjusted to reflect the spatial distribution of population.

Usage

```
dissim(x, data, nb, adjust = FALSE, p2n.args, n2m.args,
       wVECT.args, v2n.args, verbose = FALSE)
seg(data, nb)
```

Arguments

`x`	an optional object of class `SpatialPolygons` or `SpatialPolygonsDataFrame`.
`data`	a numeric matrix or data frame with two columns that represent mutually exclusive population groups (e.g., Asians and non-Asians). If more than two columns are given, only the first two will be used for computing the index.
`nb`	an optional `matrix` object describing the intensity of interaction between geographic units.
`adjust`	logical. If TRUE, and if 'x' is given, the index of dissimilarity is adjusted according to the suggestions of Morrill (1991) and Wong (1993), depending on packages installed on the system. See 'Details' for more information. Ignore if 'x' is not given.
`p2n.args`	an optional `list` of arguments to be passed to `poly2nb`. To avoid confusion, it is best to name all objects in the list in a way that they exactly match the argument names in `poly2nb`.
`n2m.args`	an optional `list` of arguments to be passed to `nb2mat`.
`wVECT.args`	an optional `list` of arguments to be passed to `writeVECT6`.

v2n.args	an optional list of arguments to be passed to vect2neigh.
verbose	logical. If TRUE, print the current stage of the computation and time spent on each job to the screen.

Details

dissim calculates the index of dissimilarity for 'data'. If 'data' is missing, it attempts to extract the data from 'x'. If 'x' is not given, or if it is not a valid SpatialPolygonsDataFrame object, the function stops with an error.

When 'x' is given and 'adjust' is set to TRUE, the index is adjusted according to the suggestions of Morrill (1991) and Wong (1993). However, this automatic adjustment requires a number of packages to be installed on the user's system: **spdep** for Morrill's D(adj) and **spgrass6** and **rgdal** for Morrill's D(w) and D(s). Note that, for D(w) and D(s), GRASS commands should be accessible from within R.

Alternatively, the index value can be adjusted using a user-specified weighting matrix 'nb'. 'nb' must be a square matrix (i.e., one that has the same number of rows and columns) but does not have to be symmetric. If 'nb' is not specified, the function calculates the traditional index of dissimilarity proposed by Duncan and Duncan (1955).

If 'nb' is a rook-based contiguity matrix standardised by the total number of neighbours, the index is adjusted as described in Morrill (1991). See the example code below and nb2mat in **spdep** for more information regarding how to construct such a matrix. For Wong's D(w) and D(s), see https://sites.google.com/site/hongseongyun/seg. Note that the sum of all elements in 'nb' should equal one.

seg is a simplified version of dissim and will be deprecated later.

Value

dissim returns a list containing the following elements:

d	index of dissimilarity.
dm	index of dissimilarity adjusted according to Morrill (1991). NA if not calculated.
dw	index of dissimilarity adjusted according to Wong (1991). NA if not calculated.
ds	index of dissimilarity adjusted according to Wong (1991). NA if not calculated.
user	index of dissimilarity adjusted using the user-specified weighting matrix 'nb'. NA if 'nb' is missing.

seg returns a single numeric value between 0 and 1, indicating the degree of segregation; 0 for no segregation, and 1 for complete segregation.

Author(s)

Seong-Yun Hong

References

Duncan, O. D., & Duncan, B. (1955). A methodological analysis of segregation indexes. *American Sociological Review*, **20**, 210-217.

Morrill, R. L. (1991). On the measure of geographic segregation. *Geography Research Forum*, **11**, 25-36.

Wong, D. W. S. (1993). Spatial indices of segregation. *Urban Studies*, **30**, 559-572.

See Also

spseg, nb2mat

Examples

```
if (require(spdep)) { # package 'spdep' is required

  # uses the idealised landscapes in 'segdata'
  data(segdata)
  grd <- GridTopology(cellcentre.offset=c(0.5,0.5),
                      cellsize=c(1,1), cells.dim=c(10,10))
  grd.sp <- as.SpatialPolygons.GridTopology(grd)
  grd.nb <- nb2mat(poly2nb(grd.sp, queen = FALSE), style = "B")
  grd.nb <- grd.nb / sum(grd.nb)

  d <- rep(NA, 8); m <- rep(NA, 8)
  for (i in 1:8) {
    idx <- 2 * i
    d[i] <- seg(segdata[,(idx-1):idx])
    m[i] <- seg(segdata[,(idx-1):idx], grd.nb)
    full <- segdata[,(idx-1)] == 100
    half <- segdata[,(idx-1)] == 50
    plot(grd.sp)
    plot(grd.sp[full,], col = "Black", add = TRUE)
    if (any(half))
      plot(grd.sp[half,], col = "Grey", add = TRUE)
    text(5, 11.5, labels = paste("D = ", round(d[i], 2),
                                 ", D(adj) = ", round(m[i], 2), sep = ""))
  }
}
```

`isp` *Index of Spatial Proximity*

Description

Computes the index of spatial proximity developed by White (1983). This measure estimates the level of clustering by comparing the average distance between members of the same group with that between all individuals (regardless of the groups to which they belong). The results may change drastically depending on the definition of distance.

Usage

```
isp(x, data, nb, fun, verbose = FALSE, ...)
whiteseg(x, data, nb, fun, verbose = FALSE, ...)
```

Arguments

x	a numeric matrix or data frame with coordinates (each row is a point), or an object of class Spatial.
data	an object of class matrix, or one that can be coerced to that class. The number of rows in 'data' should equal the number of geographic units in 'x', and the number of columns should be greater than one (i.e., at least two population groups are required). This can be missing if 'x' has a data frame attached to it.
nb	an optional matrix object indicating the distances between the geographic units.
fun	a function for the calculation of proximity. The function should take a numeric vector as an argument (distance) and return a vector of the same length (proximity). If this is not specified, a negative exponential function is used by default.
verbose	logical. If TRUE, print the current stage of the computation and time spent on each job to the screen.
...	optional arguments to be passed to dist when calculating the distances between the geographic units in 'x'. Ignored if 'nb' is given. See help(dist) for available options.

Details

'nb' must be a square matrix (i.e., one that has the same number of rows and columns) but does not have to be symmetric. If 'nb' is not given, whiteseg attempts to create a distance matrix of 'x' using the function dist in **stats** and use it as 'nb'. The optional arguments in '...' will be passed to dist.

Value

A single numeric value indicating the degree of segregation; a value of 1 indicates absence of segregation, and values greater than 1.0 indicate clustering. If the index value is less than one, it indicates an unusual form of segregation (i.e., people live closer to other population groups).

Note

The function name whiteseg() was changed in version 0.4-3 to isp(). The old function name will be deprecated from 0.6-1.

Author(s)

Seong-Yun Hong

References

White, M. J. (1983). The measurement of spatial segregation. *The American Journal of Sociology*, **88**, 1008-1018.

See Also

seg, dist

Examples

```
# uses the idealised landscapes in 'segdata'
data(segdata)
grd <- GridTopology(cellcentre.offset=c(0.5,0.5),
                    cellsize=c(1,1), cells.dim=c(10,10))
grd.sp <- as.SpatialPolygons.GridTopology(grd)

d <- rep(NA, 8) # index of dissimilarity
p <- rep(NA, 8) # index of spatial proximity
for (i in 1:8) {
  idx <- 2 * i
  d[i] <- seg(segdata[,(idx-1):idx])
  p[i] <- whiteseg(grd.sp, data = segdata[,(idx-1):idx])
  full <- segdata[,(idx-1)] == 100
  half <- segdata[,(idx-1)] == 50
  plot(grd.sp)
  plot(grd.sp[full,], col = "Black", add = TRUE)
  if (any(half))
    plot(grd.sp[half,], col = "Grey", add = TRUE)
  text(5, 11.5, labels =
       paste("D = ", round(d[i], 2), ", P = ", round(p[i], 2), sep = ""))
}
```

---

localenv *Local Population Composition*

Description

localenv calculates the local population composition at each data point from a matrix of coordinates, or an object of class Spatial or ppp.

Usage

```
localenv(x, data, power = 2, useExp = TRUE, scale = FALSE, maxdist, sprel,
         tol = .Machine$double.eps)
```

Arguments

x	a numeric matrix or data frame with coordinates (each row is a point), or an object of class Spatial or ppp.
data	an object of class matrix, or one that can be coerced to that class. The number of rows in 'data' should equal the number of points in 'x', and the number of columns should be greater than one (at least two population groups are required). This can be missing if 'x' has a data frame attached to it.

power	a numeric value that determines the change rate of a distance weight function. If zero, all data points have the same weight regardless of the distance. Typically 1-5.
useExp	logical. If FALSE, use a simple inverse distance function instead of a negative exponential function. See 'Details'.
scale	logical. If TRUE, the distances between points in 'x' are scaled so the weights are not affected by the measurement unit. The default is FALSE for consistency with the previous version.
maxdist	an optional numeric value specifying a search radius for the construction of each local environment. Must be a positive value, or zero.
sprel	an optional object of class dist or nb. See 'Details'.
tol	a small, positive non-zero value. If 'useExp' is FALSE, this value will be added to the denominator to avoid the divide-by-zero error.

Details

At each data point in 'x', localenv calculates the weighted average of the populations of all points that are within a search radius 'maxdist'. The output from this function is an essential component to compute the spatial segregation measures.

By default, the weight of each point is calculated from a negative exponential function, which is defined as:

$$w(d) = e^{-d \times power}$$

where d is the Euclidean distance between two points.

If 'useExp' is FALSE, a simple inverse distance function is used to calculate the weight of each point:

$$w(d) = \frac{1}{(d + error)^{power}}$$

If 'maxdist' is not provided (default), all data points in the study area are used for the construction of each local environment. It is recommended to specify this parameter to speed up the calculation process.

If a distance measure other than the Euclidean distance is required to represent spatial proximity between the points, users can provide an object of class dist, which contains the distances between all pairs of the points, through an optional argument 'sprel'. One convenient way of obtaining such information may be the use of the function dist, which offers a variety of distance measures, such as Manhattan, Canberra, and Minkowski.

Or alternatively, one can supply an object of class nb to use a k-nearest neighbour averaging or polygon contiguity.

Value

An object of SegLocal-class.

Note

Note that this function is not to interpolate between data points. The calculation of each local environment involves the point itself, so the simple inverse distance function with a power of 2 or more should be used with care.

Author(s)

Seong-Yun Hong

See Also

SegLocal-class, spatseg, dist

Examples

```
# uses the idealised landscapes in 'segdata'
data(segdata)
grd <- GridTopology(cellcentre.offset=c(0.5,0.5),
                    cellsize=c(1,1), cells.dim=c(10,10))
grd.sp <- as.SpatialPolygons.GridTopology(grd)

# complete segregation:
# negative exponential function of distance
xx1 <- localenv(grd.sp, data = segdata[,1:2])
spplot(xx1, main = "Negative exponential")

# inverse distance
xx2 <- localenv(grd.sp, data = segdata[,1:2], useExp = FALSE)
spplot(xx2, main = "Inverse distance")

# inverse distance with p = 0, i.e., weight all points equally
xx3 <- localenv(grd.sp, data = segdata[,1:2], useExp = FALSE, power = 0)
spplot(xx3, main = "Inverse distance with p = 0")

## Not run:
# checkerboard pattern:
# negative exponential function with different p values
vv1 <- localenv(grd.sp, data = segdata[,5:6], power = 1)
spplot(vv1, main = "Negative exponetial with p = 1")

vv2 <- localenv(grd.sp, data = segdata[,5:6])
spplot(vv2, main = "Negative exponetial with p = 2")

vv3 <- localenv(grd.sp, data = segdata[,5:6], power = 3)
spplot(vv3, main = "Negative exponetial with p = 3")
## End(Not run)
```

segdata *Hypothetical Patterns of Segregation*

Description

This data set contains eight different spatial configurations that were used by Morrill (1991) and Wong (1993) to test their segregation measures.

Usage

```
data(segdata)
```

Format

An object of class `data.frame`. The data set contains 16 columns, representing eight idealised spatial patterns. Each column indicates the following information:

[,1]	A1	Pattern A, Group 1
[,2]	A2	Pattern A, Group 2
[,3]	B1	Pattern B, Group 1
[,4]	B2	Pattern B, Group 2
[,5]	C1	Pattern C, Group 1
[,6]	C2	Pattern C, Group 2
[,7]	D1	Pattern D, Group 1
[,8]	D2	Pattern D, Group 2
[,9]	E1	Pattern E, Group 1
[,10]	E2	Pattern E, Group 2
[,11]	F1	Pattern F, Group 1
[,12]	F2	Pattern F, Group 2
[,13]	G1	Pattern G, Group 1
[,14]	G2	Pattern G, Group 2
[,15]	H1	Pattern H, Group 1
[,16]	H2	Pattern H, Group 2

Source

Morrill, R. L. (1991). On the measure of geographic segregation. *Geography Research Forum*, **11**, 25-36.

Wong, D. W. S. (1993). Spatial Indices of Segregation. *Urban Studies*, **30**, 559-572.

Examples

```
data(segdata)
grd <- GridTopology(cellcentre.offset=c(0.5,0.5),
                    cellsize=c(1,1), cells.dim=c(10,10))
grd.sp <- as.SpatialPolygons.GridTopology(grd)
```

```
pd <- par()
par(mfrow = c(2, 4), mar = c(0, 1, 0, 1))
for (i in 1:8) {
  full <- segdata[,(2*i)-1] == 100
  half <- segdata[,(2*i)-1] == 50
  plot(grd.sp)
  plot(grd.sp[full,], col = "Black", add = TRUE)
  if (any(half))
    plot(grd.sp[half,], col = "Grey", add = TRUE)
  text(5, 11.5, labels = paste("Pattern", LETTERS[i]))
}
par(mfrow = pd$mfrow, mar = pd$mar)
```

SegDecomp *Create an Object of Class SegDecomp*

Description

Creates a new object of SegDecomp-class.

Usage

```
SegDecomp(d, coords, data, proj4string = CRS(as.character(NA)))
```

Arguments

d	a numeric vector of length three, representing the locational, compositional, and qualitative segregation, respectively.
coords	a numeric matrix or data frame with coordinates (each row is a point).
data	an object of class matrix containing the population data at each data point. The number of rows in 'data' should equal the number of points in 'coords', and the number of columns should be greater than one (i.e., at least two population groups are required).
proj4string	an optional projection string of class CRS.

Value

An object of SegDecomp-class.

Author(s)

Seong-Yun Hong

See Also

SegDecomp-class, deseg

Examples

```
# creates a random data set with 50 data points and 3 population groups
xy <- matrix(runif(100), ncol = 2)
colnames(xy) <- c("x", "y")
pop <- matrix(runif(150), ncol = 3)
colnames(pop) <- LETTERS[1:3]

# constructs an object of class 'SegDecomp'
v <- SegDecomp(d = numeric(3), coords = xy, data = pop)
is(v)
```

SegDecomp-class *Class Decomp*

Description

A class to hold results from deseg.

Objects from the Class

Objects can be created by calls to deseg, or the constructor SegDecomp.

Slots

d a numeric vector of length three, representing the locational, compositional, and qualitative segregation, respectively.

coords a numeric matrix or data frame with coordinates (each row is a point).

data an object of class matrix containing the population data at each data point. The number of rows in 'data' should equal the number of points in 'coords', and the number of columns should be greater than one (i.e., at least two population groups are required).

proj4string an optional projection string of class CRS.

Methods

coerce signature(from = "SegDecomp", to = "SpatialPoints"): coerce an object of class SegDecomp to an object of class SpatialPoints. The points have no attribute data.

coerce signature(from = "SegDecomp", to = "SpatialPointsDataFrame"): coerce an object of class SegDecomp to an object of class SpatialPointsDataFrame. The values in the slot 'data' will be used as the attribute data.

coerce signature(from = "SegDecomp", to = "SpatialPixelsDataFrame"): coerce an object of class SegDecomp to an object of class SpatialPixelsDataFrame. This method is to retrieve the kernel density estimates and save it as a SpatialPixelsDataFrame object, with a proj4string.

coerce signature(from = "SegDecomp", to = "vector"): coerce an object of class SegDecomp to an object of class vector. Use this to extract the segregation index values from the specified object.

as.list `signature(x = "SegDecomp")`: same as the above.

show `signature(object = "SegDecomp")`: print the segregation index values.

print `signature(x = "SegDecomp")`: same as show.

spplot `signature(obj = "SegDecomp")`: coerce an object of class SegDecomp to an object of class SpatialPixelsDataFrame or SpatialPointsDataFrame and display it. See `help(spplot)` for more details about the graphical parameter arguments.

Author(s)

Seong-Yun Hong

See Also

SegDecomp, deseg

Examples

```
# creates 100 regularly-spaced data points and 3 population groups
xy <- expand.grid(1:10, 1:10)
colnames(xy) <- c("x", "y")
pop <- matrix(runif(300), ncol = 3)
colnames(pop) <- LETTERS[1:3]

# randomly-generated index values
d <- runif(2, 0, 0.4)
d <- c(d, 0.9 - sum(d))

# constructs an object of class 'SegDecomp'
v <- SegDecomp(d = d, coords = as.matrix(xy), data = pop)
print(v)

# retrieves the index values
as.vector(v) # same as: as(v, "vector")

# displays the kernel estimates in the slot 'data'
spplot(v, col.regions = heat.colors(20))
```

SegLocal *Create an Object of Class SegLocal*

Description

Create a new object of SegLocal-class from a matrix of coordinates and population data.

Usage

```
SegLocal(coords, data, env, proj4string = CRS(as.character(NA)))
```

Arguments

coords	a numeric matrix or data frame with coordinates (each row is a point).
data	an object of class matrix containing the population data at each data point. The number of rows in 'data' should equal the number of points in 'coords', and the number of columns should be greater than one (i.e., at least two population groups are required).
env	an object of class matrix containing the local environment parameters. Must be the same dimensions as 'data'.
proj4string	an optional projection string of class CRS.

Value

An object of SegLocal-class.

Author(s)

Seong-Yun Hong

See Also

SegLocal-class, localenv

Examples

```
# creates a random data set with 50 data points and 3 population groups
xy <- matrix(runif(100), ncol = 2)
colnames(xy) <- c("x", "y")
pop <- matrix(runif(150), ncol = 3)
colnames(pop) <- LETTERS[1:3]

# constructs an object of class 'SegLocal'
v <- SegLocal(coords = xy, data = pop, env = pop)
is(v)
```

SegLocal-class *Class SegLocal*

Description

A class to hold the local population composition data.

Objects from the Class

Objects can be created by calls to localenv, or the constructor SegLocal.

Slots

coords a numeric matrix or data frame with coordinates (each row is a point).

data an object of class `matrix` containing the population data at each data point. The number of rows in 'data' should equal the number of points in 'coords', and the number of columns should be greater than one (i.e., at least two population groups are required).

env an object of class `matrix` containing the local environment parameters. Must be the same dimensions as 'data'.

proj4string an optional projection string of class `CRS`.

Methods

coerce `signature(from = "SegLocal", to = "SpatialPoints")`: coerce an object of class `SegLocal` to an object of class `SpatialPoints`. The points have no attribute data.

coerce `signature(from = "SegLocal", to = "SpatialPointsDataFrame")`: coerce an object of class `SegLocal` to an object of class `SpatialPointsDataFrame`. The values in the slot 'env' will be used as the attribute data.

coerce `signature(from = "SegLocal", to = "SpatialPixelsDataFrame")`: coerce an object of class `SegLocal` to an object of class `SpatialPixelsDataFrame`. The values in the slot 'env' will be used as the attribute data. May not work when the points are irregularly spaced.

coerce `signature(from = "SpatialPointsDataFrame", to = "SegLocal")`: coerce an object of class `SpatialPointsDataFrame` to an object of class `SegLocal`.

coerce `signature(from = "SpatialPolygonsDataFrame", to = "SegLocal")`: coerce an object of class `SpatialPolygonsDataFrame` to an object of class `SegLocal`.

show `signature(object = "SegLocal")`: show the number of points and data columns in an object of class `SegLocal`.

print `signature(x = "SegLocal")`: same as `show`.

plot `signature(x = "SegLocal")`: draw a plot, or plots, of points in an object of class `SegLocal`. Use an optional argument 'which.col' to specify a column of the data that determines the points' sizes. See the examples below for demonstration.

points `signature(x = "SegLocal")`: draw points in an object of class `SegLocal` on an active graphic device.

spplot `signature(obj = "SegLocal")`: coerce an object of class `SegLocal` to an object of class `SpatialPixelsDataFrame` or `SpatialPointsDataFrame` and display it. See `help(spplot)` for more details about the graphical parameter arguments.

summary `signature(object = "SegLocal")`: summarise the population compositions of points and local environments in an object of class `SegLocal`.

update `signature(object = "SegLocal")`: update an existing object of class `SegLocal`.

Author(s)

Seong-Yun Hong

See Also

`SegLocal`, `localenv`

Examples

```
# creates 100 regularly-spaced data points and 3 population groups
xy <- expand.grid(1:10, 1:10)
colnames(xy) <- c("x", "y")
pop <- matrix(runif(300), ncol = 3)
colnames(pop) <- LETTERS[1:3]

# constructs an object of class 'SegLocal'
v <- SegLocal(coords = as.matrix(xy), data = pop, env = pop)
summary(v)

# updates the map projection information
v <- update(v, proj4string = CRS("+proj=nzmg +datum=nzgd49"))
summary(v)

# displays the (randomly-generated) local population data
par(mfrow = c(1, 3))
plot(v, main = paste("Data", colnames(pop)), xlab = "x", ylab = "y")
par(mfrow = c(1, 1))

# coverts the object to class 'Spatial'
plot(as(v, "SpatialPoints"))
spplot(v, col.regions = heat.colors(20))
v.sp <- as(v, "SpatialPixelsDataFrame")
is(v.sp)
```

`SegSpatial` *Create an Object of Class SegSpatial*

Description

Creates a new object of SegSpatial-class.

Usage

```
SegSpatial(d, r, h, p, coords, data, env, proj4string = CRS(as.character(NA)))
```

Arguments

`d`	an object of class `numeric` containing the spatial dissimilarity index value.
`r`	an object of class `numeric` containing the spatial diversity index value.
`h`	an object of class `numeric` containing the spatial information theory index value.
`p`	an object of class `matrix` that has the spatial exposure/isolation of all population groups.
`coords, data, env, proj4string`	see SegLocal-class.

Value

An object of SegSpatial-class.

Author(s)

Seong-Yun Hong

See Also

SegSpatial-class, spseg

Examples

```
# creates a random data set with 50 data points and 3 population groups
xy <- matrix(runif(100), ncol = 2)
colnames(xy) <- c("x", "y")
pop <- matrix(runif(150), ncol = 3)
colnames(pop) <- LETTERS[1:3]

# constructs an object of class 'SegSpatial'
v <- SegSpatial(d = numeric(), r = numeric(), h = numeric(),
                p = matrix(0, 0, 0), coords = xy, data = pop, env = pop)
is(v)
```

SegSpatial-class *Class SegSpatial*

Description

A class to hold results from spatseg.

Objects from the Class

Objects can be created by calls to spseg, or the constructor SegSpatial.

Slots

d an object of class `numeric` containing the spatial dissimilarity index value.

r an object of class `numeric` containing the spatial diversity index value.

h an object of class `numeric` containing the spatial information theory index value.

p an object of class `matrix` that has the spatial exposure/isolation of all population groups.

coords, data, env, proj4string see SegLocal-class.

Methods

coerce `signature(from = "SegSpatial", to = "SpatialPoints")`: coerce an object of class `SegSpatial` to an object of class `SpatialPoints`. The points have no attribute data.

coerce `signature(from = "SegSpatial", to = "SpatialPointsDataFrame")`: coerce an object of class `SegSpatial` to an object of class `SpatialPointsDataFrame`. The values in the slot 'data' will be used as the attribute data.

coerce `signature(from = "SegSpatial", to = "SpatialPixelsDataFrame")`: coerce an object of class `SegSpatial` to an object of class `SpatialPixelsDataFrame`. The values in the slot 'data' will be used as the attribute data. May not work when the points are irregularly spaced.

coerce `signature(from = "SegSpatial", to = "list")`: retrieve the segregation index values and return it as a `list` object.

as.list `signature(x = "SegSpatial")`: same as the above.

show `signature(object = "SegSpatial")`: show the segregation index values.

print `signature(x = "SegSpatial")`: same as `show`.

spplot `signature(obj = "SegSpatial")`: coerce an object of class `SegSpatial` to an object of class `SpatialPixelsDataFrame` or `SpatialPointsDataFrame` and display it. See `help(spplot)` for more details about the graphical parameter arguments.

Extends

`SegLocal-class`.

Author(s)

Seong-Yun Hong

See Also

`SegSpatial`, `spseg`

Examples

```
# creates 100 regularly-spaced data points and 3 population groups
xy <- expand.grid(1:10, 1:10)
colnames(xy) <- c("x", "y")
pop <- matrix(runif(300), ncol = 3)
env <- matrix(runif(300), ncol = 3)
colnames(pop) <- LETTERS[1:3]
colnames(env) <- LETTERS[4:6]

# constructs an object of class 'SegSpatial'
v <- SegSpatial(d = numeric(), r = numeric(), h = numeric(),
                p = matrix(0, 0, 0),
                coords = as.matrix(xy), data = pop, env = env)
print(v)

# changes the spatial dissimilarity index value
slot(v, "d") <- runif(1)
```

```
# retrieves the index values
as.list(v)

# displays the values in the slot 'data'
spplot(v, col.regions = heat.colors(20))

# displays the values in the slot 'env'
w <- as(v, "SegLocal")
spplot(w, col.regions = heat.colors(20))
```

---

spseg *Spatial Segregation Measures*

Description

Calculates the set of spatial segregation measures developed by Reardon and O'Sullivan (2004).

Usage

```
spseg(x, data, method = "all", smoothing = "none", nrow = 100, ncol = 100,
      window, sigma, useC = TRUE, negative.rm = FALSE,
      tol = .Machine$double.eps, verbose = FALSE, ...)
spatseg(env, method = "all", useC = TRUE, negative.rm = FALSE,
        tol = .Machine$double.eps)
```

Arguments

x	a numeric matrix or data frame with coordinates (each row is a point), or an object of class Spatial or ppp.
env	an object of SegLocal-class.
data	an object of class matrix, or one that can be coerced to that class. The number of rows in 'data' should equal the number of points in 'x', and the number of columns should be greater than one (i.e., at least two population groups are required). This can be missing if 'x' has a data frame attached to it.
method	a vector of character strings indicating an measure or measures to be computed. This must be one or more of the strings "all" (default), "exposure", "information", "diversity", and "dissimilarity". Abbreviations are accepted, as long as it is clear which method is meant.
smoothing	a character string indicating how to perform spatial smoothing of the population data. This must be (an abbreviation of) one of the strings "none" (default), "kernel", or "equal".
nrow	a numeric value indicating the number of row cells in the rasterised data surface. Ignored if 'smoothing' is "none".
ncol	a numeric value indicating the number of column cells.

window	an optional object of class matrix to be passed to kernel2d. See 'Details' in deseg.
sigma	an optional numeric value specifying the kernel bandwidth to be passed to kernel2d. See also 'Details' in deseg.
useC	logical. If TRUE, calculate the segregation values in C.
negative.rm	logical. If TRUE, all geographic units where at least one group (i.e., column) has a population of zero or less will be removed to prevent -Inf or NaN in the information theory index. If FALSE, the non-positive values will be replaced with 'tol'.
tol	a small, positive non-zero value. See 'Details'.
verbose	logical. If TRUE, print the current stage of the computation and time spent on each job to the screen.
...	optional arguments to be passed to localenv to compute the population composition of each local environment.

Details

spatseg computes the set of spatial segregation measures proposed by Reardon and O'Sullivan.

spseg is a wrapper function, which calls spatseg after constructing a population density surface and its local environment parameters with user-specified options. Currently the population density surface is constructed by assuming that the population density is uniform in each census tract, or by using kernel2d in the package **splancs**. In the previous version (< 0.4.1), the function rasterize in **raster** was used in the case of the former, and density.ppp in **spatstat** for the latter.

In R, log(0) is defined as -Inf, and 0 * log(0) is NaN (not-a-number), in compliance with the IEEE Standard for Floating-Point Arithmetic. When computing the spatial information theory index, which is a spatial version of the entropy index, this is annoying because 0 * log(0) occurs quite often, especially when one or more groups are small, and/or when the grid size is very small. To work around this problem, the argument 'tol' is added, so instead of v * log(v), it calculates (v + tol) * log(v + tol), where v = 0. This makes the entropy practically zero, as 'tol' goes towards 0.

Value

An object of SegSpatial-class.

Note

The exposure/isolation index, P, is presented in a matrix form. The spatial exposure of group 'm' to group 'n' is located in the row 'm' and column 'n' of the matrix. The matrix is rarely symmetric in practice so the spatial exposure index should be interpreted with care.

The spatial isolation index values are given in the diagonal cells of the matrix; cell value at (m, m) indicates the degree of spatial isolation for group 'm' for example.

Author(s)

Seong-Yun Hong

References

Reardon, S. F. and O'Sullivan, D. (2004) Measures of spatial segregation. *Sociological Methodology*, **34**, 121-162.

Reardon, S. F., Farrell, C. R., Matthews, S. A., O'Sullivan, D., Bischoff, K., and Firebaugh, G. (2009) Race and space in the 1990s: Changes in the geographic scale of racial residential segregation, 1990-2000. *Social Science Research*, **38**, 55-70.

See Also

SegSpatial-class, localenv, kernel2d

Examples

```
# uses the idealised landscapes in 'segdata'
data(segdata)
grd <- GridTopology(cellcentre.offset=c(0.5,0.5),
                    cellsize=c(1,1), cells.dim=c(10,10))
grd.sp <- as.SpatialPolygons.GridTopology(grd)
test.df <- segdata[,1:2]

# no spatial smoothing
xx1 <- spseg(grd.sp, data = test.df)
print(xx1, digits = 3)

# plots the values in the slot 'data'
spplot(xx1, main = "No spatial smoothing")

# smoothes the data points
xx2 <- spseg(grd.sp, data = test.df, smoothing = "equal")
print(xx2, digits = 3)
spplot(xx2, main = "Equal")

# uses the kernel smoothing of the data points
xx3 <- spseg(grd.sp, data = test.df, smoothing = "kernel",
             nrow = 20, ncol = 20)
print(xx3, digits = 3)
spplot(xx3, main = "Kernel")

## Not run:
# same as the above but with a boundary polygon
w <- matrix(c(1.5, 1.5,
              1.5, 8.5,
              8.5, 8.5,
              8.5, 4.5,
              5.5, 4.5,
              5.5, 1.5), ncol = 2, byrow = TRUE)
xx4 <- spseg(grd.sp, data = segdata[,1:2], smoothing = "kernel",
             window = w, nrow = 20, ncol = 20)
print(xx4, digits = 3)
spplot(xx4, main = "Kernel with a boundary polygon")
```

```
# retrieves the index values
as.list(xx4)

# shows the values in the slot 'env'
spplot(as(xx4, "SegLocal"), main = "Local population composition")
## End(Not run)
```

Index