Package ‘smerc’

October 1, 2021

Type Package
Title Statistical Methods for Regional Counts
Version 1.5
Maintainer Joshua French <joshua.french@ucdenver.edu>

Description Implements statistical methods for analyzing the counts of areal data, with a focus on the detection of spatial clusters and clustering. The package has a heavy emphasis on spatial scan methods, which were first introduced by Kulldorff and Nagarwalla (1995) <doi:10.1002/sim.4780140809> and Kulldorff (1997) <doi:10.1080/03610929708831995>.

License GPL (>= 2)
LazyLoad yes
Depends R (>= 3.6)
Imports MESS, pbapply, randtoolbox, sp, Rcpp
Suggests crayon, lintr, maps, testthat, SpatialEpi, knitr, rmarkdown
Encoding UTF-8
RoxygenNote 7.1.2
VignetteBuilder knitr
LinkingTo Rcpp, RcppProgress
NeedsCompilation yes

Author Joshua French [aut, cre] (<https://orcid.org/0000-0002-9708-3353>), Mohammad Meysami [ctb] (<https://orcid.org/0000-0002-3322-5244>)

Repository CRAN
Date/Publication 2021-10-01 16:50:02 UTC

R topics documented:

bn.test ................................................................. 3
bn.zones ............................................................... 5
R topics documented:

cepp.sim ........................................ 6
cepp.test ........................................ 7
cepp.weights .................................... 8
clusters .......................................... 9
color.clusters .................................. 10
combine.zones .................................. 10
csg2 ............................................. 11
dc.sim ........................................... 13
dc.test .......................................... 14
dc.zones ........................................ 16
dist.ellipse ..................................... 17
distinct .......................................... 18
dmst.sim ......................................... 19
dmst.test ........................................ 20
dmst.zones ...................................... 22
edmst.sim ........................................ 24
edmst.test ....................................... 25
edmst.zones ..................................... 27
elbow_point ..................................... 28
elliptic.nn ....................................... 29
elliptic.sim ..................................... 30
elliptic.test .................................... 31
elliptic.zones ................................... 33
fast.sim ......................................... 34
fast.test ........................................ 35
fast.zones ....................................... 37
flex.sim ......................................... 38
flex.test ......................................... 39
flex.zones ....................................... 41
flex_test ......................................... 43
flex_zones ....................................... 45
knn .................................................. 46
lget ............................................... 47
mlf.test ......................................... 48
mlf.zones ........................................ 50
mlink.sim ....................................... 52
mlink.test ....................................... 53
mlink.zones ..................................... 55
mst.all ......................................... 56
mst.seq .......................................... 59
nn.cumsum ....................................... 61
nn2zones ........................................ 62
nndist ........................................... 62
nndup ............................................ 63
nnpop ............................................. 64
noz ............................................... 65
nydf .............................................. 66
nypoly ........................................... 67
### Description


### Usage

```r
bn.test(
  coords,
  cases,
  pop,
  cstar,
  ex = sum(cases)/sum(pop) * pop,
  alpha = 0.1,
)```

bn.test

longlat = FALSE,
modified = FALSE
)

Arguments

coords  An n × 2 matrix of centroid coordinates for the regions.
cases  The number of cases observed in each region.
pop  The population size associated with each region.
cstar  A non-negative integer indicating the minimum number of cases to include in each window.
ex  The expected number of cases for each region. The default is calculated under the constant risk hypothesis.
alpha  The significance level to determine whether a cluster is significant. Default is 0.10.
longlat  The default is FALSE, which specifies that Euclidean distance should be used. If longlat is TRUE, then the great circle distance is used to calculate the inter-centroid distance.
modified  A logical value indicating whether a modified version of the test should be performed. The original paper recommends computing the p-value for each cluster as 1 -ppois(cstar -1,lambda = expected). The modified version replaces cstar with cases, the observed number of cases in the region, and computes the p-value for the cluster as 1 -ppois(cases -1,lambda = ex). The default is modified = FALSE.

Value

Returns a smerc_cluster object.

Author(s)

Joshua French

References


See Also

print.smerc_cluster, summary.smerc_cluster, plot.smerc_cluster, scan.test

Examples

data(nydf)
data(nyw)
coords = with(nydf, cbind(x, y))
out = bn.test(coords = coords, cases = nydf$cases,
bn.zones

pop = nydf$pop, cstar = 6,
alpha = 0.1)

plot(out)
data(nypoly)
library(sp)
plot(nypoly, col = color.clusters(out))

bn.zones

Determine case windows (circles)

Description

bn.zones determines the case windows (circles) for the Besag-Newell method.

Usage

bn.zones(d, cases, cstar)
casewin(d, cases, cstar)

Arguments

d     An $n \times n$ square distance matrix containing the intercentroid distance between
the $n$ region centroids.
cases A vector of length $n$ containing the observed number of cases for the $n$ region
centroids.
cstar A non-negative integer indicating the minimum number of cases to include in
each window.

Details

Using the distances provided in d, for each observation, the nearest neighbors are included in in-
creasingly larger windows until at least cstar cases are included in the window. Each row of d is
matched with the same position in cases.

Value

Returns the indices of the regions in each case window as a list. For each element of the list, the
indices are ordered from nearest to farthest from each centroid (and include the starting region).

Author(s)

Joshua French

References

Statistical Society, Series A, 154, 327-333.
Examples

data(nydf)
cords = as.matrix(nydf[,c("longitude", "latitude")])
d = sp::spDists(cords, longlat = FALSE)
cwens = bn.zones(d, cases = nydf$cases, cstar = 6)

cepp.sim

Perform cepp.test on simulated data

Description

cepp.sim efficiently performs cepp.test on a simulated data set. The function is meant to be used internally by the cepp.test function, but is informative for better understanding the implementation of the test.

Usage

cepp.sim(nsim = 1, nn, ty, ex, wts, simdist = "multinomial")

Arguments

nsim A positive integer indicating the number of simulations to perform.
nn A list of nearest neighbors produced by casewin.
ty The total number of cases in the study area.
ex The expected number of cases for each region. The default is calculated under the constant risk hypothesis.
wts A list that has the weights associated with each region of each element of nn.
simdist A character string indicating whether the simulated data should come from a "multinomial" or "poisson" distribution. The default is "multinomial", which fixes the total number of cases observed in each simulated data set.

Value

A vector with the maximum test statistic for each simulated data set.

Examples

data(nydf)
cords = with(nydf, cbind(longitude, latitude))
d = sp::spDists(as.matrix(cords), longlat = TRUE)
nn = casewin(d, cases = nydf$pop, cstar = 15000)
cases = floor(nydf$cases)
ty = sum(cases)
ex = ty/sum(nydf$pop) * nydf$pop
# find smallest windows with at least n* pop
nstar = 1000
nn = casewin(d, cases = nydf$pop, cstar = nstar)
# determine ts
wts = cepp.weights(nn, nydf$pop, nstar)
tsim = cepp.sim(1, nn = nn, ty = ty, ex = ex, wts = wts)

---

cepp.test

### Cluster Evaluation Permutation Procedure Test

**Description**

cepp.test implements the Cluster Evaluation Permutation Procedure test of Turnbull et al. (1990) for finding disease clusters.

**Usage**

```r
cepp.test(
  coords,
  cases,
  pop,
  nstar,
  ex = sum(cases)/sum(pop) * pop,
  nsim = 499,
  alpha = 0.1,
  longlat = FALSE,
  simdist = "multinomial"
)
```

**Arguments**

- `coords`: An \( n \times 2 \) matrix of centroid coordinates for the regions.
- `cases`: The number of cases observed in each region.
- `pop`: The population size associated with each region.
- `nstar`: The size of the at-risk population in each window.
- `ex`: The expected number of cases for each region. The default is calculated under the constant risk hypothesis.
- `nsim`: The number of simulations from which to compute the p-value.
- `alpha`: The significance level to determine whether a cluster is significant. Default is 0.10.
- `longlat`: The default is FALSE, which specifies that Euclidean distance should be used. If `longlat` is TRUE, then the great circle distance is used to calculate the inter-centroid distance.
- `simdist`: A character string indicating whether the simulated data should come from a "multinomial" or "poisson" distribution. The default is "multinomial", which fixes the total number of cases observed in each simulated data set.
cepp.weights

Value

Returns a smerc_cluster object.

Author(s)

Joshua French

References


See Also

print.smerc_cluster, summary.smerc_cluster, plot.smerc_cluster, scan.test

Examples

data(nydf)
data(nyw)
coords = with(nydf, cbind(x, y))
cases = nydf$cases
pop = nydf$pop
out = cepp.test(coords = coords, cases = cases, pop = pop, nstar = 1000, alpha = 0.1)
plot(out)

data(nypoly)
library(sp)
plot(nypoly, col = color.clusters(out))

---

cepp.weights Compute region weights for cepp.test

Description

Compute region weights for cepp.test

Usage

cepp.weights(nn, pop, nstar)

Arguments

nn A list of nearest neighbors produced by casewin.
pop The population size associated with each region.
nstar The size of the at-risk population in each window.
clusters

Description

clusters extracts the clusters contained in a smerc_cluster object.

Usage

clusters(x)

Arguments

x An object of class smerc_cluster.

Value

A list. Each element of the list is a vector with the indices of event locations in the associated cluster.

Examples

data(nydf)
coords = with(nydf, cbind(x, y))
pop = nydf$pop
# intercentroid distances
d = sp::spDistso(coords)
# find smallest windows with cumulative population of
# at least n* = 1000
nn = casewin(d, pop, 1000)
# compute weights
w = cepp.weights(nn, pop, 1000)

clusters(out)
color.clusters  

**Color clusters**

**Description**

color.clusters is a helper function to color clusters of regions produced by an appropriate method, e.g., scan.test or uls.test. Regions that are not part of any cluster have no color.

**Usage**

color.clusters(x, col = grDevices::hcl.colors(length(x$clusters)))

**Arguments**

- **x**: An object of class scan produced by a function such as scan.test.
- **col**: A vector of colors to color the clusters in x. Should have same length as the number of clusters in x.

**Value**

Returns a vector with colors for each region/centroid for the data set used to construct x.

**Author(s)**

Joshua French

**Examples**

data(nydf)
coords = with(nydf, cbind(longitude, latitude))
out = scan.test(coords = coords, cases = floor(nydf$cases),
                pop = nydf$pop, alpha = 0.12, longlat = TRUE,
                nsim = 9)
data(nypoly)
library(sp)
plot(nypoly, col = color.clusters(out))

---

combine.zones  

**Combine distinct zones**

**Description**

combine.zones combines the elements of z1 and z2 into a single list, returning only the unique zones.
Usage

combine.zones(z1, z2)

Arguments

z1  A list of zones
z2  A list of zones

Value

A list of distinct zones

Examples

z1 = list(1:2, 1:3)
z2 = list(2:1, 1:4)
combine.zones(z1, z2)

csg2  Construct connected subgraphs

Description

csg2, lcsg2, and scsg2 construct connected subgraphs. These functions are not intended for users.
nn contains a list of nearest neighbors for each region. idx is a vector of possible vertices being
considered as a subgraph. w is a connectivity matrix relating the N vertices. w[i, j] = 1 if vertices
i and j are connected, i.e., if they share an edge. The dimensions of w are Ntimesk, where k =
length(idx). While the rows of w contain adjacency information for all N vertices, only the idx
columns of the complete adjacency matrix are used in w. See Details for discussion of scsg.

Usage

csg2(cz, cnn, cw)
lcsg2(lcz, cnn, cw)
scsg2(
    nn,
    w,
    idx = seq_along(nn),
    nlevel = NULL,
    verbose = FALSE,
    logical = FALSE
)
Arguments

- **cz**: A logical vector representing the current subgraph.
- **cnn**: The indices of the neighbors of the current vertex.
- **cw**: A binary adjacency matrix for the neighbors of the current vertex.
- **lcz**: A list of current zones (in the form of logical vectors).
- **nn**: A list of the nearest neighbors for each vertex (region).
- **w**: A binary adjacency matrix indicating connected neighbors.
- **idx**: A vector of vertices for which to construct the set of connected subgraphs.
- **nlevel**: The maximum size of each subgraph.
- **verbose**: A logical value indicating whether descriptive messages should be provided. Default is `FALSE`. If `TRUE`, this can be useful for diagnosing where the sequences of connected subgraphs is slowing down/having problems.
- **logical**: A logical value indicating whether a list of logical vectors should be returned. The default is `FALSE`, indicating that the `scsg` function should return a list of vectors with each vector containing the vertex indices included in each subgraph.

Details

`scsg2` performs a sequence of `lcsg2` calls. Starting with `lcz = list(idx[1])`, `scsg` keeps iteratively building more connected subgraphs by performing something like: `lcz1 = list(idx[1])`. `lcz2 = lcsg2(lcz1, ...)`. `lcz3 = lcsg2(lcz2, ...)`. This is done until there are no more connected subgraphs among the elements of `idx`.

Value

A list with all possible connected subgraphs based on the user-provided parameters.

Examples

```r
data(nydf)
data(nyw)
# determine 50 nn of region 1 for NY data
coords = as.matrix(nydf[,c("longitude", "latitude")])
nn3 = knn(coords, longlat = TRUE, k = 3)
z1 = scsg2(nn3, nyw)
z2 = flex.zones(coords, nyw, k = 3, longlat = TRUE)
all.equal(z1, z2)
```
**dc.sim**  
*Perform dc.test on simulated data*

**Description**

dc.sim efficiently performs dc.test on a simulated data set. The function is meant to be used internally by the dc.test function, but is informative for better understanding the implementation of the test.

**Usage**

dc.sim(nsim = 1, nn, ty, ex, w, pop, max_pop, cl = NULL)

**Arguments**

- `nsim`  A positive integer indicating the number of simulations to perform.
- `nn`  A list of distance-based nearest neighbors, preferably from the nndist function.
- `ty`  The total number of cases in the study area.
- `ex`  The expected number of cases for each region. The default is calculated under the constant risk hypothesis.
- `w`  A binary spatial adjacency matrix for the regions.
- `pop`  The population size associated with each region.
- `max_pop`  The population upperbound (in total population) for a candidate zone.
- `cl`  A cluster object created by makeCluster, or an integer to indicate number of child-processes (integer values are ignored on Windows) for parallel evaluations (see Details on performance).

**Value**

A vector with the maximum test statistic for each simulated data set.

**Examples**

data(nydf)  
data(nyw)  
(coords = with(nydf, cbind(longitude, latitude)))  
cases = floor(nydf$cases)  
pop = nydf$pop  
ty = sum(cases)  
ex = ty/sum(pop) * pop  
d = sp::spDists(coords, longlat = TRUE)  
nn = nndist(d, ubd = 0.05)  
max_pop = sum(pop) * 0.25  

tsim = dc.sim(1, nn, ty, ex, nyw, pop = pop,  
               max_pop = max_pop)
Description

`dc.test` implements the Double Connection spatial scan test of Costa et al. (2012). Starting with a single region as a current zone, new candidate zones are constructed by combining the current zone with the connected region that maximizes the resulting likelihood ratio test statistic, with the added constraint that the region must have at least two connection (i.e., shares a border with) at least two of the regions in the current zone. This procedure is repeated until adding a connected region does not increase the test statistic (or the population or distance upper bounds are reached). The same procedure is repeated for each region. The clusters returned are non-overlapping, ordered from most significant to least significant. The first cluster is the most likely to be a cluster. If no significant clusters are found, then the most likely cluster is returned (along with a warning).

Usage

```r
dc.test(
  coords,
  cases,
  pop,
  w,
  ex = sum(cases)/sum(pop) * pop,
  nsim = 499,
  alpha = 0.1,
  ubpop = 0.5,
  ubd = 1,
  longlat = FALSE,
  cl = NULL
)
```

Arguments

- `coords` An \( n \times 2 \) matrix of centroid coordinates for the regions.
- `cases` The number of cases observed in each region.
- `pop` The population size associated with each region.
- `w` A binary spatial adjacency matrix for the regions.
- `ex` The expected number of cases for each region. The default is calculated under the constant risk hypothesis.
- `nsim` The number of simulations from which to compute the p-value.
- `alpha` The significance level to determine whether a cluster is significant. Default is 0.10.
- `ubpop` The upperbound of the proportion of the total population to consider for a cluster.
A proportion in \((0, 1]\). The distance of potential clusters must be no more than \(\text{ubd} \times m\), where \(m\) is the maximum intercentroid distance between all coordinates.

The default is \(\text{FALSE}\), which specifies that Euclidean distance should be used. If \(\text{longlat}\) is \(\text{TRUE}\), then the great circle distance is used to calculate the intercentroid distance.

A cluster object created by \texttt{makeCluster}, or an integer to indicate number of child-processes (integer values are ignored on Windows) for parallel evaluations (see Details on performance).

The maximum intercentroid distance can be found by executing the command: \texttt{sp::spDists(as.matrix(coords), longlat = longlat)} based on the specified values of \(\text{coords}\) and \(\text{longlat}\).

Returns a \texttt{smerc_cluster} object.

Joshua French


\texttt{print.smerc_cluster}, \texttt{summary.smerc_cluster}, \texttt{plot.smerc_cluster}, \texttt{scan.stat}, \texttt{scan.test}

\begin{verbatim}
data(nydf)
data(nyw)
coords = with(nydf, cbind(longitude, latitude))
out = dc.test(coords = coords, cases = floor(nydf$cases),
              pop = nydf$population, w = nyw,
              alpha = 0.12, longlat = TRUE,
              nsim = 5, ubpop = 0.1, ubd = 0.2)
data(nypoly)
library(sp)
plot(nypoly, col = color.clusters(out))
\end{verbatim}
dc.zones

Determine zones for the Double Connected scan test

Description

dc.zones determines the zones for the Double Connected scan test (dc.test). The function returns the zones, as well as the associated test statistic, cases in each zone, the expected number of cases in each zone, and the population in each zone.

Usage

dc.zones(
  coords,
  cases,
  pop,
  w,
  ex = sum(cases)/sum(pop) * pop,
  ubpop = 0.5,
  ubd = 1,
  longlat = FALSE,
  cl = NULL,
  progress = TRUE
)

Arguments

  coords  An \( n \times 2 \) matrix of centroid coordinates for the regions.
  cases   The number of cases observed in each region.
  pop     The population size associated with each region.
  w       A binary spatial adjacency matrix for the regions.
  ex      The expected number of cases for each region. The default is calculated under
           the constant risk hypothesis.
  ubpop   The upperbound of the proportion of the total population to consider for a clus-
           ter.
  ubd     A proportion in \((0, 1]\). The distance of potential clusters must be no more than
           \( ubd \times m \), where \( m \) is the maximum intercentroid distance between all coordinates.
  longlat The default is FALSE, which specifies that Euclidean distance should be used. If
           longlat is TRUE, then the great circle distance is used to calculate the inter-
           centroid distance.
  cl      A cluster object created by makeCluster, or an integer to indicate number of
           child-processes (integer values are ignored on Windows) for parallel evaluations
           (see Details on performance).
  progress A logical value indicating whether a progress bar should be displayed. The
default is TRUE.
Details

Every zone considered must have a total population less than \( \text{ubpop} \times \text{sum(pop)} \). Additionally, the maximum intercentroid distance for the regions within a zone must be no more than \( \text{ubd} \times \text{the maximum intercentroid distance across all regions} \).

Value

Returns a list with elements:

- **zones**: A list contained the location ids of each potential cluster.
- **loglikrat**: The loglikelihood ratio for each zone (i.e., the log of the test statistic).
- **cases**: The observed number of cases in each zone.
- **expected**: The expected number of cases each zone.
- **pop**: The total population in each zone.

Author(s)

Joshua French

References


Examples

```r
data(nydf)
data(nyw)
coords = as.matrix(nydf[,c("longitude", "latitude")])
# find zone with max statistic starting from each individual region
all_zones = dc.zones(coords, cases = floor(nydf$cases),
nydf$pop, w = nyw, ubpop = 0.25,
ubd = .25, longlat = TRUE)
```

```r
dist.ellipse
```

Compute minor axis distance of ellipse

Description

dist.ellipse computes the length of the minor axis needed for an ellipse of a certain shape and angle to intersect each of the other coordinates from a starting coordinate.

Usage

dist.ellipse(coords, shape, angle)
Arguments

coords    An $N \times 2$ matrix of coordinates
shape     The ratio of the major axis to the minor axis of the ellipse
angle     The angle of the ellipse in the range $[0, 180)$.

Value

A matrix of distances between each coordinate and all other coordinates (and itself). Each row contains the distances for a coordinate.

Examples

data(nydf)
coords = as.matrix(nydf[,c("x", "y")])
d = dist.ellipse(coords, 4, 15)

---

distinct

Distinct elements of a list

Description

distinct takes a list of integer vectors and returns the list indices that contain unique combinations of elements. This function is NOT robust against misuse, so please use properly.

Usage

distinct(x, N = max(unlist(x)))

Arguments

x         A list of integers
N         The largest integer value across all elements of x.

Details

Assume that $k$ is the largest integer value in $x$. A vector of the largest $k$ prime numbers is obtained (call this $pri$). The algorithm takes the sum of the log of $pri[x[[i]]]$ for each element of $x$, and determines which sums are unique. This is why the elements of $x$ must be integer vectors. The prime aspect of the algorithm is critical, as it ensures that a none of the values are multiples of the others, ensuring uniqueness.

Note: this algorithm has only been applied to data sets where each element of $x[[i]]$ appears only once, though it should work for repeats also.

Value

A vector with the distinct indices.
Author(s)

Joshua French

References

Algorithm based on suggestion at https://stackoverflow.com/a/29824978.

Examples

```R
x = list(1:3, 3:1, 1:4, 4:1, c(1, 2, 4, 6), c(6, 4, 1, 2))
x[distinct(x)]
```

---

dmst.sim

Perform dmst.test on simulated data

Description

dmst.sim efficiently performs dmst.test on a simulated data set. The function is meant to be used internally by the dmst.test function, but is informative for better understanding the implementation of the test.

Usage

```R
dmst.sim(nsim = 1, nn, ty, ex, w, pop, max_pop, cl = NULL)
```

Arguments

- `nsim`: A positive integer indicating the number of simulations to perform.
- `nn`: A list of distance-based nearest neighbors, preferably from the nndist function.
- `ty`: The total number of cases in the study area.
- `ex`: The expected number of cases for each region. The default is calculated under the constant risk hypothesis.
- `w`: A binary spatial adjacency matrix for the regions.
- `pop`: The population size associated with each region.
- `max_pop`: The population upperbound (in total population) for a candidate zone.
- `cl`: A cluster object created by makeCluster, or an integer to indicate number of child-processes (integer values are ignored on Windows) for parallel evaluations (see Details on performance).

Value

A vector with the maximum test statistic for each simulated data set.
Examples

```r
data(nydf)
data(nyw)
coords = with(nydf, cbind(longitude, latitude))
cases = floor(nydf$cases)
pop = nydf$pop
ty = sum(cases)
ex = ty/sum(pop) * pop
d = sp::spDists(coords, longlat = TRUE)
nn = nndist(d, ubd = 0.05)
max_pop = sum(pop) * 0.25
tsim = dmst.sim(1, nn, ty, ex, nyw, pop = pop,
               max_pop = max_pop)
```

---

dmst.test

*Dynamic Minimum Spanning Tree spatial scan test*

Description

dmst.test implements the dynamic Minimum Spanning Tree scan test of Assuncao et al. (2006). Starting with a single region as a current zone, new candidate zones are constructed by combining the current zone with the connected region that maximizes the resulting likelihood ratio test statistic. This procedure is repeated until the population or distance upper bounds are reached. The same procedure is repeated for each region. The clusters returned are non-overlapping, ordered from most significant to least significant. The first cluster is the most likely to be a cluster. If no significant clusters are found, then the most likely cluster is returned (along with a warning).

Usage

```r
dmst.test(
    coords,
    cases,
    pop,
    w,
    ex = sum(cases)/sum(pop) * pop,
    nsim = 499,
    alpha = 0.1,
    ubpop = 0.5,
    ubd = 1,
    longlat = FALSE,
    cl = NULL
)
```

Arguments

- `coords` An \( n \times 2 \) matrix of centroid coordinates for the regions.
- `cases` The number of cases observed in each region.
dmst.test

pop
The population size associated with each region.

w
A binary spatial adjacency matrix for the regions.

ex
The expected number of cases for each region. The default is calculated under the constant risk hypothesis.

nsim
The number of simulations from which to compute the p-value.

alpha
The significance level to determine whether a cluster is significant. Default is 0.10.

ubpop
The upperbound of the proportion of the total population to consider for a cluster.

ubd
A proportion in (0, 1]. The distance of potential clusters must be no more than ubd * m, where m is the maximum intercentroid distance between all coordinates.

longlat
The default is FALSE, which specifies that Euclidean distance should be used. If longlat is TRUE, then the great circle distance is used to calculate the intercentroid distance.

c1
A cluster object created by makeCluster, or an integer to indicate number of child-processes (integer values are ignored on Windows) for parallel evaluations (see Details on performance).

Details
The maximum intercentroid distance can be found by executing the command: sp::spDists(as.matrix(coords), longlat = longlat), based on the specified values of coords and longlat.

Value
Returns a smerc_cluster object.

Author(s)
Joshua French

References

See Also
print.smerc_cluster, summary.smerc_cluster, plot.smerc_cluster, scan.stat, scan.test

Examples

```R
 data(nydf)
 data(nyw)
 coords = with(nydf, cbind(longitude, latitude))
 out = dmst.test(coords = coords, cases = floor(nydf$cases),
 pop = nydf$pop, w = nyw,
 alpha = 0.12, longlat = TRUE,
```
dmst.zones

Determine zones for the Dynamic Minimum Spanning Tree scan test

Description

dmst.zones determines the zones for the Dynamic Minimum Spanning Tree scan test (dmst.test). The function returns the zones, as well as the associated test statistic, cases in each zone, the expected number of cases in each zone, and the population in each zone.

Usage

dmst.zones(
  coords,
  cases,
  pop,
  w,
  ex = sum(cases)/sum(pop) * pop,
  ubpop = 0.5,
  ubd = 1,
  longlat = FALSE,
  cl = NULL,
  progress = TRUE
)

Arguments

- coords: An $n \times 2$ matrix of centroid coordinates for the regions.
- cases: The number of cases observed in each region.
- pop: The population size associated with each region.
- w: A binary spatial adjacency matrix for the regions.
- ex: The expected number of cases for each region. The default is calculated under the constant risk hypothesis.
- ubpop: The upperbound of the proportion of the total population to consider for a cluster.
- ubd: A proportion in $(0, 1]$. The distance of potential clusters must be no more than $ubd \times m$, where $m$ is the maximum intercentroid distance between all coordinates.
- longlat: The default is FALSE, which specifies that Euclidean distance should be used. If longlat is TRUE, then the great circle distance is used to calculate the intercentroid distance.
cl A cluster object created by `makeCluster`, or an integer to indicate number of child-processes (integer values are ignored on Windows) for parallel evaluations (see Details on performance).

progress A logical value indicating whether a progress bar should be displayed. The default is TRUE.

Details

Every zone considered must have a total population less than `ubpop * sum(pop)`. Additionally, the maximum intercentroid distance for the regions within a zone must be no more than `ubd * the maximum intercentroid distance across all regions`.

Value

Returns a list with elements:

- `zones` A list contained the location ids of each potential cluster.
- `loglikrat` The loglikelihood ratio for each zone (i.e., the log of the test statistic).
- `cases` The observed number of cases in each zone.
- `expected` The expected number of cases each zone.
- `pop` The total population in each zone.

Author(s)

Joshua French

References


Examples

data(nydf)
data(nyw)
coords = as.matrix(nydf[,c("longitude", "latitude")])
# find zone with max statistic starting from each individual region
all_zones = dmst.zones(coords, cases = floor(nydf$cases),
                     nydf$pop, w = nyw, ubpop = 0.25,
                     ubd = .25, longlat = TRUE)
edmst.sim

Perform edmst.test on simulated data

Description

edmst.sim efficiently performs edmst.test on a simulated data set. The function is meant to be used internally by the edmst.test function, but is informative for better understanding the implementation of the test.

Usage

edmst.sim(nsim = 1, nn, ty, ex, w, pop, max_pop, cl = NULL)

Arguments

- **nsim**: A positive integer indicating the number of simulations to perform.
- **nn**: A list of distance-based nearest neighbors, preferably from the nndist function.
- **ty**: The total number of cases in the study area.
- **ex**: The expected number of cases for each region. The default is calculated under the constant risk hypothesis.
- **w**: A binary spatial adjacency matrix for the regions.
- **pop**: The population size associated with each region.
- **max_pop**: The population upperbound (in total population) for a candidate zone.
- **cl**: A cluster object created by makeCluster, or an integer to indicate number of child-processes (integer values are ignored on Windows) for parallel evaluations (see Details on performance).

Value

A vector with the maximum test statistic for each simulated data set.

Examples

data(nydf)
data(nyw)
coords = with(nydf, cbind(longitude, latitude))
cases = floor(nydf$cases)
pop = nydf$pop
ty = sum(cases)
ex = ty/sum(pop) * pop
d = sp::spDists(coords, longlat = TRUE)
nn = nndist(d, ubd = 0.05)
max_pop = sum(pop) * 0.25
tsim = edmst.sim(1, nn, ty, ex, nyw, pop = pop,
                 max_pop = max_pop)
**Description**

`edmst.test` implements the early stopping dynamic Minimum Spanning Tree scan test of Costa et al. (2012). Starting with a single region as a current zone, new candidate zones are constructed by combining the current zone with the connected region that maximizes the resulting likelihood ratio test statistic. This procedure is repeated until adding a connected region does not increase the test statistic (or the population or distance upper bounds are reached). The same procedure is repeated for each region. The clusters returned are non-overlapping, ordered from most significant to least significant. The first cluster is the most likely to be a cluster. If no significant clusters are found, then the most likely cluster is returned (along with a warning).

**Usage**

```r
edmst.test(
    coords,  
cases,   
pop,     
w,       
ex = sum(cases)/sum(pop) * pop,  
nsim = 499,  
alpha = 0.1,  
uinbox = 0.5,  
ubd = 1,  
longlat = FALSE,  
sl = NULL
)
```

**Arguments**

- `coords`: An $n \times 2$ matrix of centroid coordinates for the regions.
- `cases`: The number of cases observed in each region.
- `pop`: The population size associated with each region.
- `w`: A binary spatial adjacency matrix for the regions.
- `ex`: The expected number of cases for each region. The default is calculated under the constant risk hypothesis.
- `nsim`: The number of simulations from which to compute the p-value.
- `alpha`: The significance level to determine whether a cluster is significant. Default is 0.10.
- `ubbox`: The upperbound of the proportion of the total population to consider for a cluster.
- `ubd`: A proportion in (0, 1]. The distance of potential clusters must be no more than `ubd * m`, where $m$ is the maximum intercentroid distance between all coordinates.
longlat

The default is FALSE, which specifies that Euclidean distance should be used. If longlat is TRUE, then the great circle distance is used to calculate the inter-centroid distance.

cl

A cluster object created by `makeCluster`, or an integer to indicate number of child-processes (integer values are ignored on Windows) for parallel evaluations (see Details on performance).

Details

The maximum intercentroid distance can be found by executing the command: `sp::spDists(as.matrix(coords), longlat = longlat)`, based on the specified values of coords and longlat.

Value

Returns a `smerc_cluster` object.

Author(s)

Joshua French

References


See Also

`print.smerc_cluster`, `summary.smerc_cluster`, `plot.smerc_cluster`, `scan.stat`, `scan.test`

Examples

data(nydf)
data(nyw)
coords = with(nydf, cbind(longitude, latitude))
out = edmst.test(coords = coords, cases = floor(nydf$cases),
    pop = nydf$pop, w = nyw,
    alpha = 0.12, longlat = TRUE,
    nsim = 5, ubpop = 0.1, ubd = 0.2)
data(nypoly)
library(sp)
plot(nypoly, col = color.clusters(out))
**edmst.zones**

Determine zones for the early stopping dynamic Minimum Spanning Tree scan test

**Description**

edmst.zones determines the zones for the early stopping Dynamic Minimum Spanning Tree scan test (edmst.test). The function returns the zones, as well as the associated test statistic, cases in each zone, the expected number of cases in each zone, and the population in each zone.

**Usage**

```r
edmst.zones(
  coords, 
  cases, 
  pop, 
  w, 
  ex = sum(cases)/sum(pop) * pop, 
  ubpop = 0.5, 
  ubd = 1, 
  longlat = FALSE, 
  cl = NULL, 
  progress = TRUE
)
```

**Arguments**

- `coords`: An \( n \times 2 \) matrix of centroid coordinates for the regions.
- `cases`: The number of cases observed in each region.
- `pop`: The population size associated with each region.
- `w`: A binary spatial adjacency matrix for the regions.
- `ex`: The expected number of cases for each region. The default is calculated under the constant risk hypothesis.
- `ubpop`: The upperbound of the proportion of the total population to consider for a cluster.
- `ubd`: A proportion in \((0, 1]\). The distance of potential clusters must be no more than \(ubd \times m\), where \(m\) is the maximum intercentroid distance between all coordinates.
- `longlat`: The default is `FALSE`, which specifies that Euclidean distance should be used. If `longlat` is `TRUE`, then the great circle distance is used to calculate the intercentroid distance.
- `cl`: A cluster object created by `makeCluster`, or an integer to indicate number of child-processes (integer values are ignored on Windows) for parallel evaluations (see Details on performance).
- `progress`: A logical value indicating whether a progress bar should be displayed. The default is `TRUE`.

**Description**

edmst.zones determines the zones for the early stopping Dynamic Minimum Spanning Tree scan test (edmst.test). The function returns the zones, as well as the associated test statistic, cases in each zone, the expected number of cases in each zone, and the population in each zone.
Details

Every zone considered must have a total population less than ubpop * sum(pop). Additionally, the maximum intercentroid distance for the regions within a zone must be no more than ubd * the maximum intercentroid distance across all regions.

Value

Returns a list with elements:

- zones: A list containing the location ids of each potential cluster.
- loglikrat: The loglikelihood ratio for each zone (i.e., the log of the test statistic).
- cases: The observed number of cases in each zone.
- expected: The expected number of cases each zone.
- pop: The total population in each zone.

Author(s)

Joshua French

References


Examples

data(nydf)
data(nyw)
coords = as.matrix(nydf[,c("longitude", "latitude")])
# find zone with max statistic starting from each individual region
all_zones = edmst.zones(coords, cases = floor(nydf$cases),
                         nydf$pop, w = nyw, ubpop = 0.25,
                         ubd = .25, longlat = TRUE)

---

elbow_point

Compute Elbow Point

Description

elbow_point computes the elbow point based on the maximum distance between each point and the line passing through the end points.

Usage

elbow_point(x, y)
Arguments

- **x**: A numeric vector
- **y**: A numeric vector

Value

A list with the index (idx), x-value (x) and y-value (y) of the elbow point.

Author(s)

Joshua French and Mohammad Meysami

References

https://en.wikipedia.org/wiki/Distance_from_a_point_to_a_line

See Also

optimal_ubpop

Examples

```r
# generate some data
x = c(0, 0.5, 1)
y = c(1, 0.1, 0)
# plot data (the second point is clearly the elbow)
plot(x, y)
elbow_point(x, y)
```

Description

elliptic.nn computes the nearest neighbors relationships for elliptic.test. It will provide a list of nearest neighbors, and a list of the associated shape and angle.

Usage

```r
elliptic.nn(
    coords,
    pop,
    ubpop = 0.5,
    shape = c(1, 1.5, 2, 3, 4, 5),
    nangle = c(1, 4, 6, 9, 12, 15)
)```
Arguments

- `coords`: An $n \times 2$ matrix of centroid coordinates for the regions.
- `pop`: The population size associated with each region.
- `ubpop`: The upperbound of the proportion of the total population to consider for a cluster.
- `shape`: The ratios of the major and minor axes of the desired ellipses.
- `nangle`: The number of angles (between 0 and 180) to consider for each shape.

Value

A list of nested nearest neighbors, the associated shapes and angles for each set of nn, and all of the shapes and angles you get for each zone constructed from the set of nearest neighbors.

Examples

```r
data(nydf)
coords = with(nydf, cbind(longitude, latitude))
enn = elliptic.nn(coords, nydf$pop, 0.1,
    shape = c(1, 1.5), nangle = c(1, 4))
```

---

elliptic.sim: Perform `elliptic.test` on simulated data

Description

`elliptic.sim` efficiently performs `elliptic.test` on a simulated data set. The function is meant to be used internally by the `elliptic.test` function, but is informative for better understanding the implementation of the test.

Usage

```r
elliptic.sim(nsim = 1, nn, ty, ex, a, shape_all, ein, eout, cl = NULL)
```

Arguments

- `nsim`: A positive integer indicating the number of simulations to perform.
- `nn`: A list of nearest neighbors produced by `elliptic.nn`.
- `ty`: The total number of cases in the study area.
- `ex`: The expected number of cases for each region. The default is calculated under the constant risk hypothesis.
- `a`: The penalty for the spatial scan statistic. The default is 0.5.
- `shape_all`: A vector of the shapes associated with all of the possible zones constructed from `nn`. This can be obtained from `elliptic.nn`.
elliptic.test

The expected number of cases in the zone. Conventionally, this is the estimated overall disease risk across the study area, multiplied by the total population size of the zone.

eout

The expected number of cases outside the zone. This should be ty - ein and is computed automatically if not provided.

c1

A cluster object created by makeCluster, or an integer to indicate number of child-processes (integer values are ignored on Windows) for parallel evaluations (see Details on performance).

Value

A vector with the maximum test statistic for each simulated data set.

Examples

data(nydf)

data(nyw)

coords = with(nydf, cbind(longitude, latitude))

pop = nydf$pop

enn = elliptic.nn(coords, pop, ubpop = 0.1,

shape = c(1, 1.5), nangle = c(1, 4))

cases = floor(nydf$cases)

ty = sum(cases)

ex = ty/sum(pop) * pop

yin = nn.cumsum(enn$nn, cases)

ein = nn.cumsum(enn$nn, ex)

tsim = elliptic.sim(nsim = 2, nn = enn$nn, ty = ty, ex = ex,

a = 0.5, shape_all = enn$shape_all,

ein = ein, eout = ty - ein)

elliptic.test

Elliptical Spatial Scan Test

Description

elliptic.test performs the elliptical scan test of Kulldorf et al. (2006).

Usage

elliptic.test(

c0ords,

cases,

pop,

ex = sum(cases)/sum(pop) * pop,

nsim = 499,

alpha = 0.1,

ubpop = 0.5,

shape = c(1, 1.5, 2, 3, 4, 5),
elliptic.test

nangle = c(1, 4, 6, 9, 12, 15),
a = 0.5,
cl = NULL,
type = "poisson",
min.cases = 2
)

Arguments

coords  An \( n \times 2 \) matrix of centroid coordinates for the regions.
cases  The number of cases observed in each region.
pop  The population size associated with each region.
ex  The expected number of cases for each region. The default is calculated under the constant risk hypothesis.
nsim  The number of simulations from which to compute the p-value.
alpha  The significance level to determine whether a cluster is significant. Default is 0.10.
ubpop  The upperbound of the proportion of the total population to consider for a cluster.
shape  The ratios of the major and minor axes of the desired ellipses.
nangle  The number of angles (between 0 and 180) to consider for each shape.
a  The penalty for the spatial scan statistic. The default is 0.5.
cl  A cluster object created by \texttt{makeCluster}, or an integer to indicate number of child-processes (integer values are ignored on Windows) for parallel evaluations (see Details on performance).
type  The type of scan statistic to compute. The default is "poisson". The other choice is "binomial".
min.cases  The minimum number of cases required for a cluster. The default is 2.

Details

The test is performed using the spatial scan test based on the Poisson test statistic and a fixed number of cases. Candidate zones are elliptical and extend from the observed data locations. The clusters returned are non-overlapping, ordered from most significant to least significant. The first cluster is the most likely to be a cluster. If no significant clusters are found, then the most likely cluster is returned (along with a warning).

Value

Returns a \texttt{smerc\_cluster} object.

Author(s)

Joshua French
References


See Also

print.smerc_cluster, summary.smerc_cluster, plot.smerc_cluster, scan.stat, scan.test

Examples

data(nydf)
coords = with(nydf, cbind(longitude, latitude))
out = elliptic.test(coords = coords,
cases = floor(nydf$cases),
pop = nydf$pop, ubpop = 0.1,
nsim = 2,
alpha = 0.12,
shape = 1.5, nangle = 4)

elliptic.zones

Determine zones for elliptic.test

Description

elliptic.zones constructs the elliptical zones for elliptic.test.

Usage

elliptic.zones(
  coords,
  pop,
  ubpop = 0.5,
  shape = c(1, 1.5, 2, 3, 4, 5),
  nangle = c(1, 4, 6, 9, 12, 15)
)

Arguments

  coords An n × 2 matrix of centroid coordinates for the regions.
  pop The population size associated with each region.
  ubpop The upperbound of the proportion of the total population to consider for a cluster.
  shape The ratios of the major and minor axes of the desired ellipses.
  nangle The number of angles (between 0 and 180) to consider for each shape.
Value
A list with all distinct zones, the associated shape parameters, and the associated angle parameters.

References

Examples
```r
## Not run:
data(nydf)
coords = with(nydf, cbind(longitude, latitude))
out = elliptic.zones(coords = coords, pop = nydf$pop,
shape = 1.5, nangle = 4)
## End(Not run)
```

---

**fast.sim**  
*Perform fast.test on simulated data*

**Description**

*fast.sim* efficiently performs *fast.test* on a simulated data set. The function is meant to be used internally by the *fast.test* function, but is informative for better understanding the implementation of the test.

**Usage**

```r
fast.sim(nsim = 1, ty, ex, pop, ubpop, type = "poisson", cl = NULL)
```

**Arguments**

- **nsim** A positive integer indicating the number of simulations to perform.
- **ty** The total number of cases in the study area.
- **ex** The expected number of cases for each region. The default is calculated under the constant risk hypothesis.
- **pop** The population size associated with each region.
- **ubpop** The upperbound of the proportion of the total population to consider for a cluster.
- **type** The type of scan statistic to compute. The default is "poisson". The other choice is "binomial".
- **cl** A cluster object created by *makeCluster*, or an integer to indicate number of child-processes (integer values are ignored on Windows) for parallel evaluations (see Details on performance).
Value

A vector with the maximum test statistic for each simulated data set.

Examples

data(nydf)
coords = with(nydf, cbind(longitude, latitude))
cases = floor(nydf$cases)
pop = nydf$pop
ty = sum(cases)
ex = ty/sum(pop) * pop
tsim = fast.sim(1, ty, ex, pop = pop, ubpop = 0.5)

Description

fast.test performs the fast subset scan test of Neill (2012).

Usage

fast.test(
  coords, cases, pop,
  ex = sum(cases)/sum(pop) * pop,
  nsim = 499,
  alpha = 0.1,
  ubpop = 0.5,
  longlat = FALSE,
  cl = NULL,
  type = "poisson"
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>coords</td>
<td>An $n \times 2$ matrix of centroid coordinates for the regions.</td>
</tr>
<tr>
<td>cases</td>
<td>The number of cases observed in each region.</td>
</tr>
<tr>
<td>pop</td>
<td>The population size associated with each region.</td>
</tr>
<tr>
<td>ex</td>
<td>The expected number of cases for each region. The default is calculated under the constant risk hypothesis.</td>
</tr>
<tr>
<td>nsim</td>
<td>The number of simulations from which to compute the p-value.</td>
</tr>
<tr>
<td>alpha</td>
<td>The significance level to determine whether a cluster is significant. Default is 0.10.</td>
</tr>
</tbody>
</table>
ubpop  The upperbound of the proportion of the total population to consider for a cluster.

longlat  The default is FALSE, which specifies that Euclidean distance should be used. If longlat is TRUE, then the great circle distance is used to calculate the intercentroid distance.

cl  A cluster object created by makeCluster, or an integer to indicate number of child-processes (integer values are ignored on Windows) for parallel evaluations (see Details on performance).

type  The type of scan statistic to compute. The default is "poisson". The other choice is "binomial".

Details

The test is performed using the spatial scan test based on the Poisson test statistic and a fixed number of cases. The windows are based on the Upper Level Sets proposed by Patil and Taillie (2004). The clusters returned are non-overlapping, ordered from most significant to least significant. The first cluster is the most likely to be a cluster. If no significant clusters are found, then the most likely cluster is returned (along with a warning).

Value

Returns a list of length two of class scan. The first element (clusters) is a list containing the significant, non-overlapping clusters, and has the the following components:

locids  The location ids of regions in a significant cluster.
pop  The total population in the cluster window.
cases  The observed number of cases in the cluster window.
expected  The expected number of cases in the cluster window.
smr  Standardized mortality ratio (observed/expected) in the cluster window.
rr  Relative risk in the cluster window.
loglikrat  The loglikelihood ratio for the cluster window (i.e., the log of the test statistic).
pvalue  The pvalue of the test statistic associated with the cluster window.

The second element of the list is the centroid coordinates. This is needed for plotting purposes.

Author(s)

Joshua French

References


See Also

print.smerc_cluster, summary.smerc_cluster, plot.smerc_cluster, scan.stat, scan.test
Examples

data(nydf)
coords = with(nydf, cbind(longitude, latitude))
out = fast.test(coords = coords, cases = floor(nydf$cases),
    pop = nydf$pop,
    alpha = 0.05, longlat = TRUE,
    nsim = 49, ubpop = 0.5)

Description

fast.zones determines the unique zones obtained by implementing the fast subset scan method of Neill (2012).

Usage

fast.zones(cases, pop, ubpop = 0.5, simple = TRUE)

Arguments

cases The number of cases observed in each region.
pop The population size associated with each region.
ubpop The upperbound of the proportion of the total population to consider for a cluster.
simple A logical value indicating whether a simple version of the fast zones should be returned. See Details.

Details

The simple argument determines the formatting of the returned zones. If simple = TRUE, then a vector containing the sequential indices of the regions in each successive zones is returned. If simple = FALSE, then the complete list of all zones is returned (which is the standard format of most of the other *.zones functions.

The zones returned must have a total population less than ubpop * sum(pop) of all regions in the study area.

Value

Returns a vector of regions to sequentially and cumulatively consider for clustering.

Author(s)

Joshua French
References

Examples

```r
data(nydf)
cases = nydf$cases
pop = nydf$pop
# compare output format
fast.zones(cases, pop, ubpop = 0.05)
fast.zones(cases, pop, ubpop = 0.05, simple = FALSE)
```

flex.sim

Perform flex.test on simulated data

Description

flex.sim efficiently performs flex.test on a simulated data set. The function is meant to be used internally by the flex.test function, but is informative for better understanding the implementation of the test.

Usage

```r
flex.sim(
  nsim = 1,
  zones,
  ty,
  ex,
  type = "poisson",
  ein = NULL,
  eout = NULL,
  tpop = NULL,
  popin = NULL,
  popout = NULL,
  cl = NULL
)
```

Arguments

- `nsim`: A positive integer indicating the number of simulations to perform.
- `zones`: A list of zones to compute the test statistic over for each simulated data set.
- `ty`: The total number of cases in the study area.
- `ex`: The expected number of cases for each region. The default is calculated under the constant risk hypothesis.
flex.test

Description

data(nydf)
data(nyw)
coords = with(nydf, cbind(longitude, latitude))
zones = flex.zones(coords, w = nyw, k = 3, longlat = TRUE)
cases = floor(nydf$cases)
ty = sum(cases)
ex = ty/sum(nydf$pop) * nydf$pop
ein = zones.sum(zones, ex)
ts = flex.sim(nsim = 2, zones, ty, ex, ein = ein, eout = ty - ein)

flex.test

Flexibly-shaped Spatial Scan Test

Usage

flex.test(
  coords,
  cases,
  pop,
  w,
  k = 10,
flex.test

\[
\text{ex} = \frac{\text{sum(cases)}}{\text{sum(pop)}} \times \text{pop},
\]

\text{type = "poisson",}
\text{nsim = 499,}
\text{alpha = 0.1,}
\text{longlat = FALSE,}
\text{cl = NULL,}
\text{lonlat = longlat,}
\]

Arguments

- **coords**: An \(n \times 2\) matrix of centroid coordinates for the regions.
- **cases**: The number of cases observed in each region.
- **pop**: The population size associated with each region.
- **w**: A binary spatial adjacency matrix for the regions.
- **k**: An integer indicating the maximum number of regions to include in a potential cluster. Default is 10.
- **ex**: The expected number of cases for each region. The default is calculated under the constant risk hypothesis.
- **type**: The type of scan statistic to compute. The default is "poisson". The other choice is "binomial".
- **nsim**: The number of simulations from which to compute the p-value.
- **alpha**: The significance level to determine whether a cluster is significant. Default is 0.10.
- **longlat**: The default is FALSE, which specifies that Euclidean distance should be used. If longlat is TRUE, then the great circle distance is used to calculate the inter-centroid distance.
- **cl**: A cluster object created by `makeCluster`, or an integer to indicate number of child-processes (integer values are ignored on Windows) for parallel evaluations (see Details on performance).
- **lonlat**: Deprecated in favor of longlat.
- **...**: Not used.

Details

The test is performed using the spatial scan test based on the Poisson test statistic and a fixed number of cases. The first cluster is the most likely to be a cluster. If no significant clusters are found, then the most likely cluster is returned (along with a warning).

Value

Returns a list of length two of class `scan`. The first element (clusters) is a list containing the significant, non-ovlapperng clusters, and has the following components:
flex.zones

Author(s)

Joshua French

References


See Also

print.smerc_cluster, summary.smerc_cluster, plot.smerc_cluster, scan.stat, scan.test

Examples

data(nydf)
data(nyw)
coords = with(nydf, cbind(longitude, latitude))
out = flex.test(coords = coords, cases = floor(nydf$cases),
w = nyw, k = 3,
pop = nydf$pop, nsim = 49,
alpha = 0.12, longlat = TRUE)
data(nypoly)
library(sp)
plot(nypoly, col = color.clusters(out))

Description

flex.zones determines the unique zones to consider for the flexibly shaped spatial scan test of Tango and Takahashi (2005). The algorithm uses a breadth-first search to find all subgraphs connected to each vertex (region) in the data set of size \( k \) or less.

Usage

flex.zones(
  coords,
  w,
  k = 10,
  longlat = FALSE,
  cl = NULL,
  loop = FALSE,
  verbose = FALSE,
  pfreq = 1
)
Arguments

- **coords**: An \( n \times 2 \) matrix of centroid coordinates for the regions.
- **w**: A binary spatial adjacency matrix for the regions.
- **k**: An integer indicating the maximum number of regions to include in a potential cluster. Default is 10.
- **longlat**: The default is FALSE, which specifies that Euclidean distance should be used. If longlat is TRUE, then the great circle distance is used to calculate the inter-centroid distance.
- **cl**: A cluster object created by `makeCluster`, or an integer to indicate number of child-processes (integer values are ignored on Windows) for parallel evaluations (see Details on performance).
- **loop**: A logical value indicating whether a loop should be used to implement the function instead of `pbapply`. The default is FALSE. If TRUE, then memory-saving steps are also taken.
- **verbose**: A logical value indicating whether progress messages should be provided. The default is FALSE. If both `loop` and `verbose` are TRUE, informative messages are displayed that can be useful for diagnosing where the sequences of connected subgraphs are slowing down or having problems.
- **pfreq**: The frequency that messages are reported from the loop (if `verbose = TRUE`). The default is `pfreq = 1`, meaning a message is returned for each index of the loop.

Value

Returns a list of zones to consider for clustering. Each element of the list contains a vector with the location ids of the regions in that zone.

Author(s)

Joshua French

References


Examples

data(nydf)
data(nyw)
coords = cbind(nydf$x, nydf$y)
zones = flex.zones(coords, w = nyw, k = 3)
## Not run:
# see what happens when verbose = TRUE
zones = flex.zones(coords, w = nyw, k = 3, verbose = TRUE)
## End(Not run)
**flex_test**

*Flexibly-shaped Spatial Scan Test*

**Description**

`flex_test` performs the flexibly-shaped scan test of Tango and Takahashi (2005).

**Usage**

```r
def flex_test(
    coords,
    cases,
    pop,
    w,
    k = 10,
    ex = sum(cases)/sum(pop) * pop,
    type = "poisson",
    nsim = 499,
    alpha = 0.1,
    longlat = FALSE,
    cl = NULL,
    lonlat = longlat,
    ...
)
```

**Arguments**

- `coords` An \( n \times 2 \) matrix of centroid coordinates for the regions.
- `cases` The number of cases observed in each region.
- `pop` The population size associated with each region.
- `w` A binary spatial adjacency matrix for the regions.
- `k` An integer indicating the maximum number of regions to include in a potential cluster. Default is 10.
- `ex` The expected number of cases for each region. The default is calculated under the constant risk hypothesis.
- `type` The type of scan statistic to compute. The default is "poisson". The other choice is "binomial".
- `nsim` The number of simulations from which to compute the p-value.
- `alpha` The significance level to determine whether a cluster is significant. Default is 0.10.
- `longlat` The default is `FALSE`, which specifies that Euclidean distance should be used. If `longlat` is `TRUE`, then the great circle distance is used to calculate the inter-centroid distance.
A cluster object created by `makeCluster`, or an integer to indicate number of child-processes (integer values are ignored on Windows) for parallel evaluations (see Details on performance).

Depreciated in favor of `longlat`.

Not used.

The test is performed using the spatial scan test based on the Poisson test statistic and a fixed number of cases. The first cluster is the most likely to be a cluster. If no significant clusters are found, then the most likely cluster is returned (along with a warning).

Returns a list of length two of class `scan`. The first element (clusters) is a list containing the significant, non-ovlappering clusters, and has the the following components:

### Author(s)

Joshua French

### References


### See Also

`print.smerc_cluster`, `summary.smerc_cluster`, `plot.smerc_cluster`, `scan.stat`, `scan.test`

### Examples

```r
data(nydf)
data(nyw)
coords = with(nydf, cbind(longitude, latitude))
out = flex_test(coords = coords, cases = floor(nydf$cases),
               w = nyw, k = 3,
               pop = nydf$pop, nsim = 49,
               alpha = 0.12, longlat = TRUE)

data(nypoly)
library(sp)
# plot(nypoly, col = color.clusters(out))
```
flex_zones

Determine zones for flexibly shaped spatial scan test

Description

flex_zones determines the unique zones to consider for the flexibly shaped spatial scan test of Tango and Takahashi (2005). The algorithm uses a breadth-first search to find all subgraphs connected to each vertex (region) in the data set of size \( k \) or less.

Usage

flex_zones(coords, 
            w, 
            k = 10, 
            longlat = FALSE, 
            cl = NULL, 
            loop = FALSE, 
            verbose = FALSE, 
            pfreq = 1)

Arguments

coords
   An \( n \times 2 \) matrix of centroid coordinates for the regions.

w
   A binary spatial adjacency matrix for the regions.

k
   An integer indicating the maximum number of regions to include in a potential cluster. Default is 10

longlat
   The default is FALSE, which specifies that Euclidean distance should be used. If longlat is TRUE, then the great circle distance is used to calculate the inter-centroid distance.

c1
   Ignored, but retained for backwards compatibility

loop
   A logical value indicating whether a loop should be used to implement the function instead of pbapply. The default is FALSE. If TRUE, then memory-saving steps are also taken.

verbose
   A logical value indicating whether progress messages should be provided. The default is FALSE. If both loop and verbose are TRUE, informative messages are displayed that can be useful for diagnosing where the sequences of connected subgraphs are slowing down or having problems.

pfreq
   The frequency that messages are reported from the loop (if verbose = TRUE). The default is pfreq = 1, meaning a message is returned for each index of the loop.
**Value**

Returns a list of zones to consider for clustering. Each element of the list contains a vector with the location ids of the regions in that zone.

**Author(s)**

Joshua French

**References**


**Examples**

```r
data(nydf)
data(nyw)
coords = cbind(nydf$x, nydf$y)
zones = flex_zones(coords, w = nyw, k = 3)
## Not run:
# see what happens when verbose = TRUE
zones = flex_zones(coords, w = nyw, k = 3, verbose = TRUE)
## End(Not run)
```

---

**knn**

*K nearest neighbors*

**Description**

knn returns the k nearest neighbors of the n coordinates in coords. The nearest neighbors are constructed to be self-inclusive, i.e., an observation is its closest neighbor.

**Usage**

```r
knn(coords, longlat = FALSE, k = 1, d = NULL)
```

**Arguments**

- `coords`: An n × 2 matrix of centroid coordinates for the regions.
- `longlat`: The default is FALSE, which specifies that Euclidean distance should be used. If longlat is TRUE, then the great circle distance is used to calculate the inter-centroid distance.
- `k`: An integer indicating the maximum number of regions to include in a potential cluster. Default is 10
- `d`: An n by n distance matrix. If provided, this is used instead of computing d based on coords and longlat.
**lget**

**Value**

An $n \times k$ matrix of nearest neighbors.

**Examples**

```r
data(nydf)
coords = nydf[,c("longitude", "latitude")]
knn(coords, longlat = TRUE, k = 4)
```

---

**lget**  
*Apply getElement over a list*

**Description**

`lget` or `lgetElement` applies `getElement` to a list using `lapply`. `sget` and `sgetElement` do the same thing with `sapply`.

**Usage**

```r
lget(X, name)
```

```r
lgetElement(X, name)
```

```r
sget(X, name, simplify = TRUE, USE.NAMES = TRUE)
```

```r
sgetElement(X, name, simplify = TRUE, USE.NAMES = TRUE)
```

**Arguments**

- **X**  
  A list.

- **name**  
  A literal character string or a name (possibly `backtick` quoted). For extraction, this is normally (see under ‘Environments’) partially matched to the names of the object.

- **simplify**  
  logical or character string; should the result be simplified to a vector, matrix or higher dimensional array if possible? For `sapply` it must be named and not abbreviated. The default value, `TRUE`, returns a vector or matrix if appropriate, whereas if `simplify = "array"` the result may be an array of “rank” (`=length(dim(.))`) one higher than the result of `FUN(X[[i]])`.

- **USE.NAMES**  
  logical; if `TRUE` and if `X` is character, use `X` as names for the result unless it had names already. Since this argument follows ... its name cannot be abbreviated.

**Value**

A list (lget) or vector (sget) of the same length as `X` with the name parts of each element of `X`. 
Examples

e1 = list(x = rnorm(5),
        y = letters[c(1:2, 2:1, 3)],
        z = c(TRUE, TRUE, FALSE, TRUE, TRUE))

e2 = list(x = rnorm(5),
        y = letters[c(1:4, 1)],
        z = c(FALSE, TRUE, FALSE, TRUE, FALSE))

X = list(e1, e2)
lget(X, name = "x")
sget(X, name = "y")

Description

mlf.test implements the Maxima Likelihood First scan test of Yao et al. (2011), which is actually a special case of the Dynamic Minimum Spanning Tree of Assuncao et al. (2006). Find the single region that maximizes the likelihood ratio test statistic. Starting with this single region as a current zone, new candidate zones are constructed by combining the current zone with the connected region that maximizes the likelihood ratio test statistic. This procedure is repeated until the population and/or distance upper bound is reached.

Usage

mlf.test(
  coords,
  cases,
  pop,
  w,
  ex = sum(cases)/sum(pop) * pop,
  nsim = 499,
  alpha = 0.1,
  ubpop = 0.5,
  ubd = 0.5,
  longlat = FALSE,
  cl = NULL
)

Arguments

coords          An \( n \times 2 \) matrix of centroid coordinates for the regions.
cases          The number of cases observed in each region.
pop            The population size associated with each region.
w              A binary spatial adjacency matrix for the regions.
The expected number of cases for each region. The default is calculated under the constant risk hypothesis.

nsim
The number of simulations from which to compute the p-value.

alpha
The significance level to determine whether a cluster is significant. Default is 0.10.

ubpop
The upperbound of the proportion of the total population to consider for a cluster.

ubd
A proportion in $(0, 1]$. The distance of potential clusters must be no more than $ubd \times m$, where $m$ is the maximum intercentroid distance between all coordinates.

longlat
The default is FALSE, which specifies that Euclidean distance should be used. If longlat is TRUE, then the great circle distance is used to calculate the intercentroid distance.

cl
A cluster object created by makeCluster, or an integer to indicate number of child-processes (integer values are ignored on Windows) for parallel evaluations (see Details on performance).

Details
Only a single candidate zone is ever returned because the algorithm only constructs a single sequence of starting zones, and overlapping zones are not returned. Only the zone that maximizes the likelihood ratio test statistic is returned.

Value
Returns a list of length two of class scan. The first element (clusters) is a list containing the significant, non-ovlappering clusters, and has the the following components:

locids
The location ids of regions in a significant cluster.

pop
The total population in the cluser window.

cases
The observed number of cases in the cluster window.

expected
The expected number of cases in the cluster window.

smr
Standardized mortality ratio (observed/expected) in the cluster window.

rr
Relative risk in the cluster window.

loglikrat
The loglikelihood ratio for the cluster window (i.e., the log of the test statistic).

pvalue
The pvalue of the test statistic associated with the cluster window.

w
The adjacency matrix of the cluster.

r
The maximum radius of the cluster (in terms of intercentroid distance from the starting region).

The second element of the list is the centroid coordinates. This is needed for plotting purposes.

Author(s)
Joshua French
References


See Also

`print.smerc_cluster`, `summary.smerc_cluster`, `plot.smerc_cluster`, `scan.stat`, `scan.test`

Examples

```r
data(nydf)
data(nyw)
coords = with(nydf, cbind(longitude, latitude))
out = mlf.test(coords = coords, cases = floor(nydf$cases),
              pop = nydf$pop, w = nyw,
              alpha = 0.12, longlat = TRUE,
              nsim = 10, ubpop = 0.1, ubd = 0.5)
data(nypoly)
library(sp)
plot(nypoly, col = color.clusters(out))
```

`mlf.zones` Determine zones for the maxima likelihood first algorithm.

Description

`mlf.zones` determines the most likely cluster zone obtained by implementing the maxima likelihood first scann method of Yao et al. (2011). Note that this is really just a special case of the dynamic minimum spanning tree (DMST) algorithm of Assuncao et al. (2006)

Usage

```r
mlf.zones(
  coords,
  cases,
  pop,
  w,
  ex = sum(cases)/sum(pop) * pop,
  ubpop = 0.5,
  ubd = 1,
  longlat = FALSE
)
```
Arguments

- **coords**: An $n \times 2$ matrix of centroid coordinates for the regions.
- **cases**: The number of cases observed in each region.
- **pop**: The population size associated with each region.
- **w**: A binary spatial adjacency matrix for the regions.
- **ex**: The expected number of cases for each region. The default is calculated under the constant risk hypothesis.
- **ubpop**: The upperbound of the proportion of the total population to consider for a cluster.
- **ubd**: A proportion in (0, 1]. The distance of potential clusters must be no more than $ubd \times m$, where $m$ is the maximum intercentroid distance between all coordinates.
- **longlat**: The default is FALSE, which specifies that Euclidean distance should be used. If **longlat** is TRUE, then the great circle distance is used to calculate the intercentroid distance.

Details

Each step of the mlf scan test seeks to maximize the likelihood ratio test statistic used in the original spatial scan test (Kulldorff 1997). The first zone considered is the region that maximizes this likelihood ration test statistic, providing that no more than ubpop proportion of the total population is in the zone. The second zone is the first zone and the connected region that maximizes the scan statistic, subject to the population and distance constraints. This pattern continues until no additional zones can be added due to population or distance constraints.

Every zone considered must have a total population less than ubpop * sum(pop) in the study area. Additionally, the maximum intercentroid distance for the regions within a zone must be no more than ubd * the maximum intercentroid distance across all regions.

Value

Returns a list with elements:

- **zones**: A list contained the location ids of each potential cluster.
- **loglikrat**: The loglikelihood ratio for each zone (i.e., the log of the test statistic).
- **cases**: The observed number of cases in each zone.
- **expected**: The expected number of cases each zone.
- **pop**: The total population in each zone.

Author(s)

Joshua French

References

Examples

data(nydf)
data(nyw)
coords = as.matrix(nydf[,c("x", "y")])
mlf.zones(coords, cases = floor(nydf$cases),
          pop = nydf$pop, w = nyw, longlat = TRUE)

mlink.sim Perform mlink.test on simulated data

Description

mlink.sim efficiently performs mlink.test on a simulated data set. The function is meant to be used internally by the mlink.test function, but is informative for better understanding the implementation of the test.

Usage

mlink.sim(nsim = 1, nn, ty, ex, w, pop, max_pop, cl = NULL)

Arguments

nsim A positive integer indicating the number of simulations to perform.
nn A list of distance-based nearest neighbors, preferably from the nndist function.
ty The total number of cases in the study area.
ex The expected number of cases for each region. The default is calculated under the constant risk hypothesis.
w A binary spatial adjacency matrix for the regions.
pop The population size associated with each region.
max_pop The population upperbound (in total population) for a candidate zone.
cl A cluster object created by makeCluster, or an integer to indicate number of child-processes (integer values are ignored on Windows) for parallel evaluations (see Details on performance).

Value

A vector with the maximum test statistic for each simulated data set.

Examples

data(nydf)
data(nyw)
coords = with(nydf, cbind(longitude, latitude))
cases = floor(nydf$cases)
pop = nydf$pop
ty = sum(cases)
ex = ty/sum(pop) * pop

d = sp::spDists(coords, longlat = TRUE)

nn = nndist(d, ubd = 0.05)

max_pop = sum(pop) * 0.25

nsim = mlink.sim(1, nn, ty, ex, nyw, pop = pop,
max_pop = max_pop)

mlink.test

Maximum Linkage spatial scan test

Description

mlink.test implements the Maximum Linkage spatial scan test of Costa et al. (2012). Starting with a single region as a current zone, new candidate zones are constructed by combining the current zone with the connected region that maximizes the resulting likelihood ratio test statistic, with the added constraint that the region has the maximum connections (i.e., shares a border with) with the regions in the current zone. This procedure is repeated until the population or distance upper bounds constraints are reached. The same procedure is repeated for each region. The clusters returned are non-overlapping, ordered from most significant to least significant. The first cluster is the most likely to be a cluster. If no significant clusters are found, then the most likely cluster is returned (along with a warning).

Usage

mlink.test(
coords,
cases,
pop,
w,
ex = sum(cases)/sum(pop) * pop,
nsim = 499,
alpha = 0.1,
ubpop = 0.5,
ubd = 1,
longlat = FALSE,
cl = NULL
)

Arguments

coords An n × 2 matrix of centroid coordinates for the regions.
cases The number of cases observed in each region.
pop The population size associated with each region.
w A binary spatial adjacency matrix for the regions.
ex The expected number of cases for each region. The default is calculated under the constant risk hypothesis.
**nsim**  The number of simulations from which to compute the p-value.

**alpha**  The significance level to determine whether a cluster is significant. Default is 0.10.

**ubpop**  The upperbound of the proportion of the total population to consider for a cluster.

**ubd**  A proportion in (0, 1]. The distance of potential clusters must be no more than ubd * m, where m is the maximum intercentroid distance between all coordinates.

**longlat**  The default is FALSE, which specifies that Euclidean distance should be used. If longlat is TRUE, then the great circle distance is used to calculate the intercentroid distance.

**cl**  A cluster object created by `makeCluster`, or an integer to indicate number of child-processes (integer values are ignored on Windows) for parallel evaluations (see Details on performance).

**Details**

The maximum intercentroid distance can be found by executing the command: `sp::spDists(as.matrix(coords),longlat = longlat)`, based on the specified values of `coords` and `longlat`.

**Value**

Returns a `smerc_cluster` object.

**Author(s)**

Joshua French

**References**


**See Also**

`print.smerc_cluster`, `summary.smerc_cluster`, `plot.smerc_cluster`, `scan.stat`, `scan.test`

**Examples**

data(nydf)
data(nyw)
coords = with(nydf, cbind(longitude, latitude))
out = mlink.test(coords = coords, cases = floor(nydf$cases),
pop = nydf$pop, w = nyw,
alpha = 0.12, longlat = TRUE,
nsim = 2, ubpop = 0.05, ubd = 0.1)
data(nypoly)
library(sp)
plot(nypoly, col = color.clusters(out))
Description

mlink.zones determines the zones for the Maximum Linkage scan test (mlink.test). The function returns the zones, as well as the associated test statistic, cases in each zone, the expected number of cases in each zone, and the population in each zone.

Usage

mlink.zones(
  coords,
  cases,
  pop,
  w,
  ex = sum(cases)/sum(pop) * pop,
  ubpop = 0.5,
  ubd = 1,
  longlat = FALSE,
  cl = NULL,
  progress = TRUE
)

Arguments

  coords      An n x 2 matrix of centroid coordinates for the regions.
  cases       The number of cases observed in each region.
  pop         The population size associated with each region.
  w           A binary spatial adjacency matrix for the regions.
  ex          The expected number of cases for each region. The default is calculated under
              the constant risk hypothesis.
  ubpop       The upperbound of the proportion of the total population to consider for a clus-
              ter.
  ubd         A proportion in (0, 1]. The distance of potential clusters must be no more than
              ubd * m, where m is the maximum intercentroid distance between all coordinates.
  longlat     The default is FALSE, which specifies that Euclidean distance should be used.
              If longlat is TRUE, then the great circle distance is used to calculate the inter-
              centroid distance.
  cl          A cluster object created by makeCluster, or an integer to indicate number of
              child-processes (integer values are ignored on Windows) for parallel evaluations
              (see Details on performance).
  progress    A logical value indicating whether a progress bar should be displayed. The
              default is TRUE.
Details

Every zone considered must have a total population less than \( \text{ubpop} \times \sum(\text{pop}) \). Additionally, the maximum intercentroid distance for the regions within a zone must be no more than \( \text{ubd} \times \) the maximum intercentroid distance across all regions.

Value

Returns a list with elements:

- **zones**: A list contained the location ids of each potential cluster.
- **loglikrat**: The loglikelihood ratio for each zone (i.e., the log of the test statistic).
- **cases**: The observed number of cases in each zone.
- **expected**: The expected number of cases each zone.
- **pop**: The total population in each zone.

Author(s)

Joshua French

References


Examples

```r
data(nydf)
data(nyw)
coords = as.matrix(nydf[,c("longitude", "latitude")])
# find zone with max statistic starting from each individual region
all_zones = mlink.zones(coords, cases = floor(nydf$cases),
                        nydf$pop, w = nyw, ubpop = 0.25,
                        ubd = .25, longlat = TRUE)
```

---

**mst.all**

*Minimum spanning tree for all regions*

Description

*mst.all* finds the set of connected regions that maximize the spatial scan statistic (the likelihood ratio test statistic) from each starting region, subject to relevant constraints. The function can be used to construct candidate zones for the dynamic minimum spanning tree (dmst), early stopping dynamic minimum spanning tree (edmst), double connected spatial scan test (dc), and maximum linkage (mlink) spatial scan test.
Usage

```r
mst.all(
  neighbors, cases, pop, w, ex, ty, max_pop, type = "maxonly", nlinks = "one", early = FALSE, cl = NULL, progress = FALSE
)
```

Arguments

- **neighbors**: A list containing the vector of neighbors for each region (in ascending order of distance from the region). The starting region itself is included among the neighbors.
- **cases**: The number of cases observed in each region.
- **pop**: The population size associated with each region.
- **w**: A binary spatial adjacency matrix for the regions.
- **ex**: The expected number of cases for each region. The default is calculated under the constant risk hypothesis.
- **ty**: The total number of cases in the study area.
- **max_pop**: The population upperbound (in total population) for a candidate zone.
- **type**: One of "maxonly", "pruned", or "all". See Details.
- **nlinks**: A character vector. The options are "one", "two", or "max". See Details.
- **early**: A logical value indicating whether the "early" stopping criterion should be used. If TRUE, each sequence is stopped when the next potential zone doesn’t produce a test statistic larger than the current zone. The default is FALSE.
- **cl**: A cluster object created by `makeCluster`, or an integer to indicate number of child-processes (integer values are ignored on Windows) for parallel evaluations (see Details on performance).
- **progress**: A logical value indicating whether a progress bar should be displayed. The default is TRUE.

Details

This function is not intended to be used by users directly. Consequently, it prioritizes efficiency over user friendliness.

type is a character vector indicating what should be returned by the function. If type = "maxonly", then the maximum test statistic from each starting region is returned. If type = "pruned", the
function returns a list that includes the location ids, test statistic, total cases, expected cases, and total population for the zone with the maximum test statistic for each starting region. If `type = "all"`, the function returns a list of lists that includes the location ids, test statistic, total cases, expected cases, and total population for the sequence of candidate zones associated with each starting region. If `nlinks = "one"`, then a region only needs to be connected to one other region in the current zone to be considered for inclusion in the next zone. If `nlinks = "two"`, then the region must be connected to at least two other regions in the current zone. If `nlinks = "max"`, then only regions with the maximum number of connections to the current zone are considered for inclusion in the next zone.

**Value**

Returns a list of relevant information. See Details.

**Author(s)**

Joshua French

**References**


**Examples**

```r
# load data
data(nydf)
data(nyw)

# create relevant data
coords = nydf[,c("longitude", "latitude")]
cases = floor(nydf$cases)
pop = nydf$population
w = nyw
ex = sum(cases)/sum(pop)*pop
ubpop = 0.5
ubd = 0.5
ty = sum(cases)  # total number of cases
# intercentroid distances
d = sp::spDists(as.matrix(coords), longlat = TRUE)
# upperbound for population in zone
max_pop = ubpop * sum(pop)
# upperbound for distance between centroids in zone
max_dist = ubd * max(d)
# create list of neighbors for each region
# (inclusive of region itself)
all_neighbors = nndist(d, ubd)
# find the mst max zone
```
mst.seq

Minimum spanning tree sequence

Description

mst.seq finds the sequence of connected regions that maximize the spatial scan statistic (the likelihood ratio test statistic) from a starting region. The set of connected regions at each step is a candidate zone. The zone continues to grow until no region should be added to the zone due to relevant constraints (size, connectivity, or other stopping criteria). This function is not intended to be used by users directly, but it can be quite educational for seeing the spread of the cluster. Consequently, it prioritizes efficiency over user friendliness.

Usage

mst.seq(
  start,
  neighbors,
  cases,
  pop,
  w,
  ex,
  ty,
  max_pop,
  type = "maxonly",
  nlinks = "one",
  early = FALSE
)

Arguments

start The initial region to start the candidate zone.
neighbors A vector containing the neighbors for the starting region (in ascending order of distance from the region). The staring region itself is included among the neighbors.
cases The number of cases observed in each region.
pop The population size associated with each region.
w  A binary spatial adjacency matrix for the regions.
ex  The expected number of cases for each region. The default is calculated under the constant risk hypothesis.
ty  The total number of cases in the study area.
max_pop  The population upperbound (in total population) for a candidate zone.
type  One of "maxonly", "pruned", or "all". The default is "maxonly". See Details.
nlinks  A character vector. The options are "one", "two", or "max". See Details.
early  A logical value indicating whether the "early" stopping criterion should be used. If TRUE, the sequence is stopped when the next potential zone doesn’t produce a test statistic larger than the current zone. The default is FALSE.

Details
The function can be used to construct candidate zones for the dynamic minimum spanning tree (dmsst), early stopping dynamic minimum spanning tree (edmsst), double connection spatial scan test (dc), and maximum linkage spatial scan test (mlst).

type is a character vector indicating what should be returned by the function. If type = "maxonly", then only the maximum of the log likelihood ratio test statistic across all candidate zones is returned. If type = "pruned", the function returns a list that includes the location ids, test statistic, total cases, expected cases, and total population for the zone with the maximum test statistic. If type = "all", the same information the same information is returned for the entire sequence of zones.

If nlinks = "one", then a region only needs to be connected to one other region in the current zone to be considered for inclusion in the next zone. If nlinks = "two", then the region must be connected to at least two other regions in the current zone. If nlinks = "max", then only regions with the maximum number of connections to the current zone are considered for inclusion in the next zone.

Value
Returns a list of relevant information. See Details.

Author(s)
Joshua French

Examples
# load data
data(nydf)
data(nyw)

# create relevant data
coords = nydf[,c("longitude", "latitude")]
cases = floor(nydf$cases)
pop = nydf$population
w = nyw
ex = sum(cases)/sum(pop)*pop
ubpop = 0.5
ubd = 0.5

# total number of cases

# upperbound for population in zone

# upperbound for distance between centroids in zone

# create list of neighbors for each region (inclusive of region itself)

# find the dimst max zone

mst.seq(start = 1, all_neighbors[[1]], cases, pop, w, ex, 
    ty, max_pop)

mst.seq(start = 1, all_neighbors[[1]], cases, pop, w, ex, 
    ty, max_pop, "pruned")

bigout = mst.seq(start = 1, all_neighbors[[1]], cases, pop, 
    w, ex, ty, max_pop, "all")

head(bigout)

__nn.cumsum__

_Cumulative sum over nearest neighbors_

**Description**

nn.cumsum computes the cumulative sum of y for the sequences of indices in each element of the list contained in nn.

**Usage**

nn.cumsum(nn, y, simplify = TRUE)

**Arguments**

nn A list of nearest neighbors in the format produced by nnpop.

y A numeric vector of values to be summed over.

simplify A logical value indicating whether the results should be simplified to a numeric vector. The default is TRUE.

**Value**

A vector or list, depending on the value of simplify.

**Examples**

# show nn.cumsum example for a circular scan setting
data(nydf)

cords = with(nydf, cbind(longitude, latitude))
cases = floor(nydf$cases)
d = sp::spDists(cords, longlat = TRUE)
# compute circular nearest neighbors
nn = nnpop(d, pop = nydf$pop, ubpop = 0.1)
# compute cumulative sums over all nn
cnn = nn.cumsum(nn, cases)
# compute cumulative sums over just the first set of nn


cnn1 = cumsum(cases[nn[[1]]])
# check equality
all.equal(cnn1, cnn[seq_along(cnn1)])


nn2zones


Convert nearest neighbors list to zones


Description

nn2zones converts a list of nearest neighbors to a list of zones. The list of nearest neighbors will come from functions such as \texttt{nnpop} or \texttt{knn}.

Usage

nn2zones(nn)

Arguments

\begin{itemize}
  \item \texttt{nn} A list of nearest neighbors
\end{itemize}

Value

A list of zones

Examples

data(nydf)
coords = with(nydf, cbind(x, y))
nn = knn(coords, k = 2)
nn2zones(nn)


nndist


Determine nearest neighbors based on maximum distance


Description

nndist determines the nearest neighbors for a set of observations within a certain radius.

Usage

nndist(d, ubd)
Arguments

d  An $n \times n$ square distance matrix containing the intercentroid distance between the $n$ region centroids.

ubd  A proportion in (0, 1]. The distance of potential clusters must be no more than $\text{ubd} \times m$, where $m$ is the maximum intercentroid distance between all coordinates.

Details

This function determines the nearest neighbors of each centroid based on the intercentroid distance. The number of nearest neighbors is limited by the furthest distance between the starting centroid and the farthest neighbor.

Value

Returns the indices of the nearest neighbors as a list.

Author(s)

Joshua French

Examples

data(nydf)
coords = as.matrix(nydf[,c("longitude", "latitude")])
d = as.matrix(dist(coords))
nn = nndist(d, ubd = 0.01)

Description

nndup determines the indices of duplicated elements for a nearest neighbors list created by a function such as nnpop or knn. The indices are related to the list returned by nn2zones.

Usage

nndup(nn, N = max(unlist(nn)))

Arguments

nn  A list of nearest neighbors.

N  The largest value in nn.

Value

A logical vector of indicating duplicate indices.
**Examples**

```r
nn = list(1:3, c(2:1, 4))
nndup(nn, 4)
```

**nnpop**  
*Determine nearest neighbors with population constraint*

**Description**

`scan.nn` determines the nearest neighbors for a set of observations based on the distance matrix according to a population-based upperbound.

**Usage**

```r
nnpop(d, pop, ubpop)
scan.nn(d, pop, ubpop)
```

**Arguments**

- `d` An \( n \times n \) square distance matrix containing the intercentroid distance between the \( n \) region centroids.
- `pop` The population size associated with each region.
- `ubpop` The upperbound of the proportion of the total population to consider for a cluster.

**Details**

This function determines the nearest neighbors of each centroid based on the intercentroid distance. The number of nearest neighbors is limited by the sum of the population values among the nearest neighbors. The set of nearest neighbors can contain no more than \( ubpop \times \text{sum(pop)} \) members of the population. The nearest neighbors are ordered from nearest to farthest.

**Value**

Returns the indices of the nearest neighbors as a list. For each element of the list, the indices are ordered from nearest to farthest from each centroid.

**Author(s)**

Joshua French

**Examples**

```r
data(nydf)
coords = as.matrix(nydf[,c("longitude", "latitude")])
d = as.matrix(dist(coords))
nn = scan.nn(d, pop = nydf$pop, ubpop = 0.1)
```
**Description**

Determine non-overlapping zones from a list of candidate zones.

**Usage**

```r
noz(x)
```

**Arguments**

- `x` A list containing the candidate zones.

**Details**

The function takes a list of candidate zones. Each element of the list contains a candidate zones. The candidate zones are defined by the location indices of the regions comprising the zones. Starting with the first candidate zone, the function excludes every candidate zone that intersects the first (any other candidate zone that shares indices with the first zone). Moving onto the next non-overlapping candidate zone, the process is repeated. The function returns the indices (in the list of zones) of the zones that do not overlap.

**Value**

A vector with the list indices of the non-overlapping zones.

**Author(s)**

Joshua French

**Examples**

```r
x = list(1:2, 1:3, 4:5, 4:6, 7:8)
noz(x)
```
Leukemia data for 281 regions in New York.

Description

This data set contains 281 observations related to leukemia cases in an 8 county area of the state of New York. The data were made available in Waller and Gotway (2005) and details are provided there. These data are related to a similar data set in Waller et al. (1994). The longitude and latitude coordinates are taken from the NYleukemia data set in the SpatialEpi package for plotting purposes.

Usage

data(nydf)

Format

A data frame with 281 rows and 4 columns:

- **longitude** The longitude of the region centroid. These are NOT the original values provided by Waller and Gotway (2005), but are the right ones for plotting correctly.

- **latitude** The latitude of the region centroid. These are NOT the original values provided by Waller and Gotway (2005), but are the right ones for plotting correctly.

- **population** The population (1980 census) of the region.

- **cases** The number of leukemia cases between 1978-1982.

- **x** The original 'longitude' coordinate provided by Waller and Gotway (2005).

- **y** The original 'latitude' coordinate provided by Waller and Gotway (2005).

Source


References

### nypoly

**SpatialPolygonsDataFrame for New York leukemia data.**

**Description**

A SpatialPolygonsDataFrame for the New York leukemia data in `nydf`. Note that the coordinates in the polygon have been projected to a different coordinate system (UTM, zone 18), but the order of the regions/polygons is the same as in `nydf`.

**Usage**

```r
data(nypoly)
```

**Format**

A SpatialPolygonDataFrame

**Source**


### nyw

**Adjacency matrix for New York leukemia data.**

**Description**

This data set contains a 281 x 281 adjacency matrix for the New York leukemia data in `nydf`.

**Usage**

```r
data(nyw)
```

**Format**

A matrix of dimension 281 x 281.

**Source**


**References**

optimal_ubpop  

*Optimal Population Upper Bound Statistics*

**Description**

`optimal_ubpop` computes statistics for choosing an optimal population upper bound. `ubpop_seq` is a sequence of values to consider as the optimal choice of upper bound. The smallest value must be at least \( \min(\text{pop})/\sum(\text{pop}) \) and should generally be less than or equal to 0.5.

**Usage**

```r
optimal_ubpop(
  coords,  
cases,  
pop,  
ex = sum(cases)/sum(pop) * pop,  
nsim = 499,  
alpha = 0.05,  
ubpop_seq = seq(0.01, 0.5, len = 50),  
longlat = FALSE,  
cl = NULL,  
type = "poisson",  
min.cases = 0,  
simdist = "multinomial"
)
```

**Arguments**

- `coords`  
  An \( n \times 2 \) matrix of centroid coordinates for the regions.

- `cases`  
  The number of cases observed in each region.

- `pop`  
  The population size associated with each region.

- `ex`  
  The expected number of cases for each region. The default is calculated under the constant risk hypothesis.

- `nsim`  
  The number of simulations from which to compute the p-value.

- `alpha`  
  The significance level to determine whether a cluster is significant. Default is 0.10.

- `ubpop_seq`  
  A strictly increasing numeric vector with values between \( \min(\text{pop})/\sum(\text{pop}) \) and 1. The default is `seq(0.01, 0.5, len = 50)`.

- `longlat`  
  The default is `FALSE`, which specifies that Euclidean distance should be used. If `longlat` is `TRUE`, then the great circle distance is used to calculate the inter-centroid distance.

- `cl`  
  A cluster object created by `makeCluster`, or an integer to indicate number of child-processes (integer values are ignored on Windows) for parallel evaluations (see Details on performance).
**optimal_ubpop**

- **type**: The type of scan statistic to compute. The default is "poisson". The other choice is "binomial".
- **min.cases**: The minimum number of cases required for a cluster. The default is 2.
- **simdist**: Character string indicating the simulation distribution. The default is "multinomial", which conditions on the total number of cases observed. The other options are "poisson" and "binomial"

**Value**

Returns a smerc_optimal_ubpop object. This includes:

- **ubpop_seq**: The sequence of population bounds considered
- **elbow_method**: An object with statistics related to the elbow method
- **gini_method**: An object with statistics related to the Gini method
- **elbow_ubpop**: The population upperbound suggested by the elbow method
- **gini_ubpop**: The population upperbound suggested by the Gini method

**Author(s)**

Joshua French

**References**


**See Also**

- scan.test

**Examples**

```r
data(nydf)
coords = with(nydf, cbind(longitude, latitude))
ubpop_stats = optimal_ubpop(coords = coords, cases = nydf$cases,
pop = nydf$pop, nsim = 49,
ubpop = seq(0.05, 0.5, by = 0.05))
ubpop_stats
# Not run: plot(ubpop_stats)
```
plot.smerc_cluster  

Plot object of class smerc_cluster.

Description

Plot clusters (the centroids of the regions in each cluster) in different colors. The most likely cluster is plotted with solid red circles by default. Points not in a cluster are black open circles. The other cluster points are plotted with different symbols and colors.

Usage

### S3 method for class 'smerc_cluster'

```r
plot(
  x,
  ..., 
  nclusters = length(x$clusters),
  ccol = NULL,
  cpch = NULL,
  add = FALSE,
  usemap = FALSE,
  mapargs = list()
)
```

Arguments

- `x`  
  An object of class scan to be plotted.
- `...`  
  Additional graphical parameters passed to the `plot` function.
- `nclusters`  
  Number of clusters to plot.
- `ccol`  
  Fill color of the plotted points. Default is `grDevices::hcl.colors(nclusters, palette = "viridis")`.
- `cpch`  
  Plotting character to use for points in each cluster. Default is NULL, indicating `pch = 20` for the most likely cluster and then `pch = 2, 3, ...` up to the remaining number of clusters.
- `add`  
  A logical indicating whether results should be drawn on existing map.
- `usemap`  
  Logical indicating whether the maps::map function should be used to create a plot background for the coordinates. Default is FALSE. Use TRUE if you have longitude/latitude coordinates.
- `mapargs`  
  A list of arguments for the map function.

Examples

```r
data(nydf)
coords = with(nydf, cbind(longitude, latitude))
out = scan.test(coords = coords, cases = floor(nydf$cases),
    pop = nydf$pop, nsim = 0,
```
longlat = TRUE, alpha = 1)
plot(out, nclusters = 3)
## plot output for new york state
# specify desired argument values
mapargs = list(database = "county", region = "new york",
               xlim = range(out$coords[,1]),
               ylim = range(out$coords[,2]))
# needed for "county" database (unless you execute library(maps))
data(countyMapEnv, package = "maps")
plot(out, nclusters = 3, usemap = TRUE, mapargs = mapargs)

---

**Plot object of class smerc_optimal_ubpop.**

**Description**

Plot results of `optimal_ubpop`. This is only meant for a visual summary of the results. Users will need to access the elements of the `smerc_optimal_ubpop` object `x` if they want to create a custom plot.

**Usage**

```r
## S3 method for class 'smerc_optimal_ubpop'
plot(x, ..., method = "all")
```

**Arguments**

- **x**  
  An object of class `smerc_optimal_ubpop`.
- **...**  
  Not used
- **method**  
  The method to plot. The default is "all". The other valid options are "elbow" and "gini".

**See Also**

`optimal_ubpop`

**Examples**

```r
data(nydf)
coords = with(nydf, cbind(longitude, latitude))
ubpop_stats = optimal_ubpop(coords = coords, cases = nydf$cases,
                            pop = nydf$pop, nsim = 49,
                            ubpop = seq(0.05, 0.5, by = 0.05))
## Not run: plot(ubpop_stats)
plot(ubpop_stats, method = "elbow")
plot(ubpop_stats$ubpop_seq, ubpop_stats$elbow_method$stats)
plot(ubpop_stats, method = "gini")
plot(ubpop_stats$ubpop_seq, ubpop_stats$gini_method$stats)
```
Description

Plots results of `tango.test`. If Monte Carlo simulation was not used to produce `x`, then a a density plot of the (approximate) null distribution of `tstat.chisq` is produced, along with a vertical line for the observed `tstat`. If a Monte Carlo test was used to produce `x`, then a scatterplot of the `gof.sim` versus `sa.sim` is compared to the observed values `gof` and `sa`, respectively.

Usage

```r
## S3 method for class 'tango'
plot(x, ..., obs.list = list(pch = 20), sim.list = list(pch = 2))
```

Arguments

- `x` An object of class `tango` to be plotted.
- `...` Additional graphical parameters passed to `plot` function.
- `obs.list` A list containing arguments for the `points` function, which is used to plot the `gof` and `sa` components, when appropriate.
- `sim.list` A list containing arguments for the `points` function, which is used to plot the `gof.sim` and `sa.sim` components, when appropriate.

See Also

`tango.test`

Examples

```r
data(nydf)
coords = as.matrix(nydf[,c("x", "y")])
w = dweights(coords, kappa = 1)
x1 = tango.test(nydf$cases, nydf$pop, w)
plot(x1)
x2 = tango.test(nydf$cases, nydf$pop, w, nsim = 49)
plot(x2)
```
print.smerc_cluster  

Print object of class smerc_cluster.

Description

Print smerc_cluster object

Usage

## S3 method for class 'smerc_cluster'
print(x, ..., extra = FALSE)

Arguments

x  
An object of class smerc_cluster.

...  
Arguments passed on to base::print

extra  
A logical value. Default is FALSE. TRUE indicates that extra information should be printed.

Examples

data(nydf)
coords = with(nydf, cbind(x, y))
out = scan.test(coords = coords, cases = floor(nydf$cases),
               pop = nydf$pop, nsim = 49,
               longlat = TRUE, alpha = 0.12)
out

print.smerc_optimal_ubpop  

Print object of class smerc_optimal_ubpop.

Description

Print smerc_optimal_ubpop object

Usage

## S3 method for class 'smerc_optimal_ubpop'
print(x, ...)

Arguments

x An object of class smerc_optimal_ubpop.
...
Arguments passed on to base::print

Examples

data(nydf)
coords = with(nydf, cbind(longitude, latitude))
ubpop_stats = optimal_ubpop(coords = coords, cases = nydf$cases,
                           pop = nydf$pop, nsim = 49,
                           ubpop = seq(0.05, 0.5, by = 0.05))
ubpop_stats

print.tango

Print object of class tango.

Description

Print a tango object. If the crayon package is installed, then the results are printed in color.

Usage

## S3 method for class 'tango'
print(x, ..., digits = 2)

Arguments

x An object of class tango.
...
Arguments passed on to base::print

digits Number of significant digits to print.

Examples

data(nydf)
coords = as.matrix(nydf[,c("x", "y")])
w = dweights(coords, kappa = 1)
results = tango.test(nydf$cases, nydf$pop, w, nsim = 49)
results
rflex.midp

Compute middle p-value

Description

Computes $P(Y > \text{cases}) + P(Y = \text{cases})/2$ when $Y \sim \text{Poisson}(\text{ex})$ or $Y \sim \text{Binomial}(n = \text{pop}, p = \text{ex}/\text{pop})$. This is middle p-value computed by Tango and Takahashi (2012).

Usage

rflex.midp(cases, ex, type = "poisson", pop = NULL)

Arguments

- **cases**: The number of cases observed in each region.
- **ex**: The expected number of cases for each region. The default is calculated under the constant risk hypothesis.
- **type**: The type of scan statistic to compute. The default is "poisson". The other choice is "binomial".
- **pop**: The population size associated with each region.

Value

A vector of middle p-values

Author(s)

Joshua French

References


Examples

data(nydf)
cases = floor(nydf$cases)
pop = nydf$pop
ex = pop * sum(cases)/sum(pop)
# zones for poisson model
pp = rflex.midp(cases, ex)
# zones for binomial model
bp = rflex.midp(cases, ex, type = "binomial", pop = pop)
rflex.sim

Perform rflex.test on simulated data

Description

rflex.sim efficiently performs rflex.test on a simulated data set. The function is meant to be used internally by the rflex.test function, but is informative for better understanding the implementation of the test.

Usage

rflex.sim(
  nsim = 1,
  nn,
  w,
  ex,
  alpha1 = 0.2,
  type = "poisson",
  pop = NULL,
  cl = NULL
)

Arguments

nsim A positive integer indicating the number of simulations to perform.
nn A matrix of the k nearest neighbors for the regions described by w.
w A binary spatial adjacency matrix for the regions.
ex The expected number of cases for each region. The default is calculated under the constant risk hypothesis.
alpha1 The middle p-value threshold.
type The type of scan statistic to compute. The default is "poisson". The other choice is "binomial".
pop The population size associated with each region.
cl A cluster object created by makeCluster, or an integer to indicate number of child-processes (integer values are ignored on Windows) for parallel evaluations (see Details on performance).

Value

A vector with the maximum test statistic for each simulated data set.
**rflex.test**

**Restricted Flexibly-shaped Spatial Scan Test**

**Description**

*rflex.test* performs the restricted flexibly shaped spatial scan test of Tango and Takahashi (2012).

**Usage**

```r
rflex.test(
  coords, cases, pop, w, k = 50, ex = sum(cases)/sum(pop) * pop,
  type = "poisson", nsim = 499, alpha = 0.1,
  longlat = FALSE, alpha1 = 0.2, cl = NULL
)
```

**Arguments**

- **coords**: An $n \times 2$ matrix of centroid coordinates for the regions.
- **cases**: The number of cases observed in each region.
- **pop**: The population size associated with each region.
- **w**: A binary spatial adjacency matrix for the regions.
- **k**: An integer indicating the maximum number of regions to include in a potential cluster. Default is 10.
- **ex**: The expected number of cases for each region. The default is calculated under the constant risk hypothesis.

**Examples**

```r
data(nydf)
data(nyw)
# determine knn
coords = with(nydf, cbind(longitude, latitude))
nn = knn(coords, longlat = TRUE, k = 50)
# determine expected number of cases in each region
cases = floor(nydf$cases)
pop = nydf$pop
ex = pop * sum(cases)/sum(pop)
tsim = rflex.sim(nsim = 5, nn = nn, w = nyw, ex = ex)
```
type

The type of scan statistic to compute. The default is "poisson". The other choice is "binomial".

nsim

The number of simulations from which to compute the p-value.

alpha

The significance level to determine whether a cluster is significant. Default is 0.10.

longlat

The default is FALSE, which specifies that Euclidean distance should be used. If longlat is TRUE, then the great circle distance is used to calculate the inter-centroid distance.

alpha1

The middle p-value threshold.

cl

A cluster object created by makeCluster, or an integer to indicate number of child-processes (integer values are ignored on Windows) for parallel evaluations (see Details on performance).

Details

The test is performed using the spatial scan test based on the Poisson test statistic and a fixed number of cases. The first cluster is the most likely to be a cluster. If no significant clusters are found, then the most likely cluster is returned (along with a warning).

Value

Returns a list of length two of class scan. The first element (clusters) is a list containing the significant, non-ovlappering clusters, and has the the following components:

coords

The centroid of the significant clusters.

r

The radius of the window of the clusters.

pop

The total population in the cluster window.

cases

The observed number of cases in the cluster window.

expected

The expected number of cases in the cluster window.

smr

Standardized mortality ratio (observed/expected) in the cluster window.

rr

Relative risk in the cluster window.

loglikrat

The loglikelihood ratio for the cluster window (i.e., the log of the test statistic).

cmp

The pvalue of the test statistic associated with the cluster window.

The second element of the list is the centroid coordinates. This is needed for plotting purposes.

Author(s)

Joshua French

References

rflex.zones

Determine zones for flexibly shaped spatial scan test

Description

dflex.zones determines the unique zones to consider for the flexibly shaped spatial scan test of Tango and Takahashi (2012). The algorithm uses a breadth-first search to find all subgraphs connected to each vertex (region) in the data set of size \( k \) or less with the constraint that the middle p-value of each region must be less than \( \alpha_1 \).

Usage

dflex.zones(
  nn,
  w,
  cases,
  ex,
  alpha1 = 0.2,
  type = "poisson",
  pop = NULL,
  cl = NULL,
  loop = FALSE,
  verbose = FALSE,
  pfreq = 1
)

Arguments

- **nn**
  An n by k matrix providing the k nearest neighbors of each region, presumably produced by the \texttt{knn} function.

- **w**
  A binary spatial adjacency matrix for the regions.

Examples

data(nydf)
data(nyw)
coords = with(nydf, cbind(longitude, latitude))
out = rflex.test(coords = coords, cases = floor(nydf$cases),
  w = nyw, k = 10,
  pop = nydf$pop, nsim = 49,
  alpha = 0.05, longlat = TRUE)

data(nypoly)
library(sp)
plot(nypoly, col = color.clusters(out))

See Also

- \texttt{print.smerc_cluster}
- \texttt{summary.smerc_cluster}
- \texttt{plot.smerc_cluster}
- \texttt{scan.stat}
- \texttt{scan.test}
cases: The number of cases observed in each region.
ex: The expected number of cases for each region. The default is calculated under the constant risk hypothesis.
alpha1: The middle p-value threshold.
type: The type of scan statistic to compute. The default is "poisson". The other choice is "binomial".
pop: The population size associated with each region. The default is NULL since this argument is only needed for type = "binomial".
c1: A cluster object created by `makeCluster`, or an integer to indicate number of child-processes (integer values are ignored on Windows) for parallel evaluations (see Details on performance).
loop: A logical value indicating whether a loop should be used to implement the function instead of `pbapply`. The default is FALSE. If TRUE, then memory-saving steps are also taken.
verbose: A logical value indicating whether progress messages should be provided. The default is FALSE. If both loop and verbose are TRUE, informative messages are displayed that can be useful for diagnosing where the sequences of connected subgraphs are slowing down or having problems.
pfreq: The frequency that messages are reported from the loop (if verbose = TRUE). The default is pfreq = 1, meaning a message is returned for each index of the loop.

Value

Returns a list of zones to consider for clustering. Each element of the list contains a vector with the location ids of the regions in that zone.

Author(s)

Joshua French

References


See Also

rflex.midp

Examples

data(nydf)
data(nyw)
coords = cbind(nydf$x, nydf$y)
nn = knn(coords, k = 5)
cases = floor(nydf$cases)
pop = nydf$pop
ex = pop * sum(cases)/sum(pop)
# zones for poisson model
pzones = rflex.zones(nn, w = nyw, cases = cases, ex = ex)
## Not run:
   pzones = rflex.zones(nn, w = nyw, cases = cases,
               ex = ex, verbose = TRUE)
# zones for binomial model
bzones = rflex.zones(nn, w = nyw, cases = cases, ex = ex,
               type = "binomial", pop = pop)
## End(Not run)

rflex_zones  Determine zones for flexibly shaped spatial scan test

Description

rflex_zones determines the unique zones to consider for the flexibly shaped spatial scan test of
Tango and Takahashi (2012). The algorithm uses a breadth-first search to find all subgraphs con-
ected to each vertex (region) in the data set of size $k$ or less with the constraint that the middle
p-value of each region must be less than $\alpha_1$.

Usage

rflex_zones(  
nn,
   w,
   cases,
   ex,
   alpha1 = 0.2,
   type = "poisson",
   pop = NULL,
   cl = NULL,
   loop = FALSE,
   verbose = FALSE,
   pfreq = 1
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nn</td>
<td>An n by k matrix providing the k nearest neighbors of each region, presumably produced by the \texttt{knn} function.</td>
</tr>
<tr>
<td>w</td>
<td>A binary spatial adjacency matrix for the regions.</td>
</tr>
<tr>
<td>cases</td>
<td>The number of cases observed in each region.</td>
</tr>
<tr>
<td>ex</td>
<td>The expected number of cases for each region. The default is calculated under the constant risk hypothesis.</td>
</tr>
<tr>
<td>alpha1</td>
<td>The middle p-value threshold.</td>
</tr>
</tbody>
</table>
type

The type of scan statistic to compute. The default is "poisson". The other choice is "binomial".

pop

The population size associated with each region. The default is NULL since this argument is only needed for type = "binomial".

c1

A cluster object created by `makeCluster`, or an integer to indicate number of child-processes (integer values are ignored on Windows) for parallel evaluations (see Details on performance).

loop

A logical value indicating whether a loop should be used to implement the function instead of `pbapply`. The default is FALSE. If TRUE, then memory-saving steps are also taken.

verbose

A logical value indicating whether progress messages should be provided. The default is FALSE. If both loop and verbose are TRUE, informative messages are displayed that can be useful for diagnosing where the sequences of connected subgraphs are slowing down or having problems.

pfreq

The frequency that messages are reported from the loop (if verbose = TRUE). The default is pfreq = 1, meaning a message is returned for each index of the loop.

Value

Returns a list of zones to consider for clustering. Each element of the list contains a vector with the location ids of the regions in that zone.

Author(s)

Joshua French

References


See Also

rflex.midp

Examples

data(nydf)
data(nyw)
coords = cbind(nydf$x, nydf$y)
nn = knn(coords, k = 5)
cases = floor(nydf$cases)
pop = nydf$pop
ex = pop * sum(cases)/sum(pop)
# zones for poisson model
pzones = rflex_zones(nn, w = nyw, cases = cases, ex = ex)
## Not run:
pzones = rflex_zones(nn, w = nyw, cases = cases,
Perform scan.test on simulated data

Description

scan.sim efficiently performs scan.test on a simulated data set. The function is meant to be used internally by the scan.test function, but is informative for better understanding the implementation of the test.

Usage

scan.sim(
  nsim = 1,
  nn,
  ty,
  ex,
  type = "poisson",
  ein = NULL,
  eout = NULL,
  tpop = NULL,
  popin = NULL,
  popout = NULL,
  cl = NULL,
  simdist = "multinomial",
  pop = NULL
)

Arguments

nsim A positive integer indicating the number of simulations to perform.
nn A list of nearest neighbors produced by nnpop.
ty The total number of cases in the study area.
ex The expected number of cases for each region. The default is calculated under the constant risk hypothesis.
type The type of scan statistic to compute. The default is "poisson". The other choice is "binomial".
ein The expected number of cases in the zone. Conventionally, this is the estimated overall disease risk across the study area, multiplied by the total population size of the zone.
The expected number of cases outside the zone. This should be \( \text{ty} - \text{ein} \) and is computed automatically if not provided.

tpop
The total population in the study area.

popin
The total population in the zone.

popout
The population outside the zone. This should be \( \text{tpop} - \text{popin} \) and is computed automatically if not provided.

cl
A cluster object created by \texttt{makeCluster}, or an integer to indicate number of child-processes (integer values are ignored on Windows) for parallel evaluations (see Details on performance).

simdistr
Character string indicating the simulation distribution. The default is "multinomial", which conditions on the total number of cases observed. The other options are "poisson" and "binomial".

pop
The population size associated with each region.

Value
A vector with the maximum test statistic for each simulated data set.

Examples

```r
data(nydf)
coords = with(nydf, cbind(longitude, latitude))
d = sp::spDists(as.matrix(coords), longlat = TRUE)
nn = scan.nn(d, pop = nydf$pop, ubpop = 0.1)
cases = floor(nydf$cases)
ty = sum(cases)
ex = ty/sum(nydf$pop) * nydf$pop
yin = nn.cumsum(nn, cases)
ein = nn.cumsum(nn, ex)
tsim = scan.sim(nsim = 1, nn, ty, ex, ein = ein, eout = sum(ex) - ein)
```

---

scan.stat

*Spatial scan statistic*

Description

\texttt{scan.stat} calculates the spatial scan statistic for a zone (a set of spatial regions). The statistic is the log of the likelihood ratio test statistic of the chosen distribution. If \texttt{type = "poisson"} and \( a \) is more than zero, this statistic is penalized. See references.

Usage

```r
scan.stat(
  yin,
  ein = NULL,
  eout = NULL,
)```
ty,
type = "poisson",
popin = NULL,
tpop = NULL,
a = 0,
shape = 1,
yout = NULL,
popout = NULL
)

stat.poisson(yin, yout, ein, eout, a = 0, shape = 1)
stat.binom(yin, yout, ty, popin, popout, tpop)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>yin</td>
<td>The total number of cases in the zone.</td>
</tr>
<tr>
<td>ein</td>
<td>The expected number of cases in the zone. Conventionally, this is the estimated overall disease risk across the study area, multiplied by the total population size of the zone.</td>
</tr>
<tr>
<td>eout</td>
<td>The expected number of cases outside the zone. This should be ( ty - ein ) and is computed automatically if not provided.</td>
</tr>
<tr>
<td>ty</td>
<td>The total number of cases in the study area.</td>
</tr>
<tr>
<td>type</td>
<td>The type of scan statistic to implement. The default choice are &quot;poisson&quot;. The other choice is &quot;binomial&quot;.</td>
</tr>
<tr>
<td>popin</td>
<td>The total population in the zone.</td>
</tr>
<tr>
<td>tpop</td>
<td>The total population in the study area.</td>
</tr>
<tr>
<td>a</td>
<td>A tuning parameter for the adjusted log-likelihood ratio. See details.</td>
</tr>
<tr>
<td>shape</td>
<td>The shape of the ellipse, which is the ratio of the length of the longest and shortest axes of the ellipse. The default is 1, meaning it is a circle.</td>
</tr>
<tr>
<td>yout</td>
<td>The observed number of cases outside the zone. This should be ( ty - yin ) and is computed automatically if not provided.</td>
</tr>
<tr>
<td>popout</td>
<td>The population outside the zone. This should be ( tpop - popin ) and is computed automatically if not provided.</td>
</tr>
</tbody>
</table>

Value

A vector of scan statistics.

Author(s)

Joshua French
References


Examples

# New York leukemia data
# total cases
ty = 552
# total population
tpop = 1057673

# poisson example with yin = 106 and ein = 62.13
scan.stat(yin = 106, ty = ty, ein = 62.13)

# binomial example with yin = 41 and popin = 38999
scan.stat(yin = 41, ty = ty, popin = 38999, tpop = tpop, type = "binomial")
stat.binom(41, ty - 41, ty, 38999, tpop - 38999, tpop)

scan.test

Spatial Scan Test

Description

scan.test performs the original spatial scan test of Kulldorff (1997) based on a fixed number of cases. Candidate zones are circular and extend from the observed region centroids. The clusters returned are non-overlapping, ordered from most significant to least significant. The first cluster is the most likely to be a cluster. If no significant clusters are found, then the most likely cluster is returned (along with a warning).

Usage

scan.test(
  coords,
  cases,
  pop,
  ex = sum(cases)/sum(pop) * pop,
  nsim = 499,
  alpha = 0.1,
ubpop = 0.5,
longlat = FALSE,
c1 = NULL,
type = "poisson",
min.cases = 2,
simdist = "multinomial"
)

Arguments

- coords: An $n \times 2$ matrix of centroid coordinates for the regions.
- cases: The number of cases observed in each region.
- pop: The population size associated with each region.
- ex: The expected number of cases for each region. The default is calculated under the constant risk hypothesis.
- nsim: The number of simulations from which to compute the p-value.
- alpha: The significance level to determine whether a cluster is significant. Default is 0.10.
- ubpop: The upperbound of the proportion of the total population to consider for a cluster.
- longlat: The default is FALSE, which specifies that Euclidean distance should be used. If longlat is TRUE, then the great circle distance is used to calculate the inter-centroid distance.
- c1: A cluster object created by makeCluster, or an integer to indicate number of child-processes (integer values are ignored on Windows) for parallel evaluations (see Details on performance).
- type: The type of scan statistic to compute. The default is "poisson". The other choice is "binomial".
- min.cases: The minimum number of cases required for a cluster. The default is 2.
- simdist: Character string indicating the simulation distribution. The default is "multinomial", which conditions on the total number of cases observed. The other options are "poisson" and "binomial"

Value

Returns a smerc_cluster object.

Author(s)

Joshua French

References


See Also

print.smerc_cluster, summary.smerc_cluster, plot.smerc_cluster, scan.stat

Examples

data(nydf)
coords = with(nydf, cbind(longitude, latitude))
out = scan.test(coords = coords, cases = floor(nydf$cases),
    pop = nydf$pop, nsim = 0,
    alpha = 1, longlat = TRUE)
## plot output for new york state
# specify desired argument values
mapargs = list(database = "county", region = "new york",
    xlim = range(out$coords[,1]), ylim = range(out$coords[,2]))
# needed for "state" database (unless you execute library(maps))
data(countyMapEnv, package = "maps")
plot(out, usemap = TRUE, mapargs = mapargs)
# extract detected clusters
clusters(out)

# a second example to match the results of Waller and Gotway (2005)
# in chapter 7 of their book (pp. 220-221).
# Note that the 'longitude' and 'latitude' used by them has
# been switched. When giving their input to SatScan, the coords
# were given in the order 'longitude' and 'latitude'.
# However, the SatScan program takes coordinates in the order
# 'latitude' and 'longitude', so the results are slightly different
# from the example above.
coords = with(nydf, cbind(y, x))
out2 = scan.test(coords = coords, cases = floor(nydf$cases),
    pop = nydf$pop, nsim = 0,
    alpha = 1, longlat = TRUE)
# the cases observed for the clusters in Waller and Gotway: 117, 47, 44
# the second set of results match
sget(out2$clusters, name = "cases")[1:3]

scan.zones

Determine zones for the spatial scan test

Description

scan.zones determines the unique candidate zones to consider for the circular spatial scan test of Kulldorff (1997).

Usage

scan.zones(coords, pop, ubpop = 0.5, longlat = FALSE)
**Arguments**

- **coords** An $n \times 2$ matrix of centroid coordinates for the regions.
- **pop** The population size associated with each region.
- **ubpop** The upperbound of the proportion of the total population to consider for a cluster.
- **longlat** The default is FALSE, which specifies that Euclidean distance should be used. If longlat is TRUE, then the great circle distance is used to calculate the inter-centroid distance.

**Value**

Returns a list of zones to consider for clustering. Each element of the list contains a vector with the location ids of the regions in that zone.

**Author(s)**

Joshua French

**References**


**Examples**

```r
data(nydf)
coords = cbind(nydf$longitude, nydf$latitude)
zones = scan.zones(coords = coords, pop = nydf$pop,
                   ubpop = 0.1, longlat = TRUE)
```

**Description**

scan_stat calculates the spatial scan statistic for a zone (a set of spatial regions). The statistic is the log of the likelihood ratio test statistic of the chosen distribution. If type = "poisson" and a is more than zero, this statistic is penalized. See references.

**Usage**

```r
scan_stat(
  yin,
  ein = NULL,
  eout = NULL,
  ty,
  type = "poisson",
)```
popin = NULL, 
  tpop = NULL, 
  a = 0, 
  shape = 1, 
  yout = NULL, 
  popout = NULL 
)

stat_poisson(yin, yout, ein, eout, a = 0, shape = 1)

stat_binom(yin, yout, ty, popin, popout, tpop)

Arguments

yin
  The total number of cases in the zone.
ein
  The expected number of cases in the zone. Conventionally, this is the estimated overall disease risk across the study area, multiplied by the total population size of the zone.
eout
  The expected number of cases outside the zone. This should be ty - ein and is computed automatically if not provided.
ty
  The total number of cases in the study area.
type
  The type of scan statistic to implement. The default choice are "poisson". The other choice is "binomial".
popin
  The total population in the zone.
tpop
  The total population in the study area.
a
  A tuning parameter for the adjusted log-likelihood ratio. See details.
shape
  The shape of the ellipse, which is the ratio of the length of the longest and shortest axes of the ellipse. The default is 1, meaning it is a circle.
yout
  The observed number of cases outside the zone. This should be ty - yin and is computed automatically if not provided.
popout
  The population outside the zone. This should be tpop - popin and is computed automatically if not provided.

Value

A vector of scan statistics.

Author(s)

Joshua French

References


Examples

# New York leukemia data
# total cases
ty = 552
# total population
tpop = 1057673

# poisson example with yin = 106 and ein = 62.13
scan_stat(yin = 106, ty = ty, ein = 62.13)
stat_poisson(yin = 106, yout = 552 - 106,
            ein = 62.13, eout = 552 - 62.13)

# binomial example with yin = 41 and popin = 38999
scan_stat(yin = 41, ty = ty,
          popin = 38999, tpop = tpop, type = "binomial")
stat_binom(41, ty - 41, ty, 38999, tpop - 38999, tpop)

---

**sig_noc**

Return most significant, non-overlapping zones

**Description**

*sig_noc* return the significant, non-overlapping zones order from most significant to least significant.

**Usage**

```r
sig_noc(tobs, zones, pvalue, alpha, order_by = "tobs")
```

**Arguments**

- **tobs**: The vector of observed test statistics for each zone
- **zones**: A list of zones
- **pvalue**: The p-value associated with each test statistic
- **alpha**: The significance level of the test.
- **order_by**: Either 'tobs' or 'pvalue', indicating the argument by which to order zones.

**Value**

A list with the significant, ordered, non-overlapping tobs, zones, pvalue, and idx (a vector with the relevant indices of the original zones).
Examples

tobs = c(1, 3, 2)
zones = list(1:2, 1:3, 2:3)
pvalue = c(0.5, 0.01, 0.02)

sig_noc(tobs, zones, pvalue, alpha = 0.05)

Description

S*tatistical ME*thods for R*egional C*ounts

smerc_cluster

Prepare smerc_cluster

Description

smerc_cluster prepares a smerc_cluster.

Usage

smerc_cluster(
  tobs,
  zones,
  pvalue,
  coords,
  cases,
  pop,
  ex,
  longlat,
  method,
  rel_param,
  alpha,
  w = NULL,
  d = NULL,
  a = NULL,
  shape_all = NULL,
  angle_all = NULL
)
**Arguments**

- **tobs**: The vector of observed test statistics for each zone.
- **zones**: A list of zones.
- **pvalue**: The p-value associated with each test statistic.
- **coords**: An $n \times 2$ matrix of centroid coordinates for the regions.
- **cases**: The number of cases observed in each region.
- **pop**: The population size associated with each region.
- **ex**: The expected number of cases for each region. The default is calculated under the constant risk hypothesis.
- **longlat**: The default is `FALSE`, which specifies that Euclidean distance should be used. If `longlat` is `TRUE`, then the great circle distance is used to calculate the inter-centroid distance.
- **method**: A character string indicating the method used to construct the `smerc_cluster`.
- **rel_param**: A names list with the relevant parameters associated with `method`.
- **alpha**: The significance level of the test.
- **w**: A binary spatial adjacency matrix for the regions.
- **d**: A precomputed distance matrix based on `coords`.
- **a**: A single value $\geq 0$ indicating the penalty to use for `elliptic.test`.
- **shape_all**: A vector of shape parameters associated with `zones`.
- **angle_all**: A vector of angle parameter associated with `zones`.

**Value**

A `smerc_cluster` object. The object generally has the following components:

- **clusters**: A list containing information about the significant clusters. See further details below.
- **coords**: The matrix of centroid coordinates.
- **number_of_regions**: The number of regions considered.
- **total_population**: The total population in the regions.
- **total_cases**: The total number of cases in the regions.
- **cases_per_100k**: The rate of cases per 100,000 persons.
- **method**: The name of the method applied.
- **rel_param**: A list of relevant method parameters.
- **alpha**: The significance level.
- **longlat**: A logical value indicating which type of distance was used.

Each element of the `clusters` component has:

- **locids**: The ids of the regions in the cluster.
centroid  The cluster centroid.
r  The radius of the region (from the starting region to last region of the cluster).
max_dist  The maximum intercentroid distance between all the regions in the cluster.
population  The total population in the cluster.
cases  The number of cases in the cluster.
expected  The expected number of cases in the cluster.
smr  Standardized mortality ratio (cases/expected) in the cluster.
rr  Relative risk in the cluster window. This is \( \frac{\text{cases}}{\text{pop}} / \left( \frac{\text{total_cases} - \text{cases}}{\text{total_population} - \text{population}} \right) \).
loglikrat  The log of the likelihood ratio test statistic for the cluster. Only valid for the scan-type tests.
test_statistic  The test statistic for the cluster.
pvalue  The p-value of the test statistic associated with the cluster.
w  The adjacency information for the cluster.

For elliptic.test, clusters additionally has:

semiminor_axis  The semi-minor axis length for the ellipse.
semimajor_axis  The semi-major axis length for the ellipse.
angle  The rotation angle of the ellipse.
shape  The shape of the ellipse.

---

**summary.smerc_cluster Summary of smerc_cluster object**

**Description**

Summary of object of class smerc_cluster.

**Usage**

```r
## S3 method for class 'smerc_cluster'
summary(object, ..., idx = seq_along(object$clusters), digits = 1)
```

**Arguments**

- **object**  An object of class smerc_cluster.
- **...**  Arguments passed on to `base::summary`
- **idx**  A index vector indicating the elements of object$clusters to print information for. The default is all clusters.
- **digits**  Integer indicating the number of decimal places.
Examples

data(nydf)
coords = with(nydf, cbind(x, y))
out = scan.test(coords = coords, cases = floor(nydf$cases),
    pop = nydf$pop, nsim = 49,
    longlat = TRUE, alpha = 0.12)
summary(out)

tango.stat

Tango's statistic

Description

tango.stat computes Tango's index (Tango, 1995), including both the goodness-of-fit and spatial autocorrelation components. See Waller and Gotway (2005).

Usage

tango.stat(cases, pop, w)

Arguments

cases The number of cases observed in each region.
pop The population size associated with each region.
w An $n \times n$ weights matrix.

Value

Returns a list with the test statistic (tstat), the goodness-of-fit component (gof), and the spatial autocorrelation component (sa).

Author(s)

Joshua French

References


Examples

data(nydf)
coords = as.matrix(nydf[,c("longitude", "latitude")])
w = dweights(coords, kappa = 1, type = "tango", longlat = TRUE)
tango.stat(nydf$cases, nydf$pop, w)
tango.test  

*Description*

*tango.test* performs a test for clustering proposed by Tango (1995). The test uses Tango’s chi-square approximation for significance testing by default, but also uses Monte Carlo simulation when nsim > 0.

*Usage*

`tango.test(cases, pop, w, nsim = 0)`

*Arguments*

- **cases**: The number of cases observed in each region.
- **pop**: The population size associated with each region.
- **w**: An \( n \times n \) weights matrix.
- **nsim**: The number of simulations for which to perform a Monte Carlo test of significance. Counts are simulated according to a multinomial distribution with \( \text{sum(cases)} \) total cases and class probabilities \( \text{pop/sum(pop)} \). \( \text{sum(cases)} \).

*Details*

The `dweights` function can be used to construct a weights matrix \( w \) using the method of Tango (1995), Rogerson (1999), or a basic style.

*Value*

Returns a list of class `tango` with elements:

- **tstat**: Tango’s index
- **tstat.chisq**: The approximately chi-squared statistic proposed by Tango that is derived from `tstat`
- **dfc**: The degrees of freedom of `tstat.chisq`
- **pvalue.chisq**: The p-value associated with `tstat.chisq`
- **tstat.sim**: The vector of test statistics from the simulated data if \( \text{nsim} > 0 \)
- **pvalue.sim**: The p-value associated with the Monte Carlo test of significance when \( \text{nsim} > 0 \)

Additionally, the goodness-of-fit gof and spatial autocorrelation sa components of the Tango’s index are provided (and for the simulated data sets also, if appropriate).

*Author(s)*

Joshua French
tango.weights

References


See Also
dweights

Examples

data(nydf)
coords = as.matrix(nydf[,c("x", "y")])
w = dweights(coords, kappa = 1)
results = tango.test(nydf$cases, nydf$pop, w, nsim = 49)

Description

tango.weights constructs a distance-based weights matrix. The tango.weights function can be used to construct a weights matrix $w$ using the method of Tango (1995), Rogerson (1999), or a basic style.

Usage

tango.weights(coords, kappa = 1, longlat = FALSE, type = "basic", pop = NULL)
dweights(coords, kappa = 1, longlat = FALSE, type = "basic", pop = NULL)

Arguments

coords An $n \times 2$ matrix of centroid coordinates for the regions.
kappa A positive constant related to strength of spatial autocorrelation.
longlat The default is FALSE, which specifies that Euclidean distance should be used. If longlat is TRUE, then the great circle distance is used to calculate the inter-centroid distance.
type The type of weights matrix to construct. Current options are "basic", "tango", and "rogerson". Default is "basic". See Details.
pop The population size associated with each region.
Details

coords is used to construct an $n \times n$ distance matrix $d$.

If type = "basic", then $w_{ij} = \exp(-d_{ij}/\kappa)$.

If type = "rogerson", then $w_{ij} = \exp(-d_{ij}/\kappa)/\sqrt{(\text{pop}_i/\text{pop} \ast \text{pop}_j/\text{pop})}$.

If type = "tango", then $w_{ij} = \exp(-4 \ast d_{ij}^2/\kappa^2)$.

Value

Returns an $n \times n$ matrix of weights.

Author(s)

Joshua French

References


See Also

tango.test

Examples

data(nydf)
coords = as.matrix(nydf[,c("longitude", "latitude")])
w = tango.weights(coords, kappa = 1, longlat = TRUE)

---

**uls.sim**

*Perform uls.test on simulated data*

Description

uls.sim efficiently performs uls.test on a simulated data set. The function is meant to be used internally by the uls.test function, but is informative for better understanding the implementation of the test.
Usage

uls.sim(
     nsim = 1,
     ty, ex,
     w, pop,
     ubpop,
     type = "poisson",
     check.unique = FALSE,
     cl = NULL
)

Arguments

nsim  A positive integer indicating the number of simulations to perform.
ty  The total number of cases in the study area.
ex  The expected number of cases for each region. The default is calculated under
     the constant risk hypothesis.
w  A binary spatial adjacency matrix for the regions.
pop  The population size associated with each region.
ubpop  The upperbound of the proportion of the total population to consider for a clus-
     ter.
type  The type of scan statistic to compute. The default is "poisson". The other
     choice is "binomial".
check.unique  A logical value indicating whether a check for unique values should be deter-
     mined. The default is FALSE. This is unlikely to make a practical different for
     most real data sets.
cl  A cluster object created by makeCluster, or an integer to indicate number of
     child-processes (integer values are ignored on Windows) for parallel evaluations
     (see Details on performance).

Value

A vector with the maximum test statistic for each simulated data set.

Examples

data(nydf)
data(nyw)
coords = with(nydf, cbind(longitude, latitude))
cases = floor(nydf$cases)
pop = nydf$pop
ty = sum(cases)
ex = ty/sum(pop) * pop
tsims = uls.sim(1, ty, ex, nyw, pop = pop, ubpop = 0.5)
uls.test

Upper Level Set Spatial Scan Test

Description

uls.test performs the Upper Level Set (ULS) spatial scan test of Patil and Taillie (2004). The test is performed using the spatial scan test based on a fixed number of cases. The windows are based on the Upper Level Sets proposed by Patil and Taillie (2004). The clusters returned are non-overlapping, ordered from most significant to least significant. The first cluster is the most likely to be a cluster. If no significant clusters are found, then the most likely cluster is returned (along with a warning).

Usage

uls.test(
  coords,
  cases,
  pop,
  w,
  ex = sum(cases)/sum(pop) * pop,
  nsim = 499,
  alpha = 0.1,
  ubpop = 0.5,
  longlat = FALSE,
  type = "poisson",
  check.unique = FALSE
)

Arguments

coops
  An \( n \times 2 \) matrix of centroid coordinates for the regions.
cases
  The number of cases observed in each region.
pop
  The population size associated with each region.
w
  A binary spatial adjacency matrix for the regions.
ex
  The expected number of cases for each region. The default is calculated under the constant risk hypothesis.
nsim
  The number of simulations from which to compute the p-value.
alpha
  The significance level to determine whether a cluster is significant. Default is 0.10.
ubpop
  The upperbound of the proportion of the total population to consider for a cluster.
longlat
  The default is FALSE, which specifies that Euclidean distance should be used. If longlat is TRUE, then the great circle distance is used to calculate the inter-centroid distance.
cl A cluster object created by `makeCluster`, or an integer to indicate number of child-processes (integer values are ignored on Windows) for parallel evaluations (see Details on performance).

type The type of scan statistic to compute. The default is "poisson". The other choice is "binomial".

check.unique A logical value indicating whether a check for unique values should be determined. The default is FALSE. This is unlikely to make a practical different for most real data sets.

Details

The ULS method has a special (and time consuming) construction when the observed rates aren't unique. This is unlikely to arise for real data, except with observed rates of 0, which are of little interest. The method can take substantially if this is considered.

Value

Returns a list of length two of class scan. The first element (clusters) is a list containing the significant, non-ovlappering clusters, and has the the following components:

  locids The location ids of regions in a significant cluster.
  pop The total population in the cluser window.
  cases The observed number of cases in the cluster window.
  expected The expected number of cases in the cluster window.
  smr Standardized mortality ratio (observed/expected) in the cluster window.
  rr Relative risk in the cluster window.
  loglikrat The loglikelihood ratio for the cluster window (i.e., the log of the test statistic).
  pvalue The pvalue of the test statistic associated with the cluster window.

The second element of the list is the centroid coordinates. This is needed for plotting purposes.

Author(s)

Joshua French

References


See Also

`print.smerc_cluster`, `summary.smerc_cluster`, `plot.smerc_cluster`, `scan.stat`, `scan.test`
Examples

data(nydf)
data(nyw)
coords = with(nydf, cbind(longitude, latitude))
out = uls.test(coords = coords, cases = floor(nydf$cases),
              pop = nydf$pop, w = nyw,
              alpha = 0.05, longlat = TRUE,
              nsim = 9, ubpop = 0.5)
data(nypoly)
library(sp)
plot(nypoly, col = color.clusters(out))

uls.zones

Determine sequence of ULS zones.

Description

uls.zones determines the unique zones obtained by implementing the ULS (Upper Level Set) test of Patil and Taillie (2004).

Usage

uls.zones(cases, pop, w, ubpop = 0.5, check.unique = FALSE)

Arguments

cases The number of cases observed in each region.
pop The population size associated with each region.
w A binary spatial adjacency matrix for the regions.
ubpop The upperbound of the proportion of the total population to consider for a cluster.
check.unique A logical value indicating whether a check for unique values should be determined. The default is FALSE. This is unlikely to make a practical different for most real data sets.

Details

The zones returned must have a total population less than ubpop * sum(pop) of all regions in the study area.

Value

Returns a list of zones to consider for clustering. Each element of the list contains a vector with the location ids of the regions in that zone.

Author(s)

Joshua French
zones.sum

References

Examples
```
data(nydf)
data(nyw)
uls.zones(cases = nydf$cases, pop = nydf$population, w = nyw)

zones.sum
```

Description

zones.sum computes the sum of \( y \) for the indices in each element of the list contained in \( \text{zones} \).

Usage

```
zones.sum(zones, y)
```

Arguments

- **zones**: A list of nearest neighbors in the format produced by \text{scan.zones}.
- **y**: A numeric vector of values to be summed over.

Value

A numeric vector.

Examples

```
# show nn.cumsum example for a circular scan setting
data(nydf)
coords = with(nydf, cbind(longitude, latitude))
cases = floor(nydf$cases)
zones = scan.zones(coords, pop = nydf$pop, ubpop = 0.1)
# compute cumulative sums over all nn
szones = zones.sum(zones, cases)
# compute cumulative sums over just the first set of nn
szones2 = sapply(zones, function(x) sum(cases[x]))
# check equality
all.equal(szones, szones2)
```
Index

array, 47
backtick, 47
base::print, 73, 74
base::summary, 94
bn.test, 3
bn.zones, 5
casewin, 6, 8
casewin(bn.zones), 5
cepp.sim, 6
cepp.test, 6, 7
ccepp.weights, 8
clusters, 9
color.clusters, 10
combine.zones, 10
csg2, 11
dc.sim, 13
dc.test, 13, 14, 16
dc.zones, 16
dist.ellipse, 17
distinct, 18
dmst.sim, 19
dmst.test, 19, 20, 22
dmst.zones, 22
dweights, 96, 97
dweights(tango.weights), 97
edmst.sim, 24
edmst.test, 24, 25, 27
edmst.zones, 27
eblopow, 28
eblopow.nn, 29, 30
eblopow.sim, 30
eblopow.test, 30, 31, 33, 93, 94
neblopow.zones, 33

fast.sim, 34
fast.test, 34, 35
fast.zones, 37

flex.sim, 38
flex.test, 38, 39
flex.zones, 41
flex_test, 43
flex_zones, 45

eknn, 46, 63, 79, 81

lapply, 47
lcsg2(csg2), 11
lget, 47
lgetElement(lget), 47

makeCluster, 13, 15, 16, 19, 21, 23, 24, 26, 27, 31, 32, 34, 36, 39, 40, 42, 44, 49, 52, 54, 55, 57, 68, 76, 78, 80, 82, 84, 87, 99, 101

mlf.test, 48
mlf.zones, 50
mlink.sim, 52
mlink.test, 52, 53, 55
mlink.zones, 55
mst.all, 56
mst.seq, 59

name, 47
names, 47

n.cumsum, 61
nn2zones, 62, 63
nndist, 13, 19, 24, 52, 62
nndup, 63
nnpop, 61–63, 64, 83
noz, 65
nydf, 66
nypoly, 67
nyw, 67

optimal_ubpop, 29, 68, 71

pbapply, 42, 45, 80, 82
INDEX

plot.smerc_cluster, 4, 8, 15, 21, 26, 33, 36, 41, 44, 50, 54, 70, 79, 88, 101
plot.smerc_optimal_ubpop, 71
plot.tango, 72
points, 72
print.smerc_cluster, 4, 8, 15, 21, 26, 33, 36, 41, 44, 50, 54, 73, 79, 88, 101
print.smerc_optimal_ubpop, 73
print.tango, 74
rflex.midp, 75
rflex.sim, 76
rflex.test, 76, 77
rflex.zones, 79
rflex_zones, 81
scan.nn (nnpop), 64
scan.sim, 83
scan.stat, 15, 21, 26, 33, 36, 41, 44, 50, 54, 79, 84, 88, 101
scan.test, 4, 8, 15, 21, 26, 33, 41, 44, 50, 54, 69, 79, 83, 86, 101
scan.zones, 88, 103
scan_stat, 89
scsg2 (csg2), 11
sget (lget), 47
sgetElement (lget), 47
sig_noc, 91
smerc, 92
smerc_cluster, 9, 92
stat.binom (scan.stat), 84
stat.poisson (scan.stat), 84
stat_binom (scan_stat), 89
stat_poisson (scan_stat), 89
summary.smerc_cluster, 4, 8, 15, 21, 26, 33, 36, 41, 44, 50, 54, 79, 88, 94, 101
tango.stat, 95
tango.test, 72, 96, 98
tango.weights, 97
uls.sim, 98
uls.test, 98, 100
uls.zones, 102
zones.sum, 103