Package: smacpod (via r-universe)

August 26, 2024

Type Package

Title Statistical Methods for the Analysis of Case-Control Point Data

Version 2.6.1

Maintainer Joshua French <joshua.french@ucdenver.edu>

Description Statistical methods for analyzing case-control point data.
Methods include the ratio of kernel densities, the difference
in K Functions, the spatial scan statistic, and q nearest
neighbors of cases.

License GPL (>=2)

LazyLoad yes

Depends R (>= 3.1.1)

Imports spatstat.geom, spatstat.random, spatstat.explore, smerc,
plotrix, abind, pbapply

Suggests testthat, knitr, rmarkdown

VignetteBuilder knitr

RoxygenNote 7.2.3

Encoding UTF-8

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

RemoteUrl https://github.com/jfrench/smacpod

RemoteRef HEAD

RemoteSha f2f4e86da63fd0581fe2e6ffcb26e653e89f6eef

Contents

`arg_check_alternative` *Argument check alternative*

Description

Argument check alternative

Usage

```
arg_check_alternative(alternative)
```

Arguments

`alternative`	One of "lower", "greater", "two.sided"

`circles.intersect` *Determine whether circles intersect*

Description

`circles.intersect` determines whether circles intersect with each other.

Usage

```
circles.intersect(coords, r)
```

Arguments

`coords`	A matrix of coordinates with the centroid of each circle.
`r`	A vector containing the radii of the circles. The length of `r` must equal the number of rows of `coords`.

Details

The algorithm is based on the premise that two circles intersect if, and only if, the distance between their centroids is between the sum and the difference of their radii. I have squared the respective parts of the inequality in the implemented algorithm.

Value

Returns a matrix of logical values indicating whether the circles intersect.

Author(s)

Joshua French

Examples

```
# first two circles intersect each other,
# the next two circles intersect each other
# (but not the previous ones)
# the last circles doesn't intersect any other circle
co = cbind(c(1, 2, 5, 6, 9), c(1, 2, 5, 6, 9))
r = c(1.25, 1.25, 1.25, 1.25, 1.25)
# draw circles
circles.plot(co, r)
# confirm intersections
circles.intersect(co, r)

# nested circles (don't intersect)
co = matrix(rep(0, 4), nrow = 2)
r = c(1, 1.5)
circles.plot(co, r)
circles.intersect(co, r)
```

circles.plot *Plot circles*

Description

`plot.circles` creates a plot with one or more circles (or adds them to an existing plot).

Usage

```
circles.plot(
  coords,
  r,
  add = FALSE,
  ...,
  nv = 100,
  border = NULL,
  ccol = NA,
  clty = 1,
  density = NULL,
  angle = 45,
  clwd = 1
)
```

Arguments

`coords`	A matrix of coordinates with the centroid of each circle.
`r`	A vector containing the radii of the circles. The length of `r` must equal the number of rows of `coords`.
`add`	A logical value indicating whether the circles should be added to an existing plot. Default is `FALSE`.
`...`	Additional arguments passed to the `plot` function.
`nv`	Number of vertices to draw the circle.
`border`	A vector with the desired border of each circle. The length should either be 1 (in which case the border is repeated for all circles) or should match the number of rows of `coords`.
`ccol`	A vector with the desired color of each circle. The length should either be 1 (in which case the color is repeated for all circles) or should match the number of rows of `coords`.
`clty`	A vector with the desired line type of each circle. The length should either be 1 (in which case the line type is repeated for all circles) or should match the number of rows of `coords`.
`density`	A vector with the density for a patterned fill. The length should either be 1 (in which case the density is repeated for all circles) or should match the number of rows of `coords`. See `polygon`

`angle`	A vector with the angle of a patterned fill. The length should either be 1 (in which case the angle is repeated for all circles) or should match the number of rows of `coords`. See `polygon`
`clwd`	A vector with the desired line width of each circle. The length should either be 1 (in which case the line width is repeated for all circles) or should match the number of rows of `coords`.

Author(s)

Joshua French

See Also

`draw.circle`, `polygon`

Examples

```
co = cbind(c(1, 2, 5, 6, 9), c(1, 2, 5, 6, 9))
r = c(1.25, 1.25, 1.25, 1.25, 1.25)
# draw circles
circles.plot(co, r)
circles.plot(co, r,
   ccol = c("blue", "blue", "orange", "orange", "brown"),
   density = c(10, 20, 30, 40, 50),
   angle = c(45, 135, 45, 136, 90))
```

`clusters.spscan` *Extract clusters*

Description

Extract clusters

Usage

```
## S3 method for class 'spscan'
clusters(x, idx = seq_along(x$clusters), ...)
```

Arguments

`x`	An object of class spscan from the `spscan.test`
`idx`	An index vector indicating the elements of `x$clusters` to print information for. The default is all clusters.
`...`	Currently unimplemented

Value

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

Examples

```
data(grave)
# apply scan method
out = spscan.test(grave, nsim = 99)
# print scan object
clusters(out)
```

---

gradient.color.scale *Create gradient color scale with midpoint*

Description

Create gradient color scale with midpoint

Usage

```
gradient.color.scale(
  minval,
  maxval,
  n = 11,
  low = "blue",
  mid = "white",
  high = "red",
  midpoint = 0,
  ...
)
```

Arguments

`minval`	The minimum value of the data to be colored
`maxval`	The maximum value of the data to be colored
`n`	The desired number of breaks (approximately)
`low`	The color for the low values
`mid`	The color used for the midpoint of the gradient
`high`	The color used for the high values
`midpoint`	The midpoint of the color scale
`...`	Arguments passed on to `grDevices::colorRamp` `colors` colors to interpolate; must be a valid argument to `col2rgb()`. `bias` a positive number. Higher values give more widely spaced colors at the high end. `space` a character string; interpolation in RGB or CIE Lab color spaces. `interpolate` use spline or linear interpolation. `alpha` logical: should alpha channel (opacity) values be returned? It is an error to give a true value if `space` is specified.

Value

A list with `col` and `breaks` components specifying the colors and breaks of the color scale.

References

Based on code from https://stackoverflow.com/a/10986203/5931362

Examples

```
data(grave)
lr = logrr(grave)
grad = gradient.color.scale(min(lr$v, na.rm = TRUE), max(lr$v, na.rm = TRUE))
plot(lr, col = grad$col, breaks = grad$breaks)
```

`grave` *Medieval Grave Site Data*

Description

This data set contains 143 observations of medieval grave site data stored as a `ppp` class object from the `spatstat.geom` package. The data are marked as being "affected" by a tooth deformity or "unaffected" by a tooth deformity.

Usage

```
data(grave)
```

Format

`ppp` (planar point process) class object from the `spatstat.geom` package.

Author(s)

Joshua French

Source

Waller, L.A. and Gotway, C.A. (2005). Applied Spatial Statistics for Public Health Data. Hoboken, NJ: Wiley.

See Also

`ppp`

kd *Difference of estimated K functions*

Description

kd determines the difference in estimated K functions for a set of cases and controls.

Usage

```
kd(
  x,
  case = 2,
  r = NULL,
  rmax = NULL,
  breaks = NULL,
  correction = c("border", "isotropic", "Ripley", "translate"),
  nlarge = 3000,
  domain = NULL,
  var.approx = FALSE,
  ratio = FALSE
)
```

Arguments

x	A ppp object with marks for the case and control groups.
case	The name of the desired "case" group in `levels(x$marks)`. Alternatively, the position of the name of the "case" group in `levels(x$marks)`. Since we don't know the group names, the default is 2, the second position of `levels(x$marks)`. `x$marks` is assumed to be a factor. Automatic conversion is attempted if it is not.
r	Optional. Vector of values for the argument r at which $K(r)$ should be evaluated. Users are advised *not* to specify this argument; there is a sensible default. If necessary, specify `rmax`.
rmax	Optional. Maximum desired value of the argument r.
breaks	This argument is for internal use only.
correction	Optional. A character vector containing any selection of the options `"none"`, `"border"`, `"bord.modif"`, `"isotropic"`, `"Ripley"`, `"translate"`, `"translation"`, `"rigid"`, `"none"`, `"periodic"`, `"good"` or `"best"`. It specifies the edge correction(s) to be applied. Alternatively `correction="all"` selects all options.
nlarge	Optional. Efficiency threshold. If the number of points exceeds `nlarge`, then only the border correction will be computed (by default), using a fast algorithm.
domain	Optional. Calculations will be restricted to this subset of the window. See Details of `Kest`.
var.approx	Logical. If `TRUE`, the approximate variance of $\hat{K}(r)$ under CSR will also be computed.
ratio	Logical. If `TRUE`, the numerator and denominator of each edge-corrected estimate will also be saved, for use in analysing replicated point patterns.

Details

This function relies internally on the `Kest` and `eval.fv`. The arguments are essentially the same as the `Kest` function, and the user is referred there for more details about the various arguments.

Value

Returns an `fv` object. See documentation for `Kest`.

Author(s)

Joshua French

References

Waller, L.A. and Gotway, C.A. (2005). Applied Spatial Statistics for Public Health Data. Hoboken, NJ: Wiley.

See Also

`Kest`, `eval.fv`

Examples

```
data(grave)
kd = kd(grave)
plot(kd)
```

`kdest` *Difference of estimated K functions*

Description

`kdest` computes the difference in estimated K functions for a set of cases and controls, with `KD(r)` = `K_case(r)` - `K_control(r)` denoting the estimated difference at distance `r`. If `nsim` > `0`, then pointwise tolerance envelopes for `KD(r)` are constructed under the random labeling hypothesis for each distance `r`. The `summary` function can be used to determine the distances for which `KD(r)` is above or below the tolerance envelopes. The `plot` function will plot `KD(r)` versus r, along with the tolerance envelopes, the min/max envelopes of `KD(r)` simulated under the random labeling hypothesis, and the average KD(r) under the random labeling hypothesis.

Usage

```
kdest(
  x,
  case = 2,
  nsim = 0,
  level = 0.95,
  r = NULL,
  rmax = NULL,
  breaks = NULL,
  correction = c("border", "isotropic", "Ripley", "translate"),
  nlarge = 3000,
  domain = NULL,
  var.approx = FALSE,
  ratio = FALSE
)
```

Arguments

`x`	A `ppp` object package with marks for the case and control groups. `x$marks` is assumed to be a factor. Automatic conversion is attempted if it is not.
`case`	The name of the desired "case" group in `levels(x$marks)`. Alternatively, the position of the name of the "case" group in `levels(x$marks)`. Since we don't know the group names, the default is 2, the second position of `levels(x$marks)`. `x$marks` is assumed to be a factor. Automatic conversion is attempted if it is not.
`nsim`	The number of simulated data sets from which to construct tolerance envelopes under the random labeling hypothesis. The default is 0 (i.e., no envelopes).
`level`	The level of the tolerance envelopes.
`r`	Optional. Vector of values for the argument r at which $K(r)$ should be evaluated. Users are advised *not* to specify this argument; there is a sensible default. If necessary, specify `rmax`.
`rmax`	Optional. Maximum desired value of the argument r.
`breaks`	This argument is for internal use only.
`correction`	Optional. A character vector containing any selection of the options `"none"`, `"border"`, `"bord.modif"`, `"isotropic"`, `"Ripley"`, `"translate"`, `"translation"`, `"rigid"`, `"none"`, `"periodic"`, `"good"` or `"best"`. It specifies the edge correction(s) to be applied. Alternatively `correction="all"` selects all options.
`nlarge`	Optional. Efficiency threshold. If the number of points exceeds `nlarge`, then only the border correction will be computed (by default), using a fast algorithm.
`domain`	Optional. Calculations will be restricted to this subset of the window. See Details.
`var.approx`	Logical. If `TRUE`, the approximate variance of $\hat{K}(r)$ under CSR will also be computed.
`ratio`	Logical. If `TRUE`, the numerator and denominator of each edge-corrected estimate will also be saved, for use in analysing replicated point patterns.

Details

This function relies internally on the `Kest` and `eval.fv` functions. The arguments are essentially the same as the `Kest` function, and the user is referred there for more details about the various arguments.

Value

Returns a kdenv object. See documentation for `Kest`.

Author(s)

Joshua French

References

Waller, L.A. and Gotway, C.A. (2005). Applied Spatial Statistics for Public Health Data. Hoboken, NJ: Wiley.

See Also

`Kest`, `eval.fv`

Examples

```
data(grave)
# estimate and plot KD(r)
kd1 = kdest(grave, case = "affected")
plot(kd1, iso ~ r, ylab = "difference", legend = FALSE, main = "")
kd2 = kdest(grave, case = 2, nsim = 9, level = 0.8)
kd2 # print object
summary(kd2) # summarize distances KD(r) outside envelopes
plot(kd2)
# manually add legend
legend("bottomright", legend = c("obs", "avg", "max/min env", "95% env"),
       lty = c(1, 2, 1, 2), col = c("black", "red", "darkgrey", "lightgrey"),
       lwd = c(1, 1, 10, 10))
```

`kdplus.test` *Global test of clustering using difference in K functions*

Description

`kdplus.test` performs a global test of clustering for comparing cases and controls using the method of Diggle and Chetwynd (1991). It relies on the difference in estimated K functions.

Usage

```
kdplus.test(x)
```

Arguments

`x`	A `kdenv` object from the `kdest` function.

Value

A list providing the observed test statistic (`kdplus`) and the estimate p-value `pvalue`.

Author(s)

Joshua French

References

Waller, L.A. and Gotway, C.A. (2005). Applied Spatial Statistics for Public Health Data. Hoboken, NJ: Wiley.

Diggle, Peter J., and Amanda G. Chetwynd. "Second-order analysis of spatial clustering for inhomogeneous populations." Biometrics (1991): 1155-1163.

See Also

`kdest`

Examples

```
data(grave)
# construct envelopes for differences in estimated K functions
kdenv = kdest(grave, nsim = 9)
kdplus.test(kdenv)
```

`logrr` *Log ratio of spatial densities*

Description

`logrr` computes the estimated log relative risk of cases relative to controls. The log relative risk at location s is defined as `r(s) = ln(f(s)/g(s))`. The numerator, `f(s)`, is the spatial density of the case group. The denominator, `g(s)`, is the spatial density of the control group. If `nsim > 0`, then pointwise (at each pixel) tolerance envelopes are estimated under the random labeling hypothesis. The tolerance envelopes can be used to assess pixels where the log relative risk differs significantly from zero. See Details.

Usage

```
logrr(
  x,
  sigma = NULL,
  sigmacon = NULL,
  case = 2,
  nsim = 0,
  level = 0.9,
  alternative = "two.sided",
  envelope = "pixelwise",
  ...,
  bwargs = list(),
  weights = NULL,
  edge = TRUE,
  varcov = NULL,
  at = "pixels",
  leaveoneout = TRUE,
  adjust = 1,
  diggle = FALSE,
  kernel = "gaussian",
  scalekernel = is.character(kernel),
  positive = FALSE,
  verbose = TRUE,
  return_sims = FALSE
)
```

Arguments

x	A ppp object package with marks for the case and control groups. x$marks is assumed to be a factor. Automatic conversion is attempted if it is not.
sigma	Standard deviation of isotropic smoothing kernel for cases. Either a numerical value, or a function that computes an appropriate value of sigma. If not specified, then bw.relrisk is used.
sigmacon	Standard deviation of isotropic smoothing kernel for controls. Default is the same as sigma.
case	The name of the desired "case" group in levels(x$marks). Alternatively, the position of the name of the "case" group in levels(x$marks). Since we don't know the group names, the default is 2, the second position of levels(x$marks). x$marks is assumed to be a factor. Automatic conversion is attempted if it is not.
nsim	The number of simulated data sets from which to construct tolerance envelopes under the random labeling hypothesis. The default is 0 (i.e., no envelopes).
level	The level of the tolerance envelopes.
alternative	The type of envelopes to construct. The default is "two.sided" (upper and lower envelopes). The values "less" (lower envelope) and "greater" (upper envelope) are also valid.
envelope	The type of envelope to construct. The default is "pixelwise". The other option is "simulataneous", which controls the tolerance level across the entire study area.
...	Additional arguments passed to pixellate.ppp and as.mask to determine the pixel resolution, or passed to sigma if it is a function.
bwargs	A list of arguments for the bandwidth function supplied to sigma and sigmacon, if applicable.
weights	Optional weights to be attached to the points. A numeric vector, numeric matrix, an expression, or a pixel image.
edge	Logical value indicating whether to apply edge correction.
varcov	Variance-covariance matrix of anisotropic smoothing kernel. Incompatible with sigma.

`at`	String specifying whether to compute the intensity values at a grid of pixel locations (`at="pixels"`) or only at the points of `x` (`at="points"`).
`leaveoneout`	Logical value indicating whether to compute a leave-one-out estimator. Applicable only when `at="points"`.
`adjust`	Optional. Adjustment factor for the smoothing parameter.
`diggle`	Logical. If `TRUE`, use the Jones-Diggle improved edge correction, which is more accurate but slower to compute than the default correction.
`kernel`	The smoothing kernel. A character string specifying the smoothing kernel (current options are `"gaussian"`, `"epanechnikov"`, `"quartic"` or `"disc"`), or a pixel image (object of class `"im"`) containing values of the kernel, or a `function(x,y)` which yields values of the kernel.
`scalekernel`	Logical value. If `scalekernel=TRUE`, then the kernel will be rescaled to the bandwidth determined by `sigma` and `varcov`: this is the default behaviour when `kernel` is a character string. If `scalekernel=FALSE`, then `sigma` and `varcov` will be ignored: this is the default behaviour when `kernel` is a function or a pixel image.
`positive`	Logical value indicating whether to force all density values to be positive numbers. Default is `FALSE`.
`verbose`	Logical value indicating whether to issue warnings about numerical problems and conditions.
`return_sims`	A logical value indicating whether parts of the simulated data shoudl be returned. The default is `FALSE`. This is mostly used for debugging purposes.

Details

If `nsim=0`, the `plot` function creates a heat map of the log relative risk. If `nsim > 0`, the `plot` function colors the pixels where the estimated log relative risk is outside the tolerance envelopes created under the random labeling hypothesis (i.e., pixels with potential clustering of cases or controls). Colored regions with values above 0 indicate a cluster of cases relative to controls (without controlling for multiple comparisons), i.e., a region where the the density of the cases is greater than the the density of the controls. Colored regions with values below 0 indicate a cluster of controls relative to cases (without controlling for multiple comparisons), i.e., a region where the density of the controls is greater than the density of the cases.

The `two.sided` alternative test constructs two-sided tolerance envelopes to assess whether the estimated `r(s)` deviates more than what is expected under the random labeling hypothesis. The `greater` alternative constructs an upper tolerance envelope to assess whether the estimated `r(s)` is greater than what is expected under the random labeling hypothesis, i.e., where there is clustering of cases relative to controls. The `lower` alternative constructs a lower tolerance envelope to assess whether the estimated `r(s)` is lower than what is expected under the random labeling hypothesis, i.e., where there is clustering of controls relative to cases.

If the estimated density of the case or control group becomes too small, this function may produce warnings due to numerical underflow. Increasing the bandwidth (`sigma`) may help.

Value

The function produces an object of type `logrrenv`. Its components are similar to those returned by the `density.ppp`, with the intensity values replaced by the log ratio of spatial densities of f and g.

Includes an array `simr` of dimension c(nx, ny, nsim + 1), where nx and ny are the number of x and y grid points used to estimate the spatial density. `simr[,,1]` is the log ratio of spatial densities for the observed data, and the remaining `nsim` elements in the third dimension of the array are the log ratios of spatial densities from a new ppp simulated under the random labeling hypothesis.

Author(s)

Joshua French (and a small chunk by the authors of the `density.ppp`) function for consistency with the default behavior of that function).

References

Waller, L.A. and Gotway, C.A. (2005). Applied Spatial Statistics for Public Health Data. Hoboken, NJ: Wiley.

Kelsall, Julia E., and Peter J. Diggle. "Kernel estimation of relative risk." Bernoulli (1995): 3-16.

Kelsall, Julia E., and Peter J. Diggle. "Non-parametric estimation of spatial variation in relative risk." Statistics in Medicine 14.21-22 (1995): 2335-2342.

Hegg, Alex and French, Joshua P. (2022) "Simultaneous tolerance bands of log relative risk for case-control point data. Technical report.

Examples

```
data(grave)
# estimate and plot log relative risk
r = logrr(grave, case = "affected")
plot(r)
# use scott's bandwidth
r2 = logrr(grave, case = 2, sigma = spatstat.explore::bw.scott)
plot(r2)
# construct pointwise tolerance envelopes for log relative risk
## Not run:
renv = logrr(grave, nsim = 9)
print(renv) # print information about envelopes
plot(renv) # plot results
# plot using a better gradient
grad = gradient.color.scale(min(renv$v, na.rm = TRUE), max(renv$v, na.rm = TRUE))
plot(renv, col = grad$col, breaks = grad$breaks, conlist = list(col = "lightgrey"))
## End(Not run)
```

`logrr.test` *Global test of clustering using log ratio of spatial densities*

Description

`logrr.test` performs a global test of clustering for comparing cases and controls using the log ratio of spatial densities based on the method of Kelsall and Diggle (1995).

Usage

```
logrr.test(x)
```

Arguments

`x`	An `logrrenv` object from the `logrr` function.

Value

A list providing the observed test statistic (`islogrr`) and the estimated p-value (`pvalue`).

Author(s)

Joshua French

References

Waller, L.A. and Gotway, C.A. (2005). Applied Spatial Statistics for Public Health Data. Hoboken, NJ: Wiley.

Kelsall, Julia E., and Peter J. Diggle. "Non-parametric estimation of spatial variation in relative risk." Statistics in Medicine 14.21-22 (1995): 2335-2342.

Examples

```
data(grave)
## Not run:
logrrenv = logrr(grave, nsim = 9)
logrr.test(logrrenv)
## End(Not run)
```

`nn` *Determine nearest neighbors*

Description

`nn` determines the nearest neighbors for a set of observations based on a distance matrix.

Usage

```
nn(d, k, method = "c", self = FALSE)
```

Arguments

`d`	A square distance matrix for the coordinates of interest.
`k`	The number of neighbors to return (if `method = "c"`) or the distance for which observations are considered neighbors (if `method = "d"`).
`method`	The method of determining the neighbors. The default is `"c"`, specifying that the `k` nearest neighbors (the number of neighbors) for each observation should be returned. The alternative is `"d"`, meaning that neighbors are determined by their distance from an observation. In that case, two observations are neighbors if their separation distance is less or equal to `k`.
`self`	A logical indicating whether an observation is a neighbor with itself. The default is `FALSE`.

Details

This function determine nearest neighbors in two ways: 1. number of neighbors or 2. distance.

If `method = "c"`, then `k` specifies the total number of neighbors to return for each observation.

If `method = "d"`, then `k` specifies the maximum distance for which an observation is considered a neighbor.

The function returns the neighbors for each observation.

Value

Returns a list with the nearest neighbors of each observation. For each element of the list, the indices order neighbors from nearest to farthest.

Author(s)

Joshua French

Examples

```
data(grave)
# make distance matrix
d = as.matrix(dist(cbind(grave$x, grave$y)))
# 3 nearest neighbors
nnc = nn(d, k = 3, method = "c")
# nearest neighbors within k units of each observation
nnd = nn(d, k = 200, method = "d")
```

`noc` *Determine non-overlapping clusters*

Description

Determine non-overlapping clusters from a list of potential clusters.

Usage

```
noc(x)
```

Arguments

`x`	A list containing the potential clusters.

Details

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

Value

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

Author(s)

Joshua French

Examples

```
x = list(1:2, 1:3, 4:5, 4:6, 7:8)
noc(x)
```

plot.kdenv *Plot a* kdenv *object.*

Description

Plots an object from kdest of class kdenv.

Usage

```
## S3 method for class 'kdenv'
plot(
  x,
  ...,
  shadecol1 = "darkgrey",
  shadecol2 = "lightgrey",
  main = "",
  legend = FALSE
)
```

Arguments

x	An object of class kdenv produced by kdest.
...	Additional graphical parameters passed to the plot.fv function, which is used internally for plotting.
shadecol1	Color for min/max tolerance envelopes generated under the random labeling hypothesis. The default is a dark grey.
shadecol2	Shade color for non-rejection envelopes. The default is "lightgrey".
main	A main title for the plot. The default is blank.
legend	Logical for whether a legend should automatically be displayed. Default is FALSE. See Details for an explanation of the components of the plot.

Details

The solid line indicates the observed difference in the K functions for the cases and controls. The dashed line indicates the average difference in the K functions produced from the data sets simulated under the random labeling hypothesis when executing the kdest function. The shaded areas indicate the tolerance envelopes constructed in x for tolerance level level and the min/max envelopes constructed under the random labeling hypothesis.

See Also

plot.fv

Examples

```
data(grave)
kdenv = kdest(grave, nsim = 19, level = 0.9)
plot(kdenv)
plot(kdenv, legend = TRUE)
```

---

plot.logrrenv *Plots objects produced by the logrr function.*

Description

Plots objects of class logrrenv produced by the logrr function.

Usage

```
## S3 method for class 'logrrenv'
plot(x, ..., conlist = list(), main = "")
```

Arguments

x	An object of class logrrenv.
...	Additional graphical parameters passed to the image.im function. See Details.
conlist	Additional argument passed to the contour.im function.
main	A main title for the plot. Default is blank.

Details

An important aspect of this plot is the color argument (col) used for displaying the regions outside the non-rejection envelopes. If NULL (the implicit default), then the default color palette used by image.im will be used. Simpler schemes, e.g., c("blue", "white", "orange") can suffice. See the examples.

See Also

plot.im, contour.im

Examples

```
data(grave)
## Not run:
logrrsim = logrr(grave, nsim = 9)
plot(logrrsim)
# no border or ribben (legend).  Simple color scheme.
plot(logrrsim, col = c("blue", "white", "orange"), ribbon = FALSE, box = FALSE)
# alternate color scheme
plot(logrrsim, col = topo.colors(12), conlist = list(col = "lightgrey"))
## End(Not run)
```

plot.spscan *Plots object from* spscan.test.

Description

Plots object of class scan from spscan.test.

Usage

```
## S3 method for class 'spscan'
plot(x, ..., nv = 100, border = NULL, ccol = NULL, clty = NULL, clwd = NULL)
```

Arguments

x	An object of class spscan.
...	Additional graphical parameters passed to the plot.ppp function.
nv	The number of vertices when drawing the cluster circles. Default is 100.
border	The border color of the circle. Default is NULL, meaning black.
ccol	Fill color of the circles. Default is NULL, indicating empty.
clty	Line type of circles. Default is NULL, indicting lty = 1.
clwd	Line width of circles. Default is NULL, indicating lwd = 2 for the most likely cluster and lwd = 1 for the rest.

Details

If border, ccol, clty, or clwd are specified, then the length of these vectors must match nrow(x$coords).

See Also

plot.ppp, draw.circle

Examples

```
data(grave)
out = spscan.test(grave, case = 2, alpha = 0.1, nsim = 49)
plot(out, chars = c(1, 20), main = "most likely cluster",
     border = "orange", ccol = NA)
# change color, lty, lwd of circles
set.seed(2)
out2 = spscan.test(grave, case = 2, alpha = 0.8, nsim = 49)
plot(out2, chars = c(1, 20),  border = "blue")
plot(out2, chars = c(1, 20),  border = c("blue", "orange"),
     clwd = c(3, 2), clty = c(2, 3))
```

print.kdenv *Print a* kdenv *object*

Description

Print an kdenv object produced by kdest.

Usage

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

Arguments

x	An object produced by the kdest function.
...	Not currently implemented.
extra	A logical value indicating whether extra information related to the internal fv object should be printed. The default is FALSE.

Value

Information about the kdest

Author(s)

Joshua French

print.kdenv_summary *Print a* kdenv_summary *object*

Description

Print a kdenv_summary object

Usage

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

Arguments

x	An object produced by summary.kdenv.
...	Not currently implemented.

Value

Print summary

Author(s)

Joshua French

`print.kdplus_test` *Print a* `kdplus_test` *object*

Description

Print a `kdplus_test` object produced by `kdplus.test`.

Usage

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

Arguments

`x`	An object produced by the `kdplus.test` function.
`...`	Not currently implemented.

Value

Information about the test

Author(s)

Joshua French

`print.logrrenv` *Print a* `logrrenv` *object*

Description

Print a `logrrenv` object produced by `logrr`.

Usage

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

Arguments

`x`	An object produced by the `logrr` function.
`...`	Not currently implemented.

Value

Information about the `logrrenv`

Author(s)

Joshua French

`print.logrr_test` *Print a* `logrr_test` *object*

Description

Print an `logrr_test` object produced by `logrr.test`.

Usage

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

Arguments

`x`	An object produced by the `logrr.test` function.
`...`	Not currently implemented.

Value

Information about the test

Author(s)

Joshua French

print.spscan — *Plots object from spscan.test.*

Description

Plots object of class spscan from spscan.test.

Usage

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

Arguments

x	An object of class spscan.
...	Arguments passed on to base::print
extra	A logical value. Default is FALSE. TRUE indicates that extra information should be printed.

Details

If border, ccol, clty, or clwd are specified, then the length of these vectors must match nrow(x$coords).

Examples

```
data(grave)
out = spscan.test(grave, case = 2, alpha = 0.1)
out
```

qnn.test — *q Nearest Neighbors Test*

Description

qnn.test calculates statistics related to the q nearest neighbors method of comparing case and control point patterns under the random labeling hypothesis.

Usage

```
qnn.test(x, q = 5, case = 2, nsim = 499, longlat = FALSE)
```

Arguments

`x`	A ppp object with marks for the case and control groups.
`q`	A vector of positive integers indicating the values of `q` for which to do the q nearest neighbors test.
`case`	The name of the desired "case" group in `levels(x$marks)`. Alternatively, the position of the name of the "case" group in `levels(x$marks)`. Since we don't know the group names, the default is 2, the second position of `levels(x$marks)`. `x$marks` is assumed to be a factor. Automatic conversion is attempted if it is not.
`nsim`	The number of simulations from which to compute p-value.
`longlat`	A logical value indicating whether Euclidean distance (`FALSE`) or Great Circle (WGS84 ellipsoid, `FALSE`) should be used. Default is `FALSE`, i.e., Euclidean distance.

Value

Returns a list with the following components:

`qsum`	A dataframe with the number of neighbors (q), test statistic (Tq), and p-value for each test.
`consum`	A dataframe with the contrasts (contrast), test statistic (Tcon), and p-value (pvalue) for the test of contrasts.

Author(s)

Joshua French

References

Waller, L.A., and Gotway, C.A. (2005). Applied Spatial Statistics for Public Health Data. Hoboken, NJ: Wiley.

Cuzick, J., and Edwards, R. (1990). Spatial clustering for inhomogeneous populations. Journal of the Royal Statistical Society. Series B (Methodological), 73-104.

Alt, K.W., and Vach, W. (1991). The reconstruction of "genetic kinship" in prehistoric burial complexes-problems and statistics. Classification, Data Analysis, and Knowledge Organization, 299-310.

Examples

```
data(grave)
qnn.test(grave, case = "affected", q = c(3, 5, 7, 9, 11, 13, 15))
```

smacpod *smacpod*

Description

*S*tatistical *M*thods for the *A*nalysis of *C*ase-control *Po*int *D*ata

Details

Statistical methods for analyzing case-control point data. Methods include the ratio of kernel densities, the difference in K Functions, the spatial scan statistic, and q nearest neighbors of cases.

Run

Author(s)

Maintainer: Joshua French <joshua.french@ucdenver.edu> (ORCID)

spdensity *Kernel smoothed spatial density of point pattern*

Description

`spdensity` computes a kernel smoothed spatial density function from a point pattern. This function is basically a wrapper for `density.ppp`. The `density.ppp` function computes the spatial intensity of a point pattern; the `spdensity` function scales the intensity to produce a true spatial density.

Usage

```
spdensity(
  x,
  sigma = NULL,
  ...,
  weights = NULL,
  edge = TRUE,
  varcov = NULL,
  at = "pixels",
  leaveoneout = TRUE,
  adjust = 1,
  diggle = FALSE,
  kernel = "gaussian",
  scalekernel = is.character(kernel),
  positive = FALSE,
  verbose = TRUE
)
```

Arguments

`x`	Point pattern (object of class `"ppp"`).
`sigma`	The smoothing bandwidth (the amount of smoothing). The standard deviation of the isotropic smoothing kernel. Either a numerical value, or a function that computes an appropriate value of `sigma`.
`...`	Additional arguments passed to `pixellate.ppp` and `as.mask` to determine the pixel resolution, or passed to `sigma` if it is a function.
`weights`	Optional weights to be attached to the points. A numeric vector, numeric matrix, an `expression`, or a pixel image.
`edge`	Logical value indicating whether to apply edge correction.
`varcov`	Variance-covariance matrix of anisotropic smoothing kernel. Incompatible with `sigma`.
`at`	String specifying whether to compute the intensity values at a grid of pixel locations (`at="pixels"`) or only at the points of `x` (`at="points"`).
`leaveoneout`	Logical value indicating whether to compute a leave-one-out estimator. Applicable only when `at="points"`.
`adjust`	Optional. Adjustment factor for the smoothing parameter.
`diggle`	Logical. If `TRUE`, use the Jones-Diggle improved edge correction, which is more accurate but slower to compute than the default correction.
`kernel`	The smoothing kernel. A character string specifying the smoothing kernel (current options are `"gaussian"`, `"epanechnikov"`, `"quartic"` or `"disc"`), or a pixel image (object of class `"im"`) containing values of the kernel, or a `function(x,y)` which yields values of the kernel.
`scalekernel`	Logical value. If `scalekernel=TRUE`, then the kernel will be rescaled to the bandwidth determined by `sigma` and `varcov`: this is the default behaviour when `kernel` is a character string. If `scalekernel=FALSE`, then `sigma` and `varcov` will be ignored: this is the default behaviour when `kernel` is a function or a pixel image.
`positive`	Logical value indicating whether to force all density values to be positive numbers. Default is `FALSE`.
`verbose`	Logical value indicating whether to issue warnings about numerical problems and conditions.

Value

This function produces the spatial density of `x` as an object of class `im`.

Author(s)

Joshua French

References

Waller, L.A. and Gotway, C.A. (2005). Applied Spatial Statistics for Public Health Data. Hoboken, NJ: Wiley.

See Also

density.ppp

Examples

```
data(grave)
contour(spdensity(grave))
```

---

spscan.test *Spatial Scan Test*

Description

spscan.test performs the spatial scan test of Kulldorf (1997) for case/control point data.

Usage

```
spscan.test(
  x,
  case = 2,
  nsim = 499,
  alpha = 0.1,
  maxd = NULL,
  cl = NULL,
  longlat = FALSE
)
```

Arguments

x	A ppp object with marks for the case and control groups.
case	The name of the desired "case" group in levels(x$marks). Alternatively, the position of the name of the "case" group in levels(x$marks). Since we don't know the group names, the default is 2, the second position of levels(x$marks). x$marks is assumed to be a factor. Automatic conversion is attempted if it is not.
nsim	The number of simulations from which to compute the p-value. A non-negative integer. Default is 499.
alpha	The significance level to determine whether a cluster is signficant. Default is 0.1.
maxd	The radius of the largest possible cluster to consider. Default is NULL, i.e., half the maximum 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). It can also be "future" to use a future backend (see Details), NULL (default) refers to sequential evaluation.
longlat	A logical value indicating whether Euclidean distance (FALSE) or Great Circle (WGS84 ellipsoid, FALSE) should be used. Default is FALSE, i.e., Euclidean distance.

Details

The test is performed using the random labeling hypothesis. The windows are circular 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).

Setting `cl` to a positive integer MAY speed up computations on non-Windows computers. However, parallelization does have overhead cost, and there are cases where parallelization results in slower computations.

Value

Returns a list of length two of class `spscan`. The first element (clusters) is a list containing the significant, non-overlapping 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 cluser window.
`cases`	The observed number of cases in the cluster window.
`expected`	The expected number of cases in the cluster window.
`smr`	Standarized mortaility ratio (observed/expected) in the cluster window.
`rr`	Relative risk in the cluster window.
`propcases`	Proportion of cases 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.

Various additional pieces of information are included for plotting, printing

Author(s)

Joshua French

References

Kulldorff M., Nagarwalla N. (1995) Spatial disease clusters: Detection and Inference. Statistics in Medicine 14, 799-810.

Kulldorff, M. (1997) A spatial scan statistic. Communications in Statistics – Theory and Methods 26, 1481-1496.

Waller, L.A. and Gotway, C.A. (2005). Applied Spatial Statistics for Public Health Data. Hoboken, NJ: Wiley.

Examples

```
data(grave)
# apply scan method
out = spscan.test(grave, case = "affected", nsim = 99)
# print scan object
out
print(out, extra = TRUE)
# summarize results
summary(out)
# plot results
plot(out, chars = c(1, 20), main = "most likely cluster")
# extract clusters from out
# each element of the list gives the location index of the events in each cluster
clusters(out)
# get warning if no significant cluster
out2 = spscan.test(grave, case = 2, alpha = 0.001, nsim = 0)
```

`summary.kdenv` *Summarize a* `kdenv` *object*

Description

Summarize the sequences of distances for which the difference in estimated K functions, `KD(r) = K_case(r) - K_control(r)`, falls outside the non-rejection envelopes.

Usage

```
## S3 method for class 'kdenv'
summary(object, ...)
```

Arguments

`object`	An object produced by the `kdest` function.
`...`	Not currently implemented.

Value

A list that contains the sequences of indices for which the estimated difference in KD functions is above the envelopes, below the envelopes, and the vector of distances.

Author(s)

Joshua French

summary.spscan *Summarize object from* spscan.test.

Description

Summarize object of class scan from spscan.test.

Usage

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

Arguments

object	An spscan object.
...	Arguments passed on to base::summary, base::summary
idx	An 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.

Value

Returns/prints a data frame. Each row of the data frame summarizes the centroid of each cluster, the cluster radius, the number of events in the cluster, the number of cases in the cluster, the expected number of cases in the cluster, the relative risk of the cluster (cases/events in cluster)(cases/events outside cluster), the natural logarithm of the test statistic, and the associated p-value.

Examples

```
data(grave)
out = spscan.test(grave, nsim = 99, alpha = 0.8)
summary(out)
```

Index