Package 'spNetwork'

October 30, 2021

Type Package

Title Spatial Analysis on Network

Version 0.2.1

Description Perform spatial analysis on network.
Implement several methods for spatial analysis on network: Network Kernel Density estimation, building of spatial matrices based on network distance ('listw' objects from 'spdep' package), K functions estimation
for point pattern analysis on network, k nearest neighbours on network, reachable area calculation, and graph generation
References: Okabe et al (2019) <doi:10.1080/13658810802475491>;
Okabe et al (2012, ISBN:978-0470770818);Baddeley el al (2015, ISBN:9781482210200).

License GPL-2

Encoding UTF-8

Imports spdep (>= 1.1.2), maptools (>= 0.9-5), rgeos (>= 0.5-2), sp (>= 1.3-1), raster (>= 3.5-2), igraph (>= 1.2.4.2), cubature (>= 2.0.4.1), future.apply (>= 1.4.0), methods (>= 1.7.1), ggplot2 (>= 3.3.0), progressr (>= 0.4.0), data.table (>= 1.12.8), SearchTrees (>= 0.5.2), Rcpp (>= 1.0.4.6), Rdpack (>= 2.1.1)

Depends R (>= 3.6)

Suggests future (>= 1.16.0), testthat (>= 3.0.0), knitr (>= 1.28), kableExtra (>= 1.1.0), rmarkdown (>= 2.1), RColorBrewer (>= 1.1-2), classInt (>= 0.4-3), reshape2 (>= 1.4.3), dplyr (>= 0.8.3), rlang (>= 0.4.6), rgdal (>= 1.5-18), rgl (>= 0.107.14), tmap (>= 3.3-1), FNN (>= 1.1.3), smoothr (>= 0.2.2), concaveman (>= 1.1.0), covr (>= 3.5.1)

RoxygenNote 7.1.2

VignetteBuilder knitr

URL https://jeremygelb.github.io/spNetwork/

BugReports https://github.com/JeremyGelb/spNetwork/issues

LinkingTo Rcpp, RcppProgress, RcppArmadillo, BH

SystemRequirements C++17

RdMacros Rdpack

Language en-CA

NeedsCompilation yes

Author Jeremy Gelb [aut, cre] (<https://orcid.org/0000-0002-7114-2714>),
Philippe Apparicio [ctb] (<https://orcid.org/0000-0001-6466-9342>)

Maintainer Jeremy Gelb <jeremy.gelb@ucs.inrs.ca>

Repository CRAN

Date/Publication 2021-10-30 09:50:02 UTC

R topics documented:

`aggregate_points` *Events aggregation*

Description

Function to aggregate points within a radius.

Usage

```
aggregate_points(points, maxdist, weight = "weight")
```

Arguments

`points`	The SpatialPointsDataFrame to contract (must have a weight column)
`maxdist`	The distance to use
`weight`	The name of the column to use as weight (default is "weight"). The values of the aggregated points for this column will be summed. For all the other columns, only the first value is retained.

Details

This function can be used to aggregate points within a radius. This is done by iterating over the features. For each feature, all the features in the radius are found and merged (mean of coordinates). This process is repeated until no more modification is applied.

Value

A new SpatialPointsDataFrame

Examples

```
eventsgpkg <- system.file("extdata", "events.gpkg", package = "spNetwork", mustWork = TRUE)
bike_accidents <- rgdal::readOGR(eventsgpkg,layer="bike_accidents", verbose=FALSE)
bike_accidents$weight <- 1
agg_points <- aggregate_points(bike_accidents, 5)
```

build_graph *Network generation*

Description

Generate an igraph object from a SpatialLinesDataFrame.

Usage

```
build_graph(lines, digits, line_weight, attrs = FALSE)
```

Arguments

lines	A SpatialLinesDataFrame
digits	The number of digits to keep from the coordinates
line_weight	The name of the column giving the weight of the lines
attrs	A boolean indicating if the original lines' attributes should be stored in the final object

Details

This function can be used to generate an undirected graph object (igraph object). It uses the coordinates of the linestrings extremities to create the nodes of the graph. This is why the number of digits in the coordinates is important. Too high a precision (high number of digits) might break some connections.

Value

A list containing the following elements:

- graph: an igraph object
- linelist: the dataframe used to build the graph
- lines: the original SpatialLinesDataFrame
- spvertices: a SpatialPointsDataFrame representing the vertices of the graph
- digits : the number of digits kept for the coordinates

Examples

```
networkgpkg <- system.file("extdata", "networks.gpkg", package = "spNetwork", mustWork = TRUE)
mtl_network <- rgdal::readOGR(networkgpkg,layer="mtl_network", verbose=FALSE)
mtl_network$length <- rgeos::gLength(mtl_network, byid = TRUE)
graph_result <- build_graph(mtl_network, 2, "length", attrs = TRUE)
```

`build_graph_directed` *Directed network generation*

Description

Generate a directed igraph object from a SpatialLinesDataFrame.

Usage

```
build_graph_directed(lines, digits, line_weight, direction, attrs = FALSE)
```

Arguments

`lines`	A SpatialLinesDataFrame
`digits`	The number of digits to keep from the coordinates
`line_weight`	The name of the column giving the weight of the lines
`direction`	A column name indicating authorized travelling direction on lines. if NULL, then all lines can be used in both directions. Must be the name of a column otherwise. The values of the column must be "FT" (From - To), "TF" (To - From) or "Both"
`attrs`	A boolean indicating if the original lines' attributes should be stored in the final object

Details

This function can be used to generate a directed graph object (igraph object). It uses the coordinates of the linestrings extremities to create the nodes of the graph. This is why the number of digits in the coordinates is important. Too high a precision (high number of digits) might break some connections. The column used to indicate directions can only have the following values: "FT" (From-To), "TF" (To-From) and "Both".

Value

A list containing the following elements:

- graph: an igraph object
- linelist: the dataframe used to build the graph
- lines: the original SpatialLinesDataFrame
- spvertices: a SpatialPointsDataFrame representing the vertices of the graph
- digits : the number of digits kept for the coordinates

Examples

```
networkgpkg <- system.file("extdata", "networks.gpkg", package = "spNetwork", mustWork = TRUE)
mtl_network <- rgdal::readOGR(networkgpkg,layer="mtl_network", verbose=FALSE)
mtl_network$length <- rgeos::gLength(mtl_network, byid = TRUE)
mtl_network$direction <- "Both"
mtl_network[6, "direction"] <- "TF"
mtl_network_directed <- lines_direction(mtl_network, "direction")
graph_result <- build_graph_directed(lines = mtl_network_directed,
        digits = 2,
        line_weight = "length",
        direction = "direction",
        attrs = TRUE)
```

---

bw_cvl_calc *Bandwidth selection by Cronie and Van Lieshout's Criterion*

Description

Calculate for multiple bandwidth the Cronie and Van Lieshout's Criterion to select an appropriate bandwidth in a data-driven approach.

Usage

```
bw_cvl_calc(
  bw_range,
  bw_step,
  lines,
  events,
  w,
  kernel_name,
  method,
  diggle_correction = FALSE,
  study_area = NULL,
  max_depth = 15,
  digits = 5,
  tol = 0.1,
  agg = NULL,
  sparse = TRUE,
  grid_shape = c(1, 1),
  sub_sample = 1,
  verbose = TRUE,
  check = TRUE
)
```

Arguments

bw_range	The range of the bandwidths to consider, given as a numeric vector of two values: c(bandwidth_min, bandwidth_max)
bw_step	The step between each bandwidth to calculate given as a float
lines	A SpatialLinesDataFrame representing the underlying network. The geometries must be a SpatialLinesDataFrame (may crash if some geometries are invalid) without MultiLineSring.
events	events A SpatialPointsDataFrame representing the events on the network. The points will be snapped on the network to their closest line.
w	A vector representing the weight of each event
kernel_name	The name of the kernel to use. Must be one of triangle, gaussian, tricube, cosine ,triweight, quartic, epanechnikov or uniform.
method	The method to use when calculating the NKDE, must be one of simple / discontinuous / continuous (see nkde details for more information)
diggle_correction	A Boolean indicating if the correction factor for edge effect must be used.
study_area	A SpatialPolygonsDataFrame or a SpatialPolygon representing the limits of the study area.
max_depth	when using the continuous and discontinuous methods, the calculation time and memory use can go wild if the network has many small edges (area with many of intersections and many events). To avoid it, it is possible to set here a maximum depth. Considering that the kernel is divided at intersections, a value of 10 should yield good estimates in most cases. A larger value can be used without a problem for the discontinuous method. For the continuous method, a larger value will strongly impact calculation speed.
digits	The number of digits to retain from the spatial coordinates. It ensures that topology is good when building the network. Default is 3. Too high a precision (high number of digits) might break some connections
tol	A float indicating the minimum distance between the events and the lines' extremities when adding the point to the network. When points are closer, they are added at the extremity of the lines.
agg	A double indicating if the events must be aggregated within a distance. If NULL, the events are aggregated only by rounding the coordinates.
sparse	A Boolean indicating if sparse or regular matrices should be used by the Rcpp functions. These matrices are used to store edge indices between two nodes in a graph. Regular matrices are faster, but require more memory, in particular with multiprocessing. Sparse matrices are slower (a bit), but require much less memory.
grid_shape	A vector of two values indicating how the study area must be split when performing the calculus. Default is c(1,1) (no split). A finer grid could reduce memory usage and increase speed when a large dataset is used. When using multiprocessing, the work in each grid is dispatched between the workers.
sub_sample	A float between 0 and 1 indicating the percentage of quadra to keep in the calculus. For large datasets, it may be useful to limit the bandwidth evaluation and thus reduce calculation time.

verbose	A Boolean, indicating if the function should print messages about the process.
check	A Boolean indicating if the geometry checks must be run before the operation. This might take some times, but it will ensure that the CRS of the provided objects are valid and identical, and that geometries are valid.

Details

The Cronie and Van Lieshout's Criterion (Cronie and Van Lieshout 2018) find the optimal bandwidth by minimizing the difference between the size of the observation window and the sum of the reciprocal of the estimated kernel density at the events locations. In the network case, the size of the study area is the sum of the length of each line in the network. Thus, it is important to only use the necessary parts of the network.

Value

A dataframe with two columns, one for the bandwidths and the second for the Cronie and Van Lieshout's Criterion.

References

Cronie O, Van Lieshout MNM (2018). "A non-model-based approach to bandwidth selection for kernel estimators of spatial intensity functions." *Biometrika*, **105**(2), 455–462.

Examples

```
networkgpkg <- system.file("extdata", "networks.gpkg", package = "spNetwork", mustWork = TRUE)
eventsgpkg <- system.file("extdata", "events.gpkg", package = "spNetwork", mustWork = TRUE)
mtl_network <- rgdal::readOGR(networkgpkg,layer="mtl_network", verbose=FALSE)
bike_accidents <- rgdal::readOGR(eventsgpkg,layer="bike_accidents", verbose=FALSE)
cv_scores <- bw_cvl_calc(c(200,400),50,
                              mtl_network, bike_accidents,
                              rep(1,nrow(bike_accidents)),
                              "quartic", "discontinuous",
                              diggle_correction = FALSE, study_area = NULL,
                              max_depth = 8,
                              digits=2, tol=0.1, agg=5,
                              sparse=TRUE, grid_shape=c(1,1),
                              sub_sample = 1, verbose=TRUE, check=TRUE)
```

---

bw_cvl_calc.mc *Bandwidth selection by Cronie and Van Lieshout's Criterion (multicore version)*

Description

Calculate for multiple bandiwdths the Cronie and Van Lieshout's Criterion to select an appropriate bandwidth in a data-driven approach. A plan from the package future can be used to split the work across several cores. The different cells generated in accordance with the argument grid_shape are used for the parallelization. So if only one cell is generated (grid_shape = c(1,1)), the function will use only one core. The progress bar displays the progression for the cells.

Usage

```
bw_cvl_calc.mc(
  bw_range,
  bw_step,
  lines,
  events,
  w,
  kernel_name,
  method,
  diggle_correction = FALSE,
  study_area = NULL,
  max_depth = 15,
  digits = 5,
  tol = 0.1,
  agg = NULL,
  sparse = TRUE,
  grid_shape = c(1, 1),
  sub_sample = 1,
  verbose = TRUE,
  check = TRUE
)
```

Arguments

bw_range	The range of the bandwidths to consider, given as a numeric vector of two values: c(bandwidth_min, bandwidth_max)
bw_step	The step between each bandwidth to calculate given as a float
lines	A SpatialLinesDataFrame representing the underlying network. The geometries must be a SpatialLinesDataFrame (may crash if some geometries are invalid) without MultiLineSring.
events	events A SpatialPointsDataFrame representing the events on the network. The points will be snapped on the network to their closest line.
w	A vector representing the weight of each event
kernel_name	The name of the kernel to use. Must be one of triangle, gaussian, tricube, cosine ,triweight, quartic, epanechnikov or uniform.
method	The method to use when calculating the NKDE, must be one of simple / discontinuous / continuous (see nkde details for more information)
diggle_correction	A Boolean indicating if the correction factor for edge effect must be used.

study_area	A SpatialPolygonsDataFrame or a SpatialPolygon representing the limits of the study area.
max_depth	when using the continuous and discontinuous methods, the calculation time and memory use can go wild if the network has many small edges (area with many of intersections and many events). To avoid it, it is possible to set here a maximum depth. Considering that the kernel is divided at intersections, a value of 10 should yield good estimates in most cases. A larger value can be used without a problem for the discontinuous method. For the continuous method, a larger value will strongly impact calculation speed.
digits	The number of digits to retain from the spatial coordinates. It ensures that topology is good when building the network. Default is 3. Too high a precision (high number of digits) might break some connections
tol	A float indicating the minimum distance between the events and the lines' extremities when adding the point to the network. When points are closer, they are added at the extremity of the lines.
agg	A double indicating if the events must be aggregated within a distance. If NULL, the events are aggregated only by rounding the coordinates.
sparse	A Boolean indicating if sparse or regular matrices should be used by the Rcpp functions. These matrices are used to store edge indices between two nodes in a graph. Regular matrices are faster, but require more memory, in particular with multiprocessing. Sparse matrices are slower (a bit), but require much less memory.
grid_shape	A vector of two values indicating how the study area must be split when performing the calculus. Default is c(1,1) (no split). A finer grid could reduce memory usage and increase speed when a large dataset is used. When using multiprocessing, the work in each grid is dispatched between the workers.
sub_sample	A float between 0 and 1 indicating the percentage of quadra to keep in the calculus. For large datasets, it may be useful to limit the bandwidth evaluation and thus reduce calculation time.
verbose	A Boolean, indicating if the function should print messages about the process.
check	A Boolean indicating if the geometry checks must be run before the operation. This might take some times, but it will ensure that the CRS of the provided objects are valid and identical, and that geometries are valid.

Details

For more details, see help(bw_cvl_calc)

Value

A dataframe with two columns, one for the bandwidths and the second for the Cronie and Van Lieshout's Criterion.

Examples

```
networkgpkg <- system.file("extdata", "networks.gpkg", package = "spNetwork", mustWork = TRUE)
```

```
eventsgpkg <- system.file("extdata", "events.gpkg", package = "spNetwork", mustWork = TRUE)
mtl_network <- rgdal::readOGR(networkgpkg,layer="mtl_network", verbose=FALSE)
bike_accidents <- rgdal::readOGR(eventsgpkg,layer="bike_accidents", verbose=FALSE)
future::plan(future::multisession(workers=2))
cv_scores <- bw_cvl_calc(c(200,400),50,
                              mtl_network, bike_accidents,
                              rep(1,nrow(bike_accidents)),
                              "quartic", "discontinuous",
                              diggle_correction = FALSE, study_area = NULL,
                              max_depth = 8,
                              digits=2, tol=0.1, agg=5,
                              sparse=TRUE, grid_shape=c(1,1),
                              sub_sample = 1, verbose=TRUE, check=TRUE)
## make sure any open connections are closed afterward
if (!inherits(future::plan(), "sequential")) future::plan(future::sequential)
```

`bw_cv_likelihood_calc` *Bandwidth selection by likelihood cross validation*

Description

Calculate for multiple bandwidth the cross validation likelihood to select an appropriate bandwidth in a data-driven approach

Usage

```
bw_cv_likelihood_calc(
  bw_range,
  bw_step,
  lines,
  events,
  w,
  kernel_name,
  method,
  diggle_correction = FALSE,
  study_area = NULL,
  max_depth = 15,
  digits = 5,
  tol = 0.1,
  agg = NULL,
  sparse = TRUE,
  grid_shape = c(1, 1),
  sub_sample = 1,
  verbose = TRUE,
  check = TRUE
)
```

Arguments

bw_range	The range of the bandwidths to consider, given as a numeric vector of two values: c(bandwidth_min, bandwidth_max)
bw_step	The step between each bandwidth to calculate given as a float
lines	A SpatialLinesDataFrame representing the underlying network. The geometries must be a SpatialLinesDataFrame (may crash if some geometries are invalid) without MultiLineSring.
events	events A SpatialPointsDataFrame representing the events on the network. The points will be snapped on the network to their closest line.
w	A vector representing the weight of each event
kernel_name	The name of the kernel to use. Must be one of triangle, gaussian, tricube, cosine ,triweight, quartic, epanechnikov or uniform.
method	The method to use when calculating the NKDE, must be one of simple / discontinuous / continuous (see nkde details for more information)
diggle_correction	A Boolean indicating if the correction factor for edge effect must be used.
study_area	A SpatialPolygonsDataFrame or a SpatialPolygon representing the limits of the study area.
max_depth	when using the continuous and discontinuous methods, the calculation time and memory use can go wild if the network has many small edges (area with many of intersections and many events). To avoid it, it is possible to set here a maximum depth. Considering that the kernel is divided at intersections, a value of 10 should yield good estimates in most cases. A larger value can be used without a problem for the discontinuous method. For the continuous method, a larger value will strongly impact calculation speed.
digits	The number of digits to retain from the spatial coordinates. It ensures that topology is good when building the network. Default is 3. Too high a precision (high number of digits) might break some connections
tol	A float indicating the minimum distance between the events and the lines' extremities when adding the point to the network. When points are closer, they are added at the extremity of the lines.
agg	A double indicating if the events must be aggregated within a distance. If NULL, the events are aggregated only by rounding the coordinates.
sparse	A Boolean indicating if sparse or regular matrices should be used by the Rcpp functions. These matrices are used to store edge indices between two nodes in a graph. Regular matrices are faster, but require more memory, in particular with multiprocessing. Sparse matrices are slower (a bit), but require much less memory.
grid_shape	A vector of two values indicating how the study area must be split when performing the calculus. Default is c(1,1) (no split). A finer grid could reduce memory usage and increase speed when a large dataset is used. When using multiprocessing, the work in each grid is dispatched between the workers.
sub_sample	A float between 0 and 1 indicating the percentage of quadra to keep in the calculus. For large datasets, it may be useful to limit the bandwidth evaluation and thus reduce calculation time.

verbose — A Boolean, indicating if the function should print messages about process.

check — A Boolean indicating if the geometry checks must be run before the operation. This might take some times, but it will ensure that the CRS of the provided objects are valid and identical, and that geometries are valid.

Details

The function calculates the likelihood cross validation score for several bandwidths in order to find the most appropriate one. The general idea is to find the bandwidth that would produce the most similar results if one event was removed from the dataset (leave one out cross validation). We use here the shortcut formula as described by the package spatstat (Baddeley et al. 2021).

LCV(h) = sum[i] log(lambda[-i](x[i]))

Where the sum is taken for all events x[i] and where lambda[-i](x[i]) is the leave-one-out kernel estimate at x[i] for a bandwidth h. A lower value indicates a better bandwidth.

Value

A dataframe with two columns, one for the bandwidths and the second for the cross validation score (the lower the better).

References

Baddeley A, Turner R, Rubak E (2021). *spatstat: Spatial Point Pattern Analysis, Model-Fitting, Simulation, Tests*. R package version 2.1-0, https://CRAN.R-project.org/package=spatstat.

Examples

```
networkgpkg <- system.file("extdata", "networks.gpkg", package = "spNetwork", mustWork = TRUE)
eventsgpkg <- system.file("extdata", "events.gpkg", package = "spNetwork", mustWork = TRUE)
mtl_network <- rgdal::readOGR(networkgpkg,layer="mtl_network", verbose=FALSE)
bike_accidents <- rgdal::readOGR(eventsgpkg,layer="bike_accidents", verbose=FALSE)
cv_scores <- bw_cv_likelihood_calc(c(200,800),50,
                                   mtl_network, bike_accidents,
                                   rep(1,nrow(bike_accidents)),
                                   "quartic", "simple",
                                   diggle_correction = FALSE, study_area = NULL,
                                   max_depth = 8,
                                   digits=2, tol=0.1, agg=5,
                                   sparse=TRUE, grid_shape=c(1,1),
                                   sub_sample = 1, verbose=TRUE, check=TRUE)
```

`bw_cv_likelihood_calc.mc` *Bandwidth selection by likelihood cross validation (multicore version)*

Description

Calculate for multiple bandiwdths the cross validation likelihood to select an appropriate bandwidth in a data-driven approach. A plan from the package future can be used to split the work across several cores. The different cells generated in accordance with the argument grid_shape are used for the parallelization. So if only one cell is generated (grid_shape = c(1,1)), the function will use only one core. The progress bar displays the progression for the cells.

Usage

```
bw_cv_likelihood_calc.mc(
  bw_range,
  bw_step,
  lines,
  events,
  w,
  kernel_name,
  method,
  diggle_correction = FALSE,
  study_area = NULL,
  max_depth = 15,
  digits = 5,
  tol = 0.1,
  agg = NULL,
  sparse = TRUE,
  grid_shape = c(1, 1),
  sub_sample = 1,
  verbose = TRUE,
  check = TRUE
)
```

Arguments

`bw_range`	The range of the bandwidths to consider, given as a numeric vector of two values: c(bandwidth_min, bandwidth_max)
`bw_step`	The step between each bandwidth to calculate given as a float
`lines`	A SpatialLinesDataFrame representing the underlying network. The geometries must be a SpatialLinesDataFrame (may crash if some geometries are invalid) without MultiLineSring.
`events`	events A SpatialPointsDataFrame representing the events on the network. The points will be snapped on the network to their closest line.
`w`	A vector representing the weight of each event

`kernel_name` The name of the kernel to use. Must be one of triangle, gaussian, tricube, cosine ,triweight, quartic, epanechnikov or uniform.

`method` The method to use when calculating the NKDE, must be one of simple / discontinuous / continuous (see nkde details for more information)

`diggle_correction` A Boolean indicating if the correction factor for edge effect must be used.

`study_area` A SpatialPolygonsDataFrame or a SpatialPolygon representing the limits of the study area.

`max_depth` when using the continuous and discontinuous methods, the calculation time and memory use can go wild if the network has many small edges (area with many of intersections and many events). To avoid it, it is possible to set here a maximum depth. Considering that the kernel is divided at intersections, a value of 10 should yield good estimates in most cases. A larger value can be used without a problem for the discontinuous method. For the continuous method, a larger value will strongly impact calculation speed.

`digits` The number of digits to retain from the spatial coordinates. It ensures that topology is good when building the network. Default is 3. Too high a precision (high number of digits) might break some connections

`tol` A float indicating the minimum distance between the events and the lines' extremities when adding the point to the network. When points are closer, they are added at the extremity of the lines.

`agg` A double indicating if the events must be aggregated within a distance. If NULL, the events are aggregated only by rounding the coordinates.

`sparse` A Boolean indicating if sparse or regular matrices should be used by the Rcpp functions. These matrices are used to store edge indices between two nodes in a graph. Regular matrices are faster, but require more memory, in particular with multiprocessing. Sparse matrices are slower (a bit), but require much less memory.

`grid_shape` A vector of two values indicating how the study area must be split when performing the calculus. Default is c(1,1) (no split). A finer grid could reduce memory usage and increase speed when a large dataset is used. When using multiprocessing, the work in each grid is dispatched between the workers.

`sub_sample` A float between 0 and 1 indicating the percentage of quadra to keep in the calculus. For large datasets, it may be useful to limit the bandwidth evaluation and thus reduce calculation time.

`verbose` A Boolean, indicating if the function should print messages about process.

`check` A Boolean indicating if the geometry checks must be run before the operation. This might take some times, but it will ensure that the CRS of the provided objects are valid and identical, and that geometries are valid.

Details

For more details, see help(bw_cv_likelihood_calc)

Value

A dataframe with two columns, one for the bandwidths and the second for the cross validation score (the lower the better).

Examples

```
networkgpkg <- system.file("extdata", "networks.gpkg", package = "spNetwork", mustWork = TRUE)
eventsgpkg <- system.file("extdata", "events.gpkg", package = "spNetwork", mustWork = TRUE)
mtl_network <- rgdal::readOGR(networkgpkg,layer="mtl_network", verbose=FALSE)
bike_accidents <- rgdal::readOGR(eventsgpkg,layer="bike_accidents", verbose=FALSE)
future::plan(future::multisession(workers=2))
cv_scores <- bw_cv_likelihood_calc.mc(c(200,800),50,
                                mtl_network, bike_accidents,
                                rep(1,nrow(bike_accidents)),
                                "quartic", "simple",
                                diggle_correction = FALSE, study_area = NULL,
                                max_depth = 8,
                                digits=2, tol=0.1, agg=5,
                                sparse=TRUE, grid_shape=c(1,1),
                                sub_sample = 1, verbose=TRUE, check=TRUE)
## make sure any open connections are closed afterward
if (!inherits(future::plan(), "sequential")) future::plan(future::sequential)
```

---

calc_isochrones *Isochrones calculation*

Description

Calculate isochrones on a network

Usage

```
calc_isochrones(
  lines,
  dists,
  start_points,
  mindist = 1,
  weight = NULL,
  direction = NULL
)
```

Arguments

lines	A SpatialLinesDataFrame representing the edges of the network
dists	A vector of the size of the desired isochrones
start_points	A SpatialPointsDataFrame representing the starting points if the isochrones

mindist	The minimum distance between two points. When two points are too close, they might end up snapped at the same location on a line. Default is 1.
weight	The name of the column in lines to use an edge weight. If NULL, the geographical length is used. Note that if lines are split during the network creation, the weight column is recalculated proportionally to the new lines length.
direction	The name of the column indicating authorized travelling direction on lines. if NULL, then all lines can be used in both directions (undirected). The values of the column must be "FT" (From - To), "TF" (To - From) or "Both".

Details

An isochrone is the set of reachable lines around a node in a network within a specified distance (or time). This function perform dynamic segmentation to return the part of the edges reached and not only the fully covered edges. Several start points and several distances can be given. The network can also be directed. The lines returned by the function are the most accurate representation of the isochrones. However, if polygons are required for mapping, the vignette "Calculating isochrones" shows how to create smooth polygons from the returned sets of lines.

Value

A SpatialLinesDataFrame representing the isochrones with the following columns

- point_id: the index of the point at the centre of the isochrone
- distance: the size of the isochrone

Examples

```
library(sp)
# creating a simple network
wkt_lines <- c(
 "LINESTRING (0.0 0.0, 5.0 0.0)",
 "LINESTRING (0.0 -5.0, 5.0 -5.0)",
 "LINESTRING (5.0 0.0, 5.0 5.0)",
 "LINESTRING (5.0 -5.0, 5.0 -10.0)",
 "LINESTRING (5.0 0.0, 5.0 -5.0)",
 "LINESTRING (5.0 0.0, 10.0 0.0)",
 "LINESTRING (5.0 -5.0, 10.0 -5.0)",
 "LINESTRING (10.0 0, 10.0 -5.0)",
 "LINESTRING (10.0 -10.0, 10.0 -5.0)",
 "LINESTRING (15.0 -5.0, 10.0 -5.0)",
 "LINESTRING (10.0 0.0, 15.0 0.0)",
 "LINESTRING (10.0 0.0, 10.0 5.0)")

linesdf <- data.frame(wkt = wkt_lines,
                      id = paste("l",1:length(wkt_lines),sep=""))

geoms <- do.call(rbind,lapply(1:nrow(linesdf),function(i){
 txt <- as.character(linesdf[i,]$wkt)
 geom <- rgeos::readWKT(txt,id=i)
 return(geom)
}))
```

```
lines <- SpatialLinesDataFrame(geoms, linesdf, match.ID = FALSE)

# and the definition of the starting point
start_points <- data.frame(x=c(5),
                           y=c(-2.5))
coordinates(start_points) <- cbind(start_points$x,start_points$y)

# setting the directions

lines$direction <- "Both"
lines[6,"direction"] <- "TF"

isochrones <- calc_isochrones(lines, dists = c(10,12),
                              start_points = start_points,
                              direction = "direction")
```

`closest_points` *Find closest points*

Description

Solve the nearest neighbour problem for two SpatialPointsDataFrame. This is a simple wrap-up of the FNN::knnx.index function

Usage

```
closest_points(origins, targets)
```

Arguments

`origins`	a SpatialPointsDataFrame
`targets`	a SpatialPointsDataFrame

Value

for each origin point, the index of the nearest target point

Examples

```
#This is an internal function, no example provided
eventsgpkg <- system.file("extdata", "events.gpkg", package = "spNetwork", mustWork = TRUE)
mtl_libraries <- rgdal::readOGR(eventsgpkg,layer="mtl_libraries", verbose=FALSE)
mtl_theatres <- rgdal::readOGR(eventsgpkg,layer="mtl_theatres", verbose=FALSE)
close_libs <- closest_points(mtl_theatres, mtl_libraries)
```

cosine_kernel *Cosine kernel*

Description

Function implementing the cosine kernel.

Usage

```
cosine_kernel(d, bw)
```

Arguments

d	The distance from the event
bw	The bandwidth used for the kernel

Value

The estimated density

Examples

```
#This is an internal function, no example provided
```

cross_kfunctions *Network cross k and g functions (maturing)*

Description

Calculate the cross k and g functions for a set of points on a network. (maturing)

Usage

```
cross_kfunctions(
  lines,
  pointsA,
  pointsB,
  start,
  end,
  step,
  width,
  nsim,
  conf_int = 0.05,
  digits = 2,
  tol = 0.1,
  resolution = NULL,
```

```
  agg = NULL,
  verbose = TRUE
)
```

Arguments

lines	A SpatialLinesDataFrame representing the underlying network. The geometries must be a SpatialLinesDataFrame (may crash if some geometries are invalid) without MultiLineSring
pointsA	A SpatialPointsDataFrame representing the points to which the distances are calculated.
pointsB	A SpatialPointsDataFrame representing the points from which the distances are calculated.
start	A double, the lowest distance used to evaluate the k and g functions
end	A double, the highest distance used to evaluate the k and g functions
step	A double, the step between two evaluations of the k and g function. start, end and step are used to create a vector of distances with the function seq
width	The width of each donut for the g-function. Half of the width is applied on both sides of the considered distance
nsim	An integer indicating the number of Monte Carlo simulations to perform for inference
conf_int	A double indicating the width confidence interval (default = 0.05) calculated on the Monte Carlo simulations
digits	An integer indicating the number of digits to retain from the spatial coordinates
tol	When adding the points to the network, specify the minimum distance between these points and the lines' extremities. When points are closer, they are added at the extremity of the lines
resolution	When simulating random points on the network, selecting a resolution will reduce greatly the calculation time. When resolution is null the random points can occur everywhere on the graph. If a value is specified, the edges are split according to this value and the random points can only be vertices on the new network
agg	A double indicating if the events must be aggregated within a distance. If NULL, the events are aggregated only by rounding the coordinates
verbose	A Boolean indicating if progress messages should be displayed

Details

The cross k-function is a method to characterize the dispersion of a set of points (A) around a second set of points (B). For each point in B, the numbers of other points in A in subsequent radii are calculated. This empirical cross k-function can be more or less clustered than a cross k-function obtained if the points in A were randomly located around points in B. In a network, the network distance is used instead of the Euclidean distance. This function uses Monte Carlo simulations to assess if the points are clustered or dispersed and gives the results as a line plot. If the line of the observed cross k-function is higher than the shaded area representing the values of the simulations,

then the points in A are more clustered around points in B than what we can expect from randomness and vice-versa. The function also calculates the cross g-function, a modified version of the cross k-function using rings instead of disks. The width of the ring must be chosen. The main interest is to avoid the cumulative effect of the classical k-function. Note that the cross k-function of points A around B is not necessarily the same as the cross k-function of points B around A. This function is maturing, it works as expected (unit tests) but will probably be modified in the future releases (gain speed, advanced features, etc.).

Value

A list with the following values :

- plotk A ggplot2 object representing the values of the cross k-function
- plotg A ggplot2 object representing the values of the cross g-function
- values A DataFrame with the values used to build the plots

Examples

```
networkgpkg <- system.file("extdata", "networks.gpkg", package = "spNetwork", mustWork = TRUE)
eventsgpkg <- system.file("extdata", "events.gpkg", package = "spNetwork", mustWork = TRUE)
main_network_mtl <- rgdal::readOGR(networkgpkg,layer="main_network_mtl", verbose=FALSE)
mtl_libraries <- rgdal::readOGR(eventsgpkg,layer="mtl_libraries", verbose=FALSE)
mtl_theatres <- rgdal::readOGR(eventsgpkg,layer="mtl_theatres", verbose=FALSE)
result <- cross_kfunctions(main_network_mtl, mtl_theatres, mtl_libraries,
                           start = 0, end = 2500, step = 10, width = 250,
                           nsim = 50, conf_int = 0.05, digits = 2,
                           tol = 0.1, agg = NULL, verbose = FALSE)
```

---

cross_kfunctions.mc *Network cross k and g functions (multicore, maturing)*

Description

Calculate the cross k and g functions for a set of points on a network with multicore support. (maturing)

Usage

```
cross_kfunctions.mc(
  lines,
  pointsA,
  pointsB,
  start,
  end,
  step,
```

```
  width,
  nsim,
  conf_int = 0.05,
  digits = 2,
  tol = 0.1,
  resolution = NULL,
  agg = NULL,
  verbose = TRUE
)
```

Arguments

lines	A SpatialLinesDataFrame representing the underlying network. The geometries must be a SpatialLinesDataFrame (may crash if some geometries are invalid) without MultiLineSring
pointsA	A SpatialPointsDataFrame representing the points to which the distances are calculated.
pointsB	A SpatialPointsDataFrame representing the points from which the distances are calculated.
start	A double, the lowest distance used to evaluate the k and g functions
end	A double, the highest distance used to evaluate the k and g functions
step	A double, the step between two evaluations of the k and g function. start, end and step are used to create a vector of distances with the function seq
width	The width of each donut for the g-function. Half of the width is applied on both sides of the considered distance
nsim	An integer indicating the number of Monte Carlo simulations to perform for inference
conf_int	A double indicating the width confidence interval (default = 0.05) calculated on the Monte Carlo simulations
digits	An integer indicating the number of digits to retain from the spatial coordinates
tol	When adding the points to the network, specify the minimum distance between these points and the lines' extremities. When points are closer, they are added at the extremity of the lines
resolution	When simulating random points on the network, selecting a resolution will reduce greatly the calculation time. When resolution is null the random points can occur everywhere on the graph. If a value is specified, the edges are split according to this value and the random points can only be vertices on the new network
agg	A double indicating if the events must be aggregated within a distance. If NULL, the events are aggregated only by rounding the coordinates
verbose	A Boolean indicating if progress messages should be displayed

Value

A list with the following values :

- plotk A ggplot2 object representing the values of the cross k-function
- plotg A ggplot2 object representing the values of the cross g-function
- values A DataFrame with the values used to build the plots

Examples

```
networkgpkg <- system.file("extdata", "networks.gpkg", package = "spNetwork", mustWork = TRUE)
eventsgpkg <- system.file("extdata", "events.gpkg", package = "spNetwork", mustWork = TRUE)
main_network_mtl <- rgdal::readOGR(networkgpkg,layer="main_network_mtl", verbose=FALSE)
mtl_libraries <- rgdal::readOGR(eventsgpkg,layer="mtl_libraries", verbose=FALSE)
mtl_theatres <- rgdal::readOGR(eventsgpkg,layer="mtl_theatres", verbose=FALSE)
future::plan(future::multisession(workers=2))
result <- cross_kfunctions.mc(main_network_mtl, mtl_libraries, mtl_theatres,
                           start = 0, end = 2500, step = 10, width = 250,
                           nsim = 50, conf_int = 0.05, digits = 2,
                           tol = 0.1, agg = NULL, verbose = TRUE)
## make sure any open connections are closed afterward
if (!inherits(future::plan(), "sequential")) future::plan(future::sequential)
```

---

epanechnikov_kernel *Epanechnikov kernel*

Description

Function implementing the epanechnikov kernel.

Usage

```
epanechnikov_kernel(d, bw)
```

Arguments

d	The distance from the event
bw	The bandwidth used for the kernel

Value

The estimated density

Examples

```
#This is an internal function, no example provided
```

gaussian_kernel *Gaussian kernel*

Description

Function implementing the gaussian kernel.

Usage

```
gaussian_kernel(d, bw)
```

Arguments

d	The distance from the event
bw	The bandwidth used for the kernel

Value

The estimated density

Examples

```
#This is an internal function, no example provided
```

gaussian_kernel_scaled *Scaled gaussian kernel*

Description

Function implementing the scaled gaussian kernel.

Usage

```
gaussian_kernel_scaled(d, bw)
```

Arguments

d	The distance from the event
bw	The bandwidth used for the kernel

Value

The estimated density

Examples

```
#This is an internal function, no example provided
```

graph_checking *Topological error*

Description

A utility function to find topological errors in a network.

Usage

```
graph_checking(lines, digits, max_search = 5, tol = 0.1)
```

Arguments

lines	A SpatialLinesDataFrame representing the network
digits	An integer indicating the number of digits to retain for coordinates
max_search	The maximum number of nearest neighbour to search to find close_nodes
tol	The minimum distance expected between two nodes. Under that values nodes are considered as too close and are returned in the results.

Details

This function can be used to check for three common problems in networks: disconnected components, dangle nodes and close nodes. When a network has disconnected components, this means that several unconnected graphs are composing the overall network. This can be caused by topological errors in the dataset. Dangle nodes are nodes connected to only one other node. This type of node can be normal at the border of a network, but can also be caused by topological errors. Close nodes are nodes that are not coincident, but so close that they probably should be coincident.

Value

A list with three elements. The first is a SpatialPointsDataFrame indicating for each node of the network to which component it belongs. The second is a SpatialPointsDataFrame with nodes that are too close one of each other. The second is a SpatialPointsDataFrame with the dangle nodes of the network.

Examples

```
networkgpkg <- system.file("extdata", "networks.gpkg", package = "spNetwork", mustWork = TRUE)
mtl_network <- rgdal::readOGR(networkgpkg,layer="mtl_network", verbose=FALSE)
topo_errors <- graph_checking(mtl_network, 2)
```

`kfunctions` *Network k and g functions (maturing)*

Description

Calculate the k and g functions for a set of points on a network (maturing).

Usage

```
kfunctions(
  lines,
  points,
  start,
  end,
  step,
  width,
  nsim,
  conf_int = 0.05,
  digits = 2,
  tol = 0.1,
  resolution = NULL,
  agg = NULL,
  verbose = TRUE
)
```

Arguments

`lines`	A SpatialLinesDataFrame representing the underlying network. The geometries must be a SpatialLinesDataFrame (may crash if some geometries are invalid) without MultiLineSring
`points`	A SpatialPointsDataFrame representing the points on the network. These points will be snapped on their nearest line
`start`	A double, the lowest distance used to evaluate the k and g functions
`end`	A double, the highest distance used to evaluate the k and g functions
`step`	A double, the step between two evaluations of the k and g function. start, end and step are used to create a vector of distances with the function seq
`width`	The width of each donut for the g-function. Half of the width is applied on both sides of the considered distance
`nsim`	An integer indicating the number of Monte Carlo simulations to perform for inference
`conf_int`	A double indicating the width confidence interval (default = 0.05) calculated on the Monte Carlo simulations
`digits`	An integer indicating the number of digits to retain from the spatial coordinates

tol	When adding the points to the network, specify the minimum distance between these points and the lines' extremities. When points are closer, they are added at the extremity of the lines
resolution	When simulating random points on the network, selecting a resolution will reduce greatly the calculation time. When resolution is null the random points can occur everywhere on the graph. If a value is specified, the edges are split according to this value and the random points can only be vertices on the new network
agg	A double indicating if the events must be aggregated within a distance. If NULL, the events are aggregated only by rounding the coordinates
verbose	A Boolean indicating if progress messages should be displayed

Details

The k-function is a method to characterize the dispersion of a set of points. For each point, the numbers of other points in subsequent radii are calculated. This empirical k-function can be more or less clustered than a k-function obtained if the points were randomly located in space. In a network, the network distance is used instead of the Euclidean distance. This function uses Monte Carlo simulations to assess if the points are clustered or dispersed, and gives the results as a line plot. If the line of the observed k-function is higher than the shaded area representing the values of the simulations, then the points are more clustered than what we can expect from randomness and vice-versa. The function also calculates the g-function, a modified version of the k-function using rings instead of disks. The width of the ring must be chosen. The main interest is to avoid the cumulative effect of the classical k-function. This function is maturing, it works as expected (unit tests) but will probably be modified in the future releases (gain speed, advanced features, etc.).

Value

A list with the following values :

- plotk A ggplot2 object representing the values of the k-function
- plotg A ggplot2 object representing the values of the g-function
- values A DataFrame with the values used to build the plots

Examples

```
networkgpkg <- system.file("extdata", "networks.gpkg", package = "spNetwork", mustWork = TRUE)
eventsgpkg <- system.file("extdata", "events.gpkg", package = "spNetwork", mustWork = TRUE)
main_network_mtl <- rgdal::readOGR(networkgpkg,layer="main_network_mtl", verbose=FALSE)
mtl_libraries <- rgdal::readOGR(eventsgpkg,layer="mtl_libraries", verbose=FALSE)
result <- kfunctions(main_network_mtl, mtl_libraries,
     start = 0, end = 2500, step = 10,
     width = 200, nsim = 50,
     conf_int = 0.05, tol = 0.1, agg = NULL,
     verbose = FALSE)
```

kfunctions.mc *Network k and g functions (multicore, maturing)*

Description

Calculate the k and g functions for a set of points on a network with multicore support. For details, please see the function kfunctions. (maturing)

Usage

```
kfunctions.mc(
  lines,
  points,
  start,
  end,
  step,
  width,
  nsim,
  conf_int = 0.05,
  digits = 2,
  tol = 0.1,
  resolution = 50,
  agg = NULL,
  verbose = TRUE
)
```

Arguments

lines	A SpatialLinesDataFrame representing the underlying network. The geometries must be a SpatialLinesDataFrame (may crash if some geometries are invalid) without MultiLineSring
points	A SpatialPointsDataFrame representing the points on the network. These points will be snapped on their nearest line
start	A double, the lowest distance used to evaluate the k and g functions
end	A double, the highest distance used to evaluate the k and g functions
step	A double, the step between two evaluations of the k and g function. start, end and step are used to create a vector of distances with the function seq
width	The width of each donut for the g-function. Half of the width is applied on both sides of the considered distance
nsim	An integer indicating the number of Monte Carlo simulations to perform for inference
conf_int	A double indicating the width confidence interval (default = 0.05) calculated on the Monte Carlo simulations
digits	An integer indicating the number of digits to retain from the spatial coordinates

tol	When adding the points to the network, specify the minimum distance between these points and the lines' extremities. When points are closer, they are added at the extremity of the lines
resolution	When simulating random points on the network, selecting a resolution will reduce greatly the calculation time. When resolution is null the random points can occur everywhere on the graph. If a value is specified, the edges are split according to this value and the random points can only be vertices on the new network
agg	A double indicating if the events must be aggregated within a distance. If NULL, the events are aggregated only by rounding the coordinates
verbose	A Boolean indicating if progress messages should be displayed

Details

For details, please look at the function kfunctions.

Value

A list with the following values :

- plotk A ggplot2 object representing the values of the k-function
- plotg A ggplot2 object representing the values of the g-function
- values A DataFrame with the values used to build the plots

Examples

```
networkgpkg <- system.file("extdata", "networks.gpkg", package = "spNetwork", mustWork = TRUE)
eventsgpkg <- system.file("extdata", "events.gpkg", package = "spNetwork", mustWork = TRUE)
main_network_mtl <- rgdal::readOGR(networkgpkg,layer="main_network_mtl", verbose=FALSE)
mtl_libraries <- rgdal::readOGR(eventsgpkg,layer="mtl_libraries", verbose=FALSE)
future::plan(future::multisession(workers=2))
result <- kfunctions.mc(main_network_mtl, mtl_libraries,
     start = 0, end = 2500, step = 10,
     width = 200, nsim = 50,
     conf_int = 0.05, tol = 0.1, agg = NULL,
     verbose = FALSE)
## make sure any open connections are closed afterward
if (!inherits(future::plan(), "sequential")) future::plan(future::sequential)
```

lines_center *Centre points of lines*

Description

Generate a SpatialPointsDataFrame with points at the centre of lines. The length of the lines is used to determine its centre.

Usage

```
lines_center(lines)
```

Arguments

lines	The SpatialLinesDataframe to use

Value

An object of class SpatialPointsDataFrame (package sp)

Examples

```
networkgpkg <- system.file("extdata", "networks.gpkg", package = "spNetwork", mustWork = TRUE)
mtl_network <- rgdal::readOGR(networkgpkg,layer="mtl_network", verbose=FALSE)
centers <- lines_center(mtl_network)
```

lines_direction *Unify lines direction*

Description

A function to deal with the directions of lines. It ensures that only From-To situation are present by reverting To-From lines. For the lines labelled as To-From, the order of their vertices is reverted.

Usage

```
lines_direction(lines, field)
```

Arguments

lines	A SpatialLinesDataFrame
field	Indicate a field giving information about authorized travelling direction on lines. if NULL, then all lines can be used in both directions. Must be the name of a column otherwise. The values of the column must be "FT" (From - To), "TF" (To - From) or "Both".

Value

A SpatialLinesDataFrame

Examples

```
networkgpkg <- system.file("extdata", "networks.gpkg", package = "spNetwork", mustWork = TRUE)
mtl_network <- rgdal::readOGR(networkgpkg,layer="mtl_network", verbose=FALSE)
mtl_network$length <- rgeos::gLength(mtl_network, byid = TRUE)
mtl_network$direction <- "Both"
mtl_network[6, "direction"] <- "TF"
mtl_network_directed <- lines_direction(mtl_network, "direction")
```

---

lines_points_along *Points along lines*

Description

Generate points along the lines of a SpatialLinesDataFrame.

Usage

```
lines_points_along(lines, dist)
```

Arguments

lines	The SpatialLinesDataframe to use
dist	The distance between the points along the lines

Value

An object of class SpatialLinesDataframe (package sp)

Examples

```
networkgpkg <- system.file("extdata", "networks.gpkg", package = "spNetwork", mustWork = TRUE)
mtl_network <- rgdal::readOGR(networkgpkg,layer="mtl_network", verbose=FALSE)
new_pts <- lines_points_along(mtl_network,50)
```

lixelize_lines *Cut lines into lixels*

Description

Cut a SpatialLines object into lixels with a specified minimal distance may fail if the line geometries are self intersecting.

Usage

```
lixelize_lines(lines, lx_length, mindist = NULL)
```

Arguments

lines	The SpatialLinesDataframe to modify
lx_length	The length of a lixel
mindist	The minimum length of a lixel. After cut, if the length of the final lixel is shorter than the minimum distance, then it is added to the previous lixel. if NULL, then mindist = maxdist/10. Note that the segments that are already shorter than the minimum distance are not modified.

Value

An object of class SpatialLinesDataFrame (package sp)

Examples

```
networkgpkg <- system.file("extdata", "networks.gpkg", package = "spNetwork", mustWork = TRUE)
mtl_network <- rgdal::readOGR(networkgpkg,layer="mtl_network", verbose=FALSE)
lixels <- lixelize_lines(mtl_network,150,50)
```

lixelize_lines.mc *Cut lines into lixels (multicore)*

Description

Cut a SpatialLines object into lixels with a specified minimal distance may fail if the line geometries are self intersecting with multicore support.

Usage

```
lixelize_lines.mc(
  lines,
  lx_length,
  mindist = NULL,
  verbose = TRUE,
  chunk_size = 100
)
```

Arguments

lines	The SpatialLinesDataframe to modify
lx_length	The length of a lixel
mindist	The minimum length of a lixel. After cut, if the length of the final lixel is shorter than the minimum distance, then it is added to the previous lixel. If NULL, then mindist = maxdist/10
verbose	A Boolean indicating if a progress bar must be displayed
chunk_size	The size of a chunk used for multiprocessing. Default is 100.

Value

An object of class SpatialLinesDataFrame (package sp)

Examples

```
networkgpkg <- system.file("extdata", "networks.gpkg", package = "spNetwork", mustWork = TRUE)
mtl_network <- rgdal::readOGR(networkgpkg,layer="mtl_network", verbose=FALSE)
future::plan(future::multisession(workers=2))
lixels <- lixelize_lines.mc(mtl_network,150,50)
## make sure any open connections are closed afterward
if (!inherits(future::plan(), "sequential")){
future::plan(future::sequential)
}
```

---

network_knn *K-nearest points on network*

Description

Calculate the K-nearest points for a set of points on a network.

Usage

```
network_knn(
  origins,
  lines,
  k,
  destinations = NULL,
  maxdistance = 0,
  snap_dist = Inf,
  line_weight = "length",
  direction = NULL,
  grid_shape = c(1, 1),
  verbose = FALSE,
  digits = 3,
  tol = 0.1
)
```

Arguments

origins	A SpatialPointsDataFrame, for each point, its k nearest neighbours will be found on the network.
lines	A SpatialLinesDataFrame representing the underlying network
k	An integer indicating the number of neighbours to find.
destinations	A SpatialPointsDataFrame, might be used if the neighbours must be found in a separate set of points NULL if the neighbours must be found in origins.
maxdistance	The maximum distance between two observations to consider them as neighbours. It is useful only if a grid is used, a lower value will reduce calculating time, but one must be sure that the k nearest neighbours are within this radius. Otherwise NAs will be present in the results.
snap_dist	The maximum distance to snap the start and end points on the network.
line_weight	The weighting to use for lines. Default is "length" (the geographical length), but can be the name of a column. The value is considered proportional to the geographical length of the lines.
direction	The name of a column indicating authorized traveling direction on lines. if NULL, then all lines can be used in both directions. Must be the name of a column otherwise. The values of the column must be "FT" (From - To), "TF" (To - From) or "Both".
grid_shape	A vector of length 2 indicating the shape of the grid to use for splitting the dataset. Default is c(1,1), so all the calculation is done in one go. It might be necessary to split it if the dataset is large.
verbose	A Boolean indicating if the function should print its progress
digits	The number of digits to retain from the spatial coordinates (simplification used to reduce risk of topological error)
tol	A float indicating the minimum distance between the points and the lines' extremities when adding the point to the network. When points are closer, they are added at the extremity of the lines.

Value

A list with two matrices, one with the index of the neighbours and one with the distances.

Examples

```
networkgpkg <- system.file("extdata", "networks.gpkg", package = "spNetwork", mustWork = TRUE)
eventsgpkg <- system.file("extdata", "events.gpkg", package = "spNetwork", mustWork = TRUE)
main_network_mtl <- rgdal::readOGR(networkgpkg,layer="main_network_mtl", verbose=FALSE)
 mtl_libraries <- rgdal::readOGR(eventsgpkg,layer="mtl_libraries", verbose=FALSE)
 results <- network_knn(mtl_libraries, main_network_mtl,
     k = 3, maxdistance = 1000, line_weight = "length",
     grid_shape=c(1,1), verbose = FALSE)
```

---

network_knn.mc *K-nearest points on network (multicore version)*

Description

Calculate the K-nearest points for a set of points on a network with multicore support.

Usage

```
network_knn.mc(
  origins,
  lines,
  k,
  destinations = NULL,
  maxdistance = 0,
  snap_dist = Inf,
  line_weight = "length",
  direction = NULL,
  grid_shape = c(1, 1),
  verbose = FALSE,
  digits = 3,
  tol = 0.1
)
```

Arguments

origins	A SpatialPointsDataFrame, for each point, its k nearest neighbours will be found on the network.
lines	A SpatialLinesDataFrame representing the underlying network
k	An integer indicating the number of neighbours to find.
destinations	A SpatialPointsDataFrame, might be used if the neighbours must be found in a separate set of points NULL if the neighbours must be found in origins.

maxdistance	The maximum distance between two observations to consider them as neighbours. It is useful only if a grid is used, a lower value will reduce calculating time, but one must be sure that the k nearest neighbours are within this radius. Otherwise NAs will be present in the results.
snap_dist	The maximum distance to snap the start and end points on the network.
line_weight	The weighting to use for lines. Default is "length" (the geographical length), but can be the name of a column. The value is considered proportional to the geographical length of the lines.
direction	The name of a column indicating authorized traveling direction on lines. if NULL, then all lines can be used in both directions. Must be the name of a column otherwise. The values of the column must be "FT" (From - To), "TF" (To - From) or "Both".
grid_shape	A vector of length 2 indicating the shape of the grid to use for splitting the dataset. Default is c(1,1), so all the calculation is done in one go. It might be necessary to split it if the dataset is large.
verbose	A Boolean indicating if the function should print its progress
digits	The number of digits to retain from the spatial coordinates (simplification used to reduce risk of topological error)
tol	A float indicating the minimum distance between the points and the lines' extremities when adding the point to the network. When points are closer, they are added at the extremity of the lines.

Value

A list with two matrices, one with the index of the neighbours and one with the distances.

Examples

```
networkgpkg <- system.file("extdata", "networks.gpkg", package = "spNetwork", mustWork = TRUE)
eventsgpkg <- system.file("extdata", "events.gpkg", package = "spNetwork", mustWork = TRUE)
main_network_mtl <- rgdal::readOGR(networkgpkg,layer="main_network_mtl", verbose=FALSE)
mtl_libraries <- rgdal::readOGR(eventsgpkg,layer="mtl_libraries", verbose=FALSE)
future::plan(future::multisession(workers=2))
results <- network_knn.mc(mtl_libraries, main_network_mtl,
    k = 3, maxdistance = 1000, line_weight = "length",
    grid_shape=c(1,1), verbose = FALSE)
## make sure any open connections are closed afterward
if (!inherits(future::plan(), "sequential")) future::plan(future::sequential)
```

---

network_listw *Network distance listw*

Description

Generate listw object (spdep like) based on network distances.

Usage

```
network_listw(
  origins,
  lines,
  maxdistance,
  method = "centroid",
  point_dist = NULL,
  snap_dist = Inf,
  line_weight = "length",
  mindist = 10,
  direction = NULL,
  dist_func = "inverse",
  matrice_type = "B",
  grid_shape = c(1, 1),
  verbose = FALSE,
  digits = 3,
  tol = 0.1
)
```

Arguments

origins	A SpatialLinesDataFrame, SpatialPointsDataFrame or SpatialPolygonsDataFrame for which the spatial neighbouring list will be built
lines	A SpatialLinesDataFrame representing the network
maxdistance	The maximum distance between two observations to consider them as neighbours.
method	A string indicating how the starting points will be built. If 'centroid' is used, then the centre of lines or polygons is used. If 'pointsalong' is used, then points will be placed along polygons' borders or along lines as starting and end points. If 'ends' is used (only for lines) the first and last vertices of lines are used as starting and ending points.
point_dist	A float, defining the distance between points when the method 'pointsalong' is selected.
snap_dist	The maximum distance to snap the start and end points on the network.
line_weight	The weighting to use for lines. Default is "length" (the geographical length), but can be the name of a column. The value is considered proportional to the geographical length of the lines.
mindist	The minimum distance between two different observations. It is important for it to be different from 0 when a W style is used.
direction	Indicates a field providing information about authorized travelling direction on lines. if NULL, then all lines can be used in both directions. Must be the name of a column otherwise. The values of the column must be "FT" (From - To), "TF" (To - From) or "Both".
dist_func	Indicates the function to use to convert the distance between observation in spatial weights. Can be 'identity', 'inverse', 'squared inverse' or a function with one parameter x that will be vectorized internally

matrice_type	The type of the weighting scheme. Can be 'B' for Binary, 'W' for row weighted, see the documentation of spdep::nb2listw for details
grid_shape	A vector of length 2 indicating the shape of the grid to use for splitting the dataset. Default is c(1,1), so all the calculation is done in one go. It might be necessary to split it if the dataset is large.
verbose	A Boolean indicating if the function should print its progress
digits	The number of digits to retain in the spatial coordinates (simplification used to reduce risk of topological error)
tol	A float indicating the spatial tolerance when points are added as vertices to lines.

Value

A listw object (spdep like)

Examples

```
networkgpkg <- system.file("extdata", "networks.gpkg",package = "spNetwork", mustWork = TRUE)
mtl_network <- rgdal::readOGR(networkgpkg,layer="mtl_network", verbose=FALSE)
listw <- network_listw(mtl_network,mtl_network,maxdistance=500,
        method = "centroid", line_weight = "length",
        dist_func = 'squared inverse', matrice_type='B', grid_shape = c(2,2))
```

---

network_listw.mc *Network distance listw (multicore)*

Description

Generate listw object (spdep like) based on network distances with multicore support.

Usage

```
network_listw.mc(
  origins,
  lines,
  maxdistance,
  method = "centroid",
  point_dist = NULL,
  snap_dist = Inf,
  line_weight = "length",
  mindist = 10,
  direction = NULL,
  dist_func = "inverse",
  matrice_type = "B",
  grid_shape = c(1, 1),
  verbose = FALSE,
```

```
  digits = 3,
  tol = 0.1
)
```

Arguments

origins	A SpatialLinesDataFrame, SpatialPointsDataFrame or SpatialPolygonsDataFrame for which the spatial neighbouring list will be built
lines	A SpatialLinesDataFrame representing the network
maxdistance	The maximum distance between two observations to consider them as neighbours.
method	A string indicating how the starting points will be built. If 'centroid' is used, then the centre of lines or polygons is used. If 'pointsalong' is used, then points will be placed along polygons' borders or along lines as starting and end points. If 'ends' is used (only for lines) the first and last vertices of lines are used as starting and ending points.
point_dist	A float, defining the distance between points when the method pointsalong is selected.
snap_dist	the maximum distance to snap the start and end points on the network.
line_weight	The weights to use for lines. Default is "length" (the geographical length), but can be the name of a column. The value is considered proportional with the geographical length of the lines.
mindist	The minimum distance between two different observations. It is important for it to be different from 0 when a W style is used.
direction	Indicates a field giving information about authorized travelling direction on lines. if NULL, then all lines can be used in both directions. Must be the name of a column otherwise. The values of the column must be "FT" (From - To), "TF" (To - From) or "Both".
dist_func	Indicates the function to use to convert the distance between observation in spatial weights. Can be 'identity', 'inverse', 'squared inverse' or a function with one parameter x that will be vectorized internally
matrice_type	The type of the weighting scheme. Can be 'B' for Binary, 'W' for row weighted, see the documentation of spdep::nb2listw for details
grid_shape	A vector of length 2 indicating the shape of the grid to use for splitting the dataset. Default is c(1,1), so all the calculation is done in one go. It might be necessary to split it if the dataset is large.
verbose	A Boolean indicating if the function should print its progress
digits	The number of digits to retain in the spatial coordinates (simplification used to reduce risk of topological error)
tol	A float indicating the spatial tolerance when points are added as vertices to lines.

Value

A listw object (spdep like)

Examples

```
networkgpkg <- system.file("extdata", "networks.gpkg", package = "spNetwork", mustWork = TRUE)
mtl_network <- rgdal::readOGR(networkgpkg,layer="mtl_network", verbose=FALSE)
future::plan(future::multisession(workers=2))
listw <- network_listw.mc(mtl_network,mtl_network,maxdistance=500,
        method = "centroid", line_weight = "length",
        dist_func = 'squared inverse', matrice_type='B', grid_shape = c(2,2))
## make sure any open connections are closed afterward
if (!inherits(future::plan(), "sequential")) future::plan(future::sequential)
```

---

nkde *Network Kernel density estimate*

Description

Calculate the Network Kernel Density Estimate based on a network of lines, sampling points, and events

Usage

```
nkde(
  lines,
  events,
  w,
  samples,
  kernel_name,
  bw,
  adaptive = FALSE,
  trim_bw = NULL,
  method,
  div = "bw",
  diggle_correction = FALSE,
  study_area = NULL,
  max_depth = 15,
  digits = 5,
  tol = 0.1,
  agg = NULL,
  sparse = TRUE,
  grid_shape = c(1, 1),
  verbose = TRUE,
  check = TRUE
)
```

Arguments

lines	A SpatialLinesDataFrame representing the underlying network. The geometries must be a SpatialLinesDataFrame (may crash if some geometries are invalid) without MultiLineSring.
events	events A SpatialPointsDataFrame representing the events on the network. The points will be snapped on the network to their closest line.
w	A vector representing the weight of each event
samples	A SpatialPointsDataFrame representing the locations for which the densities will be estimated.
kernel_name	The name of the kernel to use. Must be one of triangle, gaussian, tricube, cosine ,triweight, quartic, epanechnikov or uniform.
bw	The kernel bandwidth (using the scale of the lines)
adaptive	A Boolean, indicating if an adaptive bandwidth must be used
trim_bw	A float, indicating the maximum value for the adaptive bandwidth
method	The method to use when calculating the NKDE, must be one of simple / discontinuous / continuous (see nkde details for more information)
div	The divisor to use for the kernel. Must be "n" (the number of events within the radius around each sampling point), "bw" (the bandwith) "none" (the simple sum).
diggle_correction	A Boolean indicating if the correction factor for edge effect must be used.
study_area	A SpatialPolygonsDataFrame or a SpatialPolygon representing the limits of the study area.
max_depth	when using the continuous and discontinuous methods, the calculation time and memory use can go wild if the network has many small edges (area with many of intersections and many events). To avoid it, it is possible to set here a maximum depth. Considering that the kernel is divided at intersections, a value of 10 should yield good estimates in most cases. A larger value can be used without a problem for the discontinuous method. For the continuous method, a larger value will strongly impact calculation speed.
digits	The number of digits to retain from the spatial coordinates. It ensures that topology is good when building the network. Default is 3. Too high a precision (high number of digits) might break some connections
tol	A float indicating the minimum distance between the events and the lines' extremities when adding the point to the network. When points are closer, they are added at the extremity of the lines.
agg	A double indicating if the events must be aggregated within a distance. If NULL, the events are aggregated only by rounding the coordinates.
sparse	A Boolean indicating if sparse or regular matrices should be used by the Rcpp functions. These matrices are used to store edge indices between two nodes in a graph. Regular matrices are faster, but require more memory, in particular with multiprocessing. Sparse matrices are slower (a bit), but require much less memory.

grid_shape	A vector of two values indicating how the study area must be split when performing the calculus. Default is c(1,1) (no split). A finer grid could reduce memory usage and increase speed when a large dataset is used. When using multiprocessing, the work in each grid is dispatched between the workers.
verbose	A Boolean, indicating if the function should print messages about the process.
check	A Boolean indicating if the geometry checks must be run before the operation. This might take some times, but it will ensure that the CRS of the provided objects are valid and identical, and that geometries are valid.

Details

The three NKDE methods
Estimating the density of a point process is commonly done by using an ordinary two-dimensional kernel density function. However, there are numerous cases for which the events do not occur in a two-dimensional space but on a network (like car crashes, outdoor crimes, leaks in pipelines, etc.). New methods were developed to adapt the methodology to networks, three of them are available in this package.

- method="simple"This first method was presented by (Xie and Yan 2008) and proposes an intuitive solution. The distances between events and sampling points are replaced by network distances, and the formula of the kernel is adapted to calculate the density over a linear unit instead of an areal unit.
- method="discontinuous"The previous method has been criticized by (Okabe et al. 2009), arguing that the estimator proposed is biased, leading to an overestimation of density in events hot-spots. More specifically, the simple method does not conserve mass and the induced kernel is not a probability density along the network. They thus proposed a discontinuous version of the kernel function on network, which equally "divides" the mass density of an event at intersections
- method="continuous"If the discontinuous method is unbiased, it leads to a discontinuous kernel function which is a bit counter-intuitive. Okabe et al. (2009) proposed another version of the kernel, that divides the mass of the density at intersections but adjusts the density before the intersection to make the function continuous.

The three methods are available because, even though that the simple method is less precise statistically speaking, it might be more intuitive. From a purely geographical view, it might be seen as a sort of distance decay function as used in Geographically Weighted Regression.

adaptive bandwidth
It is possible to use adaptive bandwidth instead of fixed bandwidth. Adaptive bandwidths are calculated using the Abramson's smoothing regimen (Abramson 1982). To do so, an original fixed bandwidth must be specified (bw parameter), and is used to estimate the priory densitiy at event locations. These densities are then used to calculate local bandwidth. The maximum size of the local bandwidth can be limited with the parameter trim_bw. For more details, see the vignettes.

Optimization parameters
The grid_shape parameter allows to split the calculus of the NKDE according to a grid dividing the study area. It might be necessary for big dataset to reduce the memory used. If the grid_shape is

c(1,1), then a full network is built for the area. If the grid_shape is c(2,2), then the area is split in 4 rectangles. For each rectangle, the sample points falling in the rectangle are used, the events and the lines in a radius of the bandwidth length are used. The results are combined at the end and ordered to match the original order of the samples.

The geographical coordinates of the start and end of lines are used to build the network. To avoid troubles with digits, we truncate the coordinates according to the digit parameter. A minimal loss of precision is expected but results in a fast construction of the network.

To calculate the distances on the network, all the events are added as vertices. To reduce the size of the network, it is possible to reduce the number of vertices by adding the events at the extremity of the lines if they are close to them. This is controlled by the parameter tol.

In the same way, it is possible to limit the number of vertices by aggregating the events that are close to each other. In that case, the weights of the aggregated events are summed. According to an aggregation distance, a buffer is drawn around the fist event, all events falling in that buffer are aggregated to the first event, forming a new event. The coordinates of this new event are the mean of the original events coordinates. This procedure is repeated until no events are aggregated. The aggregation distance can be fixed with the parameter agg.

When using the continuous and discontinuous kernel, the density is reduced at each intersection crossed. In the discontinuous case, after 5 intersections with four directions each, the density value is divided by 243 leading to very small values. In the same situation but with the continuous NKDE, the density value is divided by approximately 7.6. The max_depth parameters allows the user to control the maximum depth of these two NKDE. The base value is 15, but a value of 10 would yield very close estimates. A lower value might have a critical impact on speed when the bandwidth is large

When using the continuous and discontinuous kernel, the connections between graph nodes are stored in a matrix. This matrix is typically sparse, and so a sparse matrix object is used to limit memory use. If the network is small (typically when the grid used to split the data has small rectangles) then a classical matrix could be used instead of a sparse one. It significantly increases speed, but could lead to memory issues.

Value

A vector of values, they are the density estimates at samplings points

References

Abramson IS (1982). "On bandwidth variation in kernel estimates-a square root law." *The annals of Statistics*, 1217–1223.

Okabe A, Satoh T, Sugihara K (2009). "A kernel density estimation method for networks, its computational method and a GIS-based tool." *International Journal of Geographical Information Science*, **23**(1), 7–32.

Xie Z, Yan J (2008). "Kernel density estimation of traffic accidents in a network space." *Computers, environment and urban systems*, **32**(5), 396–406.

Examples

```
networkgpkg <- system.file("extdata", "networks.gpkg", package = "spNetwork", mustWork = TRUE)
eventsgpkg <- system.file("extdata", "events.gpkg", package = "spNetwork", mustWork = TRUE)
mtl_network <- rgdal::readOGR(networkgpkg,layer="mtl_network", verbose=FALSE)
bike_accidents <- rgdal::readOGR(eventsgpkg,layer="bike_accidents", verbose=FALSE)
lixels <- lixelize_lines(mtl_network,200,mindist = 50)
samples <- lines_center(lixels)
densities <- nkde(mtl_network,
                  events = bike_accidents,
                  w = rep(1,nrow(bike_accidents)),
                  samples = samples,
                  kernel_name = "quartic",
                  bw = 300, div= "bw",
                  adaptive = FALSE,
                  method = "discontinuous", digits = 1, tol = 1,
                  agg = 15,
                  grid_shape = c(1,1),
                  verbose=FALSE)
```

---

nkde.mc *Network Kernel density estimate (multicore)*

Description

Calculate the Network Kernel Density Estimate based on a network of lines, sampling points, and events with multicore support.

Usage

```
nkde.mc(
  lines,
  events,
  w,
  samples,
  kernel_name,
  bw,
  adaptive = FALSE,
  trim_bw = NULL,
  method,
  div = "bw",
  diggle_correction = FALSE,
  study_area = NULL,
  max_depth = 15,
  digits = 5,
  tol = 0.1,
  agg = NULL,
```

```
  sparse = TRUE,
  grid_shape = c(1, 1),
  verbose = TRUE,
  check = TRUE
)
```

Arguments

lines	A SpatialLinesDataFrame representing the underlying network. The geometries must be a SpatialLinesDataFrame (may crash if some geometries are invalid) without MultiLineSring.
events	events A SpatialPointsDataFrame representing the events on the network. The points will be snapped on the network to their closest line.
w	A vector representing the weight of each event
samples	A SpatialPointsDataFrame representing the locations for which the densities will be estimated.
kernel_name	The name of the kernel to use. Must be one of triangle, gaussian, tricube, cosine ,triweight, quartic, epanechnikov or uniform.
bw	The kernel bandwidth (using the scale of the lines)
adaptive	A Boolean, indicating if an adaptive bandwidth must be used
trim_bw	A float, indicating the maximum value for the adaptive bandwidth
method	The method to use when calculating the NKDE, must be one of simple / discontinuous / continuous (see nkde details for more information)
div	The divisor to use for the kernel. Must be "n" (the number of events within the radius around each sampling point), "bw" (the bandwith) "none" (the simple sum).
diggle_correction	A Boolean indicating if the correction factor for edge effect must be used.
study_area	A SpatialPolygonsDataFrame or a SpatialPolygon representing the limits of the study area.
max_depth	when using the continuous and discontinuous methods, the calculation time and memory use can go wild if the network has many small edges (area with many of intersections and many events). To avoid it, it is possible to set here a maximum depth. Considering that the kernel is divided at intersections, a value of 10 should yield good estimates in most cases. A larger value can be used without a problem for the discontinuous method. For the continuous method, a larger value will strongly impact calculation speed.
digits	The number of digits to retain from the spatial coordinates. It ensures that topology is good when building the network. Default is 3. Too high a precision (high number of digits) might break some connections
tol	A float indicating the minimum distance between the events and the lines' extremities when adding the point to the network. When points are closer, they are added at the extremity of the lines.
agg	A double indicating if the events must be aggregated within a distance. If NULL, the events are aggregated only by rounding the coordinates.

sparse	A Boolean indicating if sparse or regular matrices should be used by the Rcpp functions. These matrices are used to store edge indices between two nodes in a graph. Regular matrices are faster, but require more memory, in particular with multiprocessing. Sparse matrices are slower (a bit), but require much less memory.
grid_shape	A vector of two values indicating how the study area must be split when performing the calculus. Default is c(1,1) (no split). A finer grid could reduce memory usage and increase speed when a large dataset is used. When using multiprocessing, the work in each grid is dispatched between the workers.
verbose	A Boolean, indicating if the function should print messages about the process.
check	A Boolean indicating if the geometry checks must be run before the operation. This might take some times, but it will ensure that the CRS of the provided objects are valid and identical, and that geometries are valid.

Details

For more details, see help(nkde)

Value

A vector of values, they are the density estimates at sampling points

Examples

```
networkgpkg <- system.file("extdata", "networks.gpkg", package = "spNetwork", mustWork = TRUE)
eventsgpkg <- system.file("extdata", "events.gpkg", package = "spNetwork", mustWork = TRUE)
mtl_network <- rgdal::readOGR(networkgpkg,layer="mtl_network", verbose=FALSE)
bike_accidents <- rgdal::readOGR(eventsgpkg,layer="bike_accidents", verbose=FALSE)
future::plan(future::multisession(workers=2))
lixels <- lixelize_lines(mtl_network,200,mindist = 50)
samples <- lines_center(lixels)
densities <- nkde.mc(mtl_network,
                 events = bike_accidents,
                 w = rep(1,nrow(bike_accidents)),
                 samples = samples,
                 kernel_name = "quartic",
                 bw = 300, div= "bw",
                 adaptive = FALSE, agg = 15,
                 method = "discontinuous", digits = 1, tol = 1,
                 grid_shape = c(3,3),
                 verbose=FALSE)
## make sure any open connections are closed afterward
if (!inherits(future::plan(), "sequential")) future::plan(future::sequential)
```

quartic_kernel *Quartic kernel*

Description

Function implementing the quartic kernel.

Usage

```
quartic_kernel(d, bw)
```

Arguments

d	The distance from the event
bw	The bandwidth used for the kernel

Value

The estimated density

Examples

```
#This is an internal function, no example provided
```

simplify_network *Simplify a network*

Description

Simplify a network by applying two corrections: Healing edges and Removing mirror edges (experimental).

Usage

```
simplify_network(
  lines,
  digits = 3,
  heal = TRUE,
  mirror = TRUE,
  keep_shortest = TRUE,
  verbose = TRUE
)
```

Arguments

lines	A SpatialLinesDataFrame
digits	An integer indicating the number of digits to keep in coordinates
heal	A boolean indicating if the healing operation must be performed
mirror	A booleans indicating if the mirror edges must be removed
keep_shortest	A boolean, if TRUE, then the shortest line is keeped from mirror edges. if FALSE, then the longest line is keeped.
verbose	A boolean indicating if messages and progressbar should be displayed

Details

Healing is the operation to merge two connected linestring if the are intersecting at one extremity and do not intersect any other linestring. It helps to reduce the complexity of the network and thus can reduce calculation time. Removing mirror edges is the operation to remove edges that have the same extremities. If two edges start at the same point and end at the same point, they do not add information in the network and one can be removed to simplify the network. One can decide to keep the longest of the two edges or the shortest. NOTE: the edge healing does not consider lines directions currently!

Value

A SpatialLinesDataFrame

Examples

```
library(spNetwork)
networkgpkg <- system.file("extdata", "networks.gpkg",package = "spNetwork", mustWork = TRUE)
lines <- mtl_network <- rgdal::readOGR(networkgpkg,layer="mtl_network",verbose = FALSE)
edited_lines <- simplify_network(lines, digits = 3, verbose = FALSE)
```

---

split_graph_components

Split graph components

Description

Function to split the results of build_graph and build_graph_directed into their sub components

Usage

```
split_graph_components(graph_result)
```

Arguments

graph_result	A list typically obtained from the function build_graph or build_graph_directed

Value

A list of lists, the graph_result split for each graph component

Examples

```
networkgpkg <- system.file("extdata", "networks.gpkg", package = "spNetwork", mustWork = TRUE)
mtl_network <- rgdal::readOGR(networkgpkg,layer="mtl_network", verbose=FALSE)
mtl_network$length <- rgeos::gLength(mtl_network, byid = TRUE)
graph_result <- build_graph(mtl_network, 2, "length", attrs = TRUE)
sub_elements <- split_graph_components(graph_result)
```

---

split_lines_at_vertex *Split lines at vertices in a SpatialLinesDataFrame*

Description

Split lines (SpatialLines) at their nearest vertices (SpatialPoints), may fail if the line geometries are self intersecting.

Usage

```
split_lines_at_vertex(lines, points, nearest_lines_idx, mindist)
```

Arguments

lines	The SpatialLinesDataframe to split
points	The SpatialPoints to add to as vertex to the lines
nearest_lines_idx	For each point, the index of the nearest line
mindist	The minimum distance between one point and the extremity of the line to add the point as a vertex.

Value

An object of class SpatialLinesDataFrame (package sp)

Examples

```
# reading the data
networkgpkg <- system.file("extdata", "networks.gpkg", package = "spNetwork", mustWork = TRUE)
mtl_network <- rgdal::readOGR(networkgpkg,layer="mtl_network", verbose=FALSE)
eventsgpkg <- system.file("extdata", "events.gpkg", package = "spNetwork", mustWork = TRUE)
bike_accidents <- rgdal::readOGR(eventsgpkg,layer="bike_accidents", verbose=FALSE)
# aggregating points within a 5 metres radius
bike_accidents$weight <- 1
agg_points <- aggregate_points(bike_accidents, 5)
mtl_network$LineID <- 1:nrow(mtl_network)
```

```
# snapping point to lines
snapped_points <- snapPointsToLines2(agg_points,
    mtl_network,
    "LineID"
)
# splitting lines
new_lines <- split_lines_at_vertex(mtl_network, snapped_points,
    snapped_points$nearest_line_id, 1)
```

`triangle_kernel` *triangle kernel*

Description

Function implementing the triangle kernel.

Usage

```
triangle_kernel(d, bw)
```

Arguments

`d`	The distance from the event
`bw`	The bandwidth used for the kernel

Value

The estimated density

Examples

```
#This is an internal function, no example provided
```

`tricube_kernel` *Tricube kernel*

Description

Function implementing the tricube kernel.

Usage

```
tricube_kernel(d, bw)
```

Arguments

d	The distance from the event
bw	The bandwidth used for the kernel

Value

The estimated density

Examples

```
#This is an internal function, no example provided
```

---

triweight_kernel *Triweight kernel*

Description

Function implementing the triweight kernel.

Usage

```
triweight_kernel(d, bw)
```

Arguments

d	The distance from the event
bw	The bandwidth used for the kernel

Value

The estimated density

Examples

```
#This is an internal function, no example provided
```

`uniform_kernel` *Uniform kernel*

Description

Function implementing the uniform kernel.

Usage

```
uniform_kernel(d, bw)
```

Arguments

`d`	The distance from the event
`bw`	The bandwidth used for the kernel

Value

The estimated density

Examples

```
#This is an internal function, no example provided
```

Index