Package: SpatialGraph (via r-universe)

September 16, 2024

Version 1.0-4

Type Package

Title The SpatialGraph Class and Utilities

Date 2023-07-21

Imports igraph, methods, pracma, sf, shape, sp, splancs

Author Javier Garcia-Pintado

Maintainer Javier Garcia-Pintado <jgarciapintado@marum.de>

Description Provision of the S4 SpatialGraph class built on top of objects provided by 'igraph' and 'sp' packages, and associated utilities. See the documentation of the SpatialGraph-class within this package for further description. An example of how from a few points one can arrive to a SpatialGraph is provided in the function sl2sg().

License GPL (>=2)

URL https://github.com/garciapintado/SpatialGraph

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

RemoteUrl https://github.com/garciapintado/spatialgraph

RemoteRef HEAD

RemoteSha b0557352b0cef9101dbcf6bf6a01a7c80f7d7e65

Contents

`SpatialGraph-package` *The SpatialGraph Class and Utilities*

Description

Provision of the S4 SpatialGraph class built on top of objects provided by 'igraph' and 'sp' packages, and associated utilities. See the documentation of the SpatialGraph-class within this package for further description. An example of how from a few points one can arrive to a SpatialGraph is provided in the function sl2sg().

Details

The DESCRIPTION file: This package was not yet installed at build time.

Index: This package was not yet installed at build time.
see the documentation of the function sl2sg in this package to get a start. A case study making use if this package is Garcia-Pintado et al (2015)

Author(s)

Javier Garcia-Pintado

Maintainer: Javier Garcia-Pintado <jgarciapintado@marum.de>

References

The first published application of this package is Garcia-Pintado, J. et al. (2015). Satellite-supported flood forecasting in river networks: a real case study. J. Hydrol. 523, 705-724.

attSGe *Add or Modify attributes in SpatialGraph edges*

Description

Add or Modify attributes in SpatialGraph edges

Usage

```
attSGe(SG, att, eID, val, default)
```

Arguments

SG	SpatialGraph
att	name of the field [column] in the edge dataframe to be added/modified
eID	edge identifiers [row.names of the edge data.frame]
val	values corresponding the eID above
default	default values for edges not considered in eID above

Value

A SpatialGraph

distSG *Calculate across-network distance for a set of sparse points*

Description

This function obtains the across-network distance for a set of sparse points, by using the distance slot in a SpatialGraph. The calculation is supported by a previously calculated between vertex distance matrix [via a call to the library igraph by the function distSGv]. The SpatialGraph is considered as undirected for distance calculation. If euc=TRUE [default], the distance between two points is defined within this function as the maximum of both the minimum along-network distance and the Euclidean distance. The distance itself between the points in x,y and the network is neglected in the function for the along-network distance. Both, x and y, are SpatialPointsDataFrame objects, which must contain at least the fields eID and chain, which describe their relationship with the SpatialGraph object defined by SG. These can be obtained with either the function pointsSLDFchain or pointsToLines (the latter is faster, but depends on GEOS)

Usage

```
distSG(SG, x, y = NULL, euc = TRUE, wei = NULL, getpath = FALSE)
```

Arguments

`SG` SpatialGraph

`x` `SpatialPointsDataFrame`

`y` `SpatialPointsDataFrame`

`euc` boolean scalar, whether to use Euclidean distance as minimum threshold for resulting distances

`wei` if not null, field in `SG@e` with a variable to obtain a state-related weight. See details below.

`getpath` if TRUE (and wei != NULL), `eID` identifiers for each path from `x` to `y` elements is returned

Details

The application of state-related weights in this version is a simple state-dependent weight matrix related to some field in `SG@e` [i.e. the edges in the input `SpatialGraph`]. The only current calculation evaluates the path between queried points (x,y), and along the path, for every junction and jump into a new edge, the ratio for the evaluated state variable (taken as the highest value divided by the lowest value) between the two edges at the junction is obtained. Currently a maximum ratio equal to 10.0 is hard-coded. The product of ratios along the path gives the weight.

Value

If `wei=NULL`, a matrix of distances between `x` and `y`. If `wei` is not `NULL`, a list with a distance matrix and weight matrix (plus a matrix with eID identifiers for the path, if `getpath=TRUE`) is returned.

Author(s)

Javier Garcia-Pintado, e-mail: `<jgarciapintado@marum.de>`

Examples

```
if (1 > 2) { # not run
  dem <- readGDAL(file.path(system.file('external',package='hydrosim'),
                 'watershed1','IDRISI_maps','dem','dem.rst'))   # SpatialGridDataFrame
  plotGmeta(layer=dem, xlim=662500 + 2500 * c(-1,+1),
            ylim=4227500 + 2500 * c(-1,1), zlim='strloc', as.na=0)

  # generate some crossing lines
  zz <- list()
  zz[[1]] <- digitGmeta(layer=dem, type='Lines', ID=1)
  zz[[2]] <- digitGmeta(layer=dem, type='Lines', ID=2)
  zz[[3]] <- digitGmeta(layer=dem, type='Lines', ID=3)
  SL <- SpatialLines(zz)
  SG <- sl2sg(SL, getpath=TRUE)
  points(SG@v, cex=2)                   # plot SpatialGraph vertices

  apath <- SG@path[[1,2]]                # iteratively plot a path as an example
  for (iv in 1:length(apath$v)) {
    points(SG@v[apath$v[iv],], cex=2,pch=2)
    if (iv == length(apath$v))
      break
    lines(SG@e[apath$e[iv],],col='blue',lwd=2,lty=2)
    Sys.sleep(1)
  }

  # sample a few points [as a matrix] close to some edges
  xy    <- digit()                    # sample locations
  xych  <- pointsToLines(xy, SG@e)    # SpatialPointsDataFrame mapping
  points(xy, col='blue', pch=3)
  points(xych, col='darkgreen', pch=19)

  # along-network distance
  xyndis <- distSG(SG, xych)

  # state-dependent weighted along-network distance
 SG@e@data$wxs <- 3+round(runif(nrow(SG@e@data)),2)      # [m2] foo wetted cross-section areas
  SG@e@data

  xywdis <- distSG(SG, xych, wei='wxs')
 xywdis <- xywdis$dis * xywdis$wei     # Schur weight application into distance estimation
}
```

`distSGv` *Calculate the distance slot in a SpatialGraph*

Description

Calculate the distance slot in a SpatialGraph. This is done via a call to the library `igraph`, which does the calculation. Distances are undirected.

Usage

```
distSGv(SG, getpath = FALSE)
```

Arguments

`SG`	SpatialGraph
`getpath`	boolean. Whether to calculate the `SG@path` slot

Value

A SpatialGraph with the slot `dist` (and `path` if requested) recalculated

explodeSLDF *Explode Lines in a SpatialLinesDataFrame*

Description

explode Lines in a SpatialLinesDataFrame, so that each single Line, within each Lines slot, is upgraded as a new 1-Line Lines slot

Usage

```
explodeSLDF(SLDF, FID)
```

Arguments

`SLDF` a SpatialLinesDataFrame

`FID` if not NULL, field name, within the attribute table considered as additional unique identifier, so that incremental numeric values will added to this field to avoid duplicate values

Value

a SpatialLinesDataFrame

pointLineD *Euclidean distance from a set of points to a line segment*

Description

`pointLineD` returns a list with a number of components from a points to line segment analysis

Usage

```
pointLineD(xy, xyp)
```

Arguments

`xy` 2 x 2 [x,y] matrix defining the start and end of the segment

`xyp` p x 2 [x,y] matrix with a point set

Details

`pointLineD` conduct a detailed points to segment distance analysis, returned as a list

Value

A list with the input components `xy` and `xyp`, and the aditional components: `d`, point-line distance (distance between the points in `xyp` and their perpendicular projections of the line); `dc`, diferential chainage over [x0,y0] (> 0 if the projection goes in the segment direction); `cross`, boolean vector indicating whether the perpendicular projection of the points crosses the segment, or not

See Also

`Spatial-class`

`pointOnLine` *Snap a points to a line*

Description

This function snaps a point to a line based on the minimum distance between the point and the line

Usage

```
pointOnLine(cool, coop)
```

Arguments

`cool` 2-col matrix giving the coordinates of the line

`coop` 2-length vector repsenting the point

Value

A 4-length vector, with 'x','y' [coordinates of the point snapped to the line], 'd' [distance from the input point to the new snapped point], and 'chain' [accumulated along-line distance from the starting of the line to the snapped point]

Author(s)

Javier Garcia-Pintado

See Also

`Spatial-class`

pointOnSegment *Snap a points to a segment*

Description

This function snaps a point to a segment based on the minimum distance between the point and the segment

Usage

```
pointOnSegment(s, p)
```

Arguments

s	[2,2] matrix giving the coordinates of the line, one point per row
p	2-length vector repsenting the point

Value

A 4-length vector, with 'x','y' [coordinates of the point snapped to the segment], 'd' [distance from the input point to the new snapped point], and 'chain' [distance from the starting of the segment to the snapped point]

Author(s)

Javier Garcia-Pintado

See Also

Spatial-class

pointsPolylineD *closest points in a polyline to a set of points*

Description

pointsPolylineD returns a list with a number of components from a points to polyline analysis

Usage

```
pointsPolylineD(xy, xyp)
```

Arguments

xy	n x 2 [x,y] matrix defining the polyline
xyp	p x 2 [x,y] matrix with a point set

Details

pointsPolylineD conducts a detailed points to polyline distance analysis. First the distance from the set of points to the lines defined by every single segment in the polyline is obtained by succesive calls to pointLineD, then the distance to every single node in the polyline are also obtained. The lower distance is chosen.

Value

A data.frame with the columns: inode is the index of the first node in the closest segment to each point, x0 and y0 are the corresponding coordinates of those nodes, xc and yc are the coordinates of the point in the polyline closest to each point in xyp, these may be but are not necessarily one the polyline nodes, dis it the distance from each point tho the polyline, chain0 is the chainage of x0,y0 with the polyline, and dc is the differential chainage from xc,yc to x0,y0

See Also

Spatial-class

pointsSLDFchain *Obtain chainage from sparse points along a SpatialLinesDataFrame*

Description

For a set of points, obtains the closest Line object in a SpatialLinesDataFrame. The function assumes that each Feature (entry in the DataFrame part of the SpatialLinesDataFrame) just contains one Line (i.e. one polyline). The within-polyine chainage (that is, distance from the initial point of the poyline to the mapping of the point into the polyline) is also returned. If mask is NULL, each point in the set is assigned a line in SLDF by Euclidean distance. If mask is provided, the match between mask and the SLmsk field in SLDF is used instead for polyline assignation.

Usage

```
pointsSLDFchain(SLDF, xy, SLmsk='FEAT_ID', mask=NULL, type='SpatialPointsDataFrame')
```

Arguments

SLDF	SpatialLinesDataFrame
xy	REAL [n,2] matrix of points, or a SpatialPointsDataFrame
SLmsk	is !is.null(mask) this is the field in the SLDF data.frame matching the values in mask
mask	REAL, OPT, [n] a vector indicating to which line in SLDF is related each point
type	character. Either ’SpatialPointsDataFrame’ or ’mapping’. In the latter case, just the chainage in line feature identifiers are returned

Value

A data.frame with two columns, ’chai’, and ’eIDs’, where ’eIDs’ are the row names of the data.frame component of the input SpatialLinesDataFrame

Author(s)

Javier Garcia-Pintado, e-mail: `<jgarciapintado@marum.de>`

`pointsToLines` *Snap a set of points to a set of lines*

Description

This function snaps a set of points to a set of lines based on the minimum distance of each point to any of the lines

Usage

```
pointsToLines(points, lines, withAttrs = TRUE, withDis = TRUE, withChain = TRUE)
```

Arguments

`points` An object of the class SpatialPoints or SpatialPointsDataFrame, or a 2-col matrix of [x,y] coordinates

`lines` An object of the class SpatialLines or SpatialLinesDataFrame

`withAttrs` Boolean value for preserving (TRUE) or getting rid (FALSE) of the original point attributes. Default: TRUE. This parameter is optional

`withDis` Boolean value for including distance from source points to snapped-to-lines points

`withChain` Boolean value for including the chainage of the snapped points in their corresponding lines

Value

A SpatialPointsDataFrame object as defined by the R package ’sp’. This object contains the snapped points, therefore all of them lie on the lines. The returned object contains the fields ’lid’, ’eID’, and ’chain’, providing information about the relationship between the source data points, the snapped data points, and its location within the network: ’lid’, and ’eID’ are the line index and line ID, respectively, of the lines in which the new snapped points lie; ’dis’ is the distance between the input points and the snapped points, and ’chain’ is the chainage of the snapped point within the corresponding line

Author(s)

Javier Garcia-Pintado

See Also

Spatial-class

polylineChainage *Obtain the chainage of nodes along a polyline*

Description

Obtain the chainage of nodes along a polyline [2-col matrix]

Usage

```
polylineChainage(xy)
```

Arguments

xy	a 2-column matrix representing the polyline nodes

Details

polylineChainage calculates a vector of chainage values [along-polyline distances] from each node in a polyline to the initial node

Value

A vector

See Also

polylineLength

polylineLength *Obtain the length of a polyline*

Description

Obtain the length a polyline [2-col matrix]

Usage

```
polylineLength(xy)
```

Arguments

xy	a 2-column matrix representing the polyline nodes

Details

`polylineLength` calculates the [along-polyline] length of the polyline

Value

A scalar

See Also

`polylineChainage`

`revSGe` *Reverse Lines in a SpatialGraph*

Description

A SpatialGraph contains a `SpatialLinesDataFrame`, describing the network topology. The input `eID` indicates the identifiers of a set of lines (edges) in the network to be reversed. Note `eID` does not refer to the line index within `SG@e`, but to the Feature Identifiers, as extracted from `row.names(SG@e@data)`

Usage

```
revSGe(SG, eID)
```

Arguments

`SG`	`SpatialGraph`
`eID`	vector of Feature Identifiers for lines to be reversed

Details

Note `eID` does not refer to the line index within `SG@e`, but to the Feature Identifiers, as extracted from `row.names(SG@e@data)`. Accordingly to the reversed coordiantes, the corresponding fields ["v0","v1"], are interchanged.

Value

A SpatialGraph

rotation *Rotate 2D points*

Description

rotate points, counterclockwise for positive angles, and clockwise for negative ones

Usage

```
rotation(coords, radian)
```

Arguments

`coords`	2-col matrix of [x,y] coordinates
`radian`	rotation angle

Value

a 2-col matrix with the points rotated around [0,0]

routeSDG *Accumulate sources/sinks along a directed SpatialGraph*

Description

Assume a SpatialGraph is directed and conduct an accumulation of source/sink values at nodes across the network. The accumulation assumes no delay in transmission

Usage

```
routeSDG(SDG, FUN='cumsum', ifld='inflow')
```

Arguments

`SDG`	`SpatialGraph`, assumed as directed
`FUN`	name of a function to be applied for the routing
`ifld`	name on the field in the `SpatialPointDataFrame` vertex slot to be used used as source/sink

Details

The `SpatialGraph`, used as input, must have the `ifld` field to be used as input, in the vertices slot `v` (a SpatialPointsDataFrame). The accumulated output is provided as the new field `ofld` in `v`. The edges slot `e` serves to route the input across the network

Value

A `SpatialGraph` with the added `ofld` field in the vertex slot

`sg2igraph` *Map a SpatialGraph into an igraph*

Description

The vertex and edge information in a `SpatialGraph` is mapped into an `igraph` object

Usage

```
sg2igraph(sg, directed=FALSE)
```

Arguments

`sg`	`SpatialGraph`
`directed`	whether the resulting `igraph` is directed

Details

It is assumed that the `SpatialGraph`, used as input, is correct (i.e.g all records in `sg@e@data` have the two first field correctly identifying the field 'ID' in `sg@v`. It is also assumed that the `sg@e@data` data.frame has the fields `div` and `len`. These two are highly useful to conduct network operations on the resulting `igraph`

Value

An `igraph`

`sgChVIDs` *Change vertex IDs in a SpatialGraph*

Description

Change the field "ID" in the vertex slot, `v`, of a SpatialGraph. The fields `v0` and `v1` of the edge slot, `e`, are accordingly updated

Usage

```
  sgChVIDs(obj, IDa, IDp = NULL)
```

Arguments

`obj`	A SpatialGraph object
`IDa`	A vector indicating the updated vertex IDs
`IDp`	A vector indicating the prior vertex IDs

Details

If `IDp` is not provided, it is assumed that the vector of updated indexes is sorted equally to the order in which the vertices are stored in the slot `v` of the SpatialGraph. If `IDp` is provided, the mapping IDp -> IDa is used for reclassifying the vertices.

Value

A SpatialGraph object

`sl2sg` *Map a SpatialLinesDataFrame into a SpatialGraph*

Description

This function is the major workhorse to map an input `SpatialLinesDataFrame`, as defined in the package sp, into a `SpatialGraph` by using the spatial connectivity. Input is first exploded by using `explodeSLDF`, and then all vertices in the `SpatialGraph` are automatically generated according to crossings in the input polylines.

Usage

```
sl2sg(SL, clipd = NULL, getdist = TRUE, getpath = FALSE)
```

Arguments

`SL`	`SpatialLinesDataFrame` as defined in package sp
`clipd`	distance threshold for clipping features, If NULL, a value of 1.0E-04 of the domain side size is used
`getdist`	calculate the `dist` slot in the returned `SpatialGraph`
`getpath`	calculate the `path` slot in the returned `SpatialGraph`

Details

A `SpatialGraph` is generated

Value

A `SpatialGraph`

Author(s)

Javier Garcia-Pintado, e-mail: <j.garcia-pintado@marum.de>

Examples

```
 #   x    y
 # create list of Line objects
if (1 > 2) {
 library(sp)
 library(SpatialGraph)
 zz <- list()
 zz[[1]] <- Line(matrix(
  c(661750, 4229150,
    662650, 4229450,
    663550, 4227650,
    663550, 4226850), ncol=2, byrow=TRUE))
 zz[[2]] <- Line(matrix(
  c(660250, 4229650,
    661050, 4226450,
    662550, 4225350,
    664850, 4225850,
    664650, 4229150,
    662350, 4228850), ncol=2, byrow=TRUE))
 # upgrade Line as Lines
 for (i in 1:length(zz)) {
   zz[[i]] <- Lines(list(zz[[i]]), ID=i)
 }
 # as SpatialLines
 SL <- sp::SpatialLines(zz)
 # as SpatialGraph including path calculation
 SG <- sl2sg(SL, getpath=TRUE)

 plot(SL, axes=TRUE)
 points(SG@v, cex=2)
 lines(SG@e, lwd=2)
 points(SG@v, cex=2, col='grey', pch=19)
 text(SG@v, labels=SG@v$ID)
 # label edges and directions
 textSGe(SG)
 # show a distance matrix between nodes
 SG@dist
 # show path from node 1 to 3
 SG@path[1,3]
}
```

SpatialGraph *Create a SpatialGraph object*

Description

A SpatialGraph object is created

Usage

```
SpatialGraph(v, e, dist = NULL, path = NULL)
```

Arguments

`v`	`SpatialPointsDataFrame`
`e`	`SpatialLinesDataFrame`
`dist`	along-network (symmetric) distance matrix
`path`	matrix of lists with paths corresponding to `dist`. While distances between vertex couples are symmetric, the path matrix is not symmetric as individual path to from source vertex to destination vertex. Each list in the matrix has two S3 components (v,e) describing vertices (including bounds) and edges along the path. Thus it is always one less edge than then number of vertices in the path

Value

`SpatialGraph` returns an object of class SpatialGraph-class

`SpatialGraph-class` *Class "SpatialGraph"*

Description

Class for spatial networks

Objects from the Class

Objects can be created by calls to the function SpatialGraph

Slots

- `v`**:** Object of class `"SpatialPointsDataFrame"`, whose data.frame must contain the "ID" field as unique identifier
- `e`**:** Object of class `"SpatialLinesDataFrame"`, whose data.frame must contain the fields `v0` and `v1` matching the unique identifiers "ID" in the slot `v` data.frame
- `dist`**:** Matrix, representing the undirected along-graph distance between all vertices in the network
- `path`**:** list with variable length arrays describing the minimum distance path between vertices

Author(s)

Javier Garcia-Pintado, e-mail: `<j.garcia-pintado@reading.ac.uk>`

splitPolyline *Split a polyline into a number of transects*

Description

`splitPolyline` returns a list with a number of transects along a polyline

Usage

```
splitPolyline(xy, xyp, dmax)
```

Arguments

`xy`	2-column [x,y] matrix defining the polyline nodes
`xyp`	2-column [x,y] matrix with a point set
`dmax`	maximum distance between points in `xy` and the polyline, for these to be considered for poyline splitting

Details

`splitPolyline` obtain the closest points in a polyline to a given input set of points. Those closest points are used to divide the polyline in a number of transects. The indivudual transects are clipped to the input point dataset, so the different transects are continuous in space. Note that if the input points is quite appart from the polyline, the output seqence of transect may substantially differ form the input polyline at rupture zones

Value

A list in which each element is a matrix representing an individual polyline

See Also

`Spatial-class`

splitSLDF *Split 1-Line Lines in a SpatialLines or a SpatialLinesDataFrame by intersection with a point dataset*

Description

`splitSLDF` divides the 1-Line Lines in the `SpatialLines` or the `SpatialLinesDataFrame` at intersections with the input point dataset

Usage

```
splitSLDF(SLDF, SPDF, dmax=NULL)
```

Arguments

SLDF	length-1 SpatialLinesDataFrame or SpatialLines object
SPDF	SpatialPointsDataFrame
dmax	maximum distance between points in SPDF and the polylines in SLDF, for these to be considered for poyline splitting

Details

splitPolyline obtain the closest points in the SpatialLinesDataFrame to a given input set of points. Those closest points are used to divide the polylines in a number of transects. The individual transects are clipped to the input point dataset, so the different transects are continuous in space. Note that if the input points is quite appart from the polyline, the output sequence of transects may substantially differ form the input polyline at rupture zones. The input parameter dmax is provided as a mean to avoid too strange splitting results. Setting dmax to a ver low value will reduce the spureous results, but also the input points need to be closer to the lines for the adequate recognition of splitting points

Value

A SpatialLinesDataFrame or a SpatialLines, according to the input

See Also

Spatial-class

---

textSGe *Label edges in a SpatialGraph plot*

Description

A SpatialGraph contains a SpatialLinesDataFrame, describing the network topology. This function adds line IDs and direction arrows to an existing plot of a SpatialGraph.

Usage

```
textSGe(SG, acol='wheat', tcol='navyblue', arr.length=0.4)
```

Arguments

SG	SpatialGraph
acol	color of the graph direction arrows
tcol	color of the text for graph edge IDs
arr.length	length of the direction arrows

Value

Arrows and edge IDs added to a SpatialGraph plot

Index