Package 'bnspatial'

October 12, 2022

Title Spatial Implementation of Bayesian Networks and Mapping

Version 1.1.1

Date 2020-01-16

Maintainer Dario Masante <dario.masante@gmail.com>

URL http://github.com/dariomasante/bnspatial

BugReports https://github.com/dariomasante/bnspatial/issues

Imports raster, rgdal, sf, gRbase, gRain, doParallel, foreach, utils

Suggests knitr, rmarkdown

Description Allows spatial implementation of Bayesian networks and mapping in geographical space. It makes maps of expected value (or most likely state) given known and unknown conditions, maps of uncertainty measured as coefficient of variation or Shannon index (entropy), maps of probability associated to any states of any node of the network. Some additional features are provided as well: parallel processing options, data discretization routines and function wrappers designed for users with minimal knowledge of the R language. Outputs can be exported to any common GIS format.

License GPL-3

LazyLoad yes

LazyData yes

NeedsCompilation no

RoxygenNote 7.0.2

VignetteBuilder knitr

Author Dario Masante [aut, cre]

Depends R (>= 3.5.0)

Repository CRAN

Date/Publication 2020-01-17 08:10:02 UTC

R topics documented:

`aoi` *Build area of interest (A.O.I.)*

Description

This function creates a spatial object (raster or vector) defining the area of interest, by taking a bounding box or a spatial object, or unioning the input spatial objects if more than one are provided. When `msk` is a list of rasters, extent is set equal to their combined extent (union) and resolution to the finest resolution among them.

Usage

```
aoi(msk, mskSub = NULL, xy = FALSE, bbox = NULL)
```

Arguments

`msk`	a character (path to raster or vector file), or a bounding box as numeric (xmin,xmax,ymin,ymax), or one or more (as list of) rasters of class "RasterLayer", or a single object of class "sf" or "SpatialPolygonsDataFrame". The reference data (raster or vector) to be used as mask. All model outputs will have the same extent (outline) as this object. All locations with no data (i.e. NA) cells in `msk` input will be ignored as well.
`mskSub`	vector of values, for raster data only. The subset values from `msk` which should be considered to build the area of interest. All other values will be ignored and masked out during modelling.
`xy`	logical. Should return a two column matrix of x and y coordinates of cells centre (raster data) or the IDs of features? Defaults to FALSE, returning an object of class RasterLayer or sf.
`bbox`	numeric of four elements, the coordinates defining a rectangle (bounding box) to limit the area of interest. Must be ordered as xmin, xmax, ymin, ymax. Coordinates must be in the same reference system as spatial data.

Details

If rasters are used, all model outputs will have the same resolution and same extent as inherited from msk. All locations with no data (i.e. NA) cells from msk will be ignored as well.

Value

An object of class RasterLayer or sf, or a matrix of coordinates of mask cells (raster only). In the former case, valid cells (i.e. the area of interest) will have value 1, NA otherwise.

See Also

extractByMask

Examples

```
## Make a mask from a group of input layers:
list2env(ConwyData, environment())

network <- LandUseChange
spatialData <- c(ConwyLU, ConwySlope, ConwyStatus)
m <- aoi(spatialData)
m

## Plot mask
library(raster)
m <- aoi(ConwyLU)
plot(m)

## Make mask from a subset of values and plot
m <- aoi(ConwyLU, mskSub=c(2,3))
plot(m)

## Return coordinates of valid mask locations
coord <- aoi(ConwyLU, xy=TRUE)
head(coord)

## Using a bounding box
aoi(ConwyLU, bbox=c(270000, 284950, 347000, 365000))

## For vectorial spatial data. Note xy=TRUE shall return the features IDs
Conwy = sf::st_read(system.file("extdata", "Conwy.shp", package = "bnspatial"))
aoi(Conwy, bbox=c(270000, 284950, 347000, 365000))
```

---

bnspatial *Spatialize the Bayesian network*

Description

This function wraps most package functions, to ease the spatial implementation of Bayesian networks with minimal coding.

Usage

```
bnspatial(
  network,
  target,
  spatialData,
  lookup,
  msk = NULL,
  what = c("class", "entropy"),
  midvals = NULL,
  targetState = NULL,
  spatial = TRUE,
  inparallel = FALSE,
  export = FALSE,
  path = NULL,
  field = NULL,
  verbose = TRUE,
  ...,
  exportRaster = FALSE
)
```

Arguments

network	The Bayesian network. An object of class grain, or a character (the path to the *.net* file to be loaded)
target	character. The node of interest to be modelled and mapped.
spatialData	character with path to one or more raster files or to a single spatial vector file, or a list of objects of class 'RasterLayer' (for raster), or a single object of class 'sf' or 'SpatialPolygonsDataFrame' (for spatial vector). The spatial data associated to given network nodes, provided as file paths or as (list of) spatial objects. Items must be ordered accordingly to the corresponding nodes listed in lookup, or provided as named list, where names match exactly to the corresponding nodes name.
lookup	character (path to file) or a formatted list. This argument can be provided as path to a comma separated file or a formatted list (see setClasses)
msk	a character (path to raster or vector file), or a bounding box as numeric (xmin,xmax,ymin,ymax), or one or more (as list of) rasters of class "RasterLayer", or a single object of class "sf" or "SpatialPolygonsDataFrame". The reference data (raster or vector) to be used as mask. All model outputs will have the same extent (outline) as this object. All locations with no data (i.e. NA) cells in msk input will be ignored as well.
what	character. The required output, one or more of these are valid: • "class" returns the relatively most likely state.

- "entropy" calculates the Shannon index and returns the entropy, given the node probabilities.
- "probability" returns an object for each state of the target node, with its probability.
- "expected" gives the expected value for the target node (see Details). Only valid for target nodes of continuous values. midValues argument must be provided.
- "variation" returns the coefficient of variation, as a measure of uncertainty. Only valid for target nodes of continuous values.

midvals	vector of length equal to the number of states of the target node. Applies only if the target node is a continuous variable, in which case midvals must contain the mid values for each of the intervals
targetState	character. One or more states of interest from the target node. Applies only when argument what includes 'probability'. Default is set to all states of the node.
spatial	logical. Should the output be spatially explicit -i.e. a georeferenced raster or spatial vector? Default is TRUE, returning an object of class "RasterLayer" or "sf" for polygons. If FALSE, returns a data frame with one row for each valid cell/feature from msk and in columns the output required by what argument.
inparallel	logical or integer. Should the function use parallel processing facilities? Default is FALSE: a single process will be launched. If TRUE, all cores/processors but one will be used. Alternatively, an integer can be provided to dictate the number of cores/processors to be used.
export	Logical or character. Should the spatial output be exported to file? Applies only if argument spatial=TRUE. When export=TRUE, output will be exported in .tif (raster) or .shp (vector) format. For rasters, a character specifying another extension can be provided, in which case the raster will be exported in that format. Only formats listed by writeFormats are valid. Argument exportRaster is deprecated.
path	The directory to store the output files, when export is not FALSE. Default is the working directory as from getwd(). File names are set by a default naming convention, see Details.
field	character. Only for spatial vector data (e.g. shapefile), the field/column names in the attribute table corresponding to the nodes, ordered accordingly.
verbose	logical. If verbose = TRUE a summary of class boundaries and associated nodes and data will be printed to screen for checks.
...	Additional arguments to force one or more nodes to a state (i.e. fixing evidence). If the node is associated to any input spatial data, the latter will be ignored, thus resulting spatially equal everywhere. Node name must be provided as argument and the associated fixed state as character; both node and state names must be typed exactly as their names in the network.
exportRaster	deprecated, use export instead.

Details

bnspatial

The expected value is calculated by summing the mid values of target node states weighted by their probability: `p1 * midVal_1 + p2 * midval_2 + ... + pn * midval_n`
When a spatial object is exported to a file, the file name is set by default, accordingly to the following naming convention:

- `"class"` <*target node name*>_Class.<*file format -default .tif*>
- `"entropy"` <*target node name*>_ShanEntropy.<*file format -default .tif*>
- `"probability"` <*target node name*>_Probability_.<*targetState*>.<*file format -default .tif*>
- `"expected"` <*target node name*>_ExpectedValue.<*file format -default .tif*>
- `"variation"` <*target node name*>_CoefVariation.<*file format -default .tif*>

An additional comma separated file (.csv) is written to the same directory when `"class"`, providing a key to interpret the spatial object values and the state they refer to.

Value

A list of "RasterLayer" or "SpatialPolygonsDataFrame" objects or a data.frame, depending on input arguments: see `mapTarget`. Some basic information about discretization and network/data link are printed on screen during execution.

See Also

`setClasses`; `mapTarget`; `linkNode`; `loadNetwork`

Examples

```
list2env(ConwyData, environment())

network <- LandUseChange
spatialData <- c(ConwyLU, ConwySlope, ConwyStatus)
lookup <- LUclasses

bn <- bnspatial(network, 'FinalLULC', spatialData, lookup)
bn
```

ConwyData *Land use change data*

Description

Data derived from the Conwy catchment in North Wales (UK), widely modified for demonstration purposes. Once loaded, the data consist of several objects:

- LandUseChange An object of class `grain`. The Bayesian network, built for demonstration purposes.

- ConwyLU An object of class RasterLayer. A simplified version of the current land use map from the Conwy catchment (Wales, UK). It includes three classes: arable (raster value 3), forest (2), other (1).
- ConwySlope An object of class RasterLayer. A raster of slope derived from a digital elevation model at 50 meters resolution, units are degrees.
- ConwyStatus An object of class RasterLayer. The land ownership type (dummy data), divided into three possible classes: public (raster value 4), private (3), protected (1).
- evidence A matrix. The collection of available spatial data (see above) as extracted from each location (i.e. cell) in the catchment, where the latter is represented by the raster object ConwyLU. Each value from the spatial data was discretized through dataDiscretize or bulkDiscretize functions, then assigned to the corresponding state from the Bayesian network (LandUseChange).
- LUclasses A list with the classification of input spatial data (its corresponding states and values). The list is formatted accordingly to bnspatial functions requirement and as returned by functions importClasses and setClasses.

Usage

```
data(ConwyData)
```

Format

A dataset in native RData format.

Examples

```
library(bnspatial)
data(ConwyData)
list2env(ConwyData, environment())
ls()

## The network nodes and states
LandUseChange$universe$levels

## Lookup list relating raster values and network nodes
LUclasses

## Table of evidence extracted from input spatial data
head(evidence, 12)

## The input spatial data (raster format)
par(mfrow=c(2,2))
raster::plot(ConwyLU)
raster::plot(ConwySlope)
raster::plot(ConwyStatus)

## The input spatial data (vector format)
Conwy <- sf::st_read(system.file("extdata", "Conwy.shp", package = "bnspatial"), quiet = TRUE)
# plot(Conwy) # May be slow to show up
```

dataDiscretize *Discretize data*

Description

These functions discretize continuous input data into classes. Classes can be defined by the user or, if the user provides the number of expected classes, calculated from quantiles (default option) or by equal intervals.
`dataDiscretize` processes a single variable at a time, provided as vector. `bulkDiscretize` discretizes multiple input rasters, optionally by using parallel processing.

Usage

```
dataDiscretize(
  data,
  classBoundaries = NULL,
  classStates = NULL,
  method = "quantile"
)

bulkDiscretize(formattedLst, xy, inparallel = FALSE)
```

Arguments

`data`	numeric vector. The continuous data to be discretized.
`classBoundaries`	numeric vector or single integer. Interval boundaries to be used for data discretization. Outer values (minimum and maximum) required. `-Inf` or `Inf` are allowed, in which case data minimum and maximum will be used to evaluate the mid values of outer classes. Alternatively, a single integer to indicate the number of classes, to split by quantiles (default) or equal intervals.
`classStates`	vector. The state labels to be assigned to the discretized data.
`method`	character. What splitting method should be used? This argument is ignored if a vector of values is passed to `classBoundaries`. • `quantile` splits data into quantiles (default). • `equal` splits data into equally sized intervals based on data minimum and maximum.
`formattedLst`	A formatted list as returned by `linkNode` and `linkMultiple`
`xy`	matrix. A matrix of spatial coordinates; first column is x (longitude), second column is y (latitude) of locations (in rows).
`inparallel`	logical or integer. Should the function use parallel processing facilities? Default is FALSE: a single process will be launched. If TRUE, all cores/processors but one will be used. Alternatively, an integer can be provided to dictate the number of cores/processors to be used.

Details

dataDiscretize

Value

`dataDiscretize` returns a named list of 4 vectors:

- `$discreteData`the discretized data, labels are applied accordingly if `classStates` argument is provided
- `$classBoundaries`the class boundaries, i.e. values splitting the classes
- `$midValues`the mid point for each class (the mean of its lower and upper boundaries)
- `$classStates`the labels assigne to each class

`bulkDataDiscretize` returns a matrix: in columns each node associated to input spatial data, in rows their discretized values at coordinates specified by argument `xy`.

Examples

```
s <- runif(30)

# Split by user defined values. Values out of boundaries are set to NA:
dataDiscretize(s, classBoundaries = c(0.2, 0.5, 0.8))

# Split by quantiles (default):
dataDiscretize(s, classStates = c('a', 'b', 'c'))

# Split by equal intervals:
dataDiscretize(s, classStates = c('a', 'b', 'c'), method = "equal")

# When -Inf and Inf are provided as external boundaries, $midValues of outer classes
# are calculated on the minimum and maximum values:
dataDiscretize(s, classBoundaries=c(0, 0.5, 1), classStates=c("first", "second"))[c(2,3)]
dataDiscretize(s, classBoundaries=c(-Inf, 0.5, Inf), classStates=c("first", "second"))[c(2,3)]

## Discretize multiple spatial data by location
list2env(ConwyData, environment())

network <- LandUseChange
spatialData <- c(ConwyLU, ConwySlope, ConwyStatus)

# Link multiple spatial data to the network nodes and discretize
spDataLst <- linkMultiple(spatialData, network, LUclasses, verbose = FALSE)
coord <- aoi(ConwyLU, xy=TRUE)
head( bulkDiscretize(spDataLst, coord) )
```

`extractByMask` *Extract raster values by mask*

Description

This function extracts the values from a given input raster based on a mask.

Usage

```
extractByMask(layer, msk, spatial = FALSE, rast = NULL)
```

Arguments

`layer`	an object of class "RasterLayer" (package raster). The raster from which data will be extracted
`msk`	an object of class "RasterLayer" or a two column matrix of coordinates. The reference raster (or coordinates) to be used as mask for extraction.
`spatial`	logical. Should the output be spatially explicit -i.e. a georeferenced raster? Default is FALSE, returning a vector of extracted values from `rast`. If TRUE an object of class "RasterLayer" is returned.
`rast`	deprecated, use `layer` instead.

Details

When input data given to `rast` does not match the resolution and extent of a raster mask argument, the latter is preferred. The function will therefore return a vector of n elements, one for each non NA cell in the mask. Input raster cells falling inside mask cells, but not over their cells centre will be ignored.

Value

a vector, or an object of class "RasterLayer". The values from the input raster (`rast` argument) at coordinates provided as matrix, or those overlapping with non NA cells in the mask raster. If `spatial == TRUE` an object of class "RasterLayer" is returned.

See Also

`aoi`

Examples

```
data(ConwyData)
list2env(ConwyData, environment())

m <- aoi(msk=ConwyLU, mskSub=c(2,3))
head( extractByMask(ConwySlope, msk=m), 20)
```

```
# Extract making a raster
library(raster)
plot( extractByMask(ConwySlope, msk=m, spatial=TRUE) )
```

`linkNode` *Link nodes to spatial data*

Description

`linkNode` links a node of the Bayesian network to the corresponding spatial data, returning a list of objects, including the spatial data and relevant info about the node.
`linkMultiple` operates on multiple nodes and related spatial data.

Usage

```
linkNode(
  layer,
  network,
  node,
  intervals,
  categorical = NULL,
  field = NULL,
  verbose = TRUE,
  spatial = TRUE
)

linkMultiple(spatialData, network, lookup, field = NULL, verbose = TRUE)
```

Arguments

`layer`	character (path to file) or an object of class "RasterLayer", "sf" or "SpatialPolygonsDataFrame". The spatial data corresponding to the network node as by argument `node`.
`network`	The Bayesian network. An object of class `grain`, or a character (the path to the *.net* file to be loaded)
`node`	character. The network node to be coupled with the file/object indicated by `layer` argument
`intervals`	A list of numeric vectors. For categorical variables the spatial data values associated to each state of the node, for continuous variables the boundary values dividing into the corresponding states, including upper and lower boundaries.
`categorical`	logical, or NULL. Is the node a categorical variable? If NULL the function will attempt to assign the logical value by looking at `intervals` argument.
`field`	character. Only for spatial vector data (e.g. shapefile), the field/column names in the attribute table corresponding to the nodes, ordered accordingly.
`verbose`	logical. If `verbose = TRUE` a summary of class boundaries and associated nodes and data will be printed to screen for checks.

spatial	logical. Should the output list include the input as raster or spatial vector? Default is TRUE, returning the list with an object of class "RasterLayer" or "sf" for each node associated. If FALSE, returns only the values.
spatialData	character with path to one or more raster files or to a single spatial vector file, or a list of objects of class 'RasterLayer' (for raster), or a single object of class 'sf' or 'SpatialPolygonsDataFrame' (for spatial vector). The spatial data associated to given network nodes, provided as file paths or as (list of) spatial objects. Items must be ordered accordingly to the corresponding nodes listed in lookup, or provided as named list, where names match exactly to the corresponding nodes name.
lookup	character (path to file) or a formatted list. This argument can be provided as path to a comma separated file or a formatted list (see setClasses)

Details

In future releases, this function may be rewritten to provide an S4/S3 object.

Value

linkNode returns a list of objects, including the spatial data and the related node information. linkMultiple returns a list of lists. Each element of the list includes the spatial data and summary information for each of the input nodes.

See Also

dataDiscretize; setClasses

Examples

```
## Load data into global environment
list2env(ConwyData, environment())
lookup <- LUclasses

network <- LandUseChange
ln <- linkNode(layer=ConwyLU, network, node='CurrentLULC', intervals=c(2, 3, 1))
ln

## Link the Bayesian network to multiple spatial data at once, using a lookup list
spatialData <- c(ConwyLU, ConwySlope, ConwyStatus)
linkMultiple(spatialData, network, lookup, verbose = FALSE)

## Method for spatial vectorial data (i.e. class 'sf' or 'SpatialPolygon')
spatialData <- system.file("extdata", "Conwy.shp", package = "bnspatial")
lst <- linkMultiple(spatialData, network, lookup, field= c('LU', 'Slope', 'Status'))
lst
```

`loadNetwork` *Load a Bayesian network*

Description

This function loads the Bayesian network from a native gRain object of class `grain` or an external file with extension *.net* (as provided by external softwares Hugin or GeNIe), optionally compiling the network.

Usage

```
loadNetwork(network, target = NULL)
```

Arguments

`network`	The Bayesian network. An object of class `grain`, or a character (the path to the *.net* file to be loaded)
`target`	character. The node of interest to be modelled and mapped.

Details

Bayesian networks built with the package bnlearn can be imported with the function `bnlearn::as.grain`, which converts them into `grain` objects.
.net file format as provided from Netica 5.24 currently does not correspond to a valid Hugin .net file.
Argument `target` has default set to NULL, but if provided the network will be compiled for faster querying.

Value

An object of class `grain`. The Bayesian network. If `target` argument is provided the network is compiled for a faster querying .

Note

Under current release, this function wraps a set of hidden functions copied in block from the gRain package, as current CRAN policy discourages accessing hidden functions with the ":::" operator. These functions will be progressively substituted by bnspatial native ones.

Examples

```
## Load from external file (.net format)
raw = system.file("extdata/LandUseChange.net", package = "bnspatial")
loadNetwork(raw)

## Compile using target node
loadNetwork(raw, 'FinalLULC')
```

`mapTarget` *Make maps for target node*

Description

This function creates the required spatial outputs for the target node.

Usage

```
mapTarget(
  target,
  statesProb,
  what = c("class", "entropy"),
  msk,
  midvals = NULL,
  targetState = NULL,
  spatial = TRUE,
  export = FALSE,
  path = getwd(),
  exportRaster = export
)
```

Arguments

`target`	character. The node of interest to be modelled and mapped.
`statesProb`	matrix. The probability matrix as returned by `queryNet` and `queryNetParallel`. Columns must be named accordingly to states of the target node. Columns are the `target` node states and rows each location considered from the area of interest.
`what`	character. The required output, one or more of these are valid: • `"class"` returns the relatively most likely state. • `"entropy"` calculates the Shannon index and returns the entropy, given the node probabilities. • `"probability"` returns an object for each state of the target node, with its probability. • `"expected"` gives the expected value for the target node (see Details). Only valid for target nodes of continuous values. `midValues` argument must be provided. • `"variation"` returns the coefficient of variation, as a measure of uncertainty. Only valid for target nodes of continuous values.
`msk`	an object of class "RasterLayer" (raster) or a spatial vector of class "sf" (vector spatial polygons). The reference spatial boundaries to be used as mask. All model outputs will have the same extent/outline as this object. All locations with no data (i.e. NA) will be ignored.

midvals	vector of length equal to the number of states of the target node. Applies only if the target node is a continuous variable, in which case midvals must contain the mid values for each of the intervals
targetState	character. One or more states of interest from the target node. Applies only when argument what includes 'probability'. Default is set to all states of the node.
spatial	logical. Should the output be spatially explicit -i.e. a georeferenced raster or spatial vector? Default is TRUE, returning an object of class "RasterLayer" or "sf" for polygons. If FALSE, returns a data frame with one row for each valid cell/feature from msk and in columns the output required by what argument.
export	Logical or character. Should the spatial output be exported to file? Applies only if argument spatial=TRUE. When export=TRUE, output will be exported in .tif (raster) or .shp (vector) format. For rasters, a character specifying another extension can be provided, in which case the raster will be exported in that format. Only formats listed by writeFormats are valid. Argument exportRaster is deprecated.
path	The directory to store the output files, when export is not FALSE. Default is the working directory as from getwd(). File names are set by a default naming convention, see Details.
exportRaster	deprecated, use export instead.

Details

mapTarget

The expected value is calculated by summing the mid values of target node states weighted by their probability: p1 * midVal_1 + p2 * midval_2 + ... + pn * midval_n
When exporting to a file, the file name is set by default, according to the following naming convention:

- "class" <*target node name*>_Class.<*file format -default .tif*>
- "entropy" <*target node name*>_ShanEntropy.<*file format -default .tif*>
- "probability" <*target node name*>_Probability_.<*targetState*>.<*file format -default .tif*>
- "expected" <*target node name*>_ExpectedValue.<*file format -default .tif*>
- "variation" <*target node name*>_CoefVariation.<*file format -default .tif*>

An additional comma separated file (.csv) is written to the same directory when "class", providing a key to interpret the values and the state they refer to.

Value

A list of objects, one for each item required in what argument. If spatial = TRUE a list of rasters of class "RasterLayer" are returned, or a single spatial vector of class "sf" with one column for each output requested. If FALSE, for raster data it returns a list of vectors with the values associated to each non NA cell in msk raster (i.e. the vectorised raster). For vector data it returns a data frame. If argument export is specified, outputs are exported to files to the directory specified in path.

See Also

bnspatial, aoi, queryNet

Examples

```
list2env(ConwyData, environment())

network <- LandUseChange
target <- 'FinalLULC'
statesProb <- queryNet(network, target, evidence)

maps <- mapTarget(target, statesProb, msk=ConwyLU)

library(raster)
plot(maps$Class)
plot(maps$Entropy)

## Returns required outputs by coordinates for each 'msk' cell in a data frame:
noMap <- mapTarget(target, statesProb, msk=ConwyLU, spatial=FALSE)
head(noMap)

## Create a probability surface for the "forest" state of target node "FinalLULC"
mp <- mapTarget('FinalLULC', statesProb, what='probability', targetState='forest', msk=ConwyLU)
plot(mp$Probability$forest)

## With spatial vector (totally made up data here, just for demo):
library(sf)
spVector <- st_read(system.file("extdata", "Conwy.shp", package = "bnspatial"))
ev <- evidence[1:nrow(spVector), ]

probs <- queryNet(network, 'FinalLULC', ev)
mp <- mapTarget('FinalLULC', statesProb=probs,
                what=c('entropy', 'probability'), targetState='forest', msk=spVector)
```

---

queryNet *Query the Bayesian network*

Description

This function queries the Bayesian network and returns the probabilities for each state of the target node. Available input variables are set as evidence.
`queryNetParallel` works as `queryNet`, but makes use of multi cores/processors facilities for big network queries, by splitting data into chunks and processing them in parallel.

Usage

```
queryNet(network, target, evidence, ...)

queryNetParallel(network, target, evidence, inparallel = TRUE, ...)
```

Arguments

network	The Bayesian network. An object of class grain, or a character (the path to the *.net* file to be loaded)
target	character. The node of interest to be modelled and mapped.
evidence	matrix or data.frame. Named columns are the known input variables; rows are the discrete states associated to them for each record (NA allowed).
...	Additional arguments to force one or more nodes to a state (i.e. fixing evidence). If the node is associated to any input spatial data, the latter will be ignored, thus resulting spatially equal everywhere. Node name must be provided as argument and the associated fixed state as character; both node and state names must be typed exactly as their names in the network.
inparallel	logical or integer. Number of cores to be used by queryNetParallel. Default is TRUE, so the maximum number available minus one is set.

Value

A matrix of probabilities: columns are the states of the target node and rows are the probabilities associated to each record (i.e. spatial locations) from evidence.

Examples

```
list2env(ConwyData, environment())

network <- LandUseChange

q <- queryNet(network, 'FinalLULC', evidence)
head(q)

## Fix a given node on a state (i.e. fixed evidence) by providing an additional argument
q <- queryNet(network, 'FinalLULC', evidence, Stakeholders = 'farmers')
head(q)

## Fix evidence for two nodes, including one of the spatial inputs (i.e. overriden by evidence)
q <- queryNet(network, 'FinalLULC', evidence, Stakeholders = 'farmers', CurrentLULC = 'forest')
head(q)
## For a programmatic approach, the arguments could be passed as named list:
# lst <- list(Stakeholders = 'farmers', CurrentLULC = 'forest')
# queryNet(network, 'FinalLULC', evidence, lst)

## Use parallel processing
q <- queryNetParallel(network, 'FinalLULC', evidence, inparallel=2)
head(q)
```

`setClasses` *Set classes or intervals*

Description

Functions `setClasses` and `importClasses` return a formatted list from given arguments, to be used for the integration and error checking of Bayesian network and input spatial variables. For `setClasses` a vector with node names and a list of vectors for both states of nodes and (optional) their boundaries in the spatial data must be provided, in the right order. For `importClasses` a formatted text file must be provided (see Details).

Usage

```
setClasses(nodes, states, classBoundaries, wr = NULL, layer = NULL)

importClasses(classFile)
```

Arguments

`nodes`	character. The nodes known and available as spatial data.
`states`	A list of characters. The states associated to each of the nodes (order must match `nodes` names).
`classBoundaries`	A list of the boundary values splitting the nodes into their corresponding states. They must be sorted in ascending order. For nominal categorical variables of raster data, `classBoundaries` must be the unique raster values associated to node states.
`wr`	character. Optional, the full path to the file to be written. Default is set to NULL, otherwise it writes the formatted list returned by `setClasses` to the specified path. Suggested file format is .txt, albeit not mandatory.
`layer`	character. Optional argument to indicate the path to files with input spatial data. If not NULL, then all nodes must have a corresponding file path, stored in the 'layer' element of output list.
`classFile`	character. A text file where for each input variable associated to a node (see Details) three lines are specified as follows: the first one indicates the node name, as in the Bayesian network; the second indicates the states associated with such node, as they are in the Bayesian network (note that underscores are not allowed); the third one contains the values associated to each state in the spatial data (for discrete variables) or the class boundaries dividing the states (for continuous variables), including minimum and maximum.

Details

As a reference for the text file format required by `importClasses`, for each node of the network:
First line: the node name.
Second line: the node states, comma separated (spaces allowed). NOTE: commas are NOT allowed

inside the state names.
Third line: interval values from the spatial data associated to the states (integer values for discrete data; interval boundaries, including endpoints, for continuous data). The same exact order as node states is required.

For example:

```
CurrentLULC
forest,other,arable
2, 1, 3
Slope
flat, moderate, steep
-Inf, 1, 7, Inf
LegalStatus
public, private, protected
4, 3, 1
```

It is possible to write the formatted file automatically using `setClasses`, by setting argument `wr` as path to the text file to be created.

Value

A formatted list, specifying states break values for continuous nodes and integer values for categorical nodes.

See Also

dataDiscretize

Examples

```
## Load classes from external formatted text file
# Not run: importClasses('LUclasses.txt')
raw = system.file("extdata/LUclasses.txt", package = "bnspatial")
importClasses(raw)

## Same as:

setClasses(c('Slope', 'CurrentLULC', 'LegalStatus'), list(c('flat', 'moderate', 'steep'),
c('forest', 'arable', 'other'), c('public', 'private', 'protected')),
list(c(-Inf, 0, 5, Inf), c(2, 3, 1), (c(4, 3, 1))))
```

Index