Package 'bnspatial'

July 27, 2016

Title Spatial Implementation of Bayesian Networks and Mapping

Version 1.0

Date 2016-07-31

Copyright Natural Environment Research Council (NERC) and Centre for Ecology and Hydrology (CEH)

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

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

Imports raster, gRbase, gRain, doParallel, foreach

Suggests knitr, rmarkdown

Description Package for the 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. Development was funded by the European Union FP7 (2007-2013), under project ROBIN (agreement 283093).

License GPL-3

LazyLoad yes

NeedsCompilation no

RoxygenNote 5.0.1

VignetteBuilder knitr

Author Dario Masante [aut, cre]

Maintainer Dario Masante <dmasan@ceh.ac.uk>

Repository CRAN

Date/Publication 2016-07-27 12:01:36

R topics documented:

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

Description

This function creates a raster defining the area of interest, by unioning the input rasters or using a user defined mask. When `msk` is specified, resolution and extent are set equal to it, otherwise to the finest resolution among input spatial data and unioning the extents of input spatial data.

Usage

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

Arguments

`msk`	a character (path to raster file), a raster (object of class "RasterLayer"), or a list of rasters. The reference raster(s) to be used as mask. All model outputs will have the same resolution and same extent as this raster(s). All locations with no data (i.e. NA) cells in `msk` will be ignored as well.
`mskSub`	vector. The subset values from `msk` which should be considered to build the area of interest. All other values will be ignored and returned as NA.
`xy`	logical. Should return a two column matrix of coordinates? If FALSEan object of class RasterLayer is returned.

Details

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 (default), or a matrix of coordinates of mask cells. 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:
data(ConwyData)
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)
```

---

bnspatial *Spatialize the Bayesian network*

Description

This function wraps most bnspatial package functions to ease the spatial implementation of Bayesian networks with minimal user interaction.

Usage

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

Arguments

network	The Bayesian network. An object of class grain (from package gRain), or a character (the path to the *.net* file to be imported)
target	character. The node of interest to be modelled and mapped.
spatialData	character or list of objects of class 'RasterLayer'. The raster files corresponding to nodes, as vector of full file paths or as list of rasters (objects of class 'RasterLayer').
lookup	character 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 file), a raster (object of class "RasterLayer"), or a list of rasters. The reference raster(s) to be used as mask. All model outputs will have the same resolution and same extent as this raster(s). All locations with no data (i.e. NA) cells in msk will be ignored as well.
what	character. The required output, one or more of these values are valid: • "class" returns the relatively most likely states. • "entropy" calculates the Shannon index and returns the entropy given the state probabilities. • "probability" returns an object for each state of the target node, with associated probability. • "expected" gives the expected value for the target node (see Details). Only valid for continuous target nodes. midValues argument must be provided. • "variation" returns the coefficient of variation, as a measure of uncertainty.
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? Default is TRUE, returning an object of class "RasterLayer". If FALSE, returns a data.frame with one row for each non NA cell in msk raster and in columns the output required by mask 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.
exportRaster	Logical or character. Should the spatial output be exported to a raster file? Applies only if argument spatial=TRUE. When exportRaster=TRUE, rasters will be exported in .tif format. 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.
path	The directory to store the output files, when exportRaster is not FALSE. Default is the working directory (getwd()). File names are set by a default naming convention, see Details.

verbose	logical. If `verbose = TRUE` a summary of class boundaries and associated nodes and data will be printed to screen for quick checks.
...	Additional arguments to fix a state (i.e. setting evidence) to one or more nodes, as known and independent from any spatial data (e.g. the case of non-spatial variables which are equal everywhere). Node name is provided as argument and the associated fixed state as character; both node and state names must be typed accordingly to their names in the network.

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 RasterLayer 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 raster values and the state they refer to.

Value

A list of "RasterLayer" 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

```
data(ConwyData)
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.
- 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)
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)
```

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

ConwyLU *Current land use map (heavily aggregated)*

Description

An object of class RasterLayer, sourced from *extdata/ConwyLU.tif*. 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 *Slope raster for Conwy catchment.*

Description

An object of class RasterLayer, sourced from *extdata/ConwySlope.tif*. A raster of slope derived from a digital elevation model at 50 meters resolution, units are degrees.

ConwyStatus *Ownership/legal status raster for Conwy catchment (dummy data).*

Description

An object of class RasterLayer, sourced from *extdata/ConwyStatus.tif*. The land ownership type (dummy data), divided into three possible classes: public (raster value 4), private (3), protected (1).

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, 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:

- $discreteDatathe discretized data, labels are applied accordingly if classStates argument is provided
- $classBoundariesthe class boundaries, i.e. values splitting the classes
- $midValuesthe mid point for each class (the mean of its lower and upper boundaries)
- $classStatesthe 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
data(ConwyData)
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) )
```

---

evidence *Evidence from extracted spatial inputs.*

Description

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).

`extractByMask` *Extract raster values by mask*

Description

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

Usage

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

Arguments

`rast`	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.

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)
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) )
```

LandUseChange *Bayesian network*

Description

Bayesian network built for demonstration purposes, inspired by works like Celio et al (2014).

linkNode *Link nodes to spatial data*

Description

`linkNode` links a node of the Bayesian network to its corresponding spatial dataset (in raster format), returning a list of objects, including the spatial data and relevant information about the node. `linkMultiple` operates on multiple rasters and nodes.

Usage

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

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

Arguments

`layer`	character (path to raster file) or an object of class "RasterLayer". The spatial data corresponding to the network node in argument `node`.
`network`	The Bayesian network. An object of class `grain` (from package gRain), or a character (the path to the *.net* file to be imported)
`node`	character. A network node associated to the file in `layer` argument
`intervals`	A list of numeric vectors. For categorical variables the raster values associated to each state of the node, for continuous variables the boundary values dividing into the corresponding states.
`categorical`	logical. Is the node a categorical variable? Default is NULL.
`verbose`	logical. If `verbose = TRUE` a summary of class boundaries and associated nodes and data will be printed to screen for quick checks.
`spatialData`	character or list of objects of class 'RasterLayer'. The raster files corresponding to nodes, as vector of full file paths or as list of rasters (objects of class 'RasterLayer').
`lookup`	character 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 object instead of a list.

Value

`linkNode` returns a list of objects, including the spatial data and summary information about each node.
`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

```
data(ConwyData)
network <- LandUseChange
lst <- linkNode(layer=ConwyLU, network, node='CurrentLULC', intervals=c(2, 3, 1))
lst

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

---

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` (from package gRain), or a character (the path to the *.net* file to be imported)
`target`	character. The node of interest to be modelled and mapped.

Details

Bayesian networks from the package bnlearn can be imported via the function as.grain, from package gRain.
.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')
```

LUclasses *Lookup list, linking input spatial data and the Bayesian network*

Description

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`.

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 = colnames(statesProb), spatial = TRUE,
  exportRaster = FALSE, path = getwd())
```

Arguments

target	character. The node of interest to be modelled and mapped.
statesProb	matrix. The probability matrix as returned by queryNet and queryNetParallel. Named columns required, accordingly to states. 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 values are valid: • "class" returns the relatively most likely states. • "entropy" calculates the Shannon index and returns the entropy given the state probabilities. • "probability" returns an object for each state of the target node, with associated probability. • "expected" gives the expected value for the target node (see Details). Only valid for continuous target nodes. midValues argument must be provided. • "variation" returns the coefficient of variation, as a measure of uncertainty.
msk	an object of class "RasterLayer". The reference raster to be used as mask. All model outputs will have the same resolution and same extent as this raster. All locations with no data (i.e. NA) cells in this raster will be ignored as well.
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? Default is TRUE, returning an object of class "RasterLayer". If FALSE, returns a data.frame with one row for each non NA cell in msk raster and in columns the output required by mask argument.

`exportRaster` Logical or character. Should the spatial output be exported to a raster file? Applies only if argument `spatial=TRUE`. When `exportRaster=TRUE`, rasters will be exported in .tif format. 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.

`path` The directory to store the output files, when `exportRaster` is not FALSE. Default is the working directory (`getwd()`). File names are set by a default naming convention, see Details.

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 a RasterLayer 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 raster 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, if FALSE a list of vectors with values associated to each non NA cell in msk raster (i.e. the vectorised raster). If argument `exportRaster` is specified, outputs are exported to files to the directory specified in `path`.

See Also

`bnspatial`, `aoi`, `queryNet`

Examples

```
data(ConwyData)
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)
```

`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` (from package gRain), or a character (the path to the *.net* file to be imported)
`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 fix a state (i.e. setting evidence) to one or more nodes, as known and independent from any spatial data (e.g. the case of non-spatial variables which are equal everywhere). Node name is provided as argument and the associated fixed state as character; both node and state names must be typed accordingly to their names in the network.
`inparallel`	logical or integer. Number of cores/processors 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 target node states and rows are the probabilities associated to each record from argument `evidence` (e.g. spatial locations).

Examples

```
data(ConwyData)
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)

## 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)

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 numeric. The boundary values splitting the nodes into their corresponding states. They must be sorted in ascending order. For nominal categorical variables, `classBoundaries` must be the unique raster values associated to node states.
`wr`	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.

`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).
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 `w` 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