Package 'gmgm'

February 25, 2026

Type Package

Title Gaussian Mixture Graphical Model Learning and Inference

Version 1.1.3

Description Gaussian mixture graphical models include Bayesian networks and dynamic Bayesian networks (their temporal extension) whose local probability distributions are described by Gaussian mixture models. They are powerful tools for graphically and quantitatively representing nonlinear dependencies between continuous variables. This package provides a complete framework to create, manipulate, learn the structure and the parameters, and perform inference in these models. Most of the algorithms are described in the PhD thesis of Roos (2018) <https://theses.hal.science/tel-01943718>.

Depends R (>= 3.5.0)

Imports dplyr (>= 1.0.5), ggplot2 (>= 3.2.1), purrr (>= 0.3.3), rlang (>= 0.4.10), stats (>= 3.5.0), stringr (>= 1.4.0), tidyr (>= 1.0.0), visNetwork (>= 2.0.8)

Suggests testthat (>= 2.3.2)

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.2.1

NeedsCompilation no

Author Jérémy Roos [aut, cre, cph], RATP Group [fnd, cph]

Maintainer Jérémy Roos <jeremy.roos@gmail.com>

Repository CRAN

Date/Publication 2026-02-25 21:40:03 UTC

Contents

gmgm-package *Gaussian mixture graphical model learning and inference*

Description

This package provides a complete framework to deal with Gaussian mixture graphical models, which covers Bayesian networks and dynamic Bayesian networks (their temporal extension) whose local probability distributions are described by Gaussian mixture models. It includes a wide range of functions for:

- creating or modifying the structure of Gaussian mixture models (add_var, gmm, merge_comp, remove_var, rename_var, reorder, split_comp) or graphical models (add_arcs, add_nodes, gmbn, gmdbn, relevant, remove_arcs, remove_nodes, rename_nodes);
- describing or visualizing Gaussian mixture models (conditional, ellipses, summary.gmm) or graphical models (network, structure, summary.gmbn, summary.gmdbn);
- computing densities, expectations, or sampling Gaussian mixture models (density, expectation, sampling);
- computing scores of Gaussian mixture models (AIC.gmm, BIC.gmm, logLik.gmm) or graphical models (AIC.gmbn, AIC.gmdbn, BIC.gmbn, BIC.gmdbn, logLik.gmbn, logLik.gmdbn);
- learning the structure and/or the parameters of Gaussian mixture models (em, smem, stepwise) or graphical models (param_em, param_learn, struct_em, struct_learn);
- performing inference in Gaussian mixture graphical models (aggregation, filtering, inference, particles, prediction, propagation, smoothing).

Descriptions of these functions are provided in this manual with related references. Most of the algorithms are described in the PhD thesis of Roos (2018, in french). To better handle this package, two real-world datasets are provided (data_air, data_body) with examples of Gaussian mixture models and graphical models (gmbn_body, gmdbn_air, gmm_body).

References

Roos, J. (2018). *Prévision a court terme des flux de voyageurs : une approche par les réseaux bayésiens*. PhD thesis, University of Lyon.

add_arcs *Add arcs to a Gaussian mixture graphical model*

Description

This function adds arcs to a Gaussian mixture graphical model. For each added arc, a variable related to the start node is added to the Gaussian mixture model describing the local distribution over the end node and its parents, with mean 0 and variance 1 for each mixture component.

Usage

```
add_arcs(gmgm, arcs)
```

Arguments

gmgm	An object of class gmbn or gmdbn.
arcs	A data frame containing the added arcs. The column from describes the start node, the column to the end node and the column lag the time lag between them. Missing values in from or to are interpreted as "all possible nodes", which allows to quickly define large set of arcs that share common attributes. Missing values in lag are replaced by 0. If gmgm is a gmdbn object, the same arcs are added to each of its gmbn elements. This constraint can be overcome by passing a list of data frames named after some of these elements (b_1, ...) and containing arcs specifically added to them. The arcs whose time lags exceed the maximum temporal depth of their gmbn element are not taken into account.

Value

The gmbn or gmdbn object after adding the arcs.

See Also

add_nodes, relevant, remove_arcs, remove_nodes, rename_nodes

Examples

```
data(gmbn_body)
gmbn_1 <- add_arcs(gmbn_body,
                   data.frame(from = c("GENDER", "AGE"),
                              to = c("GLYCO", "WEIGHT")))

data(gmdbn_air)
gmdbn_1 <- add_arcs(gmdbn_air,
                    list(b_2 = data.frame(from = "WIND", to = "NO2", lag = 1),
                         b_13 = data.frame(from = c("NO2", "NO2"),
                                           to = c("O3", "O3"), lag = c(0, 1))))
```

---

add_nodes *Add nodes to a Gaussian mixture graphical model*

Description

This function adds nodes to a Gaussian mixture graphical model. If this model is a dynamic Bayesian network, the nodes are added to each of its transition models. For each added node, a one-component univariate Gaussian mixture model is created with mean 0 and variance 1.

Usage

```
add_nodes(gmgm, nodes)
```

Arguments

gmgm	An object of class gmbn or gmdbn. If NULL, a gmbn object is created with the added nodes.
nodes	A character vector containing the added nodes.

Value

The gmbn or gmdbn object after adding the nodes.

See Also

add_arcs, relevant, remove_arcs, remove_nodes, rename_nodes

Examples

```
data(gmbn_body)
gmbn_1 <- add_nodes(gmbn_body, c("CHOL", "TRIGLY"))

data(gmdbn_air)
gmdbn_1 <- add_nodes(gmdbn_air, "PM10")
```

---

add_var *Add variables to a Gaussian mixture model*

Description

This function adds variables to a Gaussian mixture model.

Usage

```
add_var(gmm, var)
```

Arguments

gmm	An object of class gmm. If NULL, a gmm object is created with the added variables and one mixture component.
var	A character vector containing the added variables, or a data frame or numeric matrix whose columns are named after the added variables. In the first case, for each mixture component, the marginal mean vector of the added variables is 0 and the marginal covariance matrix the identity matrix. In the second case, these mean vector and covariance matrix are computed from the data (after removing the rows that contain missing values).

Value

The gmm object after adding the variables.

See Also

remove_var, rename_var

Examples

```
data(gmm_body, data_body)
gmm_1 <- add_var(gmm_body, "GENDER")
gmm_2 <- add_var(gmm_body, data_body[, "GENDER"])
```

---

aggregation *Aggregate particles to obtain inferred values*

Description

This function aggregates particles to obtain inferred values. Assuming that the particles have been propagated to a given time slice t, the weighted average of the samples is computed to estimate the state of the system at t or at previous time slices (Koller and Friedman, 2009).

Usage

```
aggregation(part, nodes, col_seq = NULL, col_weight = "weight", lag = 0)
```

Arguments

part	A data frame containing the particles propagated to time slice t, as obtained from function particles or propagation.
nodes	A character vector containing the inferred nodes.
col_seq	A character vector containing the column names of part that describe the observation sequence. If NULL (the default), all the particles belong to a single sequence.
col_weight	A character string corresponding to the column name of part that describes the particle weight.
lag	A non-negative integer vector containing the time lags $l_1, l_2, \ldots$ such that the samples of time slices $t - l_1, t - l_2, \ldots$ are aggregated.

Value

If lag has one element, a data frame (tibble) containing the aggregated values of the inferred nodes and their observation sequences (if col_seq is not NULL). If lag has two or more elements, a list of data frames (tibbles) containing these values for each time lag.

References

Koller, D. and Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press.

See Also

aggregation, particles

Examples

```
library(dplyr)
set.seed(0)
data(gmdbn_air, data_air)
evid <- data_air %>%
  group_by(DATE) %>%
  slice(1:3) %>%
  ungroup()
evid$NO2[sample.int(150, 30)] <- NA
evid$O3[sample.int(150, 30)] <- NA
evid$TEMP[sample.int(150, 30)] <- NA
evid$WIND[sample.int(150, 30)] <- NA
aggreg <- particles(data.frame(DATE = unique(evid$DATE))) %>%
  propagation(gmdbn_air, evid, col_seq = "DATE", n_times = 3) %>%
  aggregation(c("NO2", "O3", "TEMP", "WIND"), col_seq = "DATE", lag = c(0, 1))
```

AIC — *Compute the Akaike Information Criterion (AIC) of a Gaussian mixture model or graphical model*

Description

This function computes the Akaike Information Criterion (AIC) of a Gaussian mixture model or graphical model:

$$AIC = logLik - n_{par}$$

where $logLik$ is the log-likelihood and n_{par} the number of free parameters.

Usage

```
## S3 method for class 'gmm'
AIC(object, data, y = NULL, regul = 0.01, ...)

## S3 method for class 'gmbn'
AIC(object, data, col_seq = NULL, ...)

## S3 method for class 'gmdbn'
AIC(object, data, col_seq = NULL, ...)
```

Arguments

object	An object of class gmm, gmbn or gmdbn.
data	A data frame containing the data used to compute the AIC. Its columns must explicitly be named after the variables (or nodes) of object. If object is a gmm object, a numeric matrix can be passed.
y	A character vector containing the dependent variables if a conditional AIC is computed. If NULL (the default), the joint AIC is computed.
regul	A positive numeric value corresponding to the regularization constant if a penalty term is added for Bayesian regularization. If NULL, no penalty term is added. If a conditional AIC is computed, this argument is ignored.
...	Unused arguments from the generic function.
col_seq	A character vector containing the column names of data that describe the observation sequence. If NULL (the default), all the observations belong to a single sequence. If object is a temporal gmbn or gmdbn object, the observations of a same sequence must be ordered such that the tth one is related to time slice t (note that the sequences can have different lengths). If object is a non-temporal gmbn object, this argument is ignored.

Value

If object is a gmm object, a numeric value corresponding to the AIC.

If object is a gmbn or gmdbn object, a list with elements:

global	A numeric value corresponding to the global AIC.
local	For a gmbn object, a numeric vector containing the local conditional AICs. For a gmdbn object, a list of numeric vectors containing these values for each gmbn element.

See Also

BIC, logLik

Examples

```
data(gmm_body, data_body)
aic_1 <- AIC(gmm_body, data_body)
aic_2 <- AIC(gmm_body, data_body, y = "WAIST")

data(gmbn_body, data_body)
aic_3 <- AIC(gmbn_body, data_body)

data(gmdbn_air, data_air)
aic_4 <- AIC(gmdbn_air, data_air, col_seq = "DATE")
```

BIC *Compute the Bayesian Information Criterion (BIC) of a Gaussian mixture model or graphical model*

Description

This function computes the Bayesian Information Criterion (BIC) of a Gaussian mixture model or graphical model:

$$BIC = logLik - \frac{\log(n_{obs})}{2} n_{par}$$

where $logLik$ is the log-likelihood, n_{obs} the number of observations in the data and n_{par} the number of free parameters.

Usage

```
## S3 method for class 'gmm'
BIC(object, data, y = NULL, regul = 0.01, ...)

## S3 method for class 'gmbn'
BIC(object, data, col_seq = NULL, ...)

## S3 method for class 'gmdbn'
BIC(object, data, col_seq = NULL, ...)
```

Arguments

object	An object of class gmm, gmbn or gmdbn.
data	A data frame containing the data used to compute the BIC. Its columns must explicitly be named after the variables (or nodes) of object. If object is a gmm object, a numeric matrix can be passed.
y	A character vector containing the dependent variables if a conditional BIC is computed. If NULL (the default), the joint BIC is computed.
regul	A positive numeric value corresponding to the regularization constant if a penalty term is added for Bayesian regularization. If NULL, no penalty term is added. If a conditional BIC is computed, this argument is ignored.
...	Unused arguments from the generic function.
col_seq	A character vector containing the column names of data that describe the observation sequence. If NULL (the default), all the observations belong to a single sequence. If object is a temporal gmbn or gmdbn object, the observations of a same sequence must be ordered such that the tth one is related to time slice t (note that the sequences can have different lengths). If object is a non-temporal gmbn object, this argument is ignored.

Value

If `object` is a `gmm` object, a numeric value corresponding to the BIC.

If `object` is a `gmbn` or `gmdbn` object, a list with elements:

`global`	A numeric value corresponding to the global BIC.
`local`	For a `gmbn` object, a numeric vector containing the local conditional BICs. For a `gmdbn` object, a list of numeric vectors containing these values for each `gmbn` element.

See Also

`AIC`, `logLik`

Examples

```
data(gmm_body, data_body)
bic_1 <- BIC(gmm_body, data_body)
bic_2 <- BIC(gmm_body, data_body, y = "WAIST")

data(gmbn_body, data_body)
bic_3 <- BIC(gmbn_body, data_body)

data(gmdbn_air, data_air)
bic_4 <- BIC(gmdbn_air, data_air, col_seq = "DATE")
```

`conditional` *Conditionalize a Gaussian mixture model*

Description

This function conditionalizes a Gaussian mixture model (Sun *et al.*, 2006).

Usage

```
conditional(gmm, y = rownames(gmm$mu)[1])
```

Arguments

`gmm`	An object of class `gmm`.
`y`	A character vector containing the dependent variables (by default the first variable of `gmm`).

Value

A list with elements:

`alpha`	A numeric vector containing the mixture proportions.
`mu_x`	A numeric matrix containing the marginal mean vectors of the explanatory variables bound by column.
`sigma_x`	A list containing the marginal covariance matrices of the explanatory variables.
`coeff`	A list containing the regression coefficient matrices of the dependent variables on the explanatory variables.
`sigma_c`	A list containing the conditional covariance matrices.

References

Sun, S., Zhang, C. and Yu, G. (2006). A Bayesian Network Approach to Traffic Flow Forecasting. *IEEE Transactions on Intelligent Transportation Systems*, 7(1):124–132.

Examples

```
data(gmm_body)
cond <- conditional(gmm_body)
```

`data_air` *Beijing air quality dataset*

Description

This dataset includes hourly air pollutants and weather data measured at the Dongsi air quality monitoring site in Beijing (China) for 320 complete days of the year 2015. These data are taken from the Beijing Multi-Site Air Quality Dataset published in the UCI Machine Learning Repository: https://archive.ics.uci.edu/dataset/501/beijing+multi+site+air+quality+data (Zhang *et al.*, 2017).

Usage

```
data_air
```

Format

A data frame (tibble) with 7680 rows and 6 columns:

- `DATE`: day's date;
- `HOUR`: hour of the day;
- `NO2`: nitrogen dioxide concentration (µg/m³);
- `O3`: ozone concentration (µg/m³);
- `TEMP`: temperature (°C);
- `WIND`: wind speed (m/s).

References

Zhang, S., Guo, B., Dong, A., He, J., Xu, Z. and Chen, S. X. (2017). Cautionary Tales on Air-Quality Improvement in Beijing. *Proceedings of the Royal Society A*, 473.

See Also

data_body, gmbn_body, gmdbn_air, gmm_body

---

data_body *NHANES body composition dataset*

Description

This dataset includes body composition data measured in 2148 adults aged 20 to 59 years in the United States. These data are taken from the National Health and Nutrition Examination Survey (NHANES) 2017-2018: https://wwwn.cdc.gov/nchs/nhanes/continuousnhanes/default.aspx?BeginYear=2017 (Centers for Disease Control and Prevention, 2020).

Usage

```
data_body
```

Format

A data frame (tibble) with 2148 rows and 8 columns:

- ID: respondent identifier;
- GENDER: gender (0: male, 1: female);
- AGE: age (years);
- HEIGHT: height (cm);
- WEIGHT: weight (kg);
- FAT: body fat (%);
- WAIST: waist circumference (cm);
- GLYCO: glycohemoglobin (%).

References

Centers for Disease Control and Prevention (2020). National Health and Nutrition Examination Survey Data.

See Also

data_air, gmbn_body, gmdbn_air, gmm_body

density *Compute densities of a Gaussian mixture model*

Description

This function computes densities of a Gaussian mixture model.

Usage

```
density(gmm, data, y = NULL, log = FALSE)
```

Arguments

gmm	An object of class gmm.
data	A data frame or numeric matrix containing the observations whose densities are computed. Its columns must explicitly be named after the variables of gmm.
y	A character vector containing the dependent variables if conditional densities are computed. If NULL (the default), joint densities are computed.
log	A logical value indicating whether the densities are returned as log-densities.

Value

A numeric vector containing the (log-)densities.

See Also

expectation, sampling

Examples

```
data(gmm_body, data_body)
dens_1 <- density(gmm_body, data_body, log = TRUE)
dens_2 <- density(gmm_body, data_body, y = "WAIST", log = TRUE)
```

ellipses *Display the mixture components of a Gaussian mixture model*

Description

This function displays the mixture components of a Gaussian mixture model. For each pair of variables, the covariance matrices are represented by confidence ellipses.

Usage

```
ellipses(
  gmm,
  data = NULL,
  y = rownames(gmm$mu),
  x = rownames(gmm$mu),
  level = 0.95
)
```

Arguments

gmm	An object of class gmm.
data	A data frame or numeric matrix containing the data displayed with the mixture components. Its columns must explicitly be named after the variables of gmm. If NULL (the default), no data is displayed.
y	A character vector containing the variables displayed on the y-axis (by default all the variables of gmm).
x	A character vector containing the variables displayed on the x-axis (by default all the variables of gmm).
level	A numeric value in [0, 1[corresponding to the confidence level of the ellipses.

Value

A ggplot object displaying the mixture components (and the data if required).

Examples

```
set.seed(0)
data(gmm_body)
ellipses(gmm_body, sampling(gmm_body, n = 500))
```

---

em *Estimate the parameters of a Gaussian mixture model*

Description

This function estimates the parameters of a Gaussian mixture model using the expectation-maximization (EM) algorithm. Given an initial model, this algorithm iteratively updates the parameters, monotonically increasing the log-likelihood until convergence to a local maximum (Bilmes, 1998). A Bayesian regularization is applied by default to prevent that a mixture component comes down to a single point and leads to a zero covariance matrix (Ormoneit and Tresp, 1996). Although the EM algorithm only applies to the joint model, good parameters can be found for a derived conditional model. However, care should be taken as the monotonic increase of the conditional log-likelihood is not guaranteed.

Usage

```
em(
  gmm,
  data,
  regul = 0.01,
  epsilon = 1e-06,
  max_iter_em = 100,
  verbose = FALSE
)
```

Arguments

gmm	An initial object of class gmm.
data	A data frame or numeric matrix containing the data used in the EM algorithm. Its columns must explicitly be named after the variables of gmm and must not contain missing values.
regul	A positive numeric value corresponding to the regularization constant if a Bayesian regularization is applied. If NULL, no regularization is applied.
epsilon	A positive numeric value corresponding to the convergence threshold for the increase in log-likelihood.
max_iter_em	A non-negative integer corresponding to the maximum number of iterations.
verbose	A logical value indicating whether iterations in progress are displayed.

Value

A list with elements:

gmm	The final gmm object.
posterior	A numeric matrix containing the posterior probabilities for each observation.
seq_loglik	A numeric vector containing the sequence of log-likelihoods measured initially and after each iteration.

References

Bilmes, J. A. (1998). A Gentle Tutorial of the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models. Technical report, International Computer Science Institute.

Ormoneit, D. and Tresp, V. (1996). Improved Gaussian Mixture Density Estimates Using Bayesian Penalty Terms and Network Averaging. In *Advances in Neural Information Processing Systems 8*, pages 542–548.

See Also

smem, stepwise

Examples

```
data(data_body)
gmm_1 <- split_comp(add_var(NULL,
                            data_body[, c("WAIST", "AGE", "FAT", "HEIGHT",
                                          "WEIGHT")]),
                    n_sub = 3)
res_em <- em(gmm_1, data_body, verbose = TRUE)
```

`expectation` *Compute expectations of a Gaussian mixture model*

Description

This function computes expectations of a Gaussian mixture model.

Usage

```
expectation(gmm, data_x = NULL)
```

Arguments

`gmm`	An object of class `gmm`.
`data_x`	A data frame or numeric matrix containing observations of the explanatory variables if conditional expectations are computed. Its columns must explicitly be named after the explanatory variables. If `NULL` (the default), the joint expectation is computed in a one-row matrix.

Value

A numeric matrix containing the expectations.

See Also

`density`, `sampling`

Examples

```
data(gmm_body, data_body)
expect_1 <- expectation(gmm_body)
expect_2 <- expectation(gmm_body,
                        data_body[, c("WEIGHT", "FAT", "HEIGHT", "AGE")])
```

`filtering` *Perform filtering inference in a Gaussian mixture dynamic Bayesian network*

Description

This function performs filtering inference in a Gaussian mixture dynamic Bayesian network. For a sequence of T time slices, this task consists in estimating the state of the system at each time slice t (for $1 \leq t \leq T$) given all the data (the evidence) collected up to t. This function is also designed to perform fixed-lag smoothing inference, which consists in defining a time lag l such that at each time slice t (for $l + 1 \leq t \leq T$), the state at $t - l$ is estimated given the evidence collected up to t (Murphy, 2002). Filtering and fixed-lag smoothing inference are performed by sequential importance resampling, which is a particle-based approximate method (Koller and Friedman, 2009).

Usage

```
filtering(
  gmdbn,
  evid,
  nodes = names(gmdbn$b_1),
  col_seq = NULL,
  lag = 0,
  n_part = 1000,
  max_part_sim = 1e+06,
  min_ess = 1,
  verbose = FALSE
)
```

Arguments

`gmdbn`	An object of class `gmdbn`.
`evid`	A data frame containing the evidence. Its columns must explicitly be named after nodes of `gmdbn` and can contain missing values (columns with no value can be removed).
`nodes`	A character vector containing the inferred nodes (by default all the nodes of `gmdbn`).
`col_seq`	A character vector containing the column names of `evid` that describe the observation sequence. If `NULL` (the default), all the observations belong to a single sequence. The observations of a same sequence must be ordered such that the tth one is related to time slice t (note that the sequences can have different lengths).
`lag`	A non-negative integer vector containing the time lags for which fixed-lag smoothing inference is performed. If `0` (the default), filtering inference is performed.
`n_part`	A positive integer corresponding to the number of particles generated for each observation sequence.

max_part_sim	An integer greater than or equal to n_part corresponding to the maximum number of particles that can be processed simultaneously. This argument is used to prevent memory overflow, dividing evid into smaller subsets that are handled sequentially.
min_ess	A numeric value in [0, 1] corresponding to the minimum ESS (expressed as a proportion of n_part) under which the renewal step of sequential importance resampling is performed. If 1 (the default), this step is performed at each time slice.
verbose	A logical value indicating whether subsets of evid and time slices in progress are displayed.

Value

If lag has one element, a data frame (tibble) with a structure similar to evid containing the estimated values of the inferred nodes and their observation sequences (if col_seq is not NULL). If lag has two or more elements, a list of data frames (tibbles) containing these values for each time lag.

References

Koller, D. and Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press.

Murphy, K. (2002). *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, University of California.

See Also

inference, prediction, smoothing

Examples

```
set.seed(0)
data(gmdbn_air, data_air)
evid <- data_air
evid$NO2[sample.int(7680, 1536)] <- NA
evid$O3[sample.int(7680, 1536)] <- NA
evid$TEMP[sample.int(7680, 1536)] <- NA
evid$WIND[sample.int(7680, 1536)] <- NA
filt <- filtering(gmdbn_air, evid, col_seq = "DATE", lag = c(0, 1),
                  verbose = TRUE)
```

---

gmbn *Create a Gaussian mixture Bayesian network*

Description

This function creates a Gaussian mixture Bayesian network as an object of S3 class gmbn. A Bayesian network is a probabilistic graphical model that represents the conditional dependencies and independencies between random variables by a directed acyclic graph. It encodes a global joint distribution over the nodes, which decomposes into a product of local conditional distributions:

$$p(X_1, \ldots, X_n) = \prod_{i=1}^{n} p(X_i | Pa(X_i))$$

where $Pa(X_i)$ is the set of parents of X_i in the graph. In a Gaussian mixture Bayesian network, each local joint distribution over a node and its parents is described by a Gaussian mixture model, which means that the global distribution is a product of local conditional Gaussian mixture models (Davies and Moore, 2000). The gmbn class can be extended to the time factor by regarding the nodes as the state of the system at a given time slice t (denoted by $X^{(t)}$) and allowing them to have parents at previous time slices. This makes it possible to create a $(k+1)$-slice temporal Bayesian network that encodes the transition distribution $p(X^{(t)}|X^{(t-1)}, \ldots, X^{(t-k)})$ (Hulst, 2006). Finally, note that a Gaussian mixture Bayesian network can be created with functions add_nodes (by passing NULL as argument gmgm) and add_arcs, which allows to quickly initialize a gmbn object that can be passed to a learning function.

Usage

```
gmbn(...)
```

Arguments

...	Objects of class gmm describing the local joint distributions over the nodes and their parents. Each gmm object must be named after the node whose distribution it describes and contain variables named after this node and its parents. Two types of parents are accepted: other nodes (whose gmm objects must be defined) and instantiations of nodes at previous time slices (if the created gmbn object is a temporal Bayesian network). In the second case, the time lag must be added at the end of the variable name after a period . (e.g. the instantiation of a node X at time slice $t-1$ is represented by the variable X.1).

Value

A list of class gmbn containing the gmm objects passed as arguments.

References

Davies, S. and Moore, A. (2000). Mix-nets: Factored Mixtures of Gaussians in Bayesian Networks with Mixed Continuous And Discrete Variables. *In Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, pages 168–175, Stanford, CA, USA.

Hulst, J. (2006). *Modeling physiological processes with dynamic Bayesian networks*. Master's thesis, Delft University of Technology.

See Also

gmdbn, gmm

Examples

```
data(data_body)
gmbn_1 <- gmbn(
  AGE = split_comp(add_var(NULL, data_body[, "AGE"]), n_sub = 3),
  FAT = split_comp(add_var(NULL,
                           data_body[, c("FAT", "GENDER", "HEIGHT", "WEIGHT")]),
                   n_sub = 2),
  GENDER = split_comp(add_var(NULL, data_body[, "GENDER"]), n_sub = 2),
  GLYCO = split_comp(add_var(NULL, data_body[, c("GLYCO", "AGE", "WAIST")]),
                     n_sub = 2),
  HEIGHT = split_comp(add_var(NULL, data_body[, c("HEIGHT", "GENDER")])),
  WAIST = split_comp(add_var(NULL,
                             data_body[, c("WAIST", "AGE", "FAT", "HEIGHT",
                                           "WEIGHT")]),
                     n_sub = 3),
  WEIGHT = split_comp(add_var(NULL, data_body[, c("WEIGHT", "HEIGHT")]),
                      n_sub = 2)
)

library(dplyr)
data(data_air)
data <- data_air %>%
  group_by(DATE) %>%
  mutate(NO2.1 = lag(NO2), O3.1 = lag(O3), TEMP.1 = lag(TEMP),
         WIND.1 = lag(WIND)) %>%
  ungroup()
gmbn_2 <- gmbn(
  NO2 = split_comp(add_var(NULL, data[, c("NO2", "NO2.1", "WIND")]), n_sub = 3),
  O3 = split_comp(add_var(NULL,
                          data[, c("O3", "NO2", "NO2.1", "O3.1", "TEMP",
                                   "TEMP.1")]),
                  n_sub = 3),
  TEMP = split_comp(add_var(NULL, data[, c("TEMP", "TEMP.1")]), n_sub = 3),
  WIND = split_comp(add_var(NULL, data[, c("WIND", "WIND.1")]), n_sub = 3)
)
```

---

gmbn_body *Gaussian mixture Bayesian network learned from the NHANES body composition dataset*

Description

This Gaussian mixture dynamic Bayesian network is learned from the NHANES body composition dataset, following the example provided in the documentation page of function `struct_learn`.

Usage

```
gmbn_body
```

Format

A gmbn object.

See Also

data_air, data_body, gmdbn_air, gmm_body

---

gmdbn *Create a Gaussian mixture dynamic Bayesian network*

Description

This function creates a Gaussian mixture dynamic Bayesian network as an object of S3 class gmdbn. Assuming that the system evolves over time (possibly non-stationary) and denoting by $X^{(t)}$ its state at time slice t, a dynamic Bayesian network is a probabilistic graphical model that encodes the joint distribution over any finite time sequence:

$$p(X^{(1)}, \ldots, X^{(T)}) = p(X^{(1)}) \prod_{t=2}^{T} p(X^{(t)} | X^{(t-1)}, \ldots, X^{(1)})$$

It is defined by a sequence of transition models $\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_N$ associated with transition time slices $t_1 = 1 < t_2 < \cdots < t_N$, where:

- $\mathcal{B}_1$ is a Bayesian network that encodes the distribution $p(X^{(t)})$ for $1 \leq t \leq t_2 - 1$, assuming that the states at these time slices do not depend on previous states;
- for each $i \geq 2$, $\mathcal{B}_i$ is a $(k_i + 1)$-slice temporal Bayesian network (where $k_i < t_i$) that encodes the transition distribution $p(X^{(t)} | X^{(t-1)}, \ldots, X^{(t-k_i)})$ for $t_i \leq t \leq t_{i+1} - 1$ (or $t \geq t_i$ if $i = N$), assuming that the states at these time slices only depend on the k_i previous states (Hourbracq *et al.*, 2017).

In a Gaussian mixture dynamic Bayesian network, these transition models are Gaussian mixture Bayesian networks (Roos *et al.*, 2017).

Usage

```
gmdbn(...)
```

Arguments

...	Objects of class gmbn corresponding to the transition models. Each gmbn object must be named with the prefix b_ followed by its associated transition time slice (e.g. a transition model whose transition time slice is 8 is represented by the gmbn object b_8). If the first gmbn object (chronologically) is associated with a transition time slice $t \geq 2$ (i.e. b_1 is not specified), it is duplicated to create transition models associated with $1, \ldots, t - 1$ (removing the arcs whose time lags exceed the maximum temporal depths of these models).

Value

A list of class gmdbn containing the gmbn objects passed as arguments.

References

Hourbracq, M., Wuillemin, P.-H., Gonzales, C. and Baumard, P. (2017). Learning and Selection of Dynamic Bayesian Networks for Non-Stationary Processes in Real Time. *In Proceedings of the 30th International Flairs Conference*, pages 742–747, Marco Island, FL, USA.

Roos, J., Bonnevay, S. and Gavin, G. (2017). Dynamic Bayesian Networks with Gaussian Mixture Models for Short-Term Passenger Flow Forecasting. *In Proceedings of the 12th International Conference on Intelligent Systems and Knowledge Engineering*, Nanjing, China.

See Also

gmbn, gmm

Examples

```
library(dplyr)
data(data_air)
data <- data_air %>%
  group_by(DATE) %>%
  mutate(NO2.1 = lag(NO2), O3.1 = lag(O3), TEMP.1 = lag(TEMP),
         WIND.1 = lag(WIND)) %>%
  ungroup()
gmdbn_1 <- gmdbn(
  b_2 = gmbn(
    NO2 = split_comp(add_var(NULL, data[, c("NO2", "NO2.1", "WIND")]),
                     n_sub = 3),
    O3 = split_comp(add_var(NULL,
                            data[, c("O3", "NO2", "NO2.1", "O3.1", "TEMP",
                                     "TEMP.1")]),
                    n_sub = 3),
    TEMP = split_comp(add_var(NULL, data[, c("TEMP", "TEMP.1")]), n_sub = 3),
    WIND = split_comp(add_var(NULL, data[, c("WIND", "WIND.1")]), n_sub = 3)
  ),
  b_13 = gmbn(
    NO2 = split_comp(add_var(NULL, data[, c("NO2", "NO2.1", "WIND")]),
                     n_sub = 3),
    O3 = split_comp(add_var(NULL,
                            data[, c("O3", "O3.1", "TEMP", "TEMP.1", "WIND")]),
                    n_sub = 3),
    TEMP = split_comp(add_var(NULL, data[, c("TEMP", "TEMP.1")]), n_sub = 3),
    WIND = split_comp(add_var(NULL, data[, c("WIND", "WIND.1")]), n_sub = 3)
  )
)
```

gmdbn_air — *Gaussian mixture dynamic Bayesian network learned from the Beijing air quality dataset*

Description

This Gaussian mixture dynamic Bayesian network is learned from the Beijing air quality dataset, following the example provided in the documentation page of function `struct_learn`.

Usage

```
gmdbn_air
```

Format

A gmdbn object.

See Also

`data_air`, `data_body`, `gmbn_body`, `gmm_body`

gmm — *Create a Gaussian mixture model*

Description

This function creates a Gaussian mixture model as an object of S3 class `gmm`. A Gaussian mixture model is a weighted sum of multivariate Gaussian distributions:

$$p(x) = \sum_{i=1}^{M} \alpha_i \mathcal{N}(x|\mu_i, \Sigma_i)$$

where α_i is the ith mixture proportion such that $\alpha_i > 0$ and $\sum_{i=1}^{M} \alpha_i = 1$, μ_i the mean vector and Σ_i the covariance matrix of the ith mixture component (Bilmes, 1998). Since conditional distributions can be derived from joint distributions, the `gmm` class is also used to work with conditional Gaussian mixture models (see function `conditional` to explicit their parameters). Finally, note that a one-component Gaussian mixture model can be created with function `add_var` (by passing `NULL` as argument `gmm`), which allows to quickly initialize a `gmm` object that can be passed to a learning function.

Usage

```
gmm(alpha, mu, sigma, var = rownames(mu))
```

Arguments

alpha	A positive numeric vector containing the mixture proportions. If the sum of these proportions is not 1, a normalization is performed by dividing them by this sum.
mu	A numeric matrix containing the mean vectors bound by column.
sigma	A list containing the covariance matrices.
var	A character vector containing the variable names (by default the row names of mu).

Value

A list of class gmm containing the elements alpha, mu and sigma passed as arguments (completed with the variable names passed as argument var).

References

Bilmes, J. A. (1998). A Gentle Tutorial of the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models. Technical report, International Computer Science Institute.

See Also

gmbn, gmdbn

Examples

```
gmm_1 <- gmm(alpha = c(0.2, 0.5, 0.3),
             mu = matrix(c(109, 91, 44, 160, 41, 99, 87, 27, 173, 40, 86, 65,
                           35, 161, 40),
                         nrow = 5),
             sigma = list(matrix(c(208, 240, 32, 17, -6, 240, 378, 40, 55, -38,
                                   32, 40, 15, -2, 1, 17, 55, -2, 47, -13, -6,
                                   -38, 1, -13, 127),
                                 nrow = 5),
                          matrix(c(242, 270, 82, 10, 49, 270, 363, 83, 44, 19,
                                   82, 83, 38, -2, 15, 10, 44, -2, 45, -7, 49,
                                   19, 15, -7, 137),
                                 nrow = 5),
                          matrix(c(109, 102, 41, 11, 29, 102, 128, 34, 38, 10,
                                   41, 34, 36, -9, 16, 11, 38, -9, 56, -5, 29,
                                   10, 16, -5, 138),
                                 nrow = 5)),
             var = c("WAIST", "WEIGHT", "FAT", "HEIGHT", "AGE"))
```

gmm_body — *Gaussian mixture model learned from the NHANES body composition dataset*

Description

This Gaussian mixture model is learned from the NHANES body composition dataset, following the example provided in the documentation page of function `stepwise`.

Usage

```
gmm_body
```

Format

A `gmm` object.

See Also

`data_air`, `data_body`, `gmbn_body`, `gmdbn_air`

inference — *Perform inference in a Gaussian mixture Bayesian network*

Description

This function performs inference in a (non-temporal) Gaussian mixture Bayesian network. This task consists in estimating the state of the system given partial observations of it (the evidence). Inference is performed by likelihood weighting, which is a particle-based approximate method (Koller and Friedman, 2009).

Usage

```
inference(
  gmbn,
  evid,
  nodes = names(gmbn),
  n_part = 1000,
  max_part_sim = 1e+06,
  verbose = FALSE
)
```

Arguments

gmbn	A (non-temporal) object of class gmbn.
evid	A data frame containing the evidence. Its columns must explicitly be named after nodes of gmbn and can contain missing values (columns with no value can be removed).
nodes	A character vector containing the inferred nodes (by default all the nodes of gmbn).
n_part	A positive integer corresponding to the number of particles generated for each observation.
max_part_sim	An integer greater than or equal to n_part corresponding to the maximum number of particles that can be processed simultaneously. This argument is used to prevent memory overflow, dividing evid into smaller subsets that are handled sequentially.
verbose	A logical value indicating whether subsets of evid in progress are displayed.

Value

A data frame (tibble) with a structure similar to evid containing the estimated values of the inferred nodes.

References

Koller, D. and Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press.

See Also

filtering, prediction, smoothing

Examples

```
set.seed(0)
data(gmbn_body, data_body)
evid <- data_body
evid$GENDER[sample.int(2148, 430)] <- NA
evid$AGE[sample.int(2148, 430)] <- NA
evid$HEIGHT[sample.int(2148, 430)] <- NA
evid$WEIGHT[sample.int(2148, 430)] <- NA
evid$FAT[sample.int(2148, 430)] <- NA
evid$WAIST[sample.int(2148, 430)] <- NA
evid$GLYCO[sample.int(2148, 430)] <- NA
infer <- inference(gmbn_body, evid, verbose = TRUE)
```

`logLik` *Compute the log-likelihood of a Gaussian mixture model or graphical model*

Description

This function computes the log-likelihood of a Gaussian mixture model or graphical model.

Usage

```
## S3 method for class 'gmm'
logLik(object, data, y = NULL, regul = 0.01, ...)

## S3 method for class 'gmbn'
logLik(object, data, col_seq = NULL, ...)

## S3 method for class 'gmdbn'
logLik(object, data, col_seq = NULL, ...)
```

Arguments

`object`	An object of class `gmm`, `gmbn` or `gmdbn`.
`data`	A data frame containing the data used to compute the log-likelihood. Its columns must explicitly be named after the variables (or nodes) of `object`. If `object` is a `gmm` object, a numeric matrix can be passed.
`y`	A character vector containing the dependent variables if a conditional log-likelihood is computed. If `NULL` (the default), the joint log-likelihood is computed.
`regul`	A positive numeric value corresponding to the regularization constant if a penalty term is added for Bayesian regularization. If `NULL`, no penalty term is added. If a conditional log-likelihood is computed, this argument is ignored.
`...`	Unused arguments from the generic function.
`col_seq`	A character vector containing the column names of `data` that describe the observation sequence. If `NULL` (the default), all the observations belong to a single sequence. If `object` is a temporal `gmbn` or `gmdbn` object, the observations of a same sequence must be ordered such that the tth one is related to time slice t (note that the sequences can have different lengths). If `object` is a non-temporal `gmbn` object, this argument is ignored.

Value

If `object` is a `gmm` object, a numeric value corresponding to the log-likelihood.

If `object` is a `gmbn` or `gmdbn` object, a list with elements:

`global`	A numeric value corresponding to the global log-likelihood.
`local`	For a `gmbn` object, a numeric vector containing the local conditional log-likelihoods. For a `gmdbn` object, a list of numeric vectors containing these values for each `gmbn` element.

See Also

AIC, BIC

Examples

```
data(gmm_body, data_body)
loglik_1 <- logLik(gmm_body, data_body)
loglik_2 <- logLik(gmm_body, data_body, y = "WAIST")

data(gmbn_body, data_body)
loglik_3 <- logLik(gmbn_body, data_body)

data(gmdbn_air, data_air)
loglik_4 <- logLik(gmdbn_air, data_air, col_seq = "DATE")
```

---

merge_comp *Merge mixture components of a Gaussian mixture model*

Description

This function merges mixture components of a Gaussian mixture model (Zhang *et al.*, 2003).

Usage

```
merge_comp(gmm, comp = seq_along(gmm$alpha))
```

Arguments

gmm	An object of class gmm.
comp	An integer vector containing the indexes of the merged mixture components (by default all the components of gmm).

Value

The gmm object after merging the mixture components.

References

Zhang, Z., Chen, C., Sun, J. and Chan, K. L. (2003). EM algorithms for Gaussian mixtures with split-and-merge operation. *Pattern Recognition*, 36(9):1973–1983.

See Also

split_comp

Examples

```
data(gmm_body)
gmm_1 <- merge_comp(gmm_body, c(1, 2))
```

`network` *Display the graphical structure of a Gaussian mixture Bayesian network*

Description

This function displays the graphical structure of a Gaussian mixture Bayesian network.

Usage

```
network(gmbn)
```

Arguments

`gmbn`	An object of class `gmbn`.

Value

A `visNetwork` object displaying the graphical structure.

Examples

```
data(gmbn_body)
network(gmbn_body)

data(gmdbn_air)
network(gmdbn_air$b_2)
```

`param_em` *Learn the parameters of a Gaussian mixture graphical model with incomplete data*

Description

This function learns the parameters of a Gaussian mixture graphical model with incomplete data using the parametric EM algorithm. At each iteration, inference (smoothing inference for a dynamic Bayesian network) is performed to complete the data given the current estimate of the parameters (E step). The completed data are then used to update the parameters (M step), and so on. Each iteration is guaranteed to increase the log-likelihood until convergence to a local maximum (Koller and Friedman, 2009). In practice, due to the sampling process inherent in particle-based inference, it may happen that the monotonic increase no longer occurs when approaching the local maximum, resulting in an earlier termination of the algorithm.

Usage

```
param_em(
  gmgm,
  data,
  nodes = structure(gmgm)$nodes,
  col_seq = NULL,
  n_part = 1000,
  max_part_sim = 1e+06,
  min_ess = 1,
  max_iter_pem = 5,
  verbose = FALSE,
  ...
)
```

Arguments

gmgm	An object of class gmbn (non-temporal) or gmdbn.
data	A data frame containing the data used for learning. Its columns must explicitly be named after nodes of gmgm and can contain missing values (columns with no value can be removed).
nodes	A character vector containing the nodes whose local conditional models are learned (by default all the nodes of gmgm). If gmgm is a gmdbn object, the same nodes are learned for each of its gmbn elements. This constraint can be overcome by passing a list of character vectors named after some of these elements (b_1, ...) and containing learned nodes specific to them.
col_seq	A character vector containing the column names of data that describe the observation sequence. If NULL (the default), all the observations belong to a single sequence. If gmgm is a gmdbn object, the observations of a same sequence must be ordered such that the tth one is related to time slice t (note that the sequences can have different lengths). If gmgm is a gmbn object, this argument is ignored.
n_part	A positive integer corresponding to the number of particles generated for each observation (if gmgm is a gmbn object) or observation sequence (if gmgm is a gmdbn object) during inference.
max_part_sim	An integer greater than or equal to n_part corresponding to the maximum number of particles that can be processed simultaneously during inference. This argument is used to prevent memory overflow, dividing data into smaller subsets that are handle sequentially.
min_ess	A numeric value in [0, 1] corresponding to the minimum ESS (expressed as a proportion of n_part) under which the renewal step of sequential importance resampling is performed. If 1 (the default), this step is performed at each time slice. If gmgm is a gmbn object, this argument is ignored.
max_iter_pem	A non-negative integer corresponding to the maximum number of iterations.
verbose	A logical value indicating whether iterations in progress are displayed.
...	Additional arguments passed to function em.

Value

A list with elements:

gmgm	The final gmbn or gmdbn object (with the highest log-likelihood).
data	A data frame (tibble) containing the complete data used to learn the final gmbn or gmdbn object.
seq_loglik	A numeric matrix containing the sequence of log-likelihoods measured after the E and M steps of each iteration.

References

Koller, D. and Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press.

See Also

param_learn, struct_em, struct_learn

Examples

```
set.seed(0)
data(data_body)
data_1 <- data_body
data_1$GENDER[sample.int(2148, 430)] <- NA
data_1$AGE[sample.int(2148, 430)] <- NA
data_1$HEIGHT[sample.int(2148, 430)] <- NA
data_1$WEIGHT[sample.int(2148, 430)] <- NA
data_1$FAT[sample.int(2148, 430)] <- NA
data_1$WAIST[sample.int(2148, 430)] <- NA
data_1$GLYCO[sample.int(2148, 430)] <- NA
gmbn_1 <- gmbn(
  AGE = split_comp(add_var(NULL, data_1[, "AGE"]), n_sub = 3),
  FAT = split_comp(add_var(NULL,
                           data_1[, c("FAT", "GENDER", "HEIGHT", "WEIGHT")]),
                   n_sub = 2),
  GENDER = split_comp(add_var(NULL, data_1[, "GENDER"]), n_sub = 2),
  GLYCO = split_comp(add_var(NULL, data_1[, c("GLYCO", "AGE", "WAIST")]),
                     n_sub = 2),
  HEIGHT = split_comp(add_var(NULL, data_1[, c("HEIGHT", "GENDER")])),
  WAIST = split_comp(add_var(NULL,
                             data_1[, c("WAIST", "AGE", "FAT", "HEIGHT",
                                        "WEIGHT")]),
                     n_sub = 3),
  WEIGHT = split_comp(add_var(NULL, data_1[, c("WEIGHT", "HEIGHT")]), n_sub = 2)
)
res_learn_1 <- param_em(gmbn_1, data_1, verbose = TRUE)

library(dplyr)
set.seed(0)
data(data_air)
data_2 <- data_air
```

```
data_2$NO2[sample.int(7680, 1536)] <- NA
data_2$O3[sample.int(7680, 1536)] <- NA
data_2$TEMP[sample.int(7680, 1536)] <- NA
data_2$WIND[sample.int(7680, 1536)] <- NA
data_3 <- data_2 %>%
  group_by(DATE) %>%
  mutate(NO2.1 = lag(NO2), O3.1 = lag(O3), TEMP.1 = lag(TEMP),
         WIND.1 = lag(WIND)) %>%
  ungroup()
gmdbn_1 <- gmdbn(
  b_2 = gmbn(
    NO2 = split_comp(add_var(NULL, data_3[, c("NO2", "NO2.1", "WIND")]),
                     n_sub = 3),
    O3 = split_comp(add_var(NULL,
                            data_3[, c("O3", "NO2", "NO2.1", "O3.1", "TEMP",
                                       "TEMP.1")]),
                    n_sub = 3),
    TEMP = split_comp(add_var(NULL, data_3[, c("TEMP", "TEMP.1")]), n_sub = 3),
    WIND = split_comp(add_var(NULL, data_3[, c("WIND", "WIND.1")]), n_sub = 3)
  ),
  b_13 = gmbn(
    NO2 = split_comp(add_var(NULL, data_3[, c("NO2", "NO2.1", "WIND")]),
                     n_sub = 3),
    O3 = split_comp(add_var(NULL,
                            data_3[, c("O3", "O3.1", "TEMP", "TEMP.1",
                                       "WIND")]),
                    n_sub = 3),
    TEMP = split_comp(add_var(NULL, data_3[, c("TEMP", "TEMP.1")]), n_sub = 3),
    WIND = split_comp(add_var(NULL, data_3[, c("WIND", "WIND.1")]), n_sub = 3)
  )
)
res_learn_2 <- param_em(gmdbn_1, data_2, col_seq = "DATE", verbose = TRUE)
```

`param_learn` *Learn the parameters of a Gaussian mixture graphical model*

Description

This function learns the parameters of a Gaussian mixture graphical model. Using the local decomposability of the log-likelihood, this task consists in learning each local conditional model independently with the EM algorithm (Koller and Friedman, 2009).

Usage

```
param_learn(
  gmgm,
  data,
  nodes = structure(gmgm)$nodes,
  col_seq = NULL,
```

```
  verbose = FALSE,
  ...
)
```

Arguments

gmgm	An initial object of class gmbn or gmdbn.
data	A data frame containing the data used for learning. Its columns must explicitly be named after the nodes of gmgm and must not contain missing values.
nodes	A character vector containing the nodes whose local conditional models are learned (by default all the nodes of gmgm). If gmgm is a gmdbn object, the same nodes are learned for each of its gmbn elements. This constraint can be overcome by passing a list of character vectors named after some of these elements (b_1, ...) and containing learned nodes specific to them.
col_seq	A character vector containing the column names of data that describe the observation sequence. If NULL (the default), all the observations belong to a single sequence. If gmgm is a temporal gmbn or gmdbn object, the observations of a same sequence must be ordered such that the tth one is related to time slice t (note that the sequences can have different lengths). If gmgm is a non-temporal gmbn object, this argument is ignored.
verbose	A logical value indicating whether learned nodes in progress are displayed.
...	Additional arguments passed to function em.

Value

A list with elements:

gmgm	The final gmbn or gmdbn object.
evol_loglik	A list with elements: global A numeric vector containing the global log-likelihood before and after learning. local For a gmbn object, a numeric matrix containing the local conditional log-likelihoods before and after learning. For a gmdbn object, a list of numeric matrices containing these values for each gmbn element.

References

Koller, D. and Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press.

See Also

param_em, struct_em, struct_learn

Examples

```
data(data_body)
gmbn_1 <- gmbn(
  AGE = split_comp(add_var(NULL, data_body[, "AGE"]), n_sub = 3),
  FAT = split_comp(add_var(NULL,
                           data_body[, c("FAT", "GENDER", "HEIGHT", "WEIGHT")]),
                   n_sub = 2),
  GENDER = split_comp(add_var(NULL, data_body[, "GENDER"]), n_sub = 2),
  GLYCO = split_comp(add_var(NULL, data_body[, c("GLYCO", "AGE", "WAIST")]),
                     n_sub = 2),
  HEIGHT = split_comp(add_var(NULL, data_body[, c("HEIGHT", "GENDER")])),
  WAIST = split_comp(add_var(NULL,
                             data_body[, c("WAIST", "AGE", "FAT", "HEIGHT",
                                           "WEIGHT")]),
                     n_sub = 3),
  WEIGHT = split_comp(add_var(NULL, data_body[, c("WEIGHT", "HEIGHT")]),
                      n_sub = 2)
)
res_learn_1 <- param_learn(gmbn_1, data_body, verbose = TRUE)

library(dplyr)
data(data_air)
data <- data_air %>%
  group_by(DATE) %>%
  mutate(NO2.1 = lag(NO2), O3.1 = lag(O3), TEMP.1 = lag(TEMP),
         WIND.1 = lag(WIND)) %>%
  ungroup()
gmdbn_1 <- gmdbn(
  b_2 = gmbn(
    NO2 = split_comp(add_var(NULL, data[, c("NO2", "NO2.1", "WIND")]),
                     n_sub = 3),
    O3 = split_comp(add_var(NULL,
                            data[, c("O3", "NO2", "NO2.1", "O3.1", "TEMP",
                                     "TEMP.1")]),
                    n_sub = 3),
    TEMP = split_comp(add_var(NULL, data[, c("TEMP", "TEMP.1")]), n_sub = 3),
    WIND = split_comp(add_var(NULL, data[, c("WIND", "WIND.1")]), n_sub = 3)
  ),
  b_13 = gmbn(
    NO2 = split_comp(add_var(NULL, data[, c("NO2", "NO2.1", "WIND")]),
                     n_sub = 3),
    O3 = split_comp(add_var(NULL,
                            data[, c("O3", "O3.1", "TEMP", "TEMP.1", "WIND")]),
                    n_sub = 3),
    TEMP = split_comp(add_var(NULL, data[, c("TEMP", "TEMP.1")]), n_sub = 3),
    WIND = split_comp(add_var(NULL, data[, c("WIND", "WIND.1")]), n_sub = 3)
  )
)
res_learn_2 <- param_learn(gmdbn_1, data_air, col_seq = "DATE", verbose = TRUE)
```

`particles` *Initialize particles to perform inference in a Gaussian mixture graphical model*

Description

This function initializes particles to perform (approximate) inference in a Gaussian mixture graphical model. Particles consist in weighted sample sequences propagated forward in time by sampling the model and aggregated to obtain the inferred values (Koller and Friedman, 2009).

Usage

```
particles(seq = NULL, col_weight = "weight", n_part = 1000)
```

Arguments

`seq`	A data frame containing the observation sequences for which particles are initialized. If `NULL` (the default), the initialization is performed for a single sequence.
`col_weight`	A character string corresponding to the column name of the resulting data frame that describes the particle weight.
`n_part`	A positive integer corresponding to the number of particles initialized for each observation sequence.

Value

A data frame (tibble) containing the initial particles.

References

Koller, D. and Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press.

See Also

`aggregation`, `propagation`

Examples

```
data(data_air)
part <- particles(data.frame(DATE = unique(data_air$DATE)))
```

`prediction` *Perform predictive inference in a Gaussian mixture dynamic Bayesian network*

Description

This function performs predictive inference in a Gaussian mixture dynamic Bayesian network. For a sequence of T time slices, this task consists in defining a time horizon h such that at each time slice t (for $0 \leq t \leq T - h$), the state of the system at $t + h$ is estimated given all the data (the evidence) collected up to t. Although the states at $t + 1, \ldots, t + h$ are observed in the future, some information about them can be known a priori (such as contextual information or features controlled by the user). This "predicted" evidence can be taken into account when propagating the particles from t to $t + h$ in order to improve the predictions. Predictive inference is performed by sequential importance resampling, which is a particle-based approximate method (Koller and Friedman, 2009).

Usage

```
prediction(
  gmdbn,
  evid,
  evid_pred = NULL,
  nodes = names(gmdbn$b_1),
  col_seq = NULL,
  horizon = 1,
  n_part = 1000,
  max_part_sim = 1e+06,
  min_ess = 1,
  verbose = FALSE
)
```

Arguments

`gmdbn`	An object of class `gmdbn`.
`evid`	A data frame containing the evidence. Its columns must explicitly be named after nodes of `gmdbn` and can contain missing values (columns with no value can be removed).
`evid_pred`	A data frame containing the "predicted" evidence. Its columns must explicitly be named after nodes of `gmdbn` and can contain missing values (columns with no value can be removed).
`nodes`	A character vector containing the inferred nodes (by default all the nodes of `gmdbn`).
`col_seq`	A character vector containing the column names of `evid` and `evid_pred` that describe the observation sequence. If `NULL` (the default), all the observations belong to a single sequence. The observations of a same sequence must be ordered such that the tth one is related to time slice t (note that the sequences can have different lengths).

horizon	A positive integer vector containing the time horizons for which predictive inference is performed.
n_part	A positive integer corresponding to the number of particles generated for each observation sequence.
max_part_sim	An integer greater than or equal to n_part corresponding to the maximum number of particles that can be processed simultaneously. This argument is used to prevent memory overflow, dividing evid into smaller subsets that are handled sequentially.
min_ess	A numeric value in [0, 1] corresponding to the minimum ESS (expressed as a proportion of n_part) under which the renewal step of sequential importance resampling is performed. If 1 (the default), this step is performed at each time slice.
verbose	A logical value indicating whether subsets of evid and time slices in progress are displayed.

Value

If horizon has one element, a data frame with a structure similar to evid containing the predicted values of the inferred nodes and their observation sequences (if col_seq is not NULL). If horizon has two or more elements, a list of data frames (tibbles) containing these values for each time horizon.

References

Koller, D. and Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press.

See Also

filtering, inference, smoothing

Examples

```
set.seed(0)
data(gmdbn_air, data_air)
evid <- data_air
evid$NO2[sample.int(7680, 1536)] <- NA
evid$O3[sample.int(7680, 1536)] <- NA
pred <- prediction(gmdbn_air, evid, evid[, c("DATE", "TEMP", "WIND")],
                   nodes = c("NO2", "O3"), col_seq = "DATE",
                   horizon = c(1, 2), verbose = TRUE)
```

`propagation` *Propagate particles forward in time*

Description

This function propagates particles forward in time. Assuming that the particles have been propagated to a given time slice t, the aim is to propagate them to a later time slice $t + k$ according to the Gaussian mixture graphical model and to the evidence collected over time. At first, a renewal step is performed if the effective sample size (ESS) is below a given threshold (Doucet and Johansen, 2009). This step consists in randomly selecting new particles among the old ones proportionately to their current weights. Upon receiving the data (the evidence) of $t + 1$, each particle is used to generate samples for the unknown values. Its weight is then updated to the likelihood for the observed values. The higher this likelihood, the more likely the particle is selected at the next renewal step for propagation to $t + 2$, and so on (Koller and Friedman, 2009).

Usage

```
propagation(
  part,
  gmgm,
  evid = NULL,
  col_seq = NULL,
  col_weight = "weight",
  n_times = 1,
  min_ess = 1
)
```

Arguments

`part`	A data frame containing the particles propagated to time slice t, as obtained from function `particles` or `propagation`.
`gmgm`	An object of class `gmbn` or `gmdbn`. For a `gmdbn` object, the `gmbn` elements used for propagation are selected according to the temporal depth of the particles, assuming that the particles contain all the samples since the first time slice (this depth is thus considered as the current time slice).
`evid`	A data frame containing the evidence of time slices $t+1, \ldots, t+k$. Its columns must explicitly be named after nodes of `gmgm` and can contain missing values (columns with no value can be removed). If `NULL` (the default), no evidence is taken into account.
`col_seq`	A character vector containing the column names of `part` and `evid` that describe the observation sequence. If `NULL` (the default), all the particles and observations belong to a single sequence. In `evid`, the observations of a same sequence must be ordered such that the kth one is related to time slice $t + k$ (note that the sequences can have different lengths).
`col_weight`	A character string corresponding to the column name of `part` that describes the particle weight.

`n_times` A non-negative integer corresponding to the number of time slices k over which the particles are propagated.

`min_ess` A numeric value in [0, 1] corresponding to the minimum ESS (expressed as a proportion of the number of particles) under which the renewal step is performed. If 1 (the default), this step is performed at each time slice.

Value

A data frame (tibble) containing the particles supplemented with the samples of time slices $t + 1, \ldots, t + k$.

References

Doucet, A. and Johansen, A. M. (2009). A Tutorial on Particle Filtering and Smoothing: Fifteen years later. *Handbook of nonlinear filtering*, 12:656–704.

Koller, D. and Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press.

See Also

`aggregation`, `particles`

Examples

```
library(dplyr)
set.seed(0)
data(gmdbn_air, data_air)
evid <- data_air %>%
  group_by(DATE) %>%
  slice(1:3) %>%
  ungroup()
evid$NO2[sample.int(150, 30)] <- NA
evid$O3[sample.int(150, 30)] <- NA
evid$TEMP[sample.int(150, 30)] <- NA
evid$WIND[sample.int(150, 30)] <- NA
part <- particles(data.frame(DATE = unique(evid$DATE))) %>%
  propagation(gmdbn_air, evid, col_seq = "DATE", n_times = 3)
```

`relevant` *Extract the minimal sub-Gaussian mixture graphical model required to infer a subset of nodes*

Description

This function extracts the minimal sub-Gaussian mixture graphical model required to infer a subset of nodes (i.e. the sub-model relevant to these nodes). The nodes that do not contribute to inference are removed, which includes those that are d-separated from the inferred ones by the nodes whose values are observed, as well as the barren nodes (Druzdzel and Suermondt, 1994).

Usage

```
relevant(gmgm, nodes, nodes_obs = NULL, nodes_miss = NULL)
```

Arguments

gmgm	An object of class gmbn or gmdbn.
nodes	A character vector containing the inferred nodes.
nodes_obs	A character vector containing the nodes whose values are observed.
nodes_miss	A character vector containing the nodes whose values are missing. Note that if a node is neither in nodes_obs nor in nodes_miss, its observability is considered uncertain or varying. Thus, it is not treated as an observed node, nor can it be removed as a barren node.

Value

The gmbn or gmdbn object relevant to the subset of nodes.

References

Druzdzel, M. J. and Suermondt, H. J. (1994). Relevance in Probabilistic Models: "Backyards" in a "Small World". *In Working Notes of the AAAI 1994 Fall Symposium Series: Relevance*, pages 60–63, New Orleans, LA, USA.

See Also

add_arcs, add_nodes, remove_arcs, remove_nodes, rename_nodes

Examples

```
data(gmbn_body)
gmbn_1 <- relevant(gmbn_body, "AGE",
                   nodes_obs = c("FAT", "HEIGHT", "WEIGHT"),
                   nodes_miss = "GLYCO")

data(gmdbn_air)
gmdbn_1 <- do.call("gmdbn", gmdbn_air[c("b_1", "b_2")])
gmdbn_2 <- relevant(gmdbn_1, "O3", nodes_obs = "NO2")
```

---

remove_arcs *Remove arcs from a Gaussian mixture graphical model*

Description

This function removes arcs from a Gaussian mixture graphical model.

Usage

```
remove_arcs(gmgm, arcs)
```

Arguments

gmgm	An object of class gmbn or gmdbn.
arcs	A data frame containing the removed arcs. The column from describes the start node, the column to the end node and the column lag the time lag between them. Missing values in from or to are interpreted as "all possible nodes", which allows to quickly define large set of arcs that share common attributes. Missing values in lag are replaced by 0. If gmgm is a gmdbn object, the same arcs are removed from each of its gmbn elements. This constraint can be overcome by passing a list of data frames named after some of these elements (b_1, ...) and containing arcs specifically removed from them.

Value

The gmbn or gmdbn object after removing the arcs.

See Also

add_arcs, add_nodes, relevant, remove_nodes, rename_nodes

Examples

```
data(gmbn_body)
gmbn_1 <- remove_arcs(gmbn_body,
                      data.frame(from = c("HEIGHT", "AGE"),
                                 to = c("FAT", "WAIST")))

data(gmdbn_air)
gmdbn_1 <- remove_arcs(gmdbn_air,
                       list(b_2 = data.frame(from = c("NO2", "TEMP"),
                                             to = c("O3", "O3"), lag = c(1, 1)),
                            b_13 = data.frame(from = "TEMP", to = "O3",
                                              lag = 1)))
```

---

remove_nodes *Remove nodes from a Gaussian mixture graphical model*

Description

This function removes nodes from a Gaussian mixture graphical model. If this model is a dynamic Bayesian network, the nodes are removed from each of its transition models.

Usage

```
remove_nodes(gmgm, nodes)
```

Arguments

gmgm	An object of class gmbn or gmdbn.
nodes	A character vector containing the removed nodes.

Value

The gmbn or gmdbn object after removing the nodes.

See Also

add_arcs, add_nodes, relevant, remove_arcs, rename_nodes

Examples

```
data(gmbn_body)
gmbn_1 <- remove_nodes(gmbn_body, c("FAT", "GLYCO"))

data(gmdbn_air)
gmdbn_1 <- remove_nodes(gmdbn_air, "TEMP")
```

---

remove_var *Remove variables from a Gaussian mixture model*

Description

This function removes variables from a Gaussian mixture model.

Usage

```
remove_var(gmm, var)
```

Arguments

gmm	An object of class gmm.
var	A character vector containing the removed variables.

Value

The gmm object after removing the variables.

See Also

add_var, rename_var

Examples

```
data(gmm_body)
gmm_1 <- remove_var(gmm_body, "FAT")
```

rename_nodes *Rename nodes of a Gaussian mixture graphical model*

Description

This function renames nodes of a Gaussian mixture graphical model. If this model is a dynamic Bayesian network, the nodes are renamed for each of its transition models.

Usage

```
rename_nodes(gmgm, nodes, names)
```

Arguments

gmgm	An object of class gmbn or gmdbn.
nodes	A character vector containing the renamed nodes.
names	A character vector containing the respective new names of the nodes.

Value

The gmbn or gmdbn object after renaming the nodes.

See Also

add_arcs, add_nodes, relevant, remove_arcs, remove_nodes

Examples

```
data(gmbn_body)
gmbn_1 <- rename_nodes(gmbn_body, c("FAT", "GLYCO"),
                       c("BODY_FAT", "GLYCOHEMOGLOBIN"))

data(gmdbn_air)
gmdbn_1 <- rename_nodes(gmdbn_air, "TEMP", "TEMPERATURE")
```

rename_var *Rename variables of a Gaussian mixture model*

Description

This function renames variables of a Gaussian mixture model.

Usage

```
rename_var(gmm, var, names)
```

Arguments

gmm	An object of class gmm.
var	A character vector containing the renamed variables.
names	A character vector containing the respective new names of the variables.

Value

The gmm object after renaming the variables.

See Also

add_var, remove_var

Examples

```
data(gmm_body)
gmm_1 <- rename_var(gmm_body, "FAT", "BODY_FAT")
```

---

reorder *Reorder the variables and the mixture components of a Gaussian mixture model*

Description

This function reorders the variables and the mixture components of a Gaussian mixture model.

Usage

```
reorder(gmm, var = NULL, comp = NULL)
```

Arguments

gmm	An object of class gmm.
var	A character vector containing the variables in the desired order. If variables are not specified, they are added after the ordered ones. If NULL (the default), the variables are not reordered.
comp	An integer vector containing the indexes of the mixture component in the desired order. If components are not specified, they are added after the ordered ones. If NULL (the default), the components are not reordered.

Value

The reordered gmm object.

Examples

```
data(gmm_body)
gmm_1 <- reorder(gmm_body, var = c("WAIST", "AGE", "FAT", "HEIGHT", "WEIGHT"),
                 comp = c(2, 1, 3))
```

`sampling` *Sample a Gaussian mixture model*

Description

This function samples a Gaussian mixture model.

Usage

```
sampling(gmm, data_x = NULL, n = 1)
```

Arguments

`gmm`	An object of class `gmm`.
`data_x`	A data frame or numeric matrix containing observations of the explanatory variables if conditional sampling is performed. Its columns must explicitly be named after the explanatory variables. If `NULL` (the default), joint sampling is performed.
`n`	A non-negative integer corresponding to the number of samples. If conditional sampling is performed, this argument is ignored.

Value

A numeric matrix containing the samples.

See Also

`density`, `expectation`

Examples

```
set.seed(0)
data(gmm_body, data_body)
sampl_1 <- sampling(gmm_body, n = 500)
sampl_2 <- sampling(gmm_body,
                    data_body[, c("WEIGHT", "FAT", "HEIGHT", "AGE")])
```

smem *Select the number of mixture components and estimate the parameters of a Gaussian mixture model*

Description

This function selects the number of mixture components and estimates the parameters of a Gaussian mixture model using a split-and-merge EM (SMEM) algorithm. At the first iteration, the classic EM algorithm is performed to update the parameters of the initial model. Then each following iteration consists in splitting a component into two or merging two components, before re-estimating the parameters with the EM algorithm. The selected split or merge operation is the one that maximizes a scoring function (after the re-estimation process). To avoid testing all possible operations, the split and merge candidates are initially ranked according to relevant criteria (Zhang *et al.*, 2003). At first, the top-ranked split and top-ranked merge operations are tested. If neither of them increases the score, the second-ranked ones are considered, and so on. The SMEM algorithm stops if a given maximum rank is reached without improving the score.

Usage

```
smem(
  gmm,
  data,
  y = NULL,
  score = "bic",
  split = TRUE,
  merge = TRUE,
  min_comp = 1,
  max_comp = Inf,
  space = 0.5,
  max_rank = 1,
  max_iter_smem = 10,
  verbose = FALSE,
  ...
)
```

Arguments

gmm	An initial object of class gmm.
data	A data frame or numeric matrix containing the data used in the SMEM algorithm. Its columns must explicitly be named after the variables of gmm and must not contain missing values.
y	A character vector containing the dependent variables if a conditional model is estimated (which involves maximizing a conditional score). If NULL (the default), the joint model is estimated.
score	A character string ("aic", "bic" or "loglik") corresponding to the scoring function.

split	A logical value indicating whether split operations are allowed (if FALSE, no mixture component can be split).
merge	A logical value indicating whether merge operations are allowed (if FALSE, no mixture component can be merged).
min_comp	A positive integer corresponding to the minimum number of mixture components.
max_comp	A positive integer corresponding to the maximum number of mixture components.
space	A numeric value in [0, 1[corresponding to the space between two subcomponents resulting from a split.
max_rank	A positive integer corresponding to the maximum rank for testing the split and merge candidates.
max_iter_smem	A non-negative integer corresponding to the maximum number of iterations.
verbose	A logical value indicating whether iterations in progress are displayed.
...	Additional arguments passed to function em.

Value

A list with elements:

gmm	The final gmm object.
posterior	A numeric matrix containing the posterior probabilities for each observation.
seq_score	A numeric vector containing the sequence of scores measured initially and after each iteration.
seq_oper	A character vector containing the sequence of split and merge operations performed at each iteration.

References

Zhang, Z., Chen, C., Sun, J. and Chan, K. L. (2003). EM algorithms for Gaussian mixtures with split-and-merge operation. *Pattern Recognition*, 36(9):1973–1983.

See Also

em, stepwise

Examples

```
data(data_body)
gmm_1 <- add_var(NULL, c("WAIST", "AGE", "FAT", "HEIGHT", "WEIGHT"))
res_smem <- smem(gmm_1, data_body, max_comp = 3, verbose = TRUE)
```

smoothing *Perform smoothing inference in a Gaussian mixture dynamic Bayesian network*

Description

This function performs smoothing inference in a Gaussian mixture dynamic Bayesian network. For a sequence of T time slices, this task consists in estimating the state of the system at each time slice t (for $1 \leq t \leq T$) given all the data (the evidence) collected up to T. Smoothing inference is performed by sequential importance resampling, which is a particle-based approximate method (Koller and Friedman, 2009).

Usage

```
smoothing(
  gmdbn,
  evid,
  nodes = names(gmdbn$b_1),
  col_seq = NULL,
  n_part = 1000,
  max_part_sim = 1e+06,
  min_ess = 1,
  verbose = FALSE
)
```

Arguments

gmdbn	An object of class gmdbn.
evid	A data frame containing the evidence. Its columns must explicitly be named after nodes of gmdbn and can contain missing values (columns with no value can be removed).
nodes	A character vector containing the inferred nodes (by default all the nodes of gmdbn).
col_seq	A character vector containing the column names of evid that describe the observation sequence. If NULL (the default), all the observations belong to a single sequence. The observations of a same sequence must be ordered such that the tth one is related to time slice t (note that the sequences can have different lengths).
n_part	A positive integer corresponding to the number of particles generated for each observation sequence.
max_part_sim	An integer greater than or equal to n_part corresponding to the maximum number of particles that can be processed simultaneously. This argument is used to prevent memory overflow, dividing evid into smaller subsets that are handled sequentially.
min_ess	A numeric value in [0, 1] corresponding to the minimum ESS (expressed as a proportion of n_part) under which the renewal step of sequential importance

	resampling is performed. If 1 (the default), this step is performed at each time slice.
verbose	A logical value indicating whether subsets of evid and time slices in progress are displayed.

Value

A data frame (tibble) with a structure similar to evid containing the estimated values of the inferred nodes and their observation sequences (if col_seq is not NULL).

References

Koller, D. and Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press.

See Also

filtering, inference, prediction

Examples

```
set.seed(0)
data(gmdbn_air, data_air)
evid <- data_air
evid$NO2[sample.int(7680, 1536)] <- NA
evid$O3[sample.int(7680, 1536)] <- NA
evid$TEMP[sample.int(7680, 1536)] <- NA
evid$WIND[sample.int(7680, 1536)] <- NA
smooth <- smoothing(gmdbn_air, evid, col_seq = "DATE", verbose = TRUE)
```

---

split_comp *Split a mixture component of a Gaussian mixture model*

Description

This function splits a mixture component of a Gaussian mixture model using the singular value decomposition of the covariance matrix (Zhang *et al.*, 2003).

Usage

```
split_comp(gmm, comp = 1, n_sub = 2, space = 0.5)
```

Arguments

gmm	An object of class gmm.
comp	An integer corresponding to the index of the split mixture component.
n_sub	A positive integer corresponding to the number of subcomponents.
space	A numeric value in [0, 1[corresponding to the space between the subcomponents.

Value

The gmm object after splitting the mixture component.

References

Zhang, Z., Chen, C., Sun, J. and Chan, K. L. (2003). EM algorithms for Gaussian mixtures with split-and-merge operation. *Pattern Recognition*, 36(9):1973–1983.

See Also

merge_comp

Examples

```
data(gmm_body)
gmm_1 <- split_comp(gmm_body, n_sub = 3)
```

---

stepwise *Select the explanatory variables, the number of mixture components and estimate the parameters of a conditional Gaussian mixture model*

Description

This function selects the explanatory variables, the number of mixture components and estimates the parameters of a conditional Gaussian mixture model using a stepwise algorithm. At the first iteration, the SMEM algorithm is performed to update the number of components and the parameters of the initial model. Then each following iteration consists in adding or removing a candidate explanatory variable, before re-estimating the model with the SMEM algorithm. The selected add or remove operation is the one that maximizes a conditional scoring function (after the re-estimation process). The stepwise algorithm stops if none of the candidate operations improves the score.

Usage

```
stepwise(
  gmm,
  data,
  y = rownames(gmm$mu)[1],
  x_cand = setdiff(colnames(data), y),
  score = "bic",
  add = TRUE,
  remove = TRUE,
  min_x = 0,
  max_x = Inf,
  max_iter_step = 10,
  verbose = FALSE,
  ...
)
```

Arguments

gmm	An initial object of class gmm.
data	A data frame or numeric matrix containing the data used in the stepwise algorithm. Its columns must explicitly be named after the variables of gmm and the candidate explanatory variables, and must not contain missing values.
y	A character vector containing the dependent variables (by default the first variable of gmm).
x_cand	A character vector containing the candidate explanatory variables for addition or removal (by default all the column names of data except y). If variables already in gmm are not candidates, they cannot be removed.
score	A character string ("aic", "bic" or "loglik") corresponding to the scoring function.
add	A logical value indicating whether add operations are allowed (if FALSE, no variable can be added).
remove	A logical value indicating whether remove operations are allowed (if FALSE, no variable can be removed).
min_x	A non-negative integer corresponding to the minimum number of explanatory variables.
max_x	A non-negative integer corresponding to the maximum number of explanatory variables.
max_iter_step	A non-negative integer corresponding to the maximum number of iterations.
verbose	A logical value indicating whether iterations in progress are displayed.
...	Additional arguments passed to function smem.

Value

A list with elements:

gmm	The final gmm object.
posterior	A numeric matrix containing the posterior probabilities for each observation.
seq_score	A numeric vector containing the sequence of scores measured initially and after each iteration.
seq_oper	A character vector containing the sequence of add and remove operations performed at each iteration.

See Also

em, smem

Examples

```
data(data_body)
gmm_1 <- add_var(NULL, "WAIST")
res_step <- stepwise(gmm_1, data_body, verbose = TRUE, max_comp = 3)
```

structure *Provide the graphical structure of a Gaussian mixture graphical model*

Description

This function provides the graphical structure of a Gaussian mixture graphical model.

Usage

```
structure(gmgm)
```

Arguments

gmgm	An object of class gmbn or gmdbn.

Value

A list with elements:

nodes	A character vector containing the nodes.
arcs	For a gmbn object, a data frame (tibble) containing the arcs. For a gmdbn object, a list of data frames (tibbles) containing the arcs of each gmbn element.

Examples

```
data(gmbn_body)
struct_1 <- structure(gmbn_body)

data(gmdbn_air)
struct_2 <- structure(gmdbn_air)
```

struct_em *Learn the structure and the parameters of a Gaussian mixture graphical model with incomplete data*

Description

This function learns the structure and the parameters of a Gaussian mixture graphical model with incomplete data using the structural EM algorithm. At each iteration, the parametric EM algorithm is performed to complete the data and update the parameters (E step). The completed data are then used to update the structure (M step), and so on. Each iteration is guaranteed to increase the scoring function until convergence to a local maximum (Koller and Friedman, 2009). In practice, due to the sampling process inherent in particle-based inference, it may happen that the monotonic increase no longer occurs when approaching the local maximum, resulting in an earlier termination of the algorithm.

Usage

```
struct_em(
  gmgm,
  data,
  nodes = structure(gmgm)$nodes,
  arcs_cand = tibble(lag = 0),
  col_seq = NULL,
  score = "bic",
  n_part = 1000,
  max_part_sim = 1e+06,
  min_ess = 1,
  max_iter_sem = 5,
  max_iter_pem = 5,
  verbose = FALSE,
  ...
)
```

Arguments

gmgm	An object of class gmbn (non-temporal) or gmdbn.
data	A data frame containing the data used for learning. Its columns must explicitly be named after nodes of gmgm and can contain missing values (columns with no value can be removed).
nodes	A character vector containing the nodes whose local conditional models are learned (by default all the nodes of gmgm). If gmgm is a gmdbn object, the same nodes are learned for each of its gmbn elements. This constraint can be overcome by passing a list of character vectors named after some of these elements (b_1, ...) and containing learned nodes specific to them.
arcs_cand	A data frame containing the candidate arcs for addition or removal (by default all possible non-temporal arcs). The column from describes the start node, the column to the end node and the column lag the time lag between them. Missing values in from or to are interpreted as "all possible nodes", which allows to quickly define large set of arcs that share common attributes. Missing values in lag are replaced by 0. If gmgm is a gmdbn object, the same candidate arcs are used for each of its gmbn elements. This constraint can be overcome by passing a list of data frames named after some of these elements (b_1, ...) and containing candidate arcs specific to them. If arcs already in gmgm are not candidates, they cannot be removed. Therefore, setting arcs_cand to NULL is equivalent to learning only the mixture structure (and the parameters) of the model.
col_seq	A character vector containing the column names of data that describe the observation sequence. If NULL (the default), all the observations belong to a single sequence. If gmgm is a gmdbn object, the observations of a same sequence must be ordered such that the tth one is related to time slice t (note that the sequences can have different lengths). If gmgm is a gmbn object, this argument is ignored.
score	A character string ("aic", "bic" or "loglik") corresponding to the scoring function.

n_part	A positive integer corresponding to the number of particles generated for each observation (if gmgm is a gmbn object) or observation sequence (if gmgm is a gmdbn object) during inference.
max_part_sim	An integer greater than or equal to n_part corresponding to the maximum number of particles that can be processed simultaneously during inference. This argument is used to prevent memory overflow, dividing data into smaller subsets that are handle sequentially.
min_ess	A numeric value in [0, 1] corresponding to the minimum ESS (expressed as a proportion of n_part) under which the renewal step of sequential importance resampling is performed. If 1 (the default), this step is performed at each time slice. If gmgm is a gmbn object, this argument is ignored.
max_iter_sem	A non-negative integer corresponding to the maximum number of iterations.
max_iter_pem	A non-negative integer corresponding to the maximum number of iterations of the parametric EM algorithm.
verbose	A logical value indicating whether iterations in progress are displayed.
...	Additional arguments passed to function stepwise.

Value

A list with elements:

gmgm	The final gmbn or gmdbn object (with the highest score).
data	A data frame (tibble) containing the complete data used to learn the final gmbn or gmdbn object.
seq_score	A numeric matrix containing the sequence of scores measured after the E and M steps of each iteration.

References

Koller, D. and Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press.

See Also

param_em, param_learn, struct_learn

Examples

```
set.seed(0)
data(data_body)
data_1 <- data_body
data_1$GENDER[sample.int(2148, 430)] <- NA
data_1$AGE[sample.int(2148, 430)] <- NA
data_1$HEIGHT[sample.int(2148, 430)] <- NA
data_1$WEIGHT[sample.int(2148, 430)] <- NA
data_1$FAT[sample.int(2148, 430)] <- NA
data_1$WAIST[sample.int(2148, 430)] <- NA
data_1$GLYCO[sample.int(2148, 430)] <- NA
```

```
gmbn_1 <- add_nodes(NULL,
                    c("AGE", "FAT", "GENDER", "GLYCO", "HEIGHT", "WAIST",
                      "WEIGHT"))
arcs_cand_1 <- data.frame(from = c("AGE", "GENDER", "HEIGHT", "WEIGHT", NA,
                                   "AGE", "GENDER", "AGE", "FAT", "GENDER",
                                   "HEIGHT", "WEIGHT", "AGE", "GENDER",
                                   "HEIGHT"),
                          to = c("FAT", "FAT", "FAT", "FAT", "GLYCO", "HEIGHT",
                                 "HEIGHT", "WAIST", "WAIST", "WAIST", "WAIST",
                                 "WAIST", "WEIGHT", "WEIGHT", "WEIGHT"))
res_learn_1 <- struct_em(gmbn_1, data_1, arcs_cand = arcs_cand_1,
                         verbose = TRUE, max_comp = 3)

set.seed(0)
data(data_air)
data_2 <- data_air
data_2$NO2[sample.int(7680, 1536)] <- NA
data_2$O3[sample.int(7680, 1536)] <- NA
data_2$TEMP[sample.int(7680, 1536)] <- NA
data_2$WIND[sample.int(7680, 1536)] <- NA
gmdbn_1 <- gmdbn(b_2 = add_nodes(NULL, c("NO2", "O3", "TEMP", "WIND")),
                 b_13 = add_nodes(NULL, c("NO2", "O3", "TEMP", "WIND")))
arcs_cand_2 <- data.frame(from = c("NO2", "NO2", "NO2", "O3", "TEMP", "TEMP",
                                   "WIND", "WIND"),
                          to = c("NO2", "O3", "O3", "O3", NA, NA, NA, NA),
                          lag = c(1, 0, 1, 1, 0, 1, 0, 1))
res_learn_2 <- struct_em(gmdbn_1, data_2, arcs_cand = arcs_cand_2,
                         col_seq = "DATE", verbose = TRUE, max_comp = 3)
```

`struct_learn` *Learn the structure and the parameters of a Gaussian mixture graphical model*

Description

This function learns the (graphical and mixture) structure and the parameters of a Gaussian mixture graphical model. Using the local decomposability of the scoring function, this task consists in learning each local conditional model independently with the stepwise algorithm (Koller and Friedman, 2009). Note that some candidate arcs may be discarded to avoid that the global graphical structure contains cycles. To limit recourse to this action, the learning process is performed sequentially. The more arcs of a local model are likely to be part of cycles (considering the worst case where all the candidate arcs are selected), the later this local model is processed. By gradually taking into account the local structures learned over time, the number of possible cycles decreases and, with it, the number of candidate arcs to discard.

Usage

```
struct_learn(
```

```
  gmgm,
  data,
  nodes = structure(gmgm)$nodes,
  arcs_cand = tibble(lag = 0),
  col_seq = NULL,
  score = "bic",
  verbose = FALSE,
  ...
)
```

Arguments

gmgm	An initial object of class gmbn or gmdbn.
data	A data frame containing the data used for learning. Its columns must explicitly be named after the nodes of gmgm and must not contain missing values.
nodes	A character vector containing the nodes whose local conditional models are learned (by default all the nodes of gmgm). If gmgm is a gmdbn object, the same nodes are learned for each of its gmbn elements. This constraint can be overcome by passing a list of character vectors named after some of these elements (b_1, ...) and containing learned nodes specific to them.
arcs_cand	A data frame containing the candidate arcs for addition or removal (by default all possible non-temporal arcs). The column from describes the start node, the column to the end node and the column lag the time lag between them. Missing values in from or to are interpreted as "all possible nodes", which allows to quickly define large set of arcs that share common attributes. Missing values in lag are replaced by 0. If gmgm is a gmdbn object, the same candidate arcs are used for each of its gmbn elements. This constraint can be overcome by passing a list of data frames named after some of these elements (b_1, ...) and containing candidate arcs specific to them. If arcs already in gmgm are not candidates, they cannot be removed. Therefore, setting arcs_cand to NULL is equivalent to learning only the mixture structure (and the parameters) of the model.
col_seq	A character vector containing the column names of data that describe the observation sequence. If NULL (the default), all the observations belong to a single sequence. If gmgm is a temporal gmbn or gmdbn object, the observations of a same sequence must be ordered such that the tth one is related to time slice t (note that the sequences can have different lengths). If gmgm is a non-temporal gmbn object, this argument is ignored.
score	A character string ("aic", "bic" or "loglik") corresponding to the scoring function.
verbose	A logical value indicating whether learned nodes in progress are displayed.
...	Additional arguments passed to function stepwise.

Value

A list with elements:

gmgm The final gmbn or gmdbn object.

evol_score A list with elements:

global A numeric vector containing the global score before and after learning.

local For a gmbn object, a numeric matrix containing the local conditional scores before and after learning. For a gmdbn object, a list of numeric matrices containing these values for each gmbn element.

References

Koller, D. and Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press.

See Also

param_em, param_learn, struct_em

Examples

```
data(data_body)
gmbn_1 <- add_nodes(NULL,
                    c("AGE", "FAT", "GENDER", "GLYCO", "HEIGHT", "WAIST",
                      "WEIGHT"))
arcs_cand_1 <- data.frame(from = c("AGE", "GENDER", "HEIGHT", "WEIGHT", NA,
                                   "AGE", "GENDER", "AGE", "FAT", "GENDER",
                                   "HEIGHT", "WEIGHT", "AGE", "GENDER",
                                   "HEIGHT"),
                          to = c("FAT", "FAT", "FAT", "FAT", "GLYCO", "HEIGHT",
                                 "HEIGHT", "WAIST", "WAIST", "WAIST", "WAIST",
                                 "WAIST", "WEIGHT", "WEIGHT", "WEIGHT"))
res_learn_1 <- struct_learn(gmbn_1, data_body, arcs_cand = arcs_cand_1,
                            verbose = TRUE, max_comp = 3)

data(data_air)
gmdbn_1 <- gmdbn(b_2 = add_nodes(NULL, c("NO2", "O3", "TEMP", "WIND")),
                 b_13 = add_nodes(NULL, c("NO2", "O3", "TEMP", "WIND")))
arcs_cand_2 <- data.frame(from = c("NO2", "NO2", "NO2", "O3", "TEMP", "TEMP",
                                   "WIND", "WIND"),
                          to = c("NO2", "O3", "O3", "O3", NA, NA, NA, NA),
                          lag = c(1, 0, 1, 1, 0, 1, 0, 1))
res_learn_2 <- struct_learn(gmdbn_1, data_air, arcs_cand = arcs_cand_2,
                            col_seq = "DATE", verbose = TRUE, max_comp = 3)
```

---

summary *Summarize a Gaussian mixture model or graphical model*

Description

This function summarizes a Gaussian mixture model or graphical model.

Usage

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

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

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

Arguments

object	An object of class gmm, gmbn or gmdbn.
...	Unused arguments from the generic function.

Value

If object is a gmm object, an integer vector containing the number of variables, mixture components and free parameters.

If object is a gmbn or gmdbn object, a list with elements:

global	An integer vector containing the global number of nodes, arcs, mixture components and free parameters (for a gmdbn object, also the number of gmbn elements).
local	For a gmbn object, an integer matrix containing the local numbers of arcs, mixture components and free parameters. For a gmdbn object, a list of integer matrices containing these statistics for each gmbn elements.

Examples

```
data(gmm_body)
summ_1 <- summary(gmm_body)

data(gmbn_body)
summ_2 <- summary(gmbn_body)

data(gmdbn_air)
summ_3 <- summary(gmdbn_air)
```

Index