Package ‘graphicalExtremes’

April 18, 2024

Title Statistical Methodology for Graphical Extreme Value Models

Version 0.3.2

Maintainer Sebastian Engelke <sebastian.engelke@unige.ch>

Description Statistical methodology for sparse multivariate extreme value models. Methods are provided for exact simulation and statistical inference for multivariate Pareto distributions on graphical structures as described in the paper 'Graphical Models for Extremes' by Engelke and Hitz (2020) <doi:10.1111/rssb.12355>.

Depends R (>= 3.6.0)

License GPL-3

Encoding UTF-8

LazyData true

LazyDataCompression bzip2

Suggests testthat (>= 2.1.0), knitr, rmarkdown, dplyr, ggplot2, bookdown, maps

RoxygenNote 7.2.2

Imports igraph (>= 1.2.4.1), mvtnorm (>= 1.0.10), Rdpack, stats (>= 3.6.0), utils, corpcor, osqp, glmnet, glassoFast, edmcr, CVXR

URL https://github.com/sebastian-engelke/graphicalExtremes

BugReports https://github.com/sebastian-engelke/graphicalExtremes/issues

RdMacros Rdpack

VignetteBuilder knitr

NeedsCompilation no

Author Sebastian Engelke [aut, cre],
Adrien S. Hitz [aut],
Nicola Gnecco [aut],
Manuel Hentschel [aut]

Repository CRAN

Date/Publication 2024-04-18 16:43:32 UTC

R topics documented:

checkGamma *HR parameter matrix checks*

Description

Checks whether the matrix given is a valid Huesler-Reiss parameter matrix in the role of Γ, Θ, or Σ, respectively.

Usage

```
checkGamma(Gamma, alert = NULL, tol = get_small_tol(), returnBoolean = FALSE)

checkSigmaTheta(
  M,
  k,
  full,
  matrixName = "Sigma",
  tol = get_small_tol(),
  alert = NULL,
  returnBoolean = FALSE
)

checkTheta(
  Theta,
  k = NULL,
  full = FALSE,
  tol = get_small_tol(),
  alert = NULL,
  returnBoolean = FALSE
)

checkSigma(
  Sigma,
  k = NULL,
  full = FALSE,
  tol = get_small_tol(),
  alert = NULL,
  returnBoolean = FALSE
)

checkMatrix(
  M,
  name = c("Gamma", "Sigma", "Theta")[1],
  k = NULL,
  full = FALSE,
  tol = get_small_tol(),
  alert = NULL,
```

```
  returnBoolean = FALSE
)

is_valid_Gamma(M, tol = get_small_tol())

is_valid_Theta(Theta, k = NULL, full = FALSE, tol = get_small_tol())

is_valid_Sigma(Sigma, k = NULL, full = FALSE, tol = get_small_tol())
```

Arguments

`Gamma`	Numeric $d \times d$ variogram matrix.
`alert`	Passed to `get_alert_function`: `NULL` or `TRUE` to read the option value, `FALSE` to return a dummy function, or a function that takes an arbitrary number of strings as arguments (e.g. `stop()`).
`tol`	Numeric scalar. Values below this are considered as zero, when zeros are required (e.g. row-sums).
`returnBoolean`	Logical scalar, set to `TRUE` to return a boolean instead of the (adjusted) input.
`M`	Numeric matrix, Γ, Σ, or Θ.
`k`	`NULL` if the input/output matrix is Σ/Θ. Else, an integer between 1 and d indicating the value of k in Σ^k, Θ^k.
`full`	Logical. If `TRUE` and `!is.null(k)`, the input/output matrix is a $d \times d$ matrix with the kth row filled with zeros.
`matrixName`	Name of the matrix to be used in alerts/error messages.
`Theta`	Numeric $d \times d$ or $(d-1) \times (d-1)$ precision matrix.
`Sigma`	Numeric $d \times d$ or $(d-1) \times (d-1)$ covariance matrix.
`name`	Name of the input matrix, indicating which other function to call.

Details

The function `is_valid_*` are a wrapper around `check*`, with arguments `alert=FALSE` and `returnBoolean=TRUE`.

Value

For `check*`, the input matrix, passed through `ensure_matrix_symmetry_and_truncate_zeros`.

For `is_valid_*`, a boolean indicating whether the input is a valid parameter matrix.

See Also

Other input validation functions: `check_graph()`, `check_partial_matrix_and_graph()`, `ensure_matrix_symmetry()`

chi2Gamma *Transformation between χ and Γ*

Description

Transforms between the extremal correlation χ and the variogram Γ. Only valid for Huesler-Reiss distributions. Done element-wise, no checks of the entire matrix structure are performed.

Usage

```
chi2Gamma(chi)

Gamma2chi(Gamma)
```

Arguments

chi	Numeric vector or matrix with entries between 0 and 1.
Gamma	Numeric vector or matrix with non-negative entries.

Details

The formula for transformation from χ to Γ is element-wise

$$\Gamma = (2\Phi^{-1}(1 - 0.5\chi))^2,$$

where Φ^{-1} is the inverse of the standard normal distribution function.

The formula for transformation from Γ to χ is element-wise

$$\chi = 2 - 2\Phi(\sqrt{\Gamma}/2),$$

where Φ is the standard normal distribution function.

Value

Numeric vector or matrix containing the implied Γ.

Numeric vector or matrix containing the implied χ.

See Also

Other parameter matrix transformations: `Gamma2Sigma()`, `Gamma2graph()`, `par2Matrix()`

complete_Gamma *Completion of Gamma matrices*

Description

Given a graph and a (partial) variogram matrix Gamma, returns a full variogram matrix that agrees with Gamma in entries corresponding to edges of graph and whose corresponding precision matrix, obtained by Gamma2Theta(), has zeros in entries corresponding to non-edges of graph. For results on the existence and uniqueness of this completion, see Hentschel et al. (2022).

Usage

```
complete_Gamma(Gamma, graph = NULL, ...)
```

Arguments

Gamma	Numeric $d \times d$ variogram matrix.
graph	NULL or igraph::graph object. If NULL, the graph is implied by non-edge entries in Gamma being NA. Must be connected, undirected.
...	Further arguments passed to complete_Gamma_general_split() if graph is not decomposable

Details

If graph is decomposable, Gamma only needs to be specified on the edges of the graph, other entries are ignored. If graph is not decomposable, the graphical completion algorithm requires a fully specified (but non-graphical) variogram matrix Gamma to begin with. If necessary, this initial completion is computed using edmcr::npf().

Value

Completed $d \times d$ variogram matrix.

References

Hentschel M, Engelke S, Segers J (2022). "Statistical Inference for Hüsler-Reiss Graphical Models Through Matrix Completions." doi:10.48550/ARXIV.2210.14292, https://arxiv.org/abs/2210.14292.

See Also

Gamma2Theta()

Other matrix completion related topics: complete_Gamma_decomposable(), complete_Gamma_general_demo(), complete_Gamma_general_split(), complete_Gamma_general()

Examples

```
## Block graph:
Gamma <- rbind(
  c(0, .5, NA, NA),
  c(.5, 0, 1, 1.5),
  c(NA, 1, 0, .8),
  c(NA, 1.5, .8, 0)
)

complete_Gamma(Gamma)

## Alternative representation of the same completion problem:
my_graph <- igraph::graph_from_adjacency_matrix(rbind(
  c(0, 1, 0, 0),
  c(1, 0, 1, 1),
  c(0, 1, 0, 1),
  c(0, 1, 1, 0)
), mode = "undirected")
Gamma_vec <- c(.5, 1, 1.5, .8)
complete_Gamma(Gamma_vec, my_graph)

## Decomposable graph:
G <- rbind(
c(0, 5, 7, 6, NA),
c(5, 0, 14, 15, NA),
c(7, 14, 0, 5, 5),
c(6, 15, 5, 0, 6),
c(NA, NA, 5, 6, 0)
)

complete_Gamma(G)

## Non-decomposable graph:
G <- rbind(
c(0, 5, 7, 6, 6),
c(5, 0, 14, 15, 13),
c(7, 14, 0, 5, 5),
c(6, 15, 5, 0, 6),
c(6, 13, 5, 6, 0)
)
g <- igraph::make_ring(5)

complete_Gamma(G, g)
```

---

complete_Gamma_decomposable

Completion of decomposable Gamma matrices

Description

Given a decomposable `graph` and incomplete variogram matrix `Gamma`, returns the full `Gamma` matrix implied by the conditional independencies.

Usage

```
complete_Gamma_decomposable(Gamma, graph = NULL)
```

Arguments

`Gamma`	A variogram matrix that is specified on the edges of `graph` and the diagonals. All other entries are ignored (if `graph` is specified), or should be `NA` to indicate non-edges in `graph`.
`graph`	`NULL` or a decomposable `[igraph::graph]` object. If `NULL`, the structure of `NA` entries in `Gamma` is used instead.

Value

A complete variogram matrix that agrees with `Gamma` on the entries corresponding to edges in `graph` and the diagonals. The corresponding Θ matrix produced by `Gamma2Theta()` has zeros in the remaining entries.

See Also

Other matrix completion related topics: `complete_Gamma_general_demo()`, `complete_Gamma_general_split()`, `complete_Gamma_general()`, `complete_Gamma()`

`complete_Gamma_general` *Non-decomposable completion of variogram matrices*

Description

Given a non-decomposable `graph`, and (non-graphical) variogram matrix `Gamma`, modifies `Gamma` in non-edge entries, such that the resulting matrix is a variogram matrix with graphical structure described by `graph`.

Usage

```
complete_Gamma_general(
  Gamma,
  graph,
  N = 10000,
  tol = get_large_tol(),
  check_tol = 100
)
```

Arguments

Gamma	Numeric $d \times d$ variogram matrix.
graph	igraph::graph() object.
N	Maximum number of iterations.
tol	Numeric scalar. Tolerance to be used when completing submatrices.
check_tol	Numeric/integer scalar. How often to check the tolerance when completing submatrices.

Value

A completed $d \times d$ variogram matrix.

See Also

Other matrix completion related topics: complete_Gamma_decomposable(), complete_Gamma_general_demo(), complete_Gamma_general_split(), complete_Gamma()

---

complete_Gamma_general_demo

DEMO-VERSION: Completion of non-decomposable Gamma matrices

Description

Given a graph and variogram matrix Gamma, returns the full Gamma matrix implied by the conditional independencies. DEMO VERSION: Returns a lot of details and allows specifying the graph list that is used. Is way slower than other functions.

Usage

```
complete_Gamma_general_demo(Gamma, graph, N = 1000, tol = 0, gList = NULL)
```

Arguments

Gamma	A complete variogram matrix (without any graphical structure).
graph	An igraph::graph object.
N	The maximal number of iterations of the algorithm.
tol	The tolerance to use when checking for zero entries in Theta.
gList	A list of graphs to be used instead of the output from make_sep_list().

Value

A nested list, containing the following details. The "error term" is the maximal absolute value of `Theta` in a non-edge entry.

- `graph, N, tol` As in the input
- `gList` As in the input or computed by `make_sep_list()`.
- `Gamma0, Theta0, err0` Initial `Gamma`, `Theta`, and error term.
- `iterations` A nested list, containing the following infos for each performed iteration:
  - `n` Number of the iteration
  - `t` Corresponding index in `gList`
  - `g` The graph used
  - `Gamma`**,** `Theta`**,** `err` The value of `Gamma`, `Theta`, and error term after the iteration

See Also

Other matrix completion related topics: `complete_Gamma_decomposable()`, `complete_Gamma_general_split()`, `complete_Gamma_general()`, `complete_Gamma()`

`complete_Gamma_general_split` *Non-decomposable completion of variogram matrices*

Description

Given a non-decomposable `graph`, and (non-graphical) variogram matrix `Gamma`, modifies `Gamma` in non-edge entries, such that the resulting matrix is a variogram matrix with graphical structure described by `graph`. Does so by splitting `graph` at complete separators into smaller subgraphs, and calling `complete_Gamma_general` for each subgraph/submatrix, using multiple cores if available.

Usage

```
complete_Gamma_general_split(
  Gamma,
  graph,
  N = 10000,
  sub_tol = get_large_tol() * 0.001,
  check_tol = 100,
  mc_cores_overwrite = NULL,
  final_tol = get_large_tol()
)
```

Arguments

Gamma	Numeric $d \times d$ variogram matrix.
graph	`igraph::graph()` object.
N	Maximum number of iterations.
sub_tol	Numeric scalar. Tolerance to be used when completing submatrices. Should be smaller than `final_tol`.
check_tol	Numeric/integer scalar. How often to check the tolerance when completing submatrices.
mc_cores_overwrite	`NULL` or numeric/integer scalar. Maximal number of cores to use.
final_tol	Numeric scalar. Check convergence of the final result with this tolerance. Skipped if this value is < 0.

Value

A completed $d \times d$ variogram matrix.

See Also

Other matrix completion related topics: `complete_Gamma_decomposable()`, `complete_Gamma_general_demo()`, `complete_Gamma_general()`, `complete_Gamma()`

---

danube — *Upper Danube basin dataset*

Description

A dataset containing river discharge data for tributaries of the Danube.

Usage

```
danube
```

Format

A named `list` with four entries

- `data_clustered` A numeric matrix, containing pre-processed discharge data for each gauging station
- `data_raw` A numeric matrix, containing daily (raw) discharge data for each gauging station
- `info` A data frame, containing information about each gauging station
- `flow_edges` A two-column numeric matrix. Each row contains the indices (in `info`) of a pair of gauging stations that are directly connected by a river.

Details

To obtain the matrix `data_clustered`, the daily discharge data from the summer months of 1960 to 2010, given in `data_raw`, was declustered, yielding between seven and ten observations per year. Each row corresponds to one observation from this declustered time series, the *non-unique rownames* indicate which year an observation is from. Each column corresponds to one of the gauging stations, with column indices in `data_raw`/`data_clustered` corresponding to row indices in `info`. See (Asadi et al. 2015) for details on the preprocessing and declustering.

`info` is a data frame containing the following information for each of the gauging stations or its corresponding catchment area:

`RivNames` Name of the river at the gauging station

`Lat`, `Long` Coordinates of the gauging station

`Lat_Center`, `Long_Center` Coordinates of the center of the catchment corresponding to the gauging station

`Alt` Mean altitude of the catchment

`Area` Area of the catchment corresponding to the gauging station

`Slope` Mean slope of the catchment

`PlotCoordX`, `PlotCoordY` X-Y-coordinates which can be used to arrange the gauging stations when plotting a flow graph.

Source

Bavarian Environmental Agency `https://www.gkd.bayern.de`.

References

Asadi P, Davison AC, Engelke S (2015). "Extremes on river networks." *Ann. Appl. Stat.*, **9**(4), 2023 – 2050. doi:10.1214/15AOAS863.

See Also

Other danubeData: `getDanubeFlowGraph()`, `plotDanube()`

Other datasets: `flights`

Examples

```
dim(danube$data_clustered)
colnames(danube$info)
```

data2mpareto *Data standardization to multivariate Pareto scale*

Description

Transforms the `data` matrix empirically to the multivariate Pareto scale.

Usage

```
data2mpareto(data, p, na.rm = FALSE)
```

Arguments

`data`	Numeric $n \times d$ matrix, where `n` is the number of observations and `d` is the dimension.
`p`	Numeric between 0 and 1. Probability used for the quantile to threshold the data.
`na.rm`	Logical. If rows containing NAs should be removed.

Details

The columns of the `data` matrix are first transformed empirically to standard Pareto distributions. Then, only the observations where at least one component exceeds the `p`-quantile of the standard Pareto distribution are kept. Those observations are finally divided by the `p`-quantile of the standard Pareto distribution to standardize them to the multivariate Pareto scale.

If `na.rm` is `FALSE`, missing entries are left as such during the transformation of univariate marginals. In the thresholding step, missing values are considered as `-Inf`.

Value

Numeric $m \times d$ matrix, where `m` is the number of rows in the original `data` matrix that are above the threshold.

See Also

Other parameter estimation methods: `emp_chi_multdim()`, `emp_chi()`, `emp_vario()`, `emtp2()`, `fmpareto_HR_MLE()`, `fmpareto_graph_HR()`, `loglik_HR()`

Other structure estimation methods: `eglatent()`, `eglearn()`, `emst()`, `fit_graph_to_Theta()`

Examples

```
n <- 20
d <- 4
p <- .8
G <- cbind(
  c(0, 1.5, 1.5, 2),
  c(1.5, 0, 2, 1.5),
```

```
  c(1.5, 2, 0, 1.5),
  c(2, 1.5, 1.5, 0)
)

set.seed(123)
my_data <- rmstable(n, "HR", d = d, par = G)
data2mpareto(my_data, p)
```

---

eglatent *Learning extremal graph structure with latent variables*

Description

Following the methodology from Engelke and Taeb (2024), fits an extremal graph structure with latent variables.

Usage

```
eglatent(
  Gamma,
  lam1_list = c(0.1, 0.15, 0.19, 0.205),
  lam2_list = c(2),
  refit = TRUE,
  verbose = FALSE
)
```

Arguments

`Gamma`	conditionally negative semidefinite matrix. This will be typically the empirical variogram matrix.
`lam1_list`	Numeric vector of non-negative regularization parameters for eglatent. Default is `lam1_list = c(0.1, 0.15, 0.19, 0.205)`.
`lam2_list`	Numeric vector of non-negative regularization parameters for eglatent. Default is `lam2_list = c(2)`.
`refit`	Logical scalar, if TRUE then the model is refit on the estimated graph to obtain an estimate of the Gamma matrix on that graph. Default is `refit = TRUE`.
`verbose`	Logical scalar, indicating whether to print progress updates.

Value

The function fits one model for each combination of values in `lam1_list` and `lam2_list`. All returned objects have one entry per model. List consisting of: #’

`graph`	A list of `igraph::graph` objects representing the fitted graphs.
`rk`	Numeric vector containing the estimated ranks of the latent variables.

`G_est`	A list of numeric estimated $d \times d$ variogram matrices Γ corresponding to the fitted graphs.
`G_refit`	A list of numeric estimated $d \times d$ variogram matrices Γ refitted with fixed graphs corresponding to the fitted graphs.
`lambdas`	A list containing the values of `lam1_list` and `lam2_list` used for the model fit.

References

Engelke S, Taeb A (2024). "Extremal graphical modeling with latent variables." 2403.09604.

See Also

Other structure estimation methods: `data2mpareto()`, `eglearn()`, `emst()`, `fit_graph_to_Theta()`

`eglearn` *Learning extremal graph structure*

Description

Following the methodology from Engelke et al. (2022), fits an extremal graph structure using the neighborhood selection approach (see Meinshausen and Bühlmann (2006)) or graphical lasso (see Friedman et al. (2008)).

Usage

```
eglearn(
  data,
  p = NULL,
  rholist = c(0.1, 0.15, 0.19, 0.205),
  reg_method = c("ns", "glasso"),
  complete_Gamma = FALSE
)
```

Arguments

`data`	Numeric $n \times d$ matrix, where `n` is the number of observations and `d` is the dimension.
`p`	Numeric between 0 and 1 or `NULL`. If `NULL` (default), it is assumed that the `data` are already on multivariate Pareto scale. Else, `p` is used as the probability in the function `data2mpareto()` to standardize the `data`.
`rholist`	Numeric vector of non-negative regularization parameters for the lasso. Default is `rholist = c(0.1, 0.15, 0.19, 0.205)`. For details see `glasso::glassopath()`.
`reg_method`	One of `"ns", "glasso"`, for neighborhood selection and graphical lasso, respectively. Default is `reg_method = "ns"`. For details see Meinshausen and Bühlmann (2006), Friedman et al. (2008).
`complete_Gamma`	Whether you want to try fto complete Gamma matrix. Default is `complete_Gamma = FALSE`.

Value

List made of:

- `graph` A list of `igraph::graph` objects representing the fitted graphs for each `rho` in `rholist`.
- `Gamma` A list of numeric estimated $d \times d$ variogram matrices Γ corresponding to the fitted graphs, for each `rho` in `rholist`. If `complete_Gamma = FALSE` or the underlying graph is not connected, it returns `NULL`.
- `rholist` The list of penalty coefficients.
- `graph_ic` A list of `igraph::graph` objects representing the optimal graph according to the `aic`, `bic`, and `mbic` information criteria. If `reg_method = "glasso"`, it returns a list of `NULL`.
- `Gamma_ic` A list of numeric $d \times d$ estimated variogram matrices Γ corresponding to the `aic`, `bic`, and `mbic` information criteria. If `reg_method = "glasso"`, `complete_Gamma = FALSE`, or the underlying graph is not connected, it returns a list of `NULL`.

References

Engelke S, Lalancette M, Volgushev S (2022). "Learning extremal graphical structures in high dimensions." doi:10.48550/ARXIV.2111.00840, Available from https://arxiv.org/abs/2111.00840., 2111.00840, https://arxiv.org/abs/2111.00840.

Friedman J, Hastie T, Tibshirani R (2008). "Sparse inverse covariance estimation with the graphical lasso." *Biostatistics*, **9**(3), 432–441.

Meinshausen N, Bühlmann P (2006). "High-dimensional graphs and variable selection with the Lasso." *Ann. Statist.*, **34**(3), 1436 – 1462. doi:10.1214/009053606000000281.

See Also

Other structure estimation methods: `data2mpareto()`, `eglatent()`, `emst()`, `fit_graph_to_Theta()`

Examples

```
set.seed(2)
m <- generate_random_model(d=6)
y <- rmpareto(n=500, par=m$Gamma)
ret <- eglearn(y)
```

---

emp_chi *Empirical estimation of extremal correlation matrix χ*

Description

Estimates empirically the matrix of bivariate extremal correlation coefficients χ.

Usage

```
emp_chi(data, p = NULL)

emp_chi_pairwise(data, p = NULL, verbose = FALSE)
```

Arguments

data	Numeric $n \times d$ matrix, where n is the number of observations and d is the dimension.
p	Numeric scalar between 0 and 1 or NULL. If NULL (default), it is assumed that the data are already on multivariate Pareto scale. Else, p is used as the probability in data2mpareto() to standardize the data.
verbose	Print verbose progress information

Details

emp_chi_pairwise calls emp_chi for each pair of observations. This is more robust if the data contains many NAs, but can take rather long.

Value

Numeric matrix $d \times d$. The matrix contains the bivariate extremal coefficients χ_{ij}, for $i, j = 1, ..., d$.

See Also

Other parameter estimation methods: data2mpareto(), emp_chi_multdim(), emp_vario(), emtp2(), fmpareto_HR_MLE(), fmpareto_graph_HR(), loglik_HR()

Examples

```
n <- 100
d <- 4
p <- .8
Gamma <- cbind(
  c(0, 1.5, 1.5, 2),
  c(1.5, 0, 2, 1.5),
  c(1.5, 2, 0, 1.5),
  c(2, 1.5, 1.5, 0)
)

set.seed(123)
my_data <- rmstable(n, "HR", d = d, par = Gamma)
emp_chi(my_data, p)
```

emp_chi_multdim — *Empirical estimation of extremal correlation χ*

Description

Estimates the d-dimensional extremal correlation coefficient χ empirically.

Usage

```
emp_chi_multdim(data, p = NULL)
```

Arguments

`data`	Numeric $n \times d$ matrix, where `n` is the number of observations and `d` is the dimension.
`p`	Numeric scalar between 0 and 1 or `NULL`. If `NULL` (default), it is assumed that the `data` are already on multivariate Pareto scale. Else, `p` is used as the probability in `data2mpareto()` to standardize the `data`.

Value

Numeric scalar. The empirical `d`-dimensional extremal correlation coefficient χ for the `data`.

See Also

Other parameter estimation methods: `data2mpareto()`, `emp_chi()`, `emp_vario()`, `emtp2()`, `fmpareto_HR_MLE()`, `fmpareto_graph_HR()`, `loglik_HR()`

Examples

```
n <- 100
d <- 2
p <- .8
G <- cbind(
  c(0, 1.5),
  c(1.5, 0)
)

set.seed(123)
my_data <- rmstable(n, "HR", d = d, par = G)
emp_chi_multdim(my_data, p)
```

emp_vario *Estimation of the variogram matrix Γ of a Huesler-Reiss distribution*

Description

Estimates the variogram of the Huesler-Reiss distribution empirically.

Usage

```
emp_vario(data, k = NULL, p = NULL)

emp_vario_pairwise(data, k = NULL, p = NULL, verbose = FALSE)
```

Arguments

data	Numeric $n \times d$ matrix, where n is the number of observations and d is the dimension.
k	Integer between 1 and d. Component of the multivariate observations that is conditioned to be larger than the threshold p. If NULL (default), then an average over all k is returned.
p	Numeric between 0 and 1 or NULL. If NULL (default), it is assumed that the data are already on multivariate Pareto scale. Else, p is used as the probability in the function data2mpareto() to standardize the data.
verbose	Print verbose progress information

Details

emp_vario_pairwise calls emp_vario for each pair of observations. This is more robust if the data contains many NAs, but can take rather long.

Value

Numeric $d \times d$ matrix. The estimated variogram of the Huesler-Reiss distribution.

See Also

Other parameter estimation methods: data2mpareto(), emp_chi_multdim(), emp_chi(), emtp2(), fmpareto_HR_MLE(), fmpareto_graph_HR(), loglik_HR()

Examples

```
G <- generate_random_Gamma(d=5)
y <- rmpareto(n=100, par=G)
Ghat <- emp_vario(y)
```

`emst` *Fitting extremal minimum spanning tree*

Description

Fits an extremal minimum spanning tree, where the edge weights are:

- negative maximized log-likelihoods of the bivariate Huesler-Reiss distributions, if `method = "ML"`. See Engelke and Hitz (2020) for details.
- empirical extremal variogram, if `method = "vario"`. See Engelke and Volgushev (2022) for details.
- empirical extremal correlation, if `method = "chi"`. See Engelke and Volgushev (2022) for details.

Usage

```
emst(data, p = NULL, method = c("vario", "ML", "chi"), cens = FALSE)
```

Arguments

`data`	Numeric $n \times d$ matrix, where `n` is the number of observations and `d` is the dimension.
`p`	Numeric between 0 and 1 or `NULL`. If `NULL` (default), it is assumed that the `data` are already on multivariate Pareto scale. Else, `p` is used as the probability in the function `data2mpareto()` to standardize the `data`.
`method`	One of `"vario"`, `"ML"`, `"chi"`. Default is `method = "vario"`.
`cens`	Logical. This argument is considered only if `method = "ML"`. If `TRUE`, then censored likelihood contributions are used for components below the threshold. By default, `cens = FALSE`.

Value

List consisting of:

`graph`	An `igraph::graph` object. The fitted minimum spanning tree.
`Gamma`	Numeric $d \times d$ estimated variogram matrix Γ corresponding to the fitted minimum spanning tree.

References

Engelke S, Hitz AS (2020). "Graphical models for extremes (with discussion)." *J. R. Stat. Soc. Ser. B Stat. Methodol.*, **82**, 871–932.

Engelke S, Volgushev S (2022). "Structure learning for extremal tree models." *J. R. Stat. Soc. Ser. B Stat. Methodol.*. doi:10.1111/rssb.12556, Forthcoming, https://rss.onlinelibrary.wiley.com/doi/pdf/10.1111/rssb.12556.

See Also

Other structure estimation methods: data2mpareto(), eglatent(), eglearn(), fit_graph_to_Theta()

Examples

```
## Fitting a 4-dimensional HR minimum spanning tree
my_graph <- igraph::graph_from_adjacency_matrix(
  rbind(
    c(0, 1, 0, 0),
    c(1, 0, 1, 1),
    c(0, 1, 0, 0),
    c(0, 1, 0, 0)
  ),
  mode = "undirected"
)
n <- 100
Gamma_vec <- c(.5, 1.4, .8)
complete_Gamma(Gamma = Gamma_vec, graph = my_graph) ## full Gamma matrix

set.seed(123)
my_data <- rmpareto_tree(n, "HR", tree = my_graph, par = Gamma_vec)
my_fit <- emst(my_data, p = NULL, method = "ML", cens = FALSE)
```

---

emtp2 — *Performs Gaussian likelihood optimization under Laplacian matrix constraints.*

Description

This function implements a block descent algorithm to find the maximum of the Gaussian log-likelihood under the constraint that the concentration matrix is a Laplacian matrix. See Röttger et al. (2021) for details.

Usage

```
emtp2(Gamma, tol = 1e-06, verbose = TRUE, initial_point = TRUE)
```

Arguments

`Gamma`	conditionally negative semidefinite matrix. This will be typically the empirical variogram matrix.
`tol`	The convergence tolerance. The algorithm terminates when the sum of absolute differences between two iterations is below `tol`.
`verbose`	if TRUE (default) the output will be printed.
`initial_point`	if TRUE (default), the algorithm will construct an initial point before the iteration steps.

Value

A list consisting of:

`G_emtp2`	The optimal value of the variogram matrix
`it`	The number of iterations

References

Röttger F, Engelke S, Zwiernik P (2021). “Total positivity in multivariate extremes.” doi:10.48550/ARXIV.2112.14727, `https://arxiv.org/abs/2112.14727`.

See Also

Other parameter estimation methods: `data2mpareto()`, `emp_chi_multdim()`, `emp_chi()`, `emp_vario()`, `fmpareto_HR_MLE()`, `fmpareto_graph_HR()`, `loglik_HR()`

`ensure_matrix_symmetry`

Ensure numerical matrix symmetry/zero values

Description

Ensures the symmetry of a square matrix by averaging it with its transpose. Optionally verifies that the matrix was close to symmetric before.

Makes sure zeros are "numerically zero", by truncating all small values.

Usage

```
ensure_matrix_symmetry(M, checkTol = Inf, alert = NULL)

truncate_zeros(M, tol = get_small_tol())

ensure_matrix_symmetry_and_truncate_zeros(
  M,
  tol = get_small_tol(),
  checkTol = Inf
)
```

Arguments

`M`	Numeric square matrix.
`checkTol`	Positive scalar. If the maximum absolute difference between `M` and `t(M)` is larger, show a warning.
`alert`	Passed to `get_alert_function`: `NULL` or `TRUE` to read the option value, `FALSE` to return a dummy function, or a function that takes an arbitrary number of strings as arguments (e.g. `stop()`).
`tol`	All entries with absolute value below this value are truncated to zero.

Value

The adjusted value of M.

See Also

Other input validation functions: checkGamma(), check_graph(), check_partial_matrix_and_graph()

---

fit_graph_to_Theta *Experimental: Fit graph using empirical Theta matrix*

Description

Fits a graph to an empirical Gamma matrix by computing the corresponding Theta matrix using Gamma2Theta() and greedily chooses m edges that correspond to high absolute values in Theta.

Usage

```
fit_graph_to_Theta(data, m = NULL, Gamma_emp = NULL)
```

Arguments

data	The (standardized) data from which to compute Gamma
m	The number of edges to add, defaults to the number of edges in a tree
Gamma_emp	The empirical Gamma matrix (can be NULL if data is given)

Value

A list containing an [igraph::graph] object and a fitted Gamma matrix

See Also

Other structure estimation methods: data2mpareto(), eglatent(), eglearn(), emst()

---

flightCountMatrixToConnectionList
Convert flight counts to connection list

Description

Convert a numeric matrix containing flight counts between airports to a data frame containing a list of connections.

Usage

```
flightCountMatrixToConnectionList(nFlightsPerConnection, directed = TRUE)
```

Arguments

`nFlightsPerConnection`
: A square, numeric matrix with identical column- and row-names. Each entry represents the number of flights from the airport indexing the row to the airport indexing the column in some arbitrary time period.

`directed`
: Logical scalar. Whether flights A->B and B->A should be considered separately.

Value

A data frame with columns `departureAirport`, `arrivalAirport`, `nFlights`. Each row represents one connection with >=1 flights in the input matrix.

See Also

Other flight data related topics: `flights`, `getFlightDelayData()`, `getFlightGraph()`, `plotFlights()`

Examples

```
flightCountMatrixToConnectionList(flights$flightCounts[1:100, 1:100, 1])
```

---

flights — *Flights delay data*

Description

A dataset containing daily total delays of major U.S. airlines. The raw data was obtained from the U.S. Bureau of Transportation Statistics, and pre-processed as described in Hentschel et al. (2022). *Note: The CRAN version of this package contains only data from 2010-2013. The full dataset is available in the GitHub version of this package.*

Usage

```
flights
```

Format

A named `list` with three entries:

`airports` A `data.frame`, containing information about US airports

`delays` A numeric matrix, containing daily aggregated delays at the airports in the dataset

`flightCounts` A numeric array, containing yearly flight numbers between airports in the dataset

Details

`flightCounts` is a three-dimensional array, containing the number of flights in the dataset between each pair of airports, aggregated on a yearly basis. Each entry is the total number of flights between the departure airport (row) and destination airport (column) in a given year (dimension 3). This array does not contain any `NA`s, even if an airport did not operate at all in a given year, which is simply indicated by zeros.

`delays` is a three-dimensional array containing daily total positive delays, in minutes, of incoming and outgoing flights respectively. Each column corresponds to an airport in the dataset and each row corresponds to a day. The third dimension has length two, `'arrivals'` containing delays of incoming flights and `'departures'` containing delays of outgoing flights. Zeros indicate that there were flights arriving/departing at that airport on a given day, but none of them had delays. `NA`s indicate that there were no flights arriving/departing at that airport on that day at all.

`airports` is a data frame containing the following information about a number of US airports. Some entries are missing, which is indicated by `NA`s.

`IATA` 3-letter IATA code

`Name` name of the airport

`City` main city served by the airport

`Country` country or territory where the airport is located (mostly `"United States"`)

`ICAO` 4-letter ICAO code

`Latitude` latitude of the airport, in decimal degrees

`Longitude` longitude of the airport, in decimal degrees

`Altitude` altitude of the airport, in feet

`Timezone` timezone of the airport, in hours offset from UTC

`DST` Daylight savings time used at the airport. 'A'=US/Canada, 'N'=None.

`Timezone2` name of the timezone of the airport

Source

Raw delays data:

- https://www.bts.dot.gov/browse-statistical-products-and-data/bts-publications/airline-service-quality-performance-234-time

Fields/Forms used in the raw data:

- https://esubmit.rita.dot.gov/ViewReports.aspx
- https://esubmit.rita.dot.gov/On-Time-Form1.aspx
- https://esubmit.rita.dot.gov/On-Time-Form3A.aspx

Airports (includes license information):

- https://openflights.org/data

References

Hentschel M, Engelke S, Segers J (2022). "Statistical Inference for Hüsler-Reiss Graphical Models Through Matrix Completions." doi:10.48550/ARXIV.2210.14292, https://arxiv.org/abs/2210.14292.

See Also

Other flight data related topics: flightCountMatrixToConnectionList(), getFlightDelayData(), getFlightGraph(), plotFlights()

Other datasets: danube

Examples

```
# Get total number of flights in the dataset:
totalFlightCounts <- apply(flights$flightCounts, c(1,2), sum)

# Get number of flights for specific years in the dataset:
flightCounts_10_11 <- apply(flights$flightCounts[,,c('2010', '2011')], c(1,2), sum)

# Get list of connections from 2008:
connections_10 <- flightCountMatrixToConnectionList(flights$flightCounts[,,'2010'])
```

---

fmpareto_graph_HR *Parameter fitting for Huesler-Reiss graphical models*

Description

Fits the parameter matrix (variogram) of a multivariate Huesler-Reiss Pareto distribution with a given graphical structure, using maximum-likelihood estimation or the empirical variogram.

Usage

```
fmpareto_graph_HR(
  data,
  graph,
  p = NULL,
  method = c("vario", "ML"),
  handleCliques = c("average", "full", "sequential"),
  ...
)
```

Arguments

data — Numeric $n \times d$ matrix, where n is the number of observations and d is the number of dimensions.

`graph`	Undirected, connected [`igraph::graph`] object with `d` vertices, representing the graphical structure of the fitted Huesler-Reiss model.
`p`	Numeric between 0 and 1 or `NULL`. If `NULL` (default), it is assumed that the `data` is already on a multivariate Pareto scale. Else, `p` is used as the probability in the function `data2mpareto()` to standardize the `data`.
`method`	One of `c('vario', 'ML')`, with `'vario'` as default, indicating the method to be used for parameter estimation. See details.
`handleCliques`	How to handle cliques and separators in the graph. See details.
`...`	Arguments passed to `fmpareto_HR_MLE()`. Currently `cens`, `maxit`, `optMethod`, and `useTheta` are supported.

Details

If `handleCliques='average'`, the marginal parameter matrix is estimated for each maximal clique of the `graph` and then combined into a partial parameter matrix by taking the average of entries from overlapping cliques. Lastly, the full parameter matrix is computed using `complete_Gamma()`.

If `handleCliques='full'`, first the full parameter matrix is estimated using the specified `method` and then the non-edge entries are adjusted such that the final parameter matrix has the graphical structure indicated by `graph`.

If `handleCliques='sequential'`, `graph` must be decomposable, and `method='ML'` must be specified. The parameter matrix is first estimated on the (recursive) separators and then on the rest of the cliques, keeping previously estimated entries fixed.

If `method='ML'`, the computational cost is mostly influenced by the total size of the graph (if `handleCliques='full'`) or the size of the cliques, and can already take a significant amount of time for modest dimensions (e.g. `d=3`).

Value

The estimated parameter matrix.

See Also

Other parameter estimation methods: `data2mpareto()`, `emp_chi_multdim()`, `emp_chi()`, `emp_vario()`, `emtp2()`, `fmpareto_HR_MLE()`, `loglik_HR()`

`fmpareto_HR_MLE` *Parameter fitting for multivariate Huesler-Reiss Pareto distribution*

Description

Fits the parameters of a multivariate Huesler-Reiss Pareto distribution using (censored) maximum likelihood estimation.

Usage

```
fmpareto_HR_MLE(
  data,
  p = NULL,
  cens = FALSE,
  init = NULL,
  fixParams = integer(0),
  useTheta = TRUE,
  maxit = 100,
  graph = NULL,
  optMethod = "BFGS",
  nAttemptsFixInit = 3
)
```

Arguments

- `data` — Numeric $n \times d$ matrix, where `n` is the number of observations and `d` is the number of dimensions.
- `p` — Numeric scalar between 0 and 1 or `NULL`. If `NULL` (default), it is assumed that the `data` is already on a multivariate Pareto scale. Else, `p` is used as the probability in `data2mpareto()` to standardize the data.
- `cens` — Logical scalar. If true, then censored likelihood contributions are used for components below the threshold. This is computationally expensive and by default `cens = FALSE`.
- `init` — Numeric vector or numeric matrix. Initial parameter values in the optimization. If `NULL`, the empirical variogram is used instead. Otherwise should be a numeric vector with one entry per edge in `graph`, or a complete variogram/precision matrix.
- `fixParams` — Numeric or logical vector. Indices of the parameter vectors that are kept fixed (identical to `init`) during the optimization. Default is `integer(0)`.
- `useTheta` — Logical. Whether to perform the MLE optimization in terms of Theta or Gamma.
- `maxit` — Positive integer. The maximum number of iterations in the optimization.
- `graph` — Graph object from `igraph` package or `NULL` (implying the complete graph).
- `optMethod` — String. A valid optimization method used by the function stats::optim. By default, `method = "BFGS"`.
- `nAttemptsFixInit` — Numeric. If `useTheta=TRUE` and the initial parameter `init` is not valid, attempt to fix it first by making sure all off-diagonal entries are negative and then adding some random noise at most this many times.

Details

Only the parameters corresponding to edges in `graph` are optimized, the remaining entries are implied by the graphical structure. If `graph` is `NULL`, the complete graph is used. The optimization is done either in terms of the variogram (Gamma) or precision matrix (Theta), depending on the value of `useTheta`. If `graph` is non-decomposable, `useTheta=TRUE` is significantly faster, otherwise they are similar in performance.

Value

List consisting of:

`convergence`	Logical. Indicates whether the optimization converged or not.
`Gamma`	Numeric `d x d` matrix. Fitted variogram matrix.
`Theta`	Numeric `d x d` matrix. Fitted precision matrix.
`par`	Numeric vector. Optimal parameters, including fixed parameters.
`par_opt`	Numeric. Optimal parameters, excluding fixed parameters.
`nllik`	Numeric. Optimal value of the negative log-likelihood function.
`hessian`	Numeric matrix. Estimated Hessian matrix of the estimated parameters.

See Also

Other parameter estimation methods: `data2mpareto()`, `emp_chi_multdim()`, `emp_chi()`, `emp_vario()`, `emtp2()`, `fmpareto_graph_HR()`, `loglik_HR()`

`Gamma2graph` *Convert matrix to graph*

Description

Creates a graph object representing the graph structure implied by a parameter matrix.

Usage

```
Gamma2graph(Gamma, tol = get_large_tol(), check = TRUE)

Sigma2graph(Sigma, tol = get_large_tol(), k = NULL, full = FALSE, check = TRUE)

Theta2graph(Theta, tol = get_large_tol(), k = NULL, full = FALSE, check = TRUE)

partialMatrixToGraph(M)
```

Arguments

`Gamma`	Numeric $d \times d$ variogram matrix.
`tol`	Numeric scalar. Entries in the precision matrix with absolute value smaller than this are considered to be zero.
`check`	Whether to check the inputs and call `ensure_matrix_symmetry_and_truncate_zeros` on the outputs.
`Sigma`	Numeric $d \times d$ or $(d-1) \times (d-1)$ covariance matrix.
`k`	`NULL` if the input/output matrix is Σ/Θ. Else, an integer between 1 and d indicating the value of k in Σ^k, Θ^k.
`full`	Logical. If `TRUE` and `!is.null(k)`, the input/output matrix is a $d \times d$ matrix with the kth row filled with zeros.
`Theta`	Numeric $d \times d$ or $(d-1) \times (d-1)$ precision matrix.
`M`	Partial matrix with `NA` entries indicating missing edges.

Value

An igraph::graph object.

See Also

Other parameter matrix transformations: Gamma2Sigma(), chi2Gamma(), par2Matrix()

Gamma2Sigma *Conversion between Huesler-Reiss parameter matrices*

Description

Converts between different matrices that parametrize the same Huesler-Reiss distribution: Γ, Σ, Θ, Σ^k, Θ^k. The $(d-1) \times (d-1)$ matrices Σ^k and Θ^k can also be given/returned as $d \times d$ matrices with the kth row and column filled with zeros.

Usage

```
Gamma2Sigma(Gamma, k = NULL, full = FALSE, check = TRUE)

Gamma2Theta(Gamma, k = NULL, full = FALSE, check = TRUE)

Sigma2Gamma(Sigma, k = NULL, full = FALSE, check = TRUE)

Theta2Gamma(Theta, k = NULL, full = FALSE, check = TRUE)

Sigma2Theta(
  Sigma,
  k1 = NULL,
  k2 = NULL,
  full1 = FALSE,
  full2 = FALSE,
  check = TRUE
)

Theta2Sigma(
  Theta,
  k1 = NULL,
  k2 = NULL,
  full1 = FALSE,
  full2 = FALSE,
  check = TRUE
)

Theta2Theta(
  Theta,
  k1 = NULL,
```

```
  k2 = NULL,
  full1 = FALSE,
  full2 = FALSE,
  check = TRUE
)

Sigma2Sigma(
  Sigma,
  k1 = NULL,
  k2 = NULL,
  full1 = FALSE,
  full2 = FALSE,
  check = TRUE
)

Gamma2Gamma(Gamma, check = TRUE)

matrix2matrix(
  M,
  name1 = c("Gamma", "Sigma", "Theta")[1],
  name2 = c("Gamma", "Sigma", "Theta")[1],
  k1 = NULL,
  k2 = NULL,
  full1 = FALSE,
  full2 = FALSE,
  check = TRUE
)
```

Arguments

`Gamma`	Numeric $d \times d$ variogram matrix.
`k`	`NULL` if the input/output matrix is Σ/Θ. Else, an integer between 1 and d indicating the value of k in Σ^k, Θ^k.
`full`	Logical. If `TRUE` and `!is.null(k)`, the input/output matrix is a $d \times d$ matrix with the kth row filled with zeros.
`check`	Whether to check the inputs and call `ensure_matrix_symmetry_and_truncate_zeros` on the outputs.
`Sigma`	Numeric $d \times d$ or $(d-1) \times (d-1)$ covariance matrix.
`Theta`	Numeric $d \times d$ or $(d-1) \times (d-1)$ precision matrix.
`k1`	`NULL` if the input matrix is Σ/Θ. Else, an integer between 1 and d indicating the value of k in Σ^k, Θ^k.
`k2`	`NULL` if the output matrix is Σ/Θ. Else, an integer between 1 and d indicating the value of k in Σ^k, Θ^k.
`full1`	Logical. If `TRUE` and `!is.null(k1)`, the input is a $d \times d$ matrix with the kth row filled with zeros.
`full2`	Logical. If `TRUE` and `!is.null(k2)`, the output is a $d \times d$ matrix with the kth row filled with zeros.

M	Numeric matrix, Γ, Σ, or Θ.
name1	Name of the input representation.
name2	Name of the output representation.

Details

If k, k1, or k2 is NULL, the corresponding full* argument is ignored.

Gamma2Gamma only checks and returns the input.

matrix2matrix is a wrapper function that calls the corresponding conversion function implied by name1, name2.

Value

The desired parameter matrix corresponding to the specified inputs.

See Also

Other parameter matrix transformations: Gamma2graph(), chi2Gamma(), par2Matrix()

---

generate_random_chordal_graph
Generate random graphs

Description

Generate random graphs with different structures. These do not follow well-defined distributions and are mostly meant for quickly generating test models.

Usage

```
generate_random_chordal_graph(
  d,
  cMin = 2,
  cMax = 6,
  sMin = 1,
  sMax = 4,
  block_graph = FALSE,
  ...
)

generate_random_connected_graph(
  d,
  m = NULL,
  p = 2/(d + 1),
  maxTries = 1000,
  ...
```

```
)

generate_random_tree(d)

generate_random_cactus(d, cMin = 2, cMax = 6)
```

Arguments

d	Number of vertices in the graph
cMin	Minimal size of cliques/blocks (last one might be smaller if necessary)
cMax	Maximal size of cliques/blocks
sMin	Minimal size of separators
sMax	Maximal size of separators
block_graph	Force sMin == sMax == 1 to produce a block graph
...	Ignored, only allowed for compatibility
m	Number of edges in the graph (specify this or p)
p	Probability of each edge being in the graph (specify this or m)
maxTries	Maximum number of tries to produce a connected Erdoes-Renyi graph

Details

generate_random_chordal_graph generates a random chordal graph by starting with a (small) complete graph and then adding new cliques until the specified size is reached. The sizes of cliques and separators can be specified.

generate_random_connected_graph first tries to generate an Erdoes-Renyi graph, if that fails, falls back to producing a tree and adding random edges to that tree.

generate_random_cactus generates a random cactus graph (mostly useful for benchmarking).

Value

An [igraph::graph] object

See Also

Other example generation functions: generate_random_Gamma(), generate_random_graphical_Gamma(), generate_random_integer_Gamma(), generate_random_model(), generate_random_spd_matrix()

generate_random_Gamma *Generate a random Gamma matrix*

Description

Generates a valid Gamma matrix with a given dimension

Usage

```
generate_random_Gamma(d, ...)
```

Arguments

d	Size of the matrix
...	Further arguments passed to generate_random_spd_matrix()

See Also

Other example generation functions: generate_random_chordal_graph(), generate_random_graphical_Gamma(), generate_random_integer_Gamma(), generate_random_model(), generate_random_spd_matrix()

generate_random_graphical_Gamma *Generate a random Gamma matrix for a given graph*

Description

Generates a valid Gamma matrix with conditional independence structure specified by a graph

Usage

```
generate_random_graphical_Gamma(graph, ...)
```

Arguments

graph	An igraph::graph object
...	Further arguments passed to generate_random_spd_matrix()

See Also

Other example generation functions: generate_random_Gamma(), generate_random_chordal_graph(), generate_random_integer_Gamma(), generate_random_model(), generate_random_spd_matrix()

`generate_random_integer_Gamma` *Generate a random Gamma matrix containing only integers*

Description

Generates a random variogram Matrix by producing a $(d-1) \times (d-1)$ matrix `B` with random integer entries between `-b` and `b`, computing `S = B %*% t(B)`, and passing this `S` to `Sigma2Gamma()`. This process is repeated with an increasing `b` until a valid Gamma matrix is produced.

Usage

```
generate_random_integer_Gamma(d, b = 2, b_step = 1)
```

Arguments

`d`	Number of rows/columns in the output matrix
`b`	Initial `b` used in the algorithm described above
`b_step`	By how much `b` is increased in each iteration

Value

A numeric $d \times d$ variogram matrix with integer entries

See Also

Other example generation functions: `generate_random_Gamma()`, `generate_random_chordal_graph()`, `generate_random_graphical_Gamma()`, `generate_random_model()`, `generate_random_spd_matrix()`

Examples

```
generate_random_integer_Gamma(5, 2, 0.1)
```

`generate_random_model` *Generate random Huesler-Reiss Models*

Description

Generates a random connected graph and Gamma matrix with conditional independence structure corresponding to that graph.

Usage

```
generate_random_model(d, graph_type = "general", ...)
```

Arguments

d	Number of vertices in the graph
graph_type	"tree", "block", "decomposable", "complete", or "general"
...	Further arguments passed to functions generating the graph and Gamma matrix

See Also

Other example generation functions: generate_random_Gamma(), generate_random_chordal_graph(), generate_random_graphical_Gamma(), generate_random_integer_Gamma(), generate_random_spd_matrix()

Examples

```
set.seed(1)
d <- 12

generate_random_model(d, 'tree')
generate_random_model(d, 'block')
generate_random_model(d, 'decomposable')
generate_random_model(d, 'general')
generate_random_model(d, 'complete')
```

---

generate_random_spd_matrix

Generate a random symmetric positive definite matrix

Description

Generates a random $d \times d$ symmetric positive definite matrix. This is done by generating a random $d \times d$ matrix B, then computing B %*% t(B), and then normalizing the matrix to approximately single digit entries.

Usage

```
generate_random_spd_matrix(d, bMin = -10, bMax = 10, ...)
```

Arguments

d	Number of rows/columns
bMin	Minimum value of entries in B
bMax	Maximum value of entries in B
...	Ignored, only allowed for compatibility

See Also

Other example generation functions: generate_random_Gamma(), generate_random_chordal_graph(), generate_random_graphical_Gamma(), generate_random_integer_Gamma(), generate_random_model()

`getDanubeFlowGraph` *Get Danube flow graph*

Description

Returns an `igraph::graph` object representing the flow graph of the `danube` dataset.

Usage

```
getDanubeFlowGraph(stationIndices = NULL, directed = FALSE)
```

Arguments

`stationIndices`	Logical or numerical vector. Indicating which stations to include.
`directed`	Logical. Whether the graph should be directed (in the direction of flow).

Value

An `igraph::graph` object.

See Also

Other danubeData: `danube`, `plotDanube()`

`getFlightDelayData` *Get filtered flight delays*

Description

Get filtered flight delay data, containing only a selection of dates and airports. Currently, all possible selections correspond to the case study in Hentschel et al. (2022).

Usage

```
getFlightDelayData(
  what = c("delays", "IATAs", "dates"),
  airportFilter = c("all", "tcCluster", "tcAll"),
  dateFilter = c("all", "tcTrain", "tcTest", "tcAll"),
  delayFilter = c("totals", "arrivals", "departures")[1]
)
```

Arguments

`what`	Whether to get the array of delays (numerical), or just the vector of airport codes (`"IATAs"`, strings) or dates (as strings). Specify exactly one.
`airportFilter`	Which airports to include. Specify exactly one. See details below.
`dateFilter`	Which dates to include. Specify exactly one. See details below.
`delayFilter`	Which kinds of delays to include. Specify one or more. Possible values are `"arrivals"`, `"departures"`, and `"totals"` (computed as sum of arrival and departure delays).

Details

The provided lists of airports and dates correspond to the ones used in the case study of Hentschel et al. (2022). The argument `airportFilter="tcCluster"` corresponds to the airports in the analyzed "Texas Cluster", `airportFilter="tcAll"` corresponds to all airports used in the previous clustering step, `airportFilter="all"` corresponds to all airports in the dataset.

Similarly, `dateFilter="tcTrain"` selects the dates from the training set, `dateFilter="tcTest"` the ones from the test/validation set. To get the union of these sets, specify `dateFilter="tcAll"`. To get all dates in the dataset (possibly more than for "tcAll"), specify `dateFilter="all"`.

Value

If `what="IATAs"` or `what="dates"`, a character vector. If required, it can be converted to `Date` objects using `as.Date()`.

If `what="delays"`, a three-dimensional array or two-dimensional matrix, with dimensions corresponding to dates, airports, and delay types.

References

Hentschel M, Engelke S, Segers J (2022). "Statistical Inference for Hüsler-Reiss Graphical Models Through Matrix Completions." doi:10.48550/ARXIV.2210.14292, `https://arxiv.org/abs/2210.14292`.

See Also

Other flight data related topics: `flightCountMatrixToConnectionList()`, `flights`, `getFlightGraph()`, `plotFlights()`

`getFlightGraph` *Get flight graph*

Description

Convert the info from `flights$flightCounts` to an `igraph::graph` object.

Usage

```
getFlightGraph(IATAs = NULL, years = NULL, minNFlights = 1, directed = FALSE)
```

Arguments

IATAs	Character vector. IATA codes of airports to include
years	Character vector. Years to include (as strings).
minNFlights	Numerical scalar. Minimum number of flights on a connection to be included as an edge.
directed	Logical scalar. Whether flights A->B and B->A should be considered separately.

Value

An igraph::graph object containing a vertex for each airport and an edge whenever there are at least minNFlights between two airports.

See Also

Other flight data related topics: flightCountMatrixToConnectionList(), flights, getFlightDelayData(), plotFlights()

Examples

```
g <- getFlightGraph()
```

---

get_alert_function *Get alert function*

Description

Get a function that can be used to alert the user of invalid inputs. Returns the value implied by the overwrite argument, or the option "graphicalExtremes.default.alert", falling back to warning() if neither is specified.

Usage

```
get_alert_function(overwrite = NULL)
```

Arguments

overwrite	NULL or TRUE to read the option value, FALSE to return a dummy function, or a function that takes an arbitrary number of strings as arguments (e.g. stop()).

Value

A function that takes an arbitrary number of strings as arguments.

See Also

graphicalExtremes-package

Other default parameters: get_mc_cores(), get_small_tol()

---

get_mc_cores *Number of cores to be used in parallel computations*

Description

Helper function that returns the number of cores to be used in parallel computations. Will always be 1 on Windows. On other systems, this value can be set using `setOption('graphicalExtremes.mc.cores', ...)`.

Usage

```
get_mc_cores(overwrite = NULL)
```

Arguments

overwrite	Use this value (if it is valid and not on Windows)

Value

An integer to be used as number of cores

See Also

graphicalExtremes-package

Other default parameters: get_alert_function(), get_small_tol()

---

get_small_tol *Tolerances to be used in computations*

Description

Helper function that returns the tolerance to be used in internal computations.

Usage

```
get_small_tol(overwrite = NULL)

get_large_tol(overwrite = NULL)
```

Arguments

overwrite	`NULL` or numeric scalar. If specified, use this value instead of the option value.

Details

There are two different tolerances used in the package, for details see graphicalExtremes-package. The default values for these tolerances can be set using the options "graphicalExtremes.tol.small" and "graphicalExtremes.tol.large".

Value

A non-negative numerical scalar

See Also

graphicalExtremes-package

Other default parameters: get_alert_function(), get_mc_cores()

---

loglik_HR — *Compute Huesler-Reiss log-likelihood, AIC, and BIC*

Description

Computes (censored) Huesler-Reiss log-likelihood, AIC, and BIC values.

Usage

```
loglik_HR(data, p = NULL, graph = NULL, Gamma, cens = FALSE)
```

Arguments

data	Numeric $n \times d$ matrix. It contains observations following a multivariate HR Pareto distribution.
p	Numeric between 0 and 1 or NULL. If NULL (default), it is assumed that the data are already on multivariate Pareto scale. Else, p is used as the probability in the function data2mpareto() to standardize the data.
graph	An [igraph::graph] object or NULL. The graph must be undirected and connected. If no graph is specified, the complete graph is used.
Gamma	Numeric $n \times d$ matrix. It represents a variogram matrix Γ.
cens	Boolean. If true, then censored log-likelihood is computed. By default, cens = FALSE.

Value

Numeric vector c("loglik"=..., "aic"=..., "bic"=...) with the evaluated log-likelihood, AIC, and BIC values.

See Also

Other parameter estimation methods: data2mpareto(), emp_chi_multdim(), emp_chi(), emp_vario(), emtp2(), fmpareto_HR_MLE(), fmpareto_graph_HR()

`plotDanube` *Plot Danube River Flow Data*

Description

Plotting function to visualize the river flow data from the `danube` dataset. Requires `ggplot2` to be installed.

Usage

```
plotDanube(
  stationIndices = NULL,
  graph = NULL,
  directed = NULL,
  plotStations = TRUE,
  plotConnections = TRUE,
  labelStations = FALSE,
  returnGGPlot = FALSE,
  useStationVolume = FALSE,
  useConnectionVolume = FALSE,
  mapCountries = c("Germany"),
  vertexColors = NULL,
  vertexShapes = NULL,
  edgeColors = NULL,
  xyRatio = NULL,
  clipMap = 1.2,
  useLatex = FALSE,
  edgeAlpha = 0.2
)

plotDanubeIGraph(
  stationIndices = NULL,
  graph = NULL,
  directed = NULL,
  labelStations = TRUE,
  vertexColors = NULL,
  vertexShapes = NULL,
  edgeColors = NULL,
  ...
)
```

Arguments

`stationIndices`	Logical or numerical vector, indicating the stations to be plotted.
`graph`	An `igraph::graph` object or `NULL` to use the flow graph.
`directed`	Logical. Whether to consider the flow graph as directed.

`plotStations`	Logical. Whether to plot the stations.
`plotConnections`	Logical. Whether to plot the connections.
`labelStations`	Logical. Whether to label stations.
`returnGGPlot`	If `TRUE`, a `ggplot2::ggplot` object is returned and not plotted immediately.
`useStationVolume`	Logical. Whether to indicate flow volume at a station by circle size.
`useConnectionVolume`	Logical. Whether to indicate flow volume on a connection by line width.
`mapCountries`	Which country borders to show using `ggplot2::map_data('world', mapCountries)`.
`vertexColors`	Vector with color information for vertices.
`vertexShapes`	Vector with shape information for vertices.
`edgeColors`	Vector with color information for edges.
`xyRatio`	Approximate X-Y-ratio (w.r.t. distance on the ground) of the area shown in the plot.
`clipMap`	Logical or numeric scalar. Whether to ignore the map image when determining the axis limits of the plot. If it is a positive scalar, the plot limits are extended by that factor.
`useLatex`	Whether to format numbers etc. as latex code (useful when plotting to tikz).
`edgeAlpha`	Numeric scalar between 0 and 1. The alpha value to be used when plotting edges/connections.
`...`	Passed through to `igraph::plot.igraph`.

Details

The values of `vertexColors`, `vertexShapes`, and `edgeColors` are interpreted differently by `ggplot2::geom_point`/`ggplot` and `igraph::plot.igraph()`.

`plotDanube` uses a combination of `ggplot2` functions to plot the graph.

`plotDanubeIGraph` uses `igraph::plot.igraph` to plot the graph.

See Also

`plotFlights`

Other danubeData: `danube`, `getDanubeFlowGraph()`

Examples

```
# Basic plot
graphicalExtremes::plotDanube()

# Plot flow volumes
graphicalExtremes::plotDanube(
    clipMap = 1.2,
    useConnectionVolume = TRUE,
    useStationVolume = TRUE
```

```
)

# Plot other graph structures
nStations <- nrow(graphicalExtremes::danube$info)
g <- igraph::erdos.renyi.game(nStations, nStations, 'gnm')
graphicalExtremes::plotDanube(
    clipMap = 1.2,
    graph = g
)
```

---

plotFlights *Plot flight data*

Description

Plotting function to visualize the flight connections from the flights dataset. This function requires the package ggplot2 to be installed.

Usage

```
plotFlights(
  airportIndices = NULL,
  airports_sel = NULL,
  connections_sel = NULL,
  graph = NULL,
  plotAirports = TRUE,
  plotConnections = TRUE,
  labelAirports = FALSE,
  returnGGPlot = FALSE,
  useAirportNFlights = FALSE,
  useConnectionNFlights = FALSE,
  minNFlights = 0,
  map = "state",
  vertexColors = NULL,
  vertexShapes = NULL,
  edgeColors = NULL,
  xyRatio = NULL,
  clipMap = FALSE,
  useLatex = FALSE,
  edgeAlpha = 0.2
)
```

Arguments

airportIndices	The indices of the airports (w.r.t. airports_sel) to include.
airports_sel	The airports to plot. Might be further subset by arguments airportIndices, graph. If NULL, then flights$airports will be used.

connections_sel
: A three columns data frame as output by flightCountMatrixToConnectionList(). If NULL, then flights$nFlights will be used to construct one.

graph
: An optional igraph::graph object, containing a flight graph to plot. Vertices should either match the selected airports in number and order, or be named with the corresponding IATA codes of the airports they represent.

plotAirports
: Logical. Whether to plot the airports specified.

plotConnections
: Logical. Whether to plot the connections specified.

labelAirports
: Logical. Whether to show the IATA code next to each plotted airport.

returnGGPlot
: If TRUE, a ggplot2::ggplot object is returned and not plotted immediately.

useAirportNFlights
: Logical. Whether to vary the size of the circles representing airports in the plot, according to the number of flights at that airport.

useConnectionNFlights
: Logical. Whether to vary the size of the edges representing connections in the plot, according to the number of flights on that connection.

minNFlights
: Numeric scalar. Only plot connections with at least this many flights.

map
: String or data.frame or NULL. What map to use as the background image. Strings are passed to ggplot2::map_data(), data frames are assumed to be the output of ggplot2::map_data().

vertexColors
: Optional vector, named with IATA codes, to be used as colors for the vertices/airports.

vertexShapes
: Optional vector, named with IATA codes, to be used as shapes for the vertices/airports. Is coerced to character.

edgeColors
: Optional vector or symmetric matrix (character or numeric), to be used as colors for edges/connections. If this is a vector, its entries must match the plotted connections (in the order specified in connections_sel or implied by igraph::get.edgelist). If this is a matrix, its row/column names must be IATA codes, or its rows/columns match the plotted airports (in number and order).

xyRatio
: Approximate X-Y-ratio (w.r.t. distance on the ground) of the area shown in the plot.

clipMap
: Logical or numeric scalar. Whether to ignore the map image when determining the axis limits of the plot. If it is a positive scalar, the plot limits are extended by that factor.

useLatex
: Whether to format numbers etc. as latex code (useful when plotting to tikz).

edgeAlpha
: Numeric scalar between 0 and 1. The alpha value to be used when plotting edges/connections.

Value

If returnGGPlot is TRUE, a ggplot2::ggplot object, otherwise NULL.

See Also

plotDanube

Other flight data related topics: flightCountMatrixToConnectionList(), flights, getFlightDelayData(), getFlightGraph()

Examples

```
# Plot all airports in the dataset
plotFlights(plotConnections = FALSE, map = 'world')

# Plot a selection of airports
plotFlights(c('JFK', 'SFO', 'LAX'), useConnectionNFlights = TRUE, useAirportNFlights = TRUE)

# Plot airports with a custom connections graph
IATAs <- c('ACV', 'BFL', 'EUG', 'SFO', 'MRY')
graph <- igraph::make_full_graph(length(IATAs))
plotFlights(IATAs, graph=graph, clipMap = 1.5)
```

---

rmpareto *Sampling of a multivariate Pareto distribution*

Description

Simulates exact samples of a multivariate Pareto distribution.

Usage

```
rmpareto(
  n,
  model = c("HR", "logistic", "neglogistic", "dirichlet"),
  d = NULL,
  par
)
```

Arguments

n	Number of simulations.
model	The parametric model type; one of: • HR (default), • logistic, • neglogistic, • dirichlet.
d	Dimension of the multivariate Pareto distribution. In some cases this can be NULL and will be inferred from par.
par	Respective parameter for the given model, that is,

- Γ, numeric $d \times d$ variogram matrix, if `model` = `HR`.
- $\theta \in (0, 1)$, if `model` = `logistic`.
- $\theta > 0$, if `model` = `neglogistic`.
- α, numeric vector of size `d` with positive entries, if `model` = `dirichlet`.

Details

The simulation follows the algorithm in Engelke and Hitz (2020). For details on the parameters of the Huesler-Reiss, logistic and negative logistic distributions see Dombry et al. (2016), and for the Dirichlet distribution see Coles and Tawn (1991).

Value

Numeric $n \times d$ matrix of simulations of the multivariate Pareto distribution.

References

Coles S, Tawn JA (1991). "Modelling extreme multivariate events." *J. R. Stat. Soc. Ser. B Stat. Methodol.*, **53**, 377–392.

Dombry C, Engelke S, Oesting M (2016). "Exact simulation of max-stable processes." *Biometrika*, **103**, 303–317.

Engelke S, Hitz AS (2020). "Graphical models for extremes (with discussion)." *J. R. Stat. Soc. Ser. B Stat. Methodol.*, **82**, 871–932.

See Also

Other sampling functions: `rmpareto_tree()`, `rmstable_tree()`, `rmstable()`

Examples

```
## A 4-dimensional HR distribution
n <- 10
d <- 4
G <- cbind(
  c(0, 1.5, 1.5, 2),
  c(1.5, 0, 2, 1.5),
  c(1.5, 2, 0, 1.5),
  c(2, 1.5, 1.5, 0)
)

rmpareto(n, "HR", d = d, par = G)

## A 3-dimensional logistic distribution
n <- 10
d <- 3
theta <- .6
rmpareto(n, "logistic", d, par = theta)

## A 5-dimensional negative logistic distribution
```

```
n <- 10
d <- 5
theta <- 1.5
rmpareto(n, "neglogistic", d, par = theta)

## A 4-dimensional Dirichlet distribution
n <- 10
d <- 4
alpha <- c(.8, 1, .5, 2)
rmpareto(n, "dirichlet", d, par = alpha)
```

`rmpareto_tree` *Sampling of a multivariate Pareto distribution on a tree*

Description

Simulates exact samples of a multivariate Pareto distribution that is an extremal graphical model on a tree as defined in Engelke and Hitz (2020).

Usage

```
rmpareto_tree(n, model = c("HR", "logistic", "dirichlet")[1], tree, par)
```

Arguments

`n`	Number of simulations.
`model`	The parametric model type; one of: • `HR` (default), • `logistic`, • `dirichlet`.
`tree`	Graph object from `igraph` package. This object must be a tree, i.e., an undirected graph that is connected and has no cycles.
`par`	Respective parameter for the given `model`, that is, • Γ, numeric $d \times d$ variogram matrix, where only the entries corresponding to the edges of the `tree` are used, if `model` = `HR`. Alternatively, can be a vector of length `d-1` containing the entries of the variogram corresponding to the edges of the given `tree`. • $\theta \in (0, 1)$, vector of length `d-1` containing the logistic parameters corresponding to the edges of the given `tree`, if `model` = `logistic`. • a matrix of size $(d-1) \times 2$, where the rows contain the parameters vectors α of size 2 with positive entries for each of the edges in `tree`, if `model` = `dirichlet`.

Details

The simulation follows the algorithm in Engelke and Hitz (2020). For details on the parameters of the Huesler-Reiss, logistic and negative logistic distributions see Dombry et al. (2016), and for the Dirichlet distribution see Coles and Tawn (1991).

Value

Numeric $n \times d$ matrix of simulations of the multivariate Pareto distribution.

References

Coles S, Tawn JA (1991). "Modelling extreme multivariate events." *J. R. Stat. Soc. Ser. B Stat. Methodol.*, **53**, 377–392.

Dombry C, Engelke S, Oesting M (2016). "Exact simulation of max-stable processes." *Biometrika*, **103**, 303–317.

Engelke S, Hitz AS (2020). "Graphical models for extremes (with discussion)." *J. R. Stat. Soc. Ser. B Stat. Methodol.*, **82**, 871–932.

See Also

Other sampling functions: `rmpareto()`, `rmstable_tree()`, `rmstable()`

Examples

```
## A 4-dimensional HR tree model

my_tree <- igraph::graph_from_adjacency_matrix(rbind(
  c(0, 1, 0, 0),
  c(1, 0, 1, 1),
  c(0, 1, 0, 0),
  c(0, 1, 0, 0)
),
mode = "undirected"
)
n <- 10
Gamma_vec <- c(.5, 1.4, .8)
set.seed(123)
rmpareto_tree(n, "HR", tree = my_tree, par = Gamma_vec)

## A 4-dimensional Dirichlet model with asymmetric edge distributions

alpha <- cbind(c(.2, 1, .5), c(1.5, .6, .8))
rmpareto_tree(n, model = "dirichlet", tree = my_tree, par = alpha)
```

`rmstable` *Sampling of a multivariate max-stable distribution*

Description

Simulates exact samples of a multivariate max-stable distribution.

Usage

```
rmstable(n, model = c("HR", "logistic", "neglogistic", "dirichlet")[1], d, par)
```

Arguments

`n`	Number of simulations.
`model`	The parametric model type; one of: • `HR` (default), • `logistic`, • `neglogistic`, • `dirichlet`.
`d`	Dimension of the multivariate Pareto distribution.
`par`	Respective parameter for the given `model`, that is, • Γ, numeric $d \times d$ variogram matrix, if `model` = `HR`. • $\theta \in (0, 1)$, if `model` = `logistic`. • $\theta > 0$, if `model` = `neglogistic`. • α, numeric vector of size `d` with positive entries, if `model` = `dirichlet`.

Details

The simulation follows the extremal function algorithm in Dombry et al. (2016). For details on the parameters of the Huesler-Reiss, logistic and negative logistic distributions see Dombry et al. (2016), and for the Dirichlet distribution see Coles and Tawn (1991).

Value

Numeric $n \times d$ matrix of simulations of the multivariate max-stable distribution.

References

Coles S, Tawn JA (1991). "Modelling extreme multivariate events." *J. R. Stat. Soc. Ser. B Stat. Methodol.*, **53**, 377–392.

Dombry C, Engelke S, Oesting M (2016). "Exact simulation of max-stable processes." *Biometrika*, **103**, 303–317.

See Also

Other sampling functions: `rmpareto_tree()`, `rmpareto()`, `rmstable_tree()`

Examples

```
## A 4-dimensional HR distribution
n <- 10
d <- 4
G <- cbind(
  c(0, 1.5, 1.5, 2),
```

```
  c(1.5, 0, 2, 1.5),
  c(1.5, 2, 0, 1.5),
  c(2, 1.5, 1.5, 0)
)

rmstable(n, "HR", d = d, par = G)

## A 3-dimensional logistic distribution
n <- 10
d <- 3
theta <- .6
rmstable(n, "logistic", d, par = theta)

## A 5-dimensional negative logistic distribution
n <- 10
d <- 5
theta <- 1.5
rmstable(n, "neglogistic", d, par = theta)

## A 4-dimensional Dirichlet distribution
n <- 10
d <- 4
alpha <- c(.8, 1, .5, 2)
rmstable(n, "dirichlet", d, par = alpha)
```

`rmstable_tree` *Sampling of a multivariate max-stable distribution on a tree*

Description

Simulates exact samples of a multivariate max-stable distribution that is an extremal graphical model on a tree as defined in Engelke and Hitz (2020).

Usage

```
rmstable_tree(n, model = c("HR", "logistic", "dirichlet")[1], tree, par)
```

Arguments

`n`	Number of simulations.
`model`	The parametric model type; one of: • `HR` (default), • `logistic`, • `dirichlet`.
`tree`	Graph object from `igraph` package. This object must be a tree, i.e., an undirected graph that is connected and has no cycles.
`par`	Respective parameter for the given `model`, that is,

- Γ, numeric $d \times d$ variogram matrix, where only the entries corresponding to the edges of the `tree` are used, if `model` = `HR`. Alternatively, can be a vector of length `d-1` containing the entries of the variogram corresponding to the edges of the given `tree`.
- $\theta \in (0, 1)$, vector of length `d-1` containing the logistic parameters corresponding to the edges of the given `tree`, if `model` = `logistic`.
- a matrix of size $(d-1) \times 2$, where the rows contain the parameter vectors α of size 2 with positive entries for each of the edges in `tree`, if `model` = `dirichlet`.

Details

The simulation follows a combination of the extremal function algorithm in Dombry et al. (2016) and the theory in Engelke and Hitz (2020) to sample from a single extremal function. For details on the parameters of the Huesler-Reiss, logistic and negative logistic distributions see Dombry et al. (2016), and for the Dirichlet distribution see Coles and Tawn (1991).

Value

Numeric $n \times d$ matrix of simulations of the multivariate max-stable distribution.

References

Coles S, Tawn JA (1991). "Modelling extreme multivariate events." *J. R. Stat. Soc. Ser. B Stat. Methodol.*, **53**, 377–392.

Dombry C, Engelke S, Oesting M (2016). "Exact simulation of max-stable processes." *Biometrika*, **103**, 303–317.

Engelke S, Hitz AS (2020). "Graphical models for extremes (with discussion)." *J. R. Stat. Soc. Ser. B Stat. Methodol.*, **82**, 871–932.

See Also

Other sampling functions: `rmpareto_tree()`, `rmpareto()`, `rmstable()`

Examples

```
## A 4-dimensional HR tree model

my_tree <- igraph::graph_from_adjacency_matrix(rbind(
  c(0, 1, 0, 0),
  c(1, 0, 1, 1),
  c(0, 1, 0, 0),
  c(0, 1, 0, 0)
),
mode = "undirected"
)
n <- 10
Gamma_vec <- c(.5, 1.4, .8)
```

```
rmstable_tree(n, "HR", tree = my_tree, par = Gamma_vec)

## A 4-dimensional Dirichlet model with asymmetric edge distributions

alpha <- cbind(c(.2, 1, .5), c(1.5, .6, .8))
rmstable_tree(n, model = "dirichlet", tree = my_tree, par = alpha)
```

Index