Package ‘evamtools’

April 14, 2023

Type Package

Title Tools for evolutionary accumulation models or event accumulation models (evam), for now mainly cancer progression models

Version 2.1.16

Date 2022-12-08

Author Ramon Diaz-Uriarte [aut, cre]
Pablo Herrera-Nieto [aut]
Daniele Ramazzotti [ctb],
Rudolf Schill [ctb],
Hesam Montazeri [ctb],
Susana Posada-Cespedes [ctb].

Maintainer Ramon Diaz-Uriarte <`r.diaz@uam.es`>

Description Wrappers to run cancer progression models (CPMs) on cross-sectional data, including Conjuntive Bayesian Networks (CBN ---and their MC-CBN version---), Oncogenetic trees (OT), Mutual Hazard Networks (MHN), Hidden Extended Suppes-Bayes Causal Networks (H-ESBCNs ---PMCE---), and Disjunctive Bayesian Networks (DBN, from the OncoBN package). Tools to represent, graphically, the fitted models (DAGs of restrictions or matrix of hazards, as appropriate), the transition matrices and transition rate matrices (where appropriate) between genotypes and to show frequencies of genotypes sampled from the fitted models. Functions to sample from the fitted models or from random models to facilitate comparing different methods. An interactive Shiny web app allows users to easily visualize the effects of changes in genotype composition and to interactively modify and create datasets from models defined from scratch.

URL `https://github.com/rdiaz02/EvAM-Tools`

BugReports `https://github.com/rdiaz02/EvAM-Tools/issues`

Depends R (>= 4.0.0)

License AGPL-3 + file LICENSE

Encoding UTF-8

LazyData true

Imports igraph, OncoSimulR, stringr, Matrix, parallel, Oncotree , gtools, stringi , plot.matrix , DT, shinyjs, shiny , OncoBN , RhpcBLASctl , Rlinsolve, fastmatrix , graph, Rgraphviz , R.utils , plotly , magrittr , dplyr , tippy , relations

Suggests testthat (>= 3.0.0)

Enhances mccbn

Config/testthat/parallel true

Config/testthat/edition 3

NeedsCompilation yes

R topics documented:

`evam` *Runs the CPMs (or evams)*

Description

Executes all CPMS given a cross sectional data set.

Usage

```
evam(x,
     methods = c("CBN", "OT", "HESBCN", "MHN", "OncoBN", "MCCBN"),
     max_cols = 15,
     cores = detectCores(),
     paths_max = FALSE,
     mhn_opts = list(lambda = 1/nrow(x),
                     omp_threads = ifelse(cores > 1, 1, detectCores())),
     ot_opts = list(with_errors_dist_ot = TRUE),
     cbn_opts = list(
                     omp_threads = 1,
                     init_poset = "OT"
                 ),
     hesbcn_opts = list(
                     MCMC_iter = 100000,
                     seed = NULL,
                     reg = c("bic", "aic", "loglik"),
                     silent = TRUE
                 ),
     oncobn_opts = list(
                     model = "DBN",
                     algorithm = "DP",
                     k = 3,
                     epsilon = min(colMeans(x)/2),
                     silent = TRUE
                 ),
     mccbn_opts = list(
                     model = "OT-CBN",
                     tmp_dir = NULL,
                     addname = NULL,
                     silent = TRUE,
                     L = 100,
                     sampling = c("forward", "add-remove",
                               "backward", "bernoulli", "pool"),
                     max.iter = 100L,
                     update.step.size = 20L,
                     tol = 0.001,
                     max.lambda.val = 1e+06,
                     T0 = 50,
                     adap.rate = 0.3,
                     acceptance.rate = NULL,
                     step.size = NULL,
                     max.iter.asa = 10000L,
                     neighborhood.dist = 1L,
                     adaptive = TRUE,
                     thrds = 1L,
                     verbose = FALSE,
                     seed = NULL)
)
```

Arguments

`x`	cross sectional data
`methods`	Methods to use. A vector with one or more of the following strings, “OT”, “OncoBN”, “CBN”, “MCCBN”, “MHN”, “HESBCN”.
`max_cols`	Maximum number of columns to use in the analysis. If x has > max_cols, selected columns are those with the largest number of events.
`cores`	If larger than 1, use mclapply to run all methods. This is the default. If you use mclapply, MHN and MCCBN should not use OMP (i.e., the number of threads for OMP for MHN and MCCBN should be 1).
`paths_max`	If TRUE, return all paths to the maxim/maxima, with their probabilities. See details for how they are computed.
`mhn_opts`	A list with two named arguments. • lambda: The penalty for MHN. Defaults to 1/nrow(data). (These are not the lambdas as the estimated parameters for the rates of the continuous-time Markov chains for MHN or CBN or HESBCN.) • omp_threads: Number of OMP threads for MHN. Do not pass `thrds` > 1 with `cores` > 1: as with MCCBN, do not use OpenMP threads from forked process from `mclapply`.
`ot_opts`	A list with the single named argument `with_errors_dist_ot`: value for option with `with.errors` in the call to `distribution.oncotree`. A value of TRUE means to incorporate the false positive and negative errors when returning the probabilities of genotypes under OT. Note that for large models using a value of TRUE can result in very long computing times. Default is TRUE.

`cbn_opts` A named list with arguments passed to CBN.

- omp_threads: OMP threads to be used by CBN (set via the environment variable OMP_NUM_THREADS). Defaults to 1. In contrast to MCCBN and MHN, you can set this to a number larger than one even if you set `cores` to a number larger than one (i.e., if we use `mclapply`). It is unclear, though, more than 1 thread will speed things much or what is the best number of threads to use; in fact, sometimes it can even slow things down, in particular if you run multiple evams in parallel.
- cbn_init_poset: Initial poset for CBN; one of "linear" or "OT" (default).

`hesbcn_opts` Named list of arguments used in the fit of H-ESBCN (details in `https://github.com/danro9685/HESBCN`).

- MCMC_iter: Number of MCMC iterations to run; this is argument "-n, –number_samples" in the original H-ESBCN C code. Default: 100000, as in the original implementation. Note that the web app uses a larger default of 200000.
- reg: Regularization: one of `bic` (default), `aic`, `loglik`.
- seed: Seed to run the experiment
- silent: Whether to run show message showing the folder name where HES-BCN is run

`oncobn_opts` Named list of arguments used in the fit of OncoBN. See `fitCPN`.

`mccbn_opts` Named list of arguments used in the fit of MC-CBN. These are `model` (one of `OT-CBN` or `H-CBN2`). The rest are options passed to `adaptive.simulated.annealing`; see the help of `adaptive.simulated.annealing` for details. In addition, the following options:

- tmp_dir: Directory name where the oput is located. This is passed to `adaptive.simulated.annealing`, as argument `outdir`, with addname added, if provided.
- addname: String to append to the temporary directory name. Default is NULL.
- silent: Whether to show a message with the name of the directory where MCCBN is run. This `silen` is different from `mccbn_hcbn2_opts$verbose`.

Note: do not pass `thrds` > 1 with `cores` > 1: as with MHN, do not use OpenMP threads from forked process from `mclapply`.

Details

Probabilities of evolutionary paths or paths of tumor progression

Details and examples on how probabilities of paths are computed are given in Diaz-Uriarte and Vasallo, 2019 (specifically, see section 3 of file S4_Text, `https://doi.org/10.1371/journal.pcbi.1007246.s006`); see also Hosseini et al., 2019. The models used in those papers all had a single local maximum. Here we follow the same procedure also for models with possibly more than one maximum, such as H-ESBCN. Note that in all cases we assume evolution can only move uphill in fitness and never crosses fitness valleys (which excludes, for example, the scenarios considered in Weinreich and Chao, 2005).

Value

A list with named components (that should be self-explanatory). The pattern is method_component.

- OT_model: Data frame with parent and descendant edges, edge weight, and observed and predicted frequencies of genes.
- OT_f_graph: The fitness graph, as a sparse matrix, with weights obtained from the edge weights (this is not a transition rate matrix). See full documentation for details.
- OT_trans_mat: Transition matrix between genotypes. This is really an abuse of what an untimed OT provides. See full documentation for details.
- OT_predicted_genotype_freqs: Probabilities of genotypes from the OT model, as a data frame.
- OT_paths_max: If `paths_max` is TRUE, a list of two components, `paths` and `weights`. The `paths` list is a list of `igraph.vs` (igraph vertex sequences) objects, one for each path; the `weights` is vector of log-probabilities of each path. If `paths_max` is FALSE, the default, NA.
- CBN_model: Similar to the ouput from OT, but with lambdas. The lambda to be used is "rerun_lambda".
- CBN_trans_rate_mat: Transition rate matrix as a sparse matrix.
- CBN_trans_mat: Transition matrix between genotypes, obtained from the transition rate matrix using competing exponentials.
- CBN_td_trans_mat: Time-discretized transition matrix, using the uniformization method; see full documentation for details.
- CBN_predicted_genotype_freqs: Named vector of probabilities of genotypes predicted by the CBN model (under a model where sampling times are distributed as an exponential of rate 1).
- CBN_paths_max: As for OT.
- MCCBN_model: As for CBN, only with one column of λs.
- MCCBN_trans_rate_mat: As for CBN.
- MCCBN_trans_mat: As for CBN.
- MCCBN_td_trans_mat: As for CBN.
- MCCBN_predicted_genotype_freqs: As for CBN.
- MCCBN_paths_max: As for OT.
- MHN_theta: Matrix of estimated thetas (the log-Theta matrix). The values in this matrix can range from minus to plus infinity.
- MHN_trans_rate_mat: As for CBN.
- MHN_trans_mat: As for CBN.
- MHN_td_trans_mat: As for CBN.
- MHN_exp_theta: Matrix of the exponential of thetas; the matrix Θ in Schill et al. (just each theta, exponentiated; not the matrix exponential of the matrix of thetas). These are the multiplicative effects themselves.
- MHN_predicted_genotype_freqs: As for CBN.
- MHN_paths_max: As for OT.
- OncoBN_model: Similar to the ones above (but with a column named theta, instead of lambdas or edge weights), with the additional column dQuoteRelation, that can take values OR (if fitting model DBN) or AND (if fitting model CBN); Single indicates nodes with a single ancestor (where OR or AND make no difference).
- OncoBN_likelihood: Likelihood of the OncoBN model.
- OncoBN_f_graph: As for OT.
- OncoBN_trans_mat: As for OT.
- OncoBN_predicted_genotype_freqs: As for OT.

- OncoBN_fitted_model: DBN or CBN, depending on what you chose.
- OncoBN_epsilon: Epsilon (this is an argument of the call to evam, but it is evaluated after possibly having modified the input data; see below).
- OncoBN_parent_set: .
- OncoBN_paths_max: As for OT.
- HESBCN_model: As for CBN.
- HESBCN_parent_set: As for CBN.
- HESBCN_trans_rate_mat: As for CBN.
- HESBCN_trans_mat: As for CBN.
- HESBCN_td_trans_mat: As for CBN.
- HESBCN_predicted_genotype_freqs: As for CBN.
- HESBCN_paths_max: As for OT.
- original_data: The original data.
- analyzed_data: The data that were actually analyzed. Can differ from the original data because of the data pre-processing steps.
- genotype_id_ordered: A named vector, from 1:number of genotypes, with names the genotypes. This can be useful for sorting; WT has value 1, and genotypes are ordered by increasing number of mutations and, withing number of mutations, alphanumerically.
- all_options: All of the options used, as a list of lists.

Note

For some methods, such as MHN and OncoBN, some parameters tipically depend on the data (lambda and epsilon for MHN and OncoBN, respectively). Since we first examine and possibly modify the input data, the values might not be the ones you thought you entered, as the **options should be evaluated after the data are pre-processed**.

The **data pre-processing** involves, in sequence, these steps:

- Adding pseudosamples: If any gene (column) is always observed mutated (i.e, has a value of 1 for all observations), we add one observation that has no gene mutated.
- Removing genes with no mutations: Any column that has value of 0 for all observations is removed.
- Merging identical columns: Any identical columns are replaced by a single one (with a new identifier, the result of pasting the names of the fused columns separated by a _).
- No more than max_cols: If the “max_cols” argument is not NULL, and if the data set has more columns that max_cols, we keep only max_cols columns of data, those with the largest number of mutations.

Changing only some options: Often, you will want to change only some of the options. You can enter, in the list, only the options you want changed (not the remaining ones). There is one example below.

During the execution, and as messages, the elapsed time of each procedure is reported. This includes executing the model itself and possible additional operations, such as obtaining the transition rate matrix, etc. (So, for example, the time for estimating the matrix of thetas for MHN is much smaller than the reported time here, which also includes building the transition rate matrix).

By default, we do not return paths to maximum/maxima, as their number can grow very quickly with number of genotypes and you only need them if, well, you care about them.

References

OT

- Szabo, A., & Boucher, K. M. (2008). Oncogenetic Trees. In W. Tan, & L. Hanin (Eds.), Handbook of Cancer Models with Applications (pp. 1–24). : World Scientific.

- Desper, R., Jiang, F., Kallioniemi, O. P., Moch, H., Papadimitriou, C. H., & Sch\"affer, A A (1999). Inferring tree models for oncogenesis from comparative genome hybridization data. J Comput Biol, 6(1), 37–51.

CBN and MCCBN

- Beerenwinkel, N., & Sullivant, S. (2009). Markov models for accumulating mutations. Biometrika, 96(3), 645.

- Gerstung, M., Baudis, M., Moch, H., & Beerenwinkel, N. (2009). Quantifying cancer progression with conjunctive Bayesian networks. Bioinformatics, 25(21), 2809–2815. http://dx.doi.org/10.1093/bioinformatics/btp505

- Gerstung, M., Eriksson, N., Lin, J., Vogelstein, B., & Beerenwinkel, N. (2011). The Temporal Order of Genetic and Pathway Alterations in Tumorigenesis. PLoS ONE, 6(11), 27136. http://dx.doi.org/10.1371/journal.pone.0027136

- Montazeri, H., Kuipers, J., Kouyos, R., B\"oni, J\"urg, Yerly, S., Klimkait, T., Aubert, V., … (2016). Large-scale inference of conjunctive Bayesian networks. Bioinformatics, 32(17), 727–735. http://dx.doi.org/10.1093/bioinformatics/btw459

MHN

- Schill, R., Solbrig, S., Wettig, T., & Spang, R. (2020). Modelling cancer progression using Mutual Hazard Networks. Bioinformatics, 36(1), 241–249. http://dx.doi.org/10.1093/bioinformatics/btz513

HESBCN (PMCE)

- Angaroni, F., Chen, K., Damiani, C., Caravagna, G., Graudenzi, A., & Ramazzotti, D. (2021). PMCE: efficient inference of expressive models of cancer evolution with high prognostic power. Bioinformatics, 38(3), 754–762. http://dx.doi.org/10.1093/bioinformatics/btab717

(About terminology: we will often refer to HESBCN, as that is the program we use, as shown here: https://github.com/danro9685/HESBCN. H-ESBCN is part of the PMCE procedure).

OncoBN (DBN)

- Nicol, P. B., Coombes, K. R., Deaver, C., Chkrebtii, O., Paul, S., Toland, A. E., & Asiaee, A. (2021). Oncogenetic network estimation with disjunctive Bayesian networks. Computational and Systems Oncology, 1(2), 1027. http://dx.doi.org/10.1002/cso2.1027

Conditional prediction of genotypes and probabilities of paths from CPMs

- Hosseini, S., Diaz-Uriarte, Ramon, Markowetz, F., & Beerenwinkel, N. (2019). Estimating the predictability of cancer evolution. Bioinformatics, 35(14), 389–397. http://dx.doi.org/10.1093/bioinformatics/btz332

- Diaz-Uriarte, R., & Vasallo, C. (2019). Every which way? On predicting tumor evolution using cancer progression models. PLOS Computational Biology, 15(8), 1007246. http://dx.doi.org/10.1371/journal.pcbi.1007246

- Diaz-Colunga, J., & Diaz-Uriarte, R. (2021). Conditional prediction of consecutive tumor evolution using cancer progression models: What genotype comes next? PLOS Computational Biology, 17(12), 1009055. http://dx.doi.org/10.1371/journal.pcbi.1009055

Reference in details

- Weinreich, D. M., & Chao, L. (2005). Rapid evolutionary escape by large populations from local fitness peaks is likely in nature. Evolution, 59(6), 1175–1182. http://dx.doi.org/10.1111/j.0014-3820.2005.tb01769.x

See Also

sample_evam, plot_evam

Examples

```
data(every_which_way_data)
## Use a small data set for speed.
Dat1 <- every_which_way_data[[16]][1:40, 2:6]

## Use MCCBN only if installed
MCCBN_INSTALLED <- requireNamespace("mccbn", quietly = TRUE)
methods <- c("CBN", "OT", "OncoBN", "MHN", "HESBCN")
if (MCCBN_INSTALLED) {
    methods <- c(methods, "MCCBN")
}

out1 <- evam(Dat1,
            methods = methods)

## Running only some methods and changing some options
## (this example is not necessarily sensible!)
## Of course, we must use the name of the data in an option that is
## data-dependent

out2 <- evam(Dat1,
            methods = c("CBN", "OT", "OncoBN",
                        "MHN"),
            mhn_opts = list(lambda = 5/nrow(Dat1)),
            cbn_opts = list(omp_threads = 2),
            oncobn_opts = list(model = "CBN"))


## Getting paths to maximum/maxima. Using only two methods
## for faster execution
out3 <- evam(Dat1,
            methods = c("MHN", "OncoBN"),
            paths_max = TRUE)

out3$OncoBN_paths_max
out3$MHN_paths_max
```

---

evamtools-deprecated   *Deprecated functions in package 'evamtools'*

Description

These functions are provided for compatibility with older versions of 'evamtools' only, and will be defunct at the next release.

Details

The following functions are deprecated and will be made defunct; use the replacement indicated below:

- plot_CPMs: `plot_evam`
- sample_CPMs: `sample_evam`

`every_which_way_data` *Cancer data sets from Diaz-Uriarte and Vasallo, 2019; also used in Diaz-Colunga and Diaz-Uriarte, 2021.*

Description

Twenty two cancer data sets used in Diaz-Uriarte and Vasallo, 2019, as well as Diaz-Colunga and Diaz-Uriarte, 2021. The data cover six different cancer types (breast, glioblastoma, lung, ovarian, colorectal, and pancreatic cancer), use different types of features (nonsynonymous somatic mutations, copy number alterations, or both) were analyzed in terms of pathways, functional modules, genes, gene events, or mutations (yielding from 3 to 192 different features), and have samples sizes from 27 to 594.

The original sources are listed below. Most of these data sets have been used before in CPM research.

Complete details about sources, processing, and former use in CPM papers are available from S5 Text (https://doi.org/10.1371/journal.pcbi.1007246.s007) of Diaz-Uriarte and Vasallo, 2019.

Usage

```
data("every_which_way_data")
```

Format

A list of length 22. Each element of the list is a data set, with subjects in rows and genes/probes in columns.

References

Bamford S, Dawson E, Forbes S, Clements J, Pettett R, Dogan A, et al. The COSMIC (Catalogue of Somatic Mutations in Cancer) Database and Website. Br J Cancer. 2004;91(2):355–358.

Cancer Genome Atlas Research Network. Comprehensive Genomic Characterization Defines Human Glioblastoma Genes and Core Pathways. Nature. 2008;455(7216):1061–1068.

Cancer Genome Atlas Research Network. Comprehensive Genomic Characterization Defines Human Glioblastoma Genes and Core Pathways. Nature. 2008;455(7216):1061–1068.

Jones S, Zhang X, Parsons DW, Lin JCH, Leary RJ, Angenendt P, et al. Core Signaling Pathways in Human Pancreatic Cancers Revealed by Global Genomic Analyses. Science (New York, NY). 2008;321(5897):1801–6.

Parsons DW, Jones S, Zhang X, Lin JCH, Leary RJ, Angenendt P, et al. An Integrated Genomic Analysis of Human Glioblastoma Multiforme. Science. 2008;321(5897):1807–1812.

Wood LD, Parsons DW, Jones S, Lin J, Sjoblom T, Leary RJ, et al. The Genomic Landscapes of Human Breast and Colorectal Cancers. Science. 2007;318(5853):1108–1113.

Brennan CW, Verhaak RGW, McKenna A, Campos B, Noushmehr H, Salama SR, et al. The Somatic Genomic Landscape of Glioblastoma. Cell. 2013;155(2):462–477.

Ding L, Getz G, Wheeler DA, Mardis ER, McLellan MD, Cibulskis K, et al. Somatic Mutations Affect Key Pathways in Lung Adenocarcinoma. Nature. 2008;455(7216):1069–1075.

Cancer Genome Atlas Research Network. Integrated Genomic Analyses of Ovarian Carcinoma. Nature. 2011;474(7353):609–615.

Knutsen T, Gobu V, Knaus R, Padilla-Nash H, Augustud M, Strausberg RL, et al. The Interactive Online SKY/M-FISH & CGH Database and the Entrez Cancer Chromosomes Search Database: Linkage of Chromosomal Aberrations with the Genome Sequence. Genes, Chromosomes and Cancer. 2005;44(1):52–64.

Piazza R, Valletta S, Winkelmann N, Redaelli S, Spinelli R, Pirola A, et al. Recurrent SETBP1 Mutations in Atypical Chronic Myeloid Leukemia. Nature Genetics. 2013;45(1):18–24.

Cancer Genome Atlas Research Network. Comprehensive Molecular Characterization of Human Colon and Rectal Cancer. Nature. 2012;487(7407):330–337.

Diaz-Uriarte, R., & Vasallo, C. (2019). Every which way? On predicting tumor evolution using cancer progression models. PLOS Computational Biology, 15(8), 1007246. http://dx.doi.org/10.1371/journal.pcbi.1007246

Diaz-Colunga, J., & Diaz-Uriarte, R. (2021). Conditional prediction of consecutive tumor evolution using cancer progression models: What genotype comes next? PLoS Computational Biology, 17(12): e1009055. https://doi.org/10.1371/journal.pcbi.1009055

Examples

```
data(every_which_way_data)
lapply(every_which_way_data, colnames)
lapply(every_which_way_data, dim)

## Run on a piece of one of the above data sets
Dat1 <- every_which_way_data[[16]][1:40, 2:6]
out <- evam(Dat1,
            methods = c("OT", "OncoBN",
                        "MHN"))
```

`ex_mixed_and_or_xor` *Small example data set that shows, for HESBCN, both AND and OR, or AND and XOR, in some runs.*

Description

Synthetic data set used for testing and plotting. HESBCN, with some seeds, will infer both AND and OR, or AND and XOR, or the three of them.

Usage

```
data("ex_mixed_and_or_xor")
```

Format

A data frame with “genes” in columns and “patients” in rows.

Examples

```
data("ex_mixed_and_or_xor")

out_AND_OR_XOR <- evam(ex_mixed_and_or_xor,
                       methods = c("OT", "HESBCN", "MHN", "OncoBN"),
                       hesbcn_opts = list(seed = 26))

plot_evam(out_AND_OR_XOR, plot_type = "trans_mat", top_paths = 4)
```

`examples_csd` *Cross sectional data sets*

Description

A list of cross sectional data set to be used as example inputs, for instance by the Shiny web app. This file was generated by running the script /inst/miscell/examples/toy_datasets.R

Usage

```
data(examples_csd)
```

Format

A list of cross sectional data sets. Each data set includes 1) the data and 2) a name. In some cases there is also dag, or the values of a matrix. A list with 3 items:

csd csd, created by directly entering cross-sectional data.

dag dag, from models with DAGs.

matrix matrix, from MHN.

`plot_evam` *Plot results from EvAMs (CPMs)*

Description

Plots fitted EvAMs (CPMs), both fitted model and custom plots for transition rates and transition probabilities.

Usage

```
plot_evam(
  cpm_output,
  samples = NULL,
  orientation = "horizontal",
  methods = NULL,
  plot_type = "trans_mat",
  label_type = "genotype",
  fixed_vertex_size = FALSE,
  top_paths = NULL
)

plot_CPMs(
  cpm_output,
  samples = NULL,
  orientation = "horizontal",
  methods = NULL,
  plot_type = "trans_mat",
  label_type = "genotype",
  fixed_vertex_size = FALSE,
  top_paths = NULL
)
```

Arguments

`cpm_output` Output from the cpm

`samples` Output from a call to sample_evam. Necessary if you request plot type `transitions`.

`orientation` String. If it is not "vertical" it will be displayed with an horizontal layout. Optional.

`methods` Vector of strings with the names of methods that we want to plot. If NULL, all methods with output in cpm_output. The list of available methods is OT, OncoBN, CBN, MCCBN, MHN, HESBCN.

`plot_type` One of:

- trans_mat: Transition matrix between genotypes (see supporting information for OT and OncoBN). This plots the object of name `trans_mat` from the output of `evam`.
- trans_rate_mat: Transition rate matrix between genotypes; unavailable for OT and OncoBN. This plots the object of name `trans_rate_mat` from the output of `evam`.
- obs_genotype_transitions: Observed transitions during the simulation of the sampling process. This plots the object called `obs_genotype_transitions` from the output of `sample_evam`.

`label_type` Type of label to show. One of:

- genotype: Displays all genes mutated
- acquisition: Only displays the last gente mutated

`fixed_vertex_size`
Boolean. If TRUE, all nodes with have the same size; otherwise, scale them proportional to frequencies of observed data.

`top_paths` Number of most relevant paths to plot. Default NULL will plot all paths. See below, Description, for details about relevance. With many genes, and particularly for MHN (or other methods, when there are no restrictions in the order of accumulation of mutations), using NULL can lead to a very long time to plot.

Value

By default this function creates a top row with the DAG of the CPM or the log-Theta matrix for MHN. The bottom row has a custom plot for the transition matrix, or the transition rate matrix, or the observed genotype transitions.

In the bottom row plots, unless `fixed_vertex_size = TRUE`, the size of genotype nodes is proportional to the observed frequency of genotypes for all plots except for `plot_type = 'obs_genotype_transitions'`, where it is proportional to the genotype frequency as obtained by the sampling from the predictions of each method; if a node (genotype) has no observations, its size is of fixed size. Thus, for all plots except `plot_type = 'obs_genotype_transitions'` the size of the genotype nodes is the same among methods, but the size of the genotype nodes can differ between methods for `plot_type = 'obs_genotype_transitions'`.

In the top plots, in the DAGs, when a node has two or more incoming edges, color depends on the type of relationship. (With a single incoming edge, there is no difference in model behavior with type of edge and, for consistency with OT and CBN, nodes with a Single parent have edges colored the same way as nodes with two or more parents and AND relationship).

In the bottom row plots, non-observed genotypes are shown in light green, to differentiate them from the observed genotypes (shown in orange).

Note

The color and design of figures in the bottom row, depicting transition matrices, transition rate matrices, and observed genotype transitions are heavily inspired by (a blatant copy of) some of the representations in Greenbury et al., 2020.

It is easy to get the plots to display poorly (overlapping names in nodes, overlapping labels between plots, etc) if you use long gene names. For best results, try to use short gene names.

The criteria to decide which are the most relevant paths with the top_paths option is the following: 1) Get all the leaves from the graph. 2) Calculate all the paths leading from "WT" to all leaves. 3) Select n paths with highest cumulative weighted sum. The weights used depend on the type of plot (`plot_type`): for trans_mat it will be log probabilities (so the most relevant paths are the most likely paths), for trans_rate_mat it will be rates, and for obs_genotype_transitions it will be raw counts.

Plots can be more readable with a combination of top_paths and label_type. If `label_type = 'acquisition'` node labels will dissapear and edge label will be shown instead. They will display the information of the last gene mutated.

plot_CPMs has been deprecated. Use plot_evam.

References

Greenbury, S. F., Barahona, M., & Johnston, I. G. (2020). HyperTraPS: Inferring Probabilistic Patterns of Trait Acquisition in Evolutionary and Disease Progression Pathways. Cell Systems, 10(1), 39–51–10. http://dx.doi.org/10.1016/j.cels.2019.10.009

Examples

```
dB_c1 <- matrix(
 c(
     rep(c(1, 0, 0, 0, 0), 30) #A
   , rep(c(0, 0, 1, 0, 0), 30) #C
   , rep(c(1, 1, 0, 0, 0), 20) #AB
   , rep(c(0, 0, 1, 1, 0), 20) #CD
   , rep(c(1, 1, 1, 0, 0), 10) #ABC
   , rep(c(1, 0, 1, 1, 0), 10) #ACD
   , rep(c(1, 1, 0, 0, 1), 10) #ABE
   , rep(c(0, 0, 1, 1, 1), 10) #CDE
   , rep(c(1, 1, 1, 0, 1), 10) #ABCE
   , rep(c(1, 0, 1, 1, 1), 10) #ACDE
```

```
    , rep(c(1, 1, 1, 1, 0), 5) # ABCD
    , rep(c(0, 0, 0, 0, 0), 1) # WT
 ), ncol = 5, byrow = TRUE
)
colnames(dB_c1) <- LETTERS[1:5]

## Use MCCBN only if installed
MCCBN_INSTALLED <- requireNamespace("mccbn", quietly = TRUE)
methods <- c("CBN", "OT", "OncoBN", "MHN", "HESBCN")
if (MCCBN_INSTALLED) {
    methods <- c(methods, "MCCBN")
}

out <- evam(dB_c1,
            methods = methods)

plot_evam(out, plot_type = "trans_mat")

plot_evam(out, plot_type = "trans_rate_mat")

plot_evam(out, plot_type = "trans_rate_mat", top_paths=2)
plot_evam(out, plot_type = "trans_rate_mat", top_paths=2
  , label_type ="acquisition")

out_samp <- sample_evam(out, 1000, output = c("sampled_genotype_counts", "obs_genotype_transitions"))

plot_evam(out, out_samp, plot_type = "obs_genotype_transitions")

## Only showing new gene mutated respect with its parent
plot_evam(out, out_samp, plot_type = "obs_genotype_transitions",
  label_type = "acquisition")

plot_evam(out, out_samp, plot_type = "obs_genotype_transitions",
  label_type = "acquisition", top_paths = 3)


## Examples with mixed AND and OR and AND and XOR for HESBCN
data("ex_mixed_and_or_xor")

out_AND_OR_XOR <- evam(ex_mixed_and_or_xor,
                       methods = c("OT", "HESBCN", "MHN", "OncoBN"),
                       hesbcn_opts = list(seed = 26))

plot_evam(out_AND_OR_XOR,plot_type = "trans_mat",
          top_paths = 3)


## Asking for a method not in the output will give a warning
plot_evam(out_AND_OR_XOR, plot_type = "trans_mat",
          methods = c("OT", "OncoBN"),
          top_paths = 4)


## Only two methods, but one not fitted
plot_evam(out_AND_OR_XOR, methods = c("CBN", "HESBCN"),
           plot_type = "trans_mat")
```

```
## Only one method
plot_evam(out_AND_OR_XOR, methods = c("MHN"),
          plot_type = "trans_mat",
          top_paths = 5)

plot_evam(out_AND_OR_XOR, plot_type = "trans_mat", top_paths = 3)


plot_evam(out_AND_OR_XOR, methods = c("MHN", "HESBCN"),
          plot_type = "trans_mat", label_type="acquisition"
          , top_paths=3)

plot_evam(out_AND_OR_XOR, methods = c("MHN", "HESBCN"),
          plot_type = "trans_mat", label_type="genotype"
          , top_paths=3)
```

`random_evam` *Generate a random EvAM model.*

Description

Generate random EvAM (CPM) models.

Usage

```
random_evam(ngenes = NULL,
                       gene_names = NULL,
                       model = c("OT", "CBN", "HESBCN",
                                 "MHN", "OncoBN"),
                       graph_density = 0.35,
                       cbn_hesbcn_lambda_min = 1/3,
                       cbn_hesbcn_lambda_max = 3,
                       hesbcn_probs = c("AND" = 1/3,
                                        "OR" = 1/3,
                                        "XOR" = 1/3),
                       ot_oncobn_weight_min = 0,
                       ot_oncobn_weight_max = 1,
                       ot_oncobn_epos = 0.1,
                       oncobn_model = "DBN"
                       )
```

Arguments

`ngenes`	Number of genes in the model. Specify this or `gene_names`.
`gene_names`	Gene names.Specify this or `ngenes`.
`model`	One of OT, CBN, OncoBN, MHN, HESBCN.
`graph_density`	Expected number of non-entries in the adjacency matrix (all methods expect MHN) or the theta matrix (MHN). See details.
`cbn_hesbcn_lambda_min`	Smallest value of lambda for CBN and HESBCN.
`cbn_hesbcn_lambda_max`	Largest value of lambabda for CBN and HESBCN.

`hesbcn_probs`	For nodes with more than one ancestor, the probability that the dependencies are AND, OR, XOR. You must pass a named vector.
`ot_oncobn_weight_min`	Smallest possible value of the weight or theta for OT and OncoBN respectively.
`ot_oncobn_weight_max`	Largest possible value of the weight or theta for OT and OncoBN respectively.= 1,
`ot_oncobn_epos`	epsilon (OncoBN) or epos (OT) error.
`oncobn_model`	One of "DBN" or "CBN".

Details

The purpose of this function is to allow easy simulation of data under a specific model. Details follow for specific models, with explanation of parameters.

For MHN we use the same procedure as available in the original code of Schill et al. `graph_density` is `1 - sparsity`. A matrix of random thetas (from a normal 0, 1, distribution) is generated, where the number of non-zero entries is controlled by `graph_density`.

For CBN a random poset is generated by calling `random_poset` in the `mccbn` package (function exported but not documented) and generating random lambdas uniformly distributed between `cbn_hesbcn_lambda_min` and `cbn_hesbcn_lambda_max`. No specific provision is made for randomly generating from MCCBN, as the way to simulate is similar to CBN (but see also the additional documentation for details about the error models).

For H-ESBCN we follow a similar procedure as for CBN, but nodes that have two or more parents are then assigned, at random, a relationship that can be AND, OR, or XOR, as given by `hesbcn_probs`.

For OT we use a procedure similar to the one for CBN, but edge weights are uniformly distributed between `ot_oncobn_weight_min` and `ot_oncobn_weight_max`. Since under OT a node can have only one parent, for all nodes that have two or more parents, we randomly keep one of the parents. Thus, `graph_density` is often larger than the actual number of non-zero connections in the adjacency matrix.

For OncoBN we do as for CBN. If you specify `oncobn_model` to be DBN, all dependencies on two or more parents are OR dependencies; if you specify CBN, dependencies are AND dependencies. As for OT, thetas are uniformly distributed between `ot_oncobn_weight_min` and `ot_oncobn_weight_max`.

For both OT and OncoBN model, `ot_oncobn_epos` controls the probability that a gene can mutate if its requirements are not satisfied. (This is thus intrinsic to the model, and independent of observation error; see next).

In all cases, the predicted distribution of genotypes for a model is done assuming perfect compliance with the model. See the additional documentation for details about the error models. Adding observation error can be done using `obs_error > 0` when calling `sample_evam`.

Value

Random model, with the same structure as returned by function `evam`. Thus, a named list with all the returned entries from `evam` for a given method.

See Also

`sample_evam`

Examples

```
rmhn <- random_evam(model = "MHN", ngenes = 5)
rcbn <- random_evam(model = "CBN", ngenes = 5,
                             graph_density = 0.5)

## Now, obtain some data
## You can obtain a random sample, with counts of frequencies of
## genotypes and add observation noise
sample_mhn <- sample_evam(rmhn, N = 1000, obs_noise = 0.05)

## The component sampled_genotype_counts_as_data is
## a matrix that you can then pass to evam as the
## input data argument (x)
```

`runShiny` *Run the web application of evamtools*

Description

Launch the server with the web based app.

Usage

```
runShiny(host="0.0.0.0", port=3000, test.mode = FALSE)
```

Arguments

`host`	Host where the app will be listening
`port`	Port where the app will be listening
`test.mode`	See `runApp`

`sample_evam` *Obtain samples of genotypes from the EvAM (CPM) models and, optionally, counts of genotype transitions.*

Description

Obtain samples of genotypes from the CPM models and, optionally, counts of genotype transitions.

For OT and OncoBN we always obtain the absolute genotype frequencies by drawing samples of size `N`, with replacement, using as probabilities the predicted genotype frequencies.

For the remaining methods, that is also what we do, unless you request also obs_genotype_transitions and state_counts. In this case, since we need to simulate sampling from the continuous-time Markov Chain (with transition rates given by the transition rate matrix) to obtain state counts and observed genotype transitions, we use this same sampling to obtain the absolute genotype frequencies. (The results are, of course, equivalent, but sampling directly from the predicted frequencies is much faster). Note that the option to request obs_genotype_transitions was removed from the web app, as it was rarely used, but lead to confusion and could increase without good reason running times. So, from the web app, we sample without using obs_genotype_transitions.

Observed genotype transitions, if requested, are obtained by counting the transitions between pairs of genotypes when simulating from the continuous-time Markov Chain. State counts are also obtained by counting from this process how many times a genotype was visited.

Usage

```
sample_evam(cpm_output,  N,
                methods = NULL,
                output = c("sampled_genotype_counts"),
                obs_noise = 0,
                genotype_freqs_as_data = TRUE
             )

sample_CPMs(cpm_output,  N,
                methods = NULL,
                output = c("sampled_genotype_counts"),
                obs_noise = 0,
                genotype_freqs_as_data = TRUE
             )
```

Arguments

`cpm_output`	Output from calling all_methods2trans_mat
`N`	Number of samples to generate
`methods`	Vector of strings with the names of methods that we want to sample. If NULL, all methods with output in cpm_output. The list of available methods is OT, OncoBN, CBN, MCCBN, MHN, HESBCN.
`output`	A vector with one or more of the following possible outputs: sampled_genotype_counts, obs_genotype_transitions, state_counts. Even if requested, obs_genotype_transitions and state_counts are not available for OT and OncoBN.
`obs_noise`	When obtaining a sample, should we add observation noise (for example, genotyping error) to the data? If larger than 0, this `obs_noise` proportion of entries in the sampled matrix will be flipped (i.e., 0s turned to 1s and 1s turned to 0s).
`genotype_freqs_as_data`	If TRUE, return a matrix where each row is a "sampled genotype", where 0 denotes no alteration and 1 alteration in the gene of the corresponding column.

Value

A list, with a many entries as methods times number of components requested. For each method among CBN, MCCBN, HESBCN, and MHN:

- sampled_genotype_counts: Counts, or absolute genotype frequencies, obtained by sampling from the predicted frequencies. See also Description, below.
- obs_genotype_transitions: Number of observed transitions between genotypes (as a sparse matrix).
- state_counts: Number of times each genotype was visited during the transitions. Column sums of observed genotype transitions are equal to state counts.

- sampled_genotype_counts_as_data: The genotypes in a matrix of 0/1. This can directly be passed as an argument for `evam`, as the input data.

Observed genotype transitions are not the way to obtain estimates of transition probabilities. The transition probabilities given by each method are already available from the output of evam itself. These genotype transitions are the observed transitions during the simulation of the sampling process and, thus, have additional noise.

For OT and OncoBN, only the `sampled_genotype_counts` and `sampled_genotype_counts_as_data` components are available (the other two are not available).

Note

sample_CPMs has been deprecated. Use sample_evam.

See Also

`random_evam`

Examples

```
data(every_which_way_data)
Dat1 <- every_which_way_data[[16]][1:40, 2:6]
## For faster execution, use only some methods
out <- suppressMessages(evam(Dat1,
                             methods = c("CBN", "OT", "OncoBN",
                                         "MHN")))

## Sample from the predicted genotype frequencies
## only for OT
outS1_ot <- sample_evam(out, N = 1000, methods = "OT")

## Sample from the predicted genotype frequencies
## for OT and HESBCN. But the later was not in the output
## so we get a warning-
outS1_ot_2 <- sample_evam(out, N = 1000, methods = c("OT", "HESBCN"))


## Sample from the predicted genotype frequencies
## for all methods in the output out

outS1 <- sample_evam(out, N = 1000)

## Same, but adding observation error
outS1e <- sample_evam(out, N = 1000, obs_noise = 0.1)

## Only CBN and will simulate sampling from the transition
## rate matrix.

outS2 <- sample_evam(out, N = 1000, methods = "CBN",
                     output = "obs_genotype_transitions")


## No output available for OT
## For CBN and MHN simulate from the transition rate matrix

outS3 <- sample_evam(out, N = 1000, methods = c("CBN", "OT", "MHN"),
                     output = c("obs_genotype_transitions",
                                "state_counts"))


## OT sampled from the predicted genotype frequencies
## No obs_genotype_transitions available for OT
## CBN and OT simulate from the transition rate matrix, for consistency

outS4 <- sample_evam(out, N = 1000, methods = c("CBN", "OT", "MHN"),
                     output = c("obs_genotype_transitions",
                                "sampled_genotype_counts"))

## Only CBN,  will simulate sampling from the transition
## rate matrix and add observation error to the genotype frequencies.

outS5 <- sample_evam(out, N = 1000, methods = "CBN",
              output = c("obs_genotype_transitions", "sampled_genotype_counts"), obs_noise = 0.1)
```

SHINY_DEFAULTS *Defaults options for running the shiny web app*

Description

Defaults of the web app. This file was generated by running the script /inst/shiny-examples/evamtools/DEFAULTS.R You will want to rerun it (so that the RData file is created again) whenever you make changes to it.

The object is called `.ev_SHINY_dflt` to minimize the risk of overwriting.

Usage

```
data(SHINY_DEFAULTS)
```

Format

Defaults values of the shiny app

max_genes Maximun number of genes allowed

min_genes Minimum number of genes allowed

ngenes Integer of default number of genes to use when building

cpm_samples Number of patients to samples to generate csd data from a matrix using with CPM outputs

all_cpms All CPMs in evamtools

csd_samples Number of patients to samples to generate csd data from a matrix or a dag

template_data One of:

- csd_counts:Data frame with the counts of each genotype
- data:Data frame with cross sectional data.
- dag:Matrix of 10x10 with lambdas
- dag_parent_sest:List of 10 elements with "Single"
- lambdas:Vector of 10 lambdas, equals to 1
- thetas:Matrix of 10x10 with thetas
- gene_names:List with gene names
- name:String with the data set name

Index