EvAM-Tools: all additional documentation

Ramon Diaz-Uriarte[1,2,†] and Pablo Herrera Nieto[1,2]

[1]Dpt. of Biochemistry, School of Medicine, Universidad Autónoma de Madrid, Madrid, Spain
[2]Instituto de Investigaciones Biomédicas 'Alberto Sols' (UAM-CSIC), Madrid, Spain
[†]To whom correspondence should be addressed: r.diaz@uam.es

2022-04-27
Version 1e74fcd

This PDF is the concatenation of three different files:

1. **EvAM-Tools: additional documentation:** Additional documentation about EvAM-Tools. These include technical details (error models, predicting genotype frequencies, generation of random evam models), additional usage information for the Shiny web GUI, and a FAQ.

2. **Package evamtools help:** The help files for the evamtools R package, collated as a single pdf.

3. **Using OncoSimulR to get accessible genotypes and transition matrices:** How we can use OncoSimulR to get accessible genotypes and transition matrices for CBN (and MCCBN), OT, HESBCN, and OncoBN; this provides additional testing and makes the fitness models explicit.

EvAM-Tools: additional documentation

Ramon Diaz-Uriarte[1,2,†]

[1]Dpt. of Biochemistry, School of Medicine, Universidad Autónoma de Madrid, Madrid, Spain
[2]Instituto de Investigaciones Biomédicas 'Alberto Sols' (UAM-CSIC), Madrid, Spain
[†]To whom correspondence should be addressed: r.diaz@uam.es

2022-04-27
Version 1e74fcd

Contents

1 Introduction **1**

2 Predicted genotype frequencies **1**
2.1 Predicted genotype frequencies for CBN, MCCBN, MHN, H-ESBCN 1
2.2 Predicted genotype frequencies for OT and OncoBN 1

3 Probabilities of evolutionary paths and transition probabilities **2**

4 Generating random CPM/EvAM models, sampling from them, and error models **2**
4.1 Generating random CPM/EvAM models and sampling from them 2
4.2 Error models 3
4.3 Error models and simulating and sampling from OT 4

5 Random EvAM models and transitive reduction **5**

6 H-ESBCN: details and examples of using λs and computing transition rate matrices and predicted genotype frequencies **5**
6.1 Lambdas from the output: "Best Lambdas" and "lambdas_matrix" 6
6.2 Interpreting OR and XOR (and AND) 6
6.3 Predicted genotype frequencies 6
6.4 An example with OR and XOR 7
6.5 Three examples from actual analysis 9
6.6 Combining AND, OR, XOR? 11

7 Shiny EvAM-Tools web app **12**
7.1 Building the models and example data 12
7.1.1 Example data/DAGs/MHNs 12
7.2 Cross-sectional data input 12
7.2.1 Number of genes and genes without mutations 12
7.3 Naming and saving data and model 12
7.4 Tabular output from the Shiny app 12

8 FAQ **13**
8.0.1 Why haven't you used method X? 13
8.1 With OncoBN sometimes I obtain DAGs that are not transitively reduced . . . 13
8.2 In the DAG figures, why do nodes with two or more incoming edges have only a single annotated edge with a number? 13
8.3 Do sampled genotype frequencies and counts contain observation noise? And predicted genotype frequencies? 14
8.4 I want to setup my own Shiny server with different default "Advanced options" 14
8.5 How can I use the Shiny app in a local intranet with load balancing using multiple Docker instances 14
8.6 If I use the Docker image for the package (rdiaz02/evamrstudio), can I run the Shiny app? 14
8.7 Why aren't you using Shiny Server? 14
8.8 I want to create my own Docker images 15
8.8.1 What if creating the image fails because of no internet connection from the container 15
8.8.2 Cleaning the build cache and stale old images 15
8.8.3 Copying docker images from one machine to another 15

9 License and copyright **16**

1 Introduction

This document provides additional documentation about EvAM-Tools; some of the sections complement the Shiny web GUI application help, others apply to the package, or to both. You can run the web app from https://iib.uam.es/evamtools/ or download a Docker image from https://hub.docker.com/r/rdiaz02/evamshiny; to run the R package download a Docker image from https://hub.docker.com/r/rdiaz02/evamrstudio.

2 Predicted genotype frequencies

2.1 Predicted genotype frequencies for CBN, MCCBN, MHN, H-ESBCN

For CBN, MCCBN, MHN, and H-ESBCN, the transition rate matrix describes the true process that generates genotypes and this matrix can be obtained from the parameters of the model (θs for MHN, λs for the rest). Therefore, we can use the transition rate matrix to calculate the predicted probabilities of the different genotypes using standard results from continuous-time Markov Chains. In all cases here, we assume that the time of observation is exponentially distributed with rate 1 (as in Gerstung *et al.*, 2009 or Schill *et al.*, 2020)[1].

Obtaining the transition rate matrix from the model output is detailed in Montazeri *et al.* (2016) for CBN and Schill *et al.* (2020) for MHN; for H-ESBCN see section *6, "H-ESBCN: details and examples of using λs and computing transition rate matrices and predicted genotype frequencies"*.

Once we have obtained the transition rate matrix, the fastest way to obtain the predicted genotype probabilities is using equation 4 in Schill *et al.* (2020); this is implemented in the non-exported function `probs_from_trm`, and follows also what is done in the original `Generate.pTh` from Schill *et al.* (2020). `probs_from_trm` is called from function `evam`.

Instead of using that expression, we can sample from the continuous-time Markov Chain using standard procedures (e.g., ch. 5 in Wilkinson, 2019 or Algorithm 1 in Gotovos *et al.*, 2021). Sampling is what we do when you call `sample_CPMs` asking for `obs_genotype_transitions` or `state_counts` to be returned (and this sampling is implemented in the non-exported function `population_sample_from_trm`, and called, as needed, by `sample_CPMs`).

2.2 Predicted genotype frequencies for OT and OncoBN

OT and OncoBN do not return rates of a continuous-time Markov chain, but probabilities of seeing specific alterations at the time of observation. Predicted probabilities of genotypes for OT and OncoBN are obtained using the weights (OT) or θs (OncoBN), according to the expression for the probability of observing a genotype. For example, see section 2.2 in Szabo and Boucher (2008) for OT and Figure 1 and section 2.1 in Nicol *et al.* (2021) for OncoBN. For OT we can use function `distributiion.oncotree` in package `Oncotree` and for OncoBN function `Lik.genotype` from package `OncoBN`[2].

In OT observation error is already part of the model (in contrast to what happened in CBN, MHN, H-ESBCN). This is explained in section *4.2, "Error models"*.

For all methods, once we have the predicted probabilities, we can obtain a finite sample and, if we want, add observational (or genotyping) noise; see section *4.2, "Error models"*.

[1] There is code in `evamtools`, in function `population_sample_from_trm`, to obtain samples at arbitrary collections of times —i.e., not limited to times exponentially distributed with rate 1.

[2] Though for OncoBN we do not use `Lik.genotype` directly, as that would involve making the exact same repeated set of calls for every individual; see the non-exported function `DBN_prob_genotypes` in file `onco-bn-process.R`

3 Probabilities of evolutionary paths and transition probabilities

How to obtain probabilities of evolutionary paths is detailed in Hosseini *et al.* (2019) and Diaz-Uriarte and Vasallo (2019) (see S4_Text: https://doi.org/10.1371/journal.pcbi.1007246.s006, section 3). For transition probabilities see also Diaz-Colunga and Diaz-Uriarte (2021) (specifically section 1 in S1 Appendix: https://doi.org/10.1371/journal.pcbi.1009055.s001).

4 Generating random CPM/EvAM models, sampling from them, and error models

4.1 Generating random CPM/EvAM models and sampling from them

We often want to generate data under the model of a CPM. Common use cases are:

- Understand what different models imply about how the cross-sectional data looks like.
- Examine how well a method can recover the true structure when the data fulfills the assumptions of a method. For instance, we would generate data under a particular model and see if the method that implements that model can recover the true structure under different sample sizes.
- Examine how a given method works, and what type of inferences it performs, when data are generated under the model of another method. For example, what is the output from MHN if the data are really coming from an H-ESBCN model?

Addressing the above needs involves:

1. Generating a random model.
2. Obtaining the predicted genotype frequencies from that model (see *"Predicted genotype frequencies", section 2*).
3. Obtaining a finite sample from the predicted frequencies of that model.
4. Using the data to answer whichever questions we had; for example, analyze the sampled data with another or the same method, plot the genotype frequencies, etc.

We explain each one in turn below, with reference to `evamtools` functions and arguments.

1. Generating a random model.

   Function `generate_random_evam` generates random models for OT, OncoBN, CBN, MHN, OncoBN, and H-ESBCN. Details about the arguments of the function are provided in its help page. No specific provision is made for randomly generating from MCCBN, as the way to simulate is similar to CBN (generate a random poset and a random set of lambdas).

2. Obtaining the predicted genotype frequencies from that model.

   These are returned as part of the output of `generate_random_evam` (as well as part of the output of `evam`). In all cases, the predicted distribution of genotypes for a model is done assuming perfect compliance with the model; we explain the general process in *"Predicted genotype frequencies" (section 2)* and see also further details below.

3. Obtaining a finite sample from the predicted frequencies of that model.

   As the output from `generate_random_evam` is the same (except for the data components) to that from `evam` we can pass the model to function `sample_CPMs`.

   When obtaining a finite sample, we can add sampling noise to the data. For example, noise due to genotyping errors; the probability of errors is controlled by argument `obs_noise` in the call to `sample_CPMs`.

   In more detail, the process involves:

   (a) Obtaining a finite sample without errors from the predicted genotype frequencies.

   (b) If requested (i.e., if `obs_noise > 0`), flipping a fraction `obs_noise` of the observations (i.e., turning 1s to 0s and 0s to 1s).

4. Using the data to answer whichever questions we had; for example, analyze the sampled data with another or the same method, plot the genotype frequencies, etc.

   To make this simpler, function `sample_CPMs` can return the finite sample (with or without observation noise) as a typical cross-sectional data set: a matrix where each row is a "sampled genotype", in which 0 denotes no alteration and 1 alteration in the gene of the corresponding column. This data matrix can be used directly as input for CPM methods, for instance as argument `x` (the cross-sectional data) to function `evam`.

4.2 Error models

We explained above that when obtaining the predicted frequencies under a model we assume perfect compliance with the model. This is straightforward with most methods, but not with others. This summarizes the error models for each method:

CBN In H-CBN the λs describe the true underlying model that produces the true, hidden genotypes, but the observed genotypes might differ from the true ones because of observation error, for instance genotyping error (Gerstung *et al.*, 2009, p. 2810).

H-ESBCN As for CBN (Angaroni *et al.*, 2021, p. 756).

MHN As for CBN (e.g., description of the simulation process in p. 244 of Schill *et al.*, 2020).

MC-CBN With MC-CBN the model is a mixture between the CBN model and a noise component model (Montazeri *et al.*, 2016, p. i730-i731). The simulations in https://github.com/cbg-ethz/MC-CBN, however, use a procedure where observations are generated from an underlying poset with a given set of lambdas, and symmetric error is then added (see the functions `mccbn:::random_poset` and `mccbn:::random_posets`).

OncoBN The model includes a DBN (disjunctive) or CBN (conjunctive) model, as given by a DAG and a set of θs, and a "spontaneous activation model" (Nicol *et al.*, 2021, p. 3-4). The "spontaneous activation model", with parameter ϵ, represents deviations from the model and allows child mutations to appear even if the parents in the DAG have not been mutated (i.e., even if the restrictions encoded in the DAG are not satisfied).

OT There are two sources of deviations from the OT model: a) those that result from observational (or genotyping) errors, that can lead to both false positive and false negative observational errors; b) events occurring that do not respect the OT model Szabo and Boucher (2002). The second would be the same as the "spontaneous activation" in OncoBN.

The `oncotree.fit` function in the `Oncotree` package returns a `eps` component with the estimated false positive, `epos`, and false negative, `eneg`, error rates. But these are the result of combining the two sources of error Szabo and Boucher (2008): observation errors and true deviations from the model. So observation error is reflected in both `eneg` and `epos`, whereas true deviations from the model are only reflected in `epos`. In other words, the false negatives, as measured by the estimated `eneg`, are due purely to observation error. But the `epos` are not equivalent to the ϵ of OncoBN: `epos` includes both observation error (false positives) and true mutations that occur without respecting the restrictions of the OT DAG (tree).

4.3 Error models and simulating and sampling from OT

Given the above, when sampling from a model, in all cases except OT, we have always followed the same procedure: we have first generated observations under the true model and, if requested, then added noise. For OT, since `epos` reflects both observation and true deviations from the model, this is not possible. Thus, for OT there is a difference when we sample from a random model and when we sample from the predictions of a fitted model.

The fitted model for OT, when fitting a true data set, includes both `epos` and `eneg`. Predictions are obtained using function `Oncotree::distribution.oncotree` with, by default, argument `with.errors = TRUE`, which is what argument `with_errors_dist_ot = TRUE` to `evam` does. Therefore, from a fitted model, the predictions incorporate both false positive and false negative error rates, as estimated by `oncotree.fit`; as explained above, however, these estimated error rates are both the errors from the observational process (genotyping errors, for example) and true deviations from the model. When you later call `sample_CPMs` you can add an `obs_noise` with value larger than 0, but for OT, when sampling from the model fitted to observed data this might not make sense (since `epos` and `eneg` have been used already to produce the predicted genotype frequencies). Thus, if we use `with_errors_dist_ot = TRUE` in the `evam` call and then set `obs_noise = 0` when calling `sample_CPMs`, the observed data we generate should be the same (have the same distribution) as if we had used `Oncotree::generate.data` with `method = ''D1''`, `with.errors = TRUE` and `edge.weights = ''estimated''`.[3]

When sampling from a random model, using `generate_random_evam`, we generate the tree (the DAG) with density as given by argument `graph_density` and the weights from uniform distributions with limits given by `ot_oncobn_weight_min` and `ot_oncobn_weight_max`. In addition, we can set a value larger than 0 for `ot_oncobn_epos`. This will be used as the `epos`, but not `eneg`, value of the OT model. When you sample and optionally add noise, with argument `obs_noise` to function `sample_CPMs` noise is added symmetrically. Thus, we use a procedure where `ot_oncobn_epos` behaves as OncoBN's ϵ and `obs_noise` is purely symmetric observational error.

Why this difference? When you use a model fitted to real data, it is sensible to use `Oncotree`'s inferential machinery to estimate the `epos` and `eneg`. If you later want to generate samples, these already include deviations from the model and noise. However, when you simulate a model, there is no data and thus no way to estimate `epos` and `eneg`. Therefore,

[3] We can try to divide the `epos` component in a component like OncoBN's ϵ and another noise component. Then, we would first obtain the predicted distribution via `distribution.oncotree` with the ϵ-like (and with `eneg` $= 0$), sample, and add noise. If `epos` $>$ `eneg` we can do this so that the noise added is symmetrical. This is shown in function `dot_noise_gd_3` in `inst/miscell/OT_generate_data_sample_CPMs.R`.

In that same file `inst/miscell/OT_generate_data_sample_CPMs.R` we show that a model obtained from `generate_random_evam` with `ot_oncobn_eps` $= x$ and sampled using `sample_CPMs` with `obs_noise` $= y$ gives predictions with the same distribution as if we had used `Oncotree::generate.data` on that very same oncotree object but with `epos` $= x + y - (1/2)\ x\ y$ and `eneg` $= y$.

it is sensible to split errors into two distinct pieces, which is also coherent with what we do with the rest of the methods: deviations from the model, and noise.

5 Random EvAM models and transitive reduction

As of now, the generation of random EvAM models uses transitively reduced graphs (we call `mccbn::random_poset` with argument `trans_reduced = TRUE`). This does not decrease the number of models that can be expressed when using CBN. However, it can limit the range of models when we can mix AND, OR, XOR in the same model. The following examples illustrate how this makes certain models impossible.

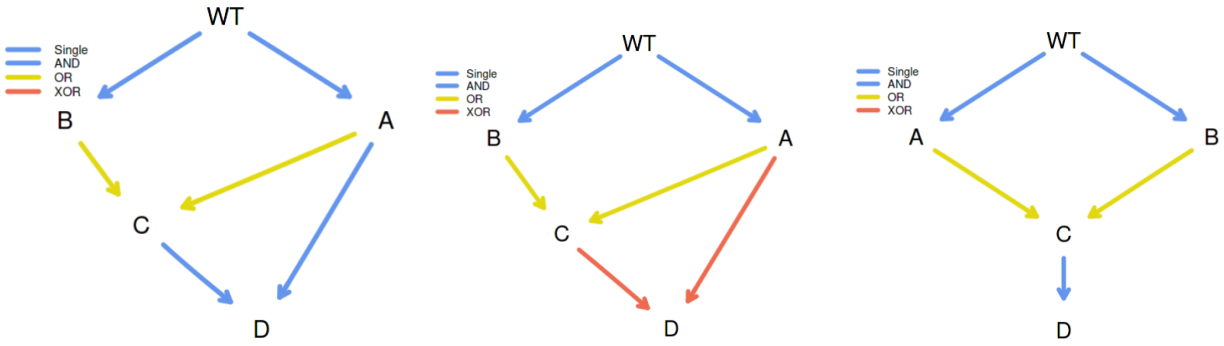

Figure 1: Non-transitively reduced DAG, OR and AND (left), non-transitively reduced DAG, OR and XOR (center), transitively reduced DAG.

- Under the left-most DAG in Fig. 1 we cannot observe genotype BCD.
- Under the center DAG in Fig. 1 we cannot observe genotype ACD.
- Under the right-most DAG, which is the transitive reduction of the above two graphs, we can observe both BCD and ACD.

Can we imagine biological scenarios where the left-most or center scenarios in Fig. 1 would apply? Yes. We don't recall seeing them in the literature, though. If this is deemed relevant, it is just a matter of changing `trans_reduced = TRUE` when we are simulating HESBCN models inside function `random_evam`.

You can of course construct the non-transitively reduced graphs "by hand" (creating the data frame with the appropriate structure) or, much simpler, using the Shiny web app.

6 H-ESBCN: details and examples of using λs and computing transition rate matrices and predicted genotype frequencies

Here I provide full details about how we interpret and use the results from the method described in Angaroni *et al.* (2021). I do this here because, in contrast to CBN or MHN, there is no existing previous code or examples that do this, and we found some potentially confusing issues. I have made a specific section so as not to break the flow of the former sections.

6.1 Lambdas from the output: "Best Lambdas" and "lambdas_matrix"

The output returned by the H-ESBCN C code contains a "Best Lambdas" vector. The output returned by function `import.hesbcn` (that we have included in the code, in file `HESBCN__import.hesbcn.R`) has an object called "lambdas_matrix" where each of the lambdas for a gene is divided by the number of parents. This can be checked in any of the examples in the PMCE repository. Code that shows three examples, with XOR, OR, AND is available under "inst/miscell/examples/HESBCN-lambdas-from-examples.R".

It is the output from "Best lambdas" (i.e., the undivided lambdas) that are "[the] rates of the Poisson processes of the continuous-time HMM, associated with the vertices of the model, which allow one to estimate the expected waiting time of a node, given that its predecessor has occurred." (p. 756). (What is the division? An operation that modifies an internal data structure, and just a temporary operation, done merely for implementation purposes. In line 95 of the code —as of current version, in https://github.com/BIMIB-DISCo/PMCE/blob/main/Utilities/R/utils.R— the divided lambdas are again summed, so the partition disappears: "curr_in_lambda = sum(hesbcn$lambdas_matrix[,curr_node])", and it is that value that is used in further downstream computations; email with the authors on 2021-07-09). The "Best lambdas" are returned by our modified `import.hesbcn` function.

6.2 Interpreting OR and XOR (and AND)

I find Figure 1C of Angaroni *et al.* (2021) possibly confusing. First, the non-confusing part: node "D" has a rate when exactly one of B XOR C has occured, and node "G" some other rate when E or F or both E and F have occurred. (Note: the figure shows τs, not λs. The comments here refer to the λs).

Now the (for me, at least) possibly confusing part: it seems that the node called "B xor C" is such that B and C have the same rates of dependencies on A; in other words, it would seem to imply that $\lambda_B = \lambda_C$. Similarly, the node called "E or F" seems to indicate that both E and F have the same rate, so $\lambda_E = \lambda_F$. But this need not be so. In fact, virtually all of the examples we have looked at, and the examples in their output, do not satisfy that the rates to the genes that are part of a XOR, OR, or AND relationship are the same. For instance, in the example above of Bladder Urothelial Carcinoma (see "inst/miscell/examples/HESBCN-lambdas-from-examples.R"), KMT2D depends on KMT2C and TP53, but the rate for KMT2C, $\lambda_{KMT2C} = 0.1991$ and that for TP53, $\lambda_{TP53} = 0.8062$.

Remember that the λ for a gene is the rate of the process until that mutation appears and is fixated, given all the dependencies of that gene are satisfied (which is, of course, the same interpretation as under CBN). Again: "[the] rates of the Poisson processes of the continuous-time HMM, associated with the vertices of the model, which allow one to estimate the expected waiting time of a node, given that its predecessor has occurred." (p. 756).

But the rate at which the parents are satisfied can differ (as it was the case for CBN). A difference with respect to CBN is that, with CBN, if a gene D depends on three genes A, B, C, regardless of the lambdas of each of A, B, C, D can only happen once all of A, B, C are present. With H-ESBCN and with OR and XOR relationships this is no longer the case: one can see D with only A, for example.

What if some genes depend with and AND, others with a XOR and other with a OR? Just apply the rules to each type of dependency: in the HESBCN model if a gene depends on a set of genes, it has the same type of dependency on all the genes of that set.

6.3 Predicted genotype frequencies

Once we have the transition rate matrix, obtaining the predicted genotype frequencies uses the same procedure as for CBN and MHN; see section *2.1, "Predicted genotype frequencies*

for CBN, MCCBN, MHN, H-ESBCN".

6.4 An example with OR and XOR

In this example:

- A, B, C depend on none.
- D depends, with an OR, on both A and B
- E depends, with an XOR, on B and C
- Transition rate matrix is shown below: rows are origin, column destination. λs are those from "Best Lambdas".

	WT	A	B	C	AB	AC	AD	BC	BD	BE	CE	ABC	ABD	ABE	ACD	ACE	BCD	BDE	ABCD	ABDE	ACDE
WT		λ_A	λ_B	λ_C																	
A					λ_B	λ_C	λ_D														
B					λ_A			λ_C	λ_D	λ_E											
C						λ_A		λ_B			λ_E										
AB												λ_C	λ_D	λ_E							
AC												λ_B			λ_D	λ_E					
AD													λ_B		λ_C						
BC												λ_A					λ_D				
BD													λ_A				λ_C	λ_E			
BE														λ_A				λ_D			
CE																λ_A					
ABC																			λ_D		
ABD																			λ_C		
ABE																				λ_D	
ACD																			λ_B		λ_E
ACE																					λ_E
BCD																			λ_A		
BDE																				λ_A	
ABCD																					
ABDE																					
ACDE																					

6.5 Three examples from actual analysis

- The code in inst/miscell/HESBCN-OR-XOR-AND-lambda-and-rates.R contains examples of how we use those lambdas (the `n`, number of steps, used is ridiculously small, and just for the sake of speed).

1. OR

    - Suppose output such as this (again, see file inst/miscell/HESBCN-OR-XOR-AND-lambda-and-rates.R for how to reproduce it).

```
$adjacency_matrix
     Root A B C D
Root    0 1 1 0 0
A       0 0 0 1 1
B       0 0 0 1 1
C       0 0 0 0 0
D       0 0 0 0 0

$lambdas_matrix
     Root     A     B     C      D
Root    0 8.083 2.585 0.000 0.0000
A       0 0.000 0.000 8.914 0.2062
B       0 0.000 0.000 8.914 0.2062
C       0 0.000 0.000 0.000 0.0000
D       0 0.000 0.000 0.000 0.0000


$parent_set
       A        B        C        D
"Single" "Single"    "XOR"     "OR"


$lambdas
[1]  8.0833  2.5854 17.8277  0.4124

$edges
  From To      Edge Lambdas Relation
1 Root  A Root -> A  8.0833   Single
2 Root  B Root -> B  2.5854   Single
3    A  C    A -> C 17.8277      XOR
4    B  C    B -> C 17.8277      XOR
5    A  D    A -> D  0.4124       OR
6    B  D    B -> D  0.4124       OR
```

    - From the above output, these are the lambdas: $\lambda_A = 8.0833, \lambda_B = 2.5854, \lambda_C = 17.8277, \lambda_D = 0.4124$.
    - Focusing only on A, B, D, to see gene D we can follow four paths.
        - The first two involve only two mutations:
            * $WT \to A \to AD$

    * $WT \rightarrow B \rightarrow BD$
    * The first is much faster, since the rate for the transition from WT to A is 8.1 compared to 2.6 of the transition B to D (from competing exponentials, the probabilities of moving to A and B are 0.76 and 0.24, respectively).
  - In the other two paths D is the third gene to appear:
    * $WT \rightarrow A \rightarrow AB \rightarrow ABD$
    * $WT \rightarrow B \rightarrow AB \rightarrow ABD$
    * These two paths take the same time, on average: both A and B need to appear (with rates given by λ_A, λ_B) and then we need D to appear (λ_D).
  - Similarly, to get to genotype "A, B, D" we can follow these paths:
    * $WT \rightarrow A \rightarrow AB \rightarrow ABD$
    * $WT \rightarrow B \rightarrow AB \rightarrow ABD$
    * $WT \rightarrow A \rightarrow AD \rightarrow ABD$
    * $WT \rightarrow B \rightarrow BD \rightarrow ABD$
    * All of them take the same expected time, as we need A, B, and D to happen, each governed by λ_A, λ_B, λ_D, respectively.
- In terms of fitness, if we used OncoSimulR (see additional document "Using OncoSimulR to get accessible genotypes and transition matrices"), we would write, for the fitness of AB: $(1 + \lambda_A)(1 + \lambda_B)$, for AD $(1 + \lambda_A)(1 + \lambda_D)$, and for ABD $(1 + \lambda_A)(1 + \lambda_B)(1 + \lambda_D)$.
  - Note, specifically, that genotypes AD and BD are not fitness equivalent, unless $\lambda_A = \lambda_B$.

2. XOR

- Using the above example, and focusing only on A, B, C, these are the only ways of seeing a C:
  - $WT \rightarrow A \rightarrow AC$
  - $WT \rightarrow B \rightarrow BC$
  - As we have a XOR, no routes can go through AB.
  - The first is much faster and common than the second ($\lambda_A = 8.1; \lambda_B = 2.6$).
  - Fitness (again, this is relevant if using, for example, OncoSimulR) of AC is $(1 + \lambda_A)(1 + \lambda_C)$ and of BC $(1 + \lambda_B)(1 + \lambda_C)$.

3. Both OR and XOR

- There is nothing new. As an example, gaining both C and D mutations.
  - $WT \rightarrow A \rightarrow AC \rightarrow ACD$
  - $WT \rightarrow B \rightarrow BC \rightarrow BCD$
  - $WT \rightarrow A \rightarrow AD \rightarrow ACD$
  - $WT \rightarrow B \rightarrow BD \rightarrow BCD$
  - There is no path going through AB since C has a XOR relationship on A and B.
  - In the first path we first need to wait for A to happen (rate λ_A) then C (λ_C) then D (λ_D).
  - Same for the second, with B instead of A. The first path is much more common than the second.

- The third path transposes the order of occurrence of D and C, but takes the same average time as the third. Note that the fitness of the final genotype is the same through both routes, only the order of steps changes.
- The fourth path transposes the order of occurrence of D and C, but takes the same average time as the fourth. Note that the fitness of the final genotype is the same through both routes, only the order of steps changes.

6.6 Combining AND, OR, XOR?

Nothing changes. Use the rules for AND where there is an AND, XOR where there is a XOR, OR where there is an OR. Again, in the HESBCN model if a gene depends on a set of genes, it has the same type of dependency on all the genes of that set.

7 Shiny EvAM-Tools web app

7.1 Building the models and example data

7.1.1 Example data/DAGs/MHNs

For each of the ways to input cross-sectional data or build models, we provide some pre-built examples. You can use these to directly run the CPMs or modify them to your liking. The MHN and DAG builders allow you start from working examples. Explanations on what is available and why follow:

Modifying pre-built examples If you modify the examples, you probably will want to give your dataset or example a distinctive name.

Cross-sectional data We provide data sets that have been generated under specific models with AND, OR, XOR. However, no DAG is shown here, and that is on purpose; use the labels as possible hints, not as the truth, and compare with datasets you can generate from models you specify under the DAG builder.

DAG Several examples to illustrate OR/AND/XOR and mixtures of the above are provided.

7.2 Cross-sectional data input

7.2.1 Number of genes and genes without mutations

When creating user data (for instance, when adding new genotypes), any gene that has no mutations is automatically excluded from the data, regardless of the setting for number of genes. This is a feature, not a bug. For example, suppose you set the number of genes to 3, but you only specify frequencies, or counts, for genotypes "A" and "A, B". The data set will only contain columns for genes A and B (since gene C has no mutations and it would be excluded during the analyses).

7.3 Naming and saving data and model

To make it easier to use and analyze different models and data sets, you can name/rename the dataset used. This is at the top of the page, under "(Re)name the data".

You can save the data and models (at the bottom of the page). If you define a DAG or an MHN model, the saved object contains the model. In the case of the DAG, that includes the generated data (if any), and several components that are used by evamtools and that provide a complete description of the model. If you use MHN, what are saved are the sampled data (if any) and the $\log\Theta$ matrix you entered.

7.4 Tabular output from the Shiny app

Transition probabilities Conditional probability of transitions to a genotype (obtained using competing exponentials from the transition rate matrix for all methods except OT and OncoBN); for OT and OncoBN this is actually an abuse of the untimed oncogenetic tree model, as explained in Diaz-Uriarte and Vasallo (2019).

Transition rates Transition rates of the continuous-time Markov chain that models the transition from one genotype to another. This option is not available for OT and OncoBN, as these do not return rates.

Predicted genotype relative frequencies The predicted frequencies of genotypes under the model. See section *4.2, "Error models"* for details about OT and OncoBN

Sampled genotype counts Counts, or absolute genotype frequencies, obtained by sampling from the predicted frequencies.

Observed genotype transitions (counts) Only if you ask for 'Sample for observed genotype transitions' in "Advanced options and CPMs to use". This is the number of observed transitions between genotypes.

See also further details in section section *4.2, "Error models"* as well as in the help for functions `evam` and `sample_CPMs`.

8 FAQ

8.0.1 Why haven't you used method X?

We have included here what we believe are the current state-of-the-art methods that have existing public implementations that run in reasonable time. After searching the literature, we have included any method that could be deemed appropriate. We have, in fact, provided access to two very recent methods: H-ESBCN and OncoBN (and their github repos show we have contributed bug reports).

Among the remaining methods available, most of them do not seem to be developed nor used anymore. For some of these methods, their authors have developed newer methods that seem to have superseded the former methods. Some other methods have dependencies on external libraries that are not open source. And, of course, we cannot provide access to methods that have no software, or have software that will run only under proprietary systems. Some further comments are provided in S4_Text in Diaz-Uriarte and Vasallo (2019) (https://doi.org/10.1371/journal.pcbi.1007246.s006).

If you think we have overlooked a method that should be included, please let us know.

8.1 With OncoBN sometimes I obtain DAGs that are not transitively reduced

Yes, that can happen. See details here https://github.com/phillipnicol/OncoBN/issues/5.

8.2 In the DAG figures, why do nodes with two or more incoming edges have only a single annotated edge with a number?

Because the number, which is the λ (CBN, HESBCN) or θ (OncoBN) is the rate (CBN, HESBCN) or probability, conditional on the assumptions indicated by the DAG being satisfied. So the λ or θ are per node, not per edge. For instance, suppose gene C depends on both A and B (there is an AND); and you see a number of 0.7. That is the λ or θ for observing C mutated when both A and B are mutated.

And why then not annotate the nodes, instead of the edges? Because in our experience:

- Annotating nodes leads to more confusing figures.
- Annotating edges shows what transitions are likely/fast, an idea not conveyed by annotating nodes.

8.3 Do sampled genotype frequencies and counts contain observation noise? And predicted genotype frequencies?

For all models except OT, predicted genotype frequencies do not have observation noise added. The OT model itself estimates noise, and thus predicted frequencies obtained from models fitted to observed data incorporate observation noise. See *"Error models" (section 4.2)*.

When we obtain a finite sample from the predicted frequencies, you can decide to add observation noise with argument `obs_noise` to function `sample_CPMs`; what happens with OT depends on whether the predictions are from a simulated model or a model fit to observed data; see details in *"Error models and simulating and sampling from OT" (section 4.3)*.

8.4 I want to setup my own Shiny server with different default "Advanced options"

In file `EvAM-Tools/evamtools/inst/shiny-examples/evamtools/ui.R` search for "Advanced options" and modify the defaults to whatever you want.

8.5 How can I use the Shiny app in a local intranet with load balancing using multiple Docker instances

This is well beyond the scope of this document and there are many options available. One that can work (and this is more or less what we actually do) is the following:

- Start multiple Docker instances (say, 20) by changing the range of ports, for example, 3010 to 3030.
- Use HAProxy (https://www.haproxy.org/) so that you have a single entry point for all requests to the service that are then distributed, with load balancing, to the 20 instances. You will want to use "sticky connections" (see the HAProxy documentation).

As said above, this is just a sketch of the basic procedure. There are many other options.

8.6 If I use the Docker image for the package (rdiaz02/evamrstudio), can I run the Shiny app?

Yes, you can. Just start a browser as explained in the README (https://github.com/rdiaz02/EvAM-Tools#how-to-run-the-r-package-from-the-docker-image). Then, once in RStudio, in the R console type `runShiny()` and you will have the interactive Shiny app open. But even if you can do it, it is not clear why you'd want to do this: if you only want to run the Shiny app, the Docker image rdiaz02/evamshiny is lighter and the steps to launch it much faster.

8.7 Why aren't you using Shiny Server?

Because we did not see it as necessary or convenient. If you want to run Shiny interactively from an R session load `evamtools` and call function `runShiny`; no need for Shiny Server.

If we want to run Shiny as a service, with Docker images it is rather straightforward to launch a bunch Docker instances and use HAProxy to access to them using load-balancing (see 8.5) or any other such similar solution.

Moreover, notice this in the Shiny Server documentation (https://shiny.rstudio.com/articles/shiny-server.html): "Shiny Server will host each app at its own web address and automatically start the app when a user visits the address. When the user leaves, Shiny Server will automatically stop the app. " That is not exactly what we want. We want the

containers to be up and running, ready to answer requests as they come with minimal latency. Moreover, a single Shiny Server would have given access to a single instance of the app (so that if two or more users access the app, one of the users has to wait while R is busy executing what the other user is running); to give users access to multiple simultaneous instances we would have needed, for example, multiple Docker images each with its own Shiny Server.

However, we might be missing something; if you think Shiny Server would allow or ease some use cases, please let us know.

8.8 I want to create my own Docker images

If you want to modify the Docker images, modify the Dockerfiles: Dockerfile-evam-rstudio (for the RStudio Dockerfile that launches RStudio) or Dockerfile-evam-shiny (well, for the Dockerfile that creates the container to run shiny).

Then, from the 'EvAM-Tools' directory run one or both of:

```
docker build -f Dockerfile-evam-shiny  --tag somename .
docker build -f Dockerfile-evam-rstudio  --tag somename .
```

You can now run these images, as explained in the README file..

Note: it is possible, and actually a better idea, to run docker without sudo; look a the Docker documentation: https://docs.docker.com/engine/security/rootless/).

8.8.1 What if creating the image fails because of no internet connection from the container

Creating the above image requires installing R packages and that might fail because the Docker container cannot connect with the internet. The following might help: https://superuser.com/a/1582710, https://superuser.com/a/1619378. In many cases, doing `sudo systemctl restart docker` might be enough.

8.8.2 Cleaning the build cache and stale old images

Sometimes (e.g., if the base containers change or you want to remove build cache) you might want to issue

```
docker builder prune
```

or the much more drastic

```
docker system prune -a
```

Please, read the documentation for both.

8.8.3 Copying docker images from one machine to another

Yes, that can be done. See here, for example: https://stackoverflow.com/a/23938978

9 License and copyright

References

Angaroni, F., Chen, K., Damiani, C., Caravagna, G., Graudenzi, A., and Ramazzotti, D. (2021). PMCE: Efficient inference of expressive models of cancer evolution with high prognostic power. *Bioinformatics*, **38**(3), 754–762.

Diaz-Colunga, J. and Diaz-Uriarte, R. (2021). Conditional prediction of consecutive tumor evolution using cancer progression models: What genotype comes next? *PLOS Computational Biology*, **17**(12), e1009055.

Diaz-Uriarte, R. and Vasallo, C. (2019). Every which way? on predicting tumor evolution using cancer progression models. *PLoS computational biology*, **15**(8), e1007246.

Gerstung, M., Baudis, M., Moch, H., and Beerenwinkel, N. (2009). Quantifying cancer progression with conjunctive bayesian networks. *Bioinformatics*, **25**(21), 2809–2815.

Gotovos, A., Burkholz, R., Quackenbush, J., and Jegelka, S. (2021). Scaling up Continuous-Time Markov Chains Helps Resolve Underspecification. *arXiv:2107.02911 [cs, stat]*.

Hosseini, S.-R., Diaz-Uriarte, R., Markowetz, F., and Beerenwinkel, N. (2019). Estimating the predictability of cancer evolution. *Bioinformatics*, **35**(14), i389–i397.

Montazeri, H., Kuipers, J., Kouyos, R., Böni, J., Yerly, S., Klimkait, T., Aubert, V., Günthard, H. F., Beerenwinkel, N., and Study, S. H. C. (2016). Large-scale inference of conjunctive bayesian networks. *Bioinformatics*, **32**(17), i727–i735.

Nicol, P. B., Coombes, K. R., Deaver, C., Chkrebtii, O., Paul, S., Toland, A. E., and Asiaee, A. (2021). Oncogenetic network estimation with disjunctive bayesian networks. *Computational and Systems Oncology*, **1**(2), e1027.

Schill, R., Solbrig, S., Wettig, T., and Spang, R. (2020). Modelling cancer progression using mutual hazard networks. *Bioinformatics*, **36**(1), 241–249.

Szabo, A. and Boucher, K. (2002). Estimating an oncogenetic tree when false negatives and positives are present. *Mathematical Biosciences*, **176**(2), 219–236.

Szabo, A. and Boucher, K. M. (2008). Oncogenetic trees. In W.-Y. Tan and L. Hanin, editors, *Handbook of Cancer Models with Applications*, pages 1–24. World Scientific.

Wilkinson, D. J. (2019). *Stochastic modelling for systems biology, 3rd ed.* Chapman and Hall/CRC.

Package 'evamtools'

April 27, 2022

Type Package

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

Version 1.1.2

Date 2022-04-27

Author Ramon Diaz-Uriarte [aut, cre]
Pablo Herrera-Nieto [aut]
Daniele Ramazzotti [ctb],
Rudolf Schill [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 , mccbn , RhpcBLASctl , Rlinsolve, fastmatrix , graph, Rgraphviz , R.utils

Suggests testthat (>= 3.0.0)

Config/testthat/parallel true

Config/testthat/edition 3

NeedsCompilation yes

R topics documented:

evam 2
evamtools-deprecated 8
every_which_way_data 8
ex_mixed_and_or_xor 10
examples_csd 10
plot_evam 11
random_evam 14
runShiny 16
sample_evam 17
SHINY_DEFAULTS 19

Index **21**

`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(
                     steps = 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.
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).
- n_steps: Number of steps to run. Default: 100000
- 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 HESBCN 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`.

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

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]
out1 <- evam(Dat1,
            methods = c("CBN", "OT", "OncoBN",
                        "MHN", "HESBCN", "MCCBN"))


## 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, with 17 different scenarios. They represents different relationships: AND, OR, and XOR, but there are also more complex set ups, like sign epitasis and reciprocal sign epistasis.

dag dag, It has a simple example of an AND relationships

matrix matrix, Simple random example

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.

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]

out <- evam(dB_c1,
            methods = c("CBN", "OT", "HESBCN", "MHN", "OncoBN", "MCCBN"))

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 lambada 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).

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 to the data? `obs_noise > 0` is the proportion of observations with error (for instance, genotyping error). If larger than 0, this 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

∗ **datasets**
every_which_way_data, 8
ex_mixed_and_or_xor, 10
examples_csd, 10
SHINY_DEFAULTS, 19
.ev_SHINY_dflt *(*SHINY_DEFAULTS*)*, 19

adaptive.simulated.annealing, *4*

distribution.oncotree, *3*

evam, 2, *11*, *16*, *18*
evamtools-deprecated, 8
every_which_way_data, 8
ex_mixed_and_or_xor, 10
examples_csd, 10

fitCPN, *4*

mclapply, *3*, *4*

plot_CPMs *(*plot_evam*)*, 11
plot_evam, *7*, *8*, 11

random_evam, 14, *18*
runApp, *16*
runShiny, 16

sample_CPMs *(*sample_evam*)*, 17
sample_evam, *7*, *8*, *11*, *12*, *16*, 17
SHINY_DEFAULTS, 19

Using OncoSimulR to get accessible genotypes and transition matrices

Ramon Diaz-Uriarte[1,2,†]

[1]Dpt. of Biochemistry, School of Medicine, Universidad Autónoma de Madrid, Madrid, Spain
[2]Instituto de Investigaciones Biomédicas 'Alberto Sols' (UAM-CSIC), Madrid, Spain
[†]To whom correspondence should be addressed: r.diaz@uam.es
https://ligarto.org/rdiaz

2022-04-27
Version 1e74fcd

Contents

1 Introduction **1**

2 Using OncoSimulR to get accessible genotypes and transition matrices **2**
2.1 Computing fitness of genotypes: for CBN (and MCCBN) and OT 2
2.2 Crucial assumption above 3
2.3 A detail about the fitness specification with OncoSimulR's DAGs and epistatic fitness specifications 4
2.4 Transition probabilities using an epistatic specification 5
2.4.1 Even more details about CBN, s, λ 6

3 What about H-ESBCN/PMCE, with AND, XOR, OR? **7**

4 OncoBN **7**

5 MHN **7**

6 Benefits of this exercise with OncoSimulR **8**

7 License and copyright **8**

1 Introduction

Here I explain how we can use OncoSimulR[1] to get accessible genotypes and transition matrices for CBN (and MCCBN), OT, HESBCN, and OncoBN. The code for using OncoSimulR is implemented in access_genots_from_oncosimul.R.

[1]A BioConductor package for forward population genetic simulation in asexual populations; it allows us to specify fitness, among other ways, using DAGs of restrictions. Repo at https://github.com/rdiaz02/OncoSimul. Citation: Diaz-Uriarte, R. (2017). OncoSimulR: Genetic simulation with arbitrary epistasis and mutator genes in asexual populations. Bioinformatics, 33(12), 1898–1899. https://doi.org/10.1093/bioinformatics/btx077.

(This document is written, on purpose, using an itemized list style, with plenty of repetition and detailed examples, to make it suitable for instance for class use.)

2 Using OncoSimulR to get accessible genotypes and transition matrices

OncoSimulR has had, for a long time, the AND, XOR, OR operations (see the help of "allFitnessEffects", under "typeDep"), if a gene depends on other genes with the same relationship for all parents. Since we can obtain the fitness of genotypes, obtaining accessible genotypes is simple:

- Use an appropriate setting for the "s"
- Use $-\infty$ for sh (so if restrictions are not satisfied, a genotype has fitness 0).
- Evaluate the fitness of genotypes.
- Call function "genots_2_fgraph_and_trans_mat".
  - This is a general function.
  - In particular, it does not assume that mutations that do not kill a genotype always increase fitness or at least do not decrease it.
  - This function returns accessible genotypes, fitness graph, and transition matrices directly from the fitness of the genotypes.

2.1 Computing fitness of genotypes: for CBN (and MCCBN) and OT

- OncoSimulR, when using DAGs, uses a model of fitness (birth rate), for a genotype with restrictions satisfied as $\Pi(1 + s_i)$.
  - Again, to emphasize the above: s_i, when using OncoSimulR with a DAG, is the selection coefficient from gene i **with its restrictions satisfied.**
- Recall that the transition probabilities can be computed from competing exponentials. For example, suppose from genotype A we can go to enotypes AB and AC. The probability of going to AB should be $\lambda_B/(\lambda_B + \lambda_C)$.
- As explained in Hosseini et al., 2019 (Hosseini, S., Diaz-Uriarte, Ramon, Markowetz, F., & Beerenwinkel, N. (2019). Estimating the predictability of cancer evolution. Bioinformatics, 35(14), 389–397. https://dx.doi.org/10.1093/bioinformatics/btz332), p. i392, let us define the selective coefficient of a mutation i as the fitness difference that it causes along the mutational pathway.
  - Using our previous example, that would be $Pr(A \rightarrow AB) = \frac{W_{AB}-W_A}{(W_{AB}-W_A)+(W_{AC}-W_A)}$, where W_x is fitness of genotype x.
  - If we write, as is common, $W_{AB} = W_A\ (1 + s_B)$, we get from the above $Pr(A \rightarrow AB) = \frac{s_B}{s_B+s_C}$.
  - (So, actually, the s_i are the relative differences, not just differences, in fitness: $s_B = \frac{W_{AB}-W_A}{W_A}$, as shown also in p.7 of the supplementary material of Weinreich, D. M., Delaney, N. F., DePristo, M. A., & Hartl, D. L. (2006). Darwinian Evolution Can Follow Only Very Few Mutational Paths to Fitter Proteins. Science, 312(5770), 111–114.https://dx.doi.org/10.1126/science.1123539 .)

- (We wrote $W_{AB} = W_A\ (1+s_B)$. This we can do as we explained what the meaning of the s_i are: selection coefficient from gene i with its restrictions satisfied. See below: Transition probabilities using an epistatic specification.)

- Note that this is the same procedure as in Weinreich et al., 2006, (Weinreich, D. M., Delaney, N. F., DePristo, M. A., & Hartl, D. L. (2006). Darwinian Evolution Can Follow Only Very Few Mutational Paths to Fitter Proteins. Science, 312(5770), 111–114.https://dx.doi.org/10.1126/science.1123539) supplementary material, p. 4): $s_{i\to j}$ "the selection coefficient for the mutation that carries allele i to allele j"[2].

  - Specifically, see equation S5b in the supplementary material of Weinreich et al., 2006, which shows the relationship between the expected value of the conditioned probability of fixation in a mutation from i to j and the expected value of the ratio of the selection coefficient for the mutation that turns i to j over the sum of selection coefficients of beneficial mutations that turn i into all other alleles; see also their figure S1 in p. 7 of the supplementary material that shows the accuracy of their expression.

- Additional note: In Gerstung et al., 2011 (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. https://dx.doi.org/10.1371/journal.pone.0027136), PLoS ONE (p.8) the relationship between λ_i and s_i is also discussed.

- And why the fitness difference with respect to the ancestor and not just a ratio of fitnesses? Suppose $W_{AC} < W_A < W_{AB}$. The procedure above correctly would say that we always transition from A to AB.

- So we do as follows:

  - Set $s_i = \lambda_i$ (for OT, we use edgeWeight instead of λ).
  - Obtain the fitness of all genotypes from OncoSimulR.
  - If so desired (e.g., to ensure the maximum fitness is a specific number), multiply all fitness values (not the WT, which is kept at 1) by the appropriate scaling factor.

  Is the above correct for OT? Strictly not as OT are untimed oncogenetic trees. (And, yes, we are aware that under OT if you have, say, both A and B descend from root, the probability of genotype A is $p_a(1-p_b)$).

2.2 Crucial assumption above

- We compute fitness above assuming that only one of two things can happen: a mutation provides a fitness benefit or it leads to death. When the requirements are satisfied, a mutation conveys a fitness increase (λ_i); otherwise, the cell with the mutation has fitness 0.

- Strictly, mutations without dependencies satisfied might not be lethal, but they should not confer any fitness advantage, so that we will never observe them (Gerstung et al., 2009, p. 2810, say "(...) mutations that need to be present before mutation i can fixate." and Gerstung and Beerenwinkel, 2010, p. 126, "with steps including both mutation and clonal expansion occurring at effective rates k j").

[2]Selection coefficient has the usual textbook definition. For example, Gillespie, 2004 (Population genetics: a concise guide, 2nd. Baltimore, Md: The Johns Hopkins University Press.), p. 63. But here we write $W_{AB} = W_A\ (1 + s_B)$, and thus if $s_B > 0$ AB is fitter than A; see also p.7 of the supplementary material of Weinreich et al., 2006

- In OncoSimulR, in addition to the s_i, it is possible to set `sh = 0`, meaning there is no penalty for not respecting the restrictions. When `sh = 0` there is also no fitness gain, either, so fitness for those genotypes ends up being the fitness of the immediate parent (there is no contribution from the gen without restrictions satisfied to the fitness of the parent genotype). Regardless, when `sh = 0`, the transition matrix does not change compared to the transition matrix we obtain if we assume that mutations to genotypes with non-satisfied dependencies lead to a fitness of 0.

- To elaborate on this point: The output from the code, with `sh = 0`, will result in more genotypes being shown as accessible. It is arguable, though, that those genotypes are not really accessible, since their fitness is never larger than the fitness of their ancestor. So the probability of transitioning to them will be 0 under the expressions above when in SSWM. We have changed the code so that now something is only shown as accessible if its fitness is strictly larger than the fitness of its ancestor.

- (Actually, in OncoSimulR, the s_h can vary by gene, so we could have different s_{hi}, but this does not affect these arguments).

2.3 A detail about the fitness specification with OncoSimulR's DAGs and epistatic fitness specifications

- We said above: "Again, to emphasize the above: s_i, when using OncoSimulR with a DAG, is the selection coefficient from gene i **with its restrictions satisfied.**"

- This also means, when using DAGs in OncoSimulR, that terms such as s_{ij} are not used in that specification: they are not needed as the DAG models do not include epistasis beyond that given by the DAG, and all these epistatic interactions we capture with the DAG and the s_i and s_h, which denote the fitness effects when restrictions are satisfied and not satisfied, respectively.

- But with OncoSimulR you can also specify fitness with the usual multiplicative expression where you specify explicitly the contribution of genes and gene interactions (e.g., s_{ij} for the effect of the interaction between genes i and j, so that fitness of the genotype with both i and j mutated would be $(1 + s_i)\ (1 + s_j)\ (1 + s_{ij})$).

- In other words, suppose j depends on i. The usual epistatic interaction fitness specification would write: $W_{ij} = (1 + s_i)\ (1 + s_j)\ (1 + s_{ij})$ and $W_j = (1 + s_j)$.

- Using the DAG, if the restriction is not satisfied, i.e., for genotype with only j: $W_j = (1 + s_h)$. If the restriction is satisfied, $W_{ij} = (1 + s_i)(1 + s_j)$. So the meaning of the s is different.

- To fully elaborate here, and to give a more complex example, suppose C depends on both A and B, according to the DAG.

  - When using the DAG, then, these are the expressions for some genotypes:
    * $W_{ABC} = (1 + s_A)(1 + s_B)(1 + s_C)$
    * $W_{AC} = (1 + s_A)(1 + s_h)$
    * (If we had gene-specific s_h, such as s_{hC}, that does not change anything fundamental, just adds a subscript)
  - If we were to use an epistatic specification:
    * $W_{ABC} = (1 + s_A)(1 + s_B)(1 + s_C)(1 + s_{AB})(1 + s_{AC})(1 + s_{BC})(1 + s_{ABC})$
    * $W_{AC} = (1 + s_A)(1 + s_C)(1 + s_{AC})$

- Therefore, the meaning of the s_i is not the same under both specifications. That is why we said "s_i, when using OncoSimulR with a DAG, is the selection coefficient from gene i **with its restrictions satisfied.**" and "terms such as s_{ij} are not used in that specification: they are not needed as the DAG models do not include epistasis beyond that given by the DAG, and all these epistatic interactions we capture (...)".

- Yes, sure, we could always re-write the s_i and s_{hi} in the DAG specification as a function of the s_i, s_{ij}, s_{ijk} in the epistatic specification. (See section Transition probabilities using an epistatic specification).

- This was just for the sake of completeness. The use of s_h and the epistatic fitness specification is fully explained in the documentation of OncoSimulR and its vignette, and is not in the scope of this document.

2.4 Transition probabilities using an epistatic specification

- Suppose B and C both depend on A. If we were to use an specification with epistasis, instead of how we have used and interpreted the s_i using the DAGs, then we would have to write $W_{AB} = W_A\ (1 + s^*_B)\ (1 + s^*_{AB})$, where now I am using $s*$ to make the sets of s clearly distinct. We can express the s_B as a function of s^*_B and s^*_{AB}. If we set $s^*_B = 0$ (similar to setting `sh = 0`) then $s_B = s^*_{AB}$. Otherwise, the expression will be $s_B = ((1 + s^*_B)\ (1 + s^*_{AB})) - 1$; and, to respect the restrictions, it must be the case that $s^*_B < 0$.

- The expressions for probabilities of transition become messier, but you end up with a ratio of

$$\frac{increase\_in\_fitness\_from\_acquiring\_B}{increase\_in\_fitness\_from\_acquiring\_B + increase\_in\_fitness\_from\_acquiring\_C}$$

  where $increase\_in\_fitness\_from\_acquiring\_B$ would include the effect of B, s^*_B, and the epistatic interaction, s^*_{AB}.

- s_B is still the relative fitness difference $\frac{W_{AB}-W_A}{W_A}$. Which is the same as saying that $((1 + s^*_B)\ (1 + s^*_{AB})) - 1 = \frac{W_{AB}-W_A}{W_A}$ is the relative fitness difference.

- This shows we can directly use the DAG fitness specification where we take the s_i as the selection coefficient from gene i with its restrictions satisfied.

- And why do we do what we do with CBN? Because it simplifies everything and fitness can be written as $\prod(1 + s_i)$ for any genotype with its restrictions satisfied.

  - If neither A nor B depend on anything, then the expression for fitness is $(1+s_A)\ (1+s_B)$ because, under CBN, there is no epistasis here so $s_{AB} = 0$ (look, for example, at the transition rate matrix in Montazeri et al., 2016, Figure 1, for the transition from genotype 1 to genotype 1,2 or from genotype 2 to genotype 1,2).
  - If B depends on A, when we consider the transition from A to B, we can use a single term, $(1 + s_X)$ to multiply $(1 + s_A)$, and that $s_X = \lambda_B$. That λ_B is the (relative) increase in fitness due to B, when B's restrictions are satisfied (for example, in Example 1 in Montazeri et al., 2016 (Large-scale inference of conjunctive Bayesian networks. Bioinformatics, 32(17), 727–735. https://dx.doi.org/10.1093/bioinformatics/btw459), see the transition rate matrix from genotype 2 to genotype 2,4[3]). You

[3]Notice that Figure 1 is correct, but the matrix in Example 1 has a typo, and is missing the entry for λ_4; or look at the transition from 1,2 to 1,2,3 and 1,2,4

can think of this s_X as the joint combination of the effect of B on its own and the epistasis of A and B; but thinking of B on its own is a moot point, since B on its own (i.e., without A, without its restrictions satisfied) is not a genotype that can be observed.

- Thus, for any genotype, do $\prod(1 + s_i)$, where $s_i = \lambda_i$ when the restrictions are satisfied.

2.4.1 Even more details about CBN, s, λ

- Remember that having $\lambda_i < 0$ makes no sense.
- Suppose a model where A and B depend on no one, D depends on A and C depends on both A and B.
- Simple case:
  - $W_{AD} = (1 + \lambda_A)(1 + s_D)(1 + s_{AD})$
  - $W_{AD} = (1 + \lambda_A)(1 + \lambda_D)$
  - So: $1 + s_{AD} = \frac{1+\lambda_D}{1+s_D}$
  - If $s_D = 0$ we get the $s_{AD} = \lambda_D$ or "the epistatic term is equal to the lambda".
  - If $s_D < 0$ then the epistatic term, $s_{AD} > \lambda_D$: it has to be large enough to compensate for the decrease in fitness from the single D.
  - This can matter if we try to generate $s_{xy...}$ from some distribution and match them to the λ.
- Beware, though, of a simple interpretation of the s_D as s_h, specially when there are more genes. An example:
  - $W_{ADC} = (1 + \lambda_A)(1 + s_D)(1 + s_{AD})(1 + s_C)(1 + s_{DC})(1 + s_{AC})(1 + s_{ACD})$
  - But we can replace the second and third terms:
    - $W_{ADC} = (1 + \lambda_A)(1 + \lambda_D)(1 + s_C)(1 + s_{DC})(1 + s_{AC})(1 + s_{ACD})$
  - OncoSimulR is NOT replacing all the extra terms by s_h.
    - If it did you would get:
      - $W_{ADC} = (1 + \lambda_A)(1 + \lambda_D)(1 + s_h)^4$
    - But what OncoSimul actually gives you is:
      - $W_{ADC} = (1 + \lambda_A)(1 + \lambda_D)(1 + s_h)$
    - Why? Because only one gene, C, has not got its restrictions satisfied.
    - In other words, the number of $(1+s_h)$ is equal to the number of genes (not genes and gene combinations) with their restrictions not satisfied.
  - In particular, note that this is not correct:
    - $W_{ADC} = (1 + \lambda_A)(1 + s_h)(1 + s_{AD})(1 + s_h)(1 + s_h)(1 + s_h)(1 + s_h)$
    - Where the first s_h would correspond to s_D and the rest to C, AC, DC, ACD.
    - And thus, it is not correct to write: $1 + s_{AD} = \frac{1+\lambda_D}{1+s_h}$
  - Of course, if $s_h < 0$ then $W_{ADC} < W_{AD}$.
- And with this same DAG, we can write either:
  - $W_{ABC} = (1 + \lambda_A)(1 + \lambda_B)(1 + \lambda_C)$
  - $W_{ABC} = (1 + \lambda_A)(1 + \lambda_B)(1 + s_C)(1 + s_{AC})(1 + s_{BC})(1 + s_{ABC})$

- As before we could do: $(1 + s_{ABC}) = \frac{1+\lambda_C}{(1+s_C)(1+s_{AC})(1+s_{BC})}$
- And this shows again that the epistatic term for ABC (i.e., when restrictions are satisfied) might have to be very large to compensate for large negative fitness effects of mutations without restrictions satisfied (e.g., s_C).

3 What about H-ESBCN/PMCE, with AND, XOR, OR?

By H-ESBCN/PMCE I mean the method described in

- 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

We can repeat what we did above, with OR and XOR replaced by, well, OR and XOR in OncoSimulR (OR and XOR are also called SM and XMPN in OncoSimulR). OncoSimulR has dealt with OR, XOR, AND, and mixtures of them since many years ago. Remember also that in the H-ESBCN model if a gene depends on a set of genes, it has the same type of dependency on all the genes of that set.

4 OncoBN

What about OncoBN, the method described in 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?

OncoBN can fit both conjunctive (AND) and disjunctive (OR, not XOR) models; for the first you specify `model = ''CBN''` and for the second `model = ''DBN''`. So it resembles CBN and HESBCN. However, the θs returned by OncoBN are not rates, as in CBN, HESBCN, or MHN, but rather probabilities of seeing specific alterations at the time of observation as in OT. So probably a better way to think of OncoBN is as an extension of OT, where nodes can have multiple parents, and the relationship of dependence can be AND or OR (but not both).

We deal with OncoBN as with any other method, but as we do with OT, we do not interpret the parameters as rates. This also means that our transition matrices (again, transition matrices, not transition rate matrices: no transition rate matrices are returned for OT or OncoBN), as for OT, are really an abuse of the untimed oncogenetic model.

When using OncoSimulR to represent OncoBN models, there is nothing new. If OncoBN was fitted specifying "CBN", we use ANDs, if it used "DBN" we use ORs when computing fitness.

5 MHN

MHN has been described in 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.

We cannot use OncoSimulR as for the rest of the modes, because the MHN model is rather peculiar if taken at face value as an evolutionary model (see 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 ; in particular, see section 1.7 of the Supporting Information: https://doi.org/10.1371/journal.pcbi.1009055.s001).

To express MHN in terms of fitness of genotypes, we would need to express it as a model where order of acquisition of mutations matters. This is possible with OncoSimulR[4], but it does not provide any additional intuition, and can lead to a huge number of fitnesses for a genotype (a genotype with k mutated loci could possibly have $k!$ different fitnesses, one for each of its $k!$ different ways of mutation its k loci).

6 Benefits of this exercise with OncoSimulR

- We make the fitness model explicit.
- We can double check the code in `evamtools` for obtaining fitness graphs and transition probabilities as some critical computations are being done with very different code.

7 License and copyright

The source of this document and the EvAM-Tools package is at https://github.com/rdiaz02/EvAM-Tools.

[4]We would need to use "order effects" for the fitness specification. See the vignette for OncoSimulR https://rdiaz02.github.io/OncoSimul/OncoSimulR.html#36_Order_effects, and the help for function `allFitnessEffects`.