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

2022-10-01
Version a206824

This PDF is the concatenation of four different files:

1. **EvAM-Tools: additional technical documentation:** Additional documentation about EvAM-Tools. These include technical details (CPMs, error models, predicting genotype frequencies, generation of random evam models), and a FAQ.

2. **EvAM-Tools: examples:** Commented examples, with both real and simulated data, that illustrate the use and utility of EvAM-Tools.

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

4. **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: methods' details and FAQ

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

2022-10-01
Version a206824

Contents

1 Introduction

This document provides additional details about EvAM-Tools and the methods included in it. 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. Another file, `evamtools_examples.pdf`, from https://rdiaz02.github.io/EvAM-Tools/pdfs/evamtools_examples.pdf, includes commented examples, with both real and simulated data, that illustrate the use and utility of EvAM-Tools.

2 Cancer Progression Models included in EvAM-Tool: details

2.1 Cancer Progression Models and cross-sectional data: overview and type of input data

In cross-sectional data a single sample is obtained from each subject or patient. That single sample represents the "observed genotype" of, for example, the tumor of that patient. Genotype can refer to single point mutations, insertions, deletions, or any other genetic modification; in fact, these models have been used to analyze point mutations, gains and losses of CGH regions, SNP alterations, pathway alteration data, etc: the granularity of the data and level of analysis depend on the question addressed, and is not inherent to the models. As is often done by Cancer Progression Models (CPM) software, we think of the cross-sectional data as being stored in a matrix, where rows are patients or subjects, and columns are genes/CGH regions/SNPs/pathways/etc; the data is a 1 if the event (or alteration or mutation) was observed and 0 if it was not.

We have used expressions such as "genotype", "mutation" and other genetic- and genomic-related terms, but nothing prevents CPMs from being used with non-genetic, non-genomic data, and thus our preference for the expression "event accumulation models". The key features that the data must have to be properly analyzed with these methods are: a) that events or alterations are (or can be reasonably assumed to be) gained one by one; b) that once gained, they are not lost (e.g., there is no back mutation); c) that we can consider the different individuals/patients in the cross-sectional data as replicate evolutionary experiments or runs where all individuals are under the same constraints (e.g., genetic constraints if we are dealing with mutations); see further details below (section *2.2, "Cancer Progression Models (CPMs): assumptions"*).

Cancer progression models (CPMs) or, more generally, event accumulation models, use these cross-sectional data to try to infer restrictions in the order of accumulation of events; for example, that a mutation on gene B is always preceded by a mutation in gene A (maybe because mutating B when A is not mutated results in a lethal state for that cell). Inferring restrictions, in the sense just explained (B only if A), is what CBN, OT, OncoBN, and H-ESBCN do. Other cancer progression models, such as MHN, instead of modeling deterministic restrictions, model promoting/inhibiting interactions between genes, for example that having a mutation in gene A makes it very likely to gain a mutation in gene B.

2.2 Cancer Progression Models (CPMs): assumptions

CPMs assume that the observations in the cross-sectional data set are independent realizations of evolutionary processes where the same constraints hold for all tumors; therefore, a cross-sectional data set is considered a set of replicate evolutionary experiments where all individuals are under the same (genetic) constraints (Gerstung *et al.*, 2011; Beerenwinkel *et al.*, 2015, 2016; Diaz-Uriarte and Vasallo, 2019). The objective of CPMs is to infer these constraints. CPMs

assume that events are gained one by one (no simultaneous acquisition of events) and that there is no back mutation so that once gained an event is not lost; CPMs also assume that the events that drive the process (driver genes if we are thinking about cancer) are known and present in the data set. Finally, CPMs assume that all subjects start the evolutionary process without any of the studied events (i.e., all subjects start the process with 0s in the matrix of subjects by alterations). If we think about cancer, this means that "CPMs assume that all tumors start cancer progression without any of the mutations considered in the study (the above matrix of subjects by driver alterations), but other mutations could be present that have caused the initial tumor growth" (Diaz-Uriarte and Vasallo, 2019); these other additional mutations that lead to the initiation of the process are absorbed in the root node from which cancer starts (Attolini *et al.*, 2010).

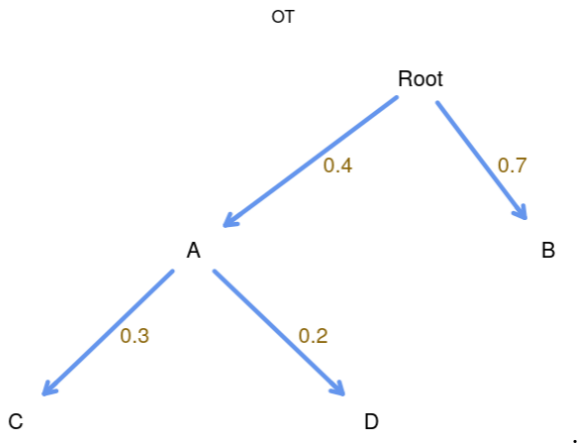
2.3 Cancer Progression Models (CPMs): details

2.3.1 Oncogenetic Trees (OT)

OTs are among the earliest formal models of accumulation of mutations in cancer. They were originally described in Desper *et al.* (1999) (see also Simon *et al.*, 2000; Radmacher *et al.*, 2001); additional references include Szabo and Boucher (2008); Szabo and Pappas (2022); Szabo and Boucher (2002). With OTs, restrictions in the accumulation of mutations (or events) are represented as a tree. Hence, a parent node can have many children, but children have a single parent: therefore, an event can only directly depend on another event. OTs are untimed models (in contrast to, for example CBN, explained in section 2.3.3): weights along edges (the π_{xy} we will use below) can be directly interpreted as probabilities of transition along the edges by the time of observation (Szabo and Boucher, 2008, p. 5). In other words, edge weights represent conditional probabilities of observing a given mutation, when the sample is taken, given the parents are observed.

As explained in (Szabo and Boucher, 2008, Definition 1, p. 4): "A pure untimed oncogenetic tree is a tree T with a probability $\pi(e)$ attached to each edge e. This tree generates observations on mutation presence/absence the following way: each edge e is independently retained with probability $\pi(e)$; the set of vertices that are still reachable from M_0 [the root of the tree, representing no alterations] gives the set of the observed genetic alterations."

To give an example, suppose a tree as follows:



In this tree, events (mutations) A and B can be acquired independently, and depend on no one (Root is M_0 in the notation of Szabo and Boucher, 2008). C and D depend on A (and are independent of each other, conditional on A). The parameters of the model, shown in brown, are:

- Probability of acquiring A, $\pi_{0A} = 0.4$; π_{0A} is the notation in Szabo and Boucher, 2008, and is the *weight* along the edge from M_0 to A.
- Probability of acquiring B, $\pi_{0B} = 0.7$.
- Probability of acquiring C, given A has already been acquired, $\pi_{AC} = 0.3$ (again, π_{AC} is the *weight* along the edge from A to C).
- Probability of acquiring D, given A has already been acquired, $\pi_{AD} = 0.2$.

According to the above model, the tumor develops as follows: starting from Root (or M_0), the tumor can gain A and B, and these are independent events. If A is gained, then the tumor can gain C and D, and these two are again independent events (once A has been gained). Therefore, the probabilities of the different genotypes or states of the tumor at the time of sampling are:

- Only Root or M_0, i.e., no events gained: $(1 - \pi_{0A})(1 - \pi_{0B})$.
- Only A occurs (i.e., genotype A): $\pi_{0A}(1 - \pi_{0B})(1 - \pi_{AC})(1 - \pi_{AD})$.
- Only B occures (i.e., we observe genotype B): $\pi_{0B}(1 - \pi_{0A})$.
- Both A and B (but no C or D), genotype AB: $\pi_{0A}\pi_{0B}(1 - \pi_{AC})(1 - \pi_{AD})$.
- A and C, genotype AC: $\pi_{0A}(1 - \pi_{0B})\pi_{AC}(1 - \pi_{AD})$.
- Both B and C but no A, genotype BC: 0 (as A needs to occur before C can occur).
- ...

The above describes the ideal scenario, without errors. OT includes a model for errors from different sources: deviations from the model (i.e., events that do not respect the pure untimed model above) and observational (e.g., genotyping) errors. Together, these two types of error cause false positive and false negative observational errors (ϵ_+, ϵ_-). These error rates are estimated by the OT algorithm and are incorporated in the computation of the predicted frequencies of genotypes according to OT (see details in *"CPMs: Error models", section 2.3.6*).

When using OT, as explained in Szabo and Boucher (p. 5 2008), the main objective is reconstructing the topology of the tree; the estimation of the edge probabilities (the weights or π_{xy}) and the error rates (ϵ_+, ϵ_-) is of secondary importance. As detailed in Szabo and Boucher (2008, p. 5), the estimation of the topology uses an "(...) algorithm [that] takes a greedy bottom-up approach: it assigns the parent of each node by finding the maximum-weight in-edge starting from the leaves." and that provides a computationally fast way of inferring the tree. The full algorithm for topology reconstruction is provided in Szabo and Boucher (2008, Section 3 and Fig. 2) (the algorithm is also provided in Figure 2 of file `ot.pdf`, part of the documentation of Szabo and Pappas, 2022); estimation of the weights is detailed in Szabo and Boucher (2008, p. 13). Sufficient conditions for the reconstruction of the true tree when there are false positive and false negative errors are given in Szabo and Boucher (2002) and sample size requirements in Szabo and Boucher (2008, p. 8)(see also Desper *et al.*, 1999).

2.3.2 OncoBN

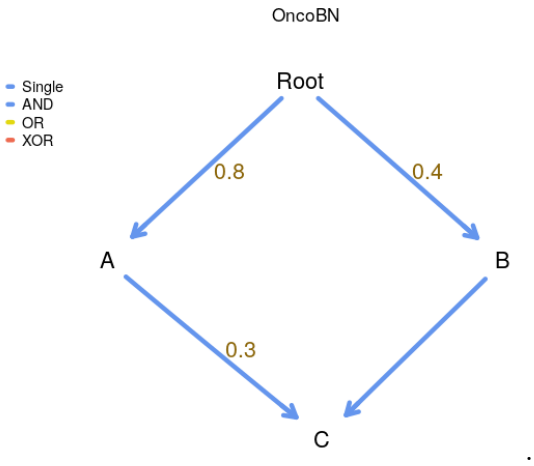
OncoBN, described in Nicol *et al.* (2021), is similar to OT in the sense of being an untimed oncogenetic model but, in contrast to OT, a node can have multiple parents. When there are multiple parents the relationships and models can be of two different kinds:

- disjunctive (OR relationship): the DBN, Disjunctive Bayesian Network model;
- conjunctive (AND relationship): the CBN, Conjunctive Bayesian Network model.

A given OncoBN be either a DBN or a CBN, but not both: it can have conjunctive or disjunctive relationships, but not both. (And note that the CBN models fitted by OncoBN are untimed, and thus the parameters do not have the same interpretation as the parameters of the CBN models discussed below, *"Conjuntive Bayesian Networks (CBN)" (section 2.3.3)*).

As explained in Nicol *et al.* (2021, p. 2), a key difference between the conjunctive (AND) and the disjunctive (OR) model is that under the conjunctive model all parent alterations that constitute the AND relationship must be present in a cell for the child mutation to occur; the disjunctive model, in contrast, allows child event to occur when just one of the parent events has taken place. According to the authors, this might make the model better for modeling intra-tumor heterogeneity.

The following DAG shows a conjunctive model fitted with OncoBN:

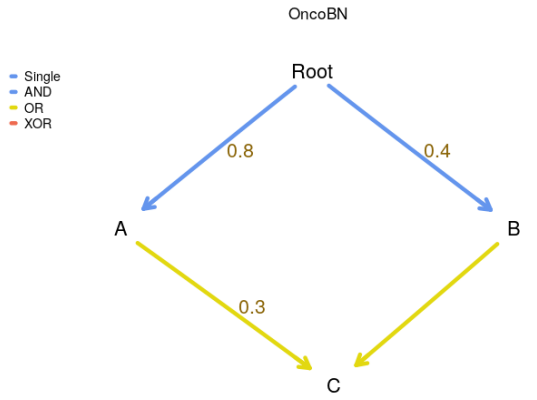


.

Note that the value of 0.3 is the value of the parameter θ_C: this is the conditional probability of C given its ancestors. (So, in contrast to OT, but similar to CBN and H-ESBCN, the parameters are not of edges, but of events). The values of θ are: $\theta_A = 0.8$, $\theta_B = 0.4$, $\theta_C = 0.3$. According to the OncoBN model the probabilities of some genotypes are:

- Only Root (i.e., only genotypes without any mutation, or "WT"): $(1 - \theta_A)(1 - \theta_B)$.
- Only A, i.e., genotype A: $\theta_A(1 - \theta_B)$.
- A and C, genotype AC: 0, since acquiring C requires also B.
- A and B (but not C), genotype AB: $\theta_A\theta_B(1 - \theta_C)$.
- All of A, B, C, genotype ABC: $\theta_A\theta_B\theta_C$.
- Only C: 0, since neither A nor B have occurred.

The next DAG is identical, except the model is a disjunctive one (notice the edges are OR edges):



.

Now, θ_C is the probability of C occurring if at least one of its ancestors has occurred. Therefore, we have the following probabilities of some of the genotypes, where those that differ from the conjunctive case have been marked in bold with an initial asterisk:

- Only Root (i.e., only genotypes without any mutation, or "WT") : $(1-\theta_A)(1-\theta_B)$.
- *** Only A:** $\theta_A(1-\theta_B)(1-\theta_C)$**.**
- *** A and C, genotype AC:** $\theta_A(1-\theta_B)\theta_C$**.**
- A and B (but not C), genotype AB: $\theta_A\theta_B(1-\theta_C)$.
- All of A, B, C, genotype ABC: $\theta_A\theta_B\theta_C$.
- Only C: 0, since neither A nor B have occurred.

The above represent the probabilities in a model without errors. OncoBN includes an error model, the "spontaneous activation model" where there is a non-zero probability of observing child events when restrictions in the DAG are not satisfied. The rate of spontaneous activation is part of the estimation procedure, and is included in the computed probabilities of the different genotypes (see details in section *2.3.6, "CPMs: Error models"*). For example, under the disjunctive model above, the probability of observing genotype C would be $(1-\theta_A)(1-\theta_B)\epsilon$, where ϵ is the spontaneous activation probability, which is set as the same for all events (Nicol *et al.*, 2021, p. 5)). (Figure 1 of Nicol *et al.*, 2021 provides another example of the role of ϵ in computing predicted probabilities).

Structure learning can be conducted using an exact procedure that uses dynamic programming, recommended when there are less than 30 events; for larger problems, approximate structure learning using genetic programming is available.

2.3.3 Conjuntive Bayesian Networks (CBN)

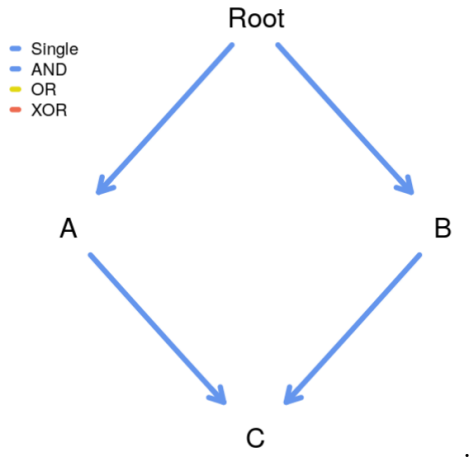
In terms of the representation of the restrictions, CBN, like OncoBN, generalizes the tree-based restriction of OT to a directed acyclic graph (DAG): a node can have multiple parents. A node with multiple parents means that all of the parents have to be present (all of the parent events must have occurred) for the children to appear; therefore, relationships are conjuntive —AND relationships between the parents (recall OncoBN can model AND and OR relationships). CBN also differs from OT and OncoBM because the CBN model is a timed model: the λs, the parameters of the models, are the rates of the exponentially distributed

times to fixation of an event given that all parents of that event have been observed (i.e., given that the event restrictions, as specified in the DAG, are satisfied: Montazeri *et al.*, 2016, p. i729; Gerstung *et al.*, 2009, section 2.2).

Specifically, T_i, the waiting time for event i to occur, is an exponentially distributed random variable with parameter λ_i conditioned on all the parent mutations, $\text{pa}(i)$, having occurred (Gerstung *et al.*, 2009); thus, T_i is defined recursively as (Gerstung *et al.*, 2009; Hosseini *et al.*, 2019):

$$T_i \sim \text{Exp}(\lambda_i) + \max_{j \in \text{pa}(i)} T_j \tag{1}$$

To give an example, suppose a DAG as follows:



Then, the time to fixation of the three mutations (not genotypes) are:

- $T_A \sim \text{Exp}(\lambda_A)$
- $T_B \sim \text{Exp}(\lambda_B)$
- $T_C \sim \text{Exp}(\lambda_C) + \max(T_A, T_B)$

and we will not observe C unless both A and B have occurred.

The λ parameters of the CBN model define the transition rate matrix between genotypes (see also Montazeri *et al.*, 2016). For the example above we have:

- Rate from WT to genotype with A mutated: λ_A.
- Rate from WT to genotype with B mutated: λ_B.
- Rate from genotype with A mutated to genotype with both A and B mutated: λ_B.
- Rate from genotype with B mutated to genotype with both A and B mutated: λ_A.
- Rate from genotype with A and B mutated to genotype with A, B, C mutated: λ_C.

In other words, this is the transition rate matrix, where only genotypes that can appear are shown (i.e., genotypes C, AC, and BC are not shown):

$$Q = \begin{array}{c} \\ WT \\ A \\ B \\ AB \\ ABC \end{array} \begin{array}{c} \begin{array}{ccccc} WT & A & B & AB & ABC \end{array} \\ \begin{pmatrix} -(\lambda_A + \lambda_B) & \lambda_A & \lambda_B & 0 & 0 \\ 0 & -\lambda_B & 0 & \lambda_B & 0 \\ 0 & 0 & -\lambda_A & -\lambda_A & 0 \\ 0 & 0 & 0 & -\lambda_C & \lambda_C \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \tag{2}$$

For parameter estimation, and since the observation times of the different individuals are unknown, it is assumed that observation time is exponentially distributed with parameter 1 (the probabilities of observing the different events are invariant under rescalings of the λ_i and the λ_s, the rate of the time to observation — Gerstung *et al.*, 2009).

In EvAM-Tools we include two versions of CBN that differ in the algorithm (and, thus, in speed and in how many observations can be analyzed) and in the error model: H-CBN, described in Gerstung *et al.* (2009, 2011), and MC-CBN, described in Montazeri *et al.* (2016). Unless qualified otherwise (i.e., saying MC-CBN), when we say "CBN" we refer to H-CBN[1].

H-CBN uses simulated annealing with a nested expectation-maximization (EM) algorithm for estimation: structure —DAG— learning is conducted with simulated annealing and parameters (λs and ϵ —the error term; see next) are estimated using the EM algorithm (section 2.3, pp. 2810–2811 of Gerstung *et al.*, 2009). As is the case for other methods, the key focus of the algorithm is inferring the DAG of restrictions (the poset); the selected DAG (poset) is the maximum likelihood one "(...) without additional model selection criterion such as the Akaike or Bayesian information criterion (AIC and BIC, respectively)" (Gerstung *et al.*, 2009, p. 2811). Briefly, the algorithm first finds the maximum likelihood estimates for λs and ϵ of a given poset; a new poset is then generated from the previous one (after addition/removal of relations from the poset), the maximum likelihood estimates of λs and ϵ computed for this new poset, and the new poset is accepted if its likelihood is larger or, if smaller, it is accepted with a probability that is a function the difference in likelihoods divided by the temperature (recall they use a simulated annealing algorithm). MC-CBN uses a Monte-Carlo EM algorithm (see Montazeri *et al.*, 2016, p. i731 for network —DAG— learning and p. i730 and Algorithm 1 in p. i731 for parameter estimation).

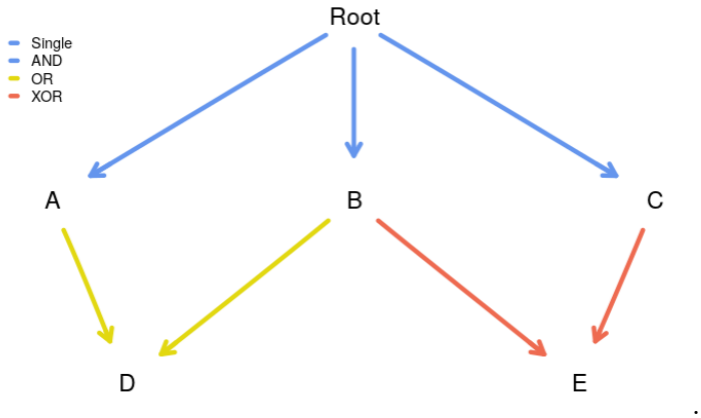
H-CBN and MC-CBN also differ in their error models. In H-CBN the λs describe the true underlying model that produces the true, hidden genotypes, but the observed genotypes might differ from the true ones because of observation error; the observation error is a Bernoulli process, in which a mutation is falsely observed with probability ϵ, which is assumed to be the same and independent across all sites (see also Sakoparnig and Beerenwinkel, 2012, p. 2319). In MC-CBN the model is a mixture between the CBN model and a noise component model, such as the independence model provided by a DAG where all mutations are direct descendants of the root (the empty poset; see details in Montazeri *et al.*, 2016, p. i731). The error models are, of course, part of the fitting algorithm.

2.3.4 Hidden Extended Suppes-Bayes Causal Networks (H-ESBCN)

H-ESBCN (Hidden Extended Suppes-Bayes Causal Networks), described in Angaroni *et al.* (2021) (and used by its authors as part of Progression Models of Cancer Evolution, PMCE), is similar to CBN in that it is a timed model, where parameters of the model, the λs are the rates of the exponentially distributed times to fixation of an event given that the parents of that event have been observed. In contrast to CBN, the dependency relationships are not limited to AND, and they can include OR and XOR. In contrast to OncoBN with respect to dependencies, H-ESBCN adds XOR relationships, but H-ESBCN allows the very same model to include AND, OR, and XOR relationships; the fitting algorithm includes automatic inference of logical formulas for these three different patterns, AND, OR, XOR.

To give an example, suppose the following DAG (we only show XOR and OR relationships, since we have already shown AND relationships in examples above, and there is nothing new with AND relationships); this example is discused, in another context, in *"An example with OR and XOR" (section 7.4)*:

[1]Note that, by default, MC-CBN is not selected as a method to be used in the web app because it is often much slower than any of the remaining methods.



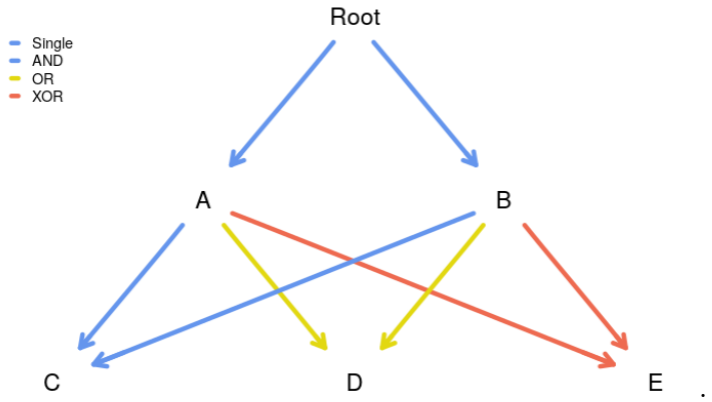
According to this DAG

- A, B, C depend on none, and their rates are, respectively, $\lambda_A, \lambda_B, \lambda_C$.
- D depends, with an OR, on both A and B: the rate of fixation of D given at least one of A or B have occurred is λ_D. Thus, we can observe genotypes AD, BD, ABD.
- E depends, with an XOR, on B and C: the rate of occurrence of E given exactly one of B XOR C has occurred is λ_E. Thus, E can only be observed in genotypes that show B XOR C, such as genotypes BE, CE, ABE, ACE; genotypes BCE or $ABCE$, in contrast, are not allowed because those genotypes have both B and C mutated.

The transition rate matrix between genotypes is shown below, where rows are origin, column destination (i.e., entries of Q_{xy} are the transition rates from x to y):

	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																					

Note that it is possible to have two (or more) parents to have dependents with different relationships. This, for example, is one of the pre-loaded DAGs in EvAM-Tools:



The error model is similar to the one of CBN *"Conjuntive Bayesian Networks (CBN)" (section 2.3.3)*, as described in Gerstung *et al.* (2009); Sakoparnig and Beerenwinkel (2012); see also *"CPMs: Error models" (section 2.3.6)*. The fitting algorithm is described in Angaroni *et al.* (2021, Sections 2.1, 2.2, pp. 756 and 757). As for other methods, its main focus is inferring the structure of the DAG, in this case the maximum a posteriori one in the framework of Suppe's probabilistic causation. A key feature of the algorithm is the attempt to automatically detect the correct logical formula (AND, OR, XOR) for the dependency. The structure searching algorithm uses MCMC from a randomly initialized structure which is modified according to eight different possible moves (Angaroni *et al.*, 2021, p. 756). To avoid fitting unneeded logic formulas, the structure learning algorithm includes regularization, which can be chosen by the user to be AIC or BIC. Estimation of the λs (and error rate) for a fixed DAG structure is then done using an EM algorithm (Angaroni *et al.*, 2021, p.757).

2.3.5 Mutual Hazard networks (MHN)

All of the methods described above share a model of deterministic dependencies for the accumulation of events (or mutations) (Schill *et al.*, 2020): an event (a mutation) can only occur if its dependencies are satisfied (though note that both OT and OncoBN, as well as MB-CBN allow for error deviations from this requirement — see *"CPMs: Error models", section 2.3.6*).

In contrast to the previous methods, with MHN (Schill *et al.*, 2020) dependencies are not deterministic and events can make other events more likely (promoting influence) or less likely (inhibiting influence). The rate of occurrence of events is modeled by a spontaneous rate of fixation and a multiplicative effect that each of these events can have on other events via pairwise interactions; these pairwise interactions are what allow MHN to model both promoting and inhibiting dependencies.

In more detail, the Markov process that governs the transition from a genotype $\mathbf{x}$ to a genotype with mutation i added to genotype $\mathbf{x}$ is specified by (Schill *et al.*, 2020, eq. 2):

$$Q_{\mathbf{x}_{+i},\mathbf{x}} = \Theta_{ii} \prod_{x_j=1} \Theta_{ij} \tag{3}$$

where x_j is 1 if gene j is already mutated in genotype $\mathbf{x}$, and $Q_{y,x}$ is the transition rate from x to y (we are using the notation in Schill *et al.*, 2020, where transition rate matrices are transposed relative to the notation in Montazeri *et al.*, 2016 that we have used when describing CBN and H-ESBCN). Θ_{ii} is the baseline hazard or the rate of i before any other

events; Θ_{ij} is the multiplicative effect of event j on the rate of event i. Therefore, equation 3 shows the transition rate as the product of the baseline hazard times the multiplicative effects of all the other mutated genes or events, j, on i.
To give a specific example, suppose the Θ matrix for a three-gene model[2] is:

$$\Theta = \begin{bmatrix} \Theta_{11} & \Theta_{12} & \Theta_{13} \\ \Theta_{21} & \Theta_{22} & \Theta_{23} \\ \Theta_{31} & \Theta_{32} & \Theta_{33} \end{bmatrix} \tag{4}$$

The following are the transition rates for some transitions:

- From WT to the genotype with the first event or gene: Θ_{11}.
- From the genotype with the first event to the genotype with the first and the second events: $\Theta_{22}\Theta_{21}$.
- From the genotype with the first event and second events to the genotype with the third event: $\Theta_{33}(\Theta_{31}\Theta_{32})$.

Note that in EvAM-Tools we show the log-Θ matrix, the matrix of θ_{ij}, where $\Theta_{ij} = e^{\theta_{ij}}$, because this makes it immediate to identify the inhibiting relationships as those with a negative sign, and it symmetrizes the effects around 0.
As can be seen, the relationships between events are inhibiting (event j inhibits event i if $\Theta_{ij} < 1$ or, equivalently, $\theta_{ij} < 0$) or promoting ($\Theta_{ij} > 1$ or, equivalently, $\theta_{ij} > 0$), but there are no deterministic restrictions (although MHN can be seen as a stochastic approximation to the deterministic dependencies of CBN: see the supplementary material of Schill *et al.*, 2020).
To fit the model, because observation time is unknown, and as is done by Gerstung *et al.* (2009), the authors assume that observation times are exponentially distributed with parameter 1. To prevent overfitting, the model fitting procedure maximizes the likelihood of the data minus an L1 penalty to try to avoid many interacting events (i.e., to promote sparsity of the fitted models): it uses a tunning parameter, λ[3] that multiplies the sum of the absolute values of the off-diagonal entries of the log Θ matrix (Schill *et al.*, 2020, eq. 6). The default value of λ is 1/number of rows of the data set. The authors provide an efficient implementation of their method that uses a Quasi-Newton algorithm.

2.3.6 CPMs: Error models

We have mentioned error models when describing each procedure. We put together those details here, to allow for easier understanding of the similarities and differences between methods. (Methods are not ordered as above but, rather, by increasing complexity of the error model).

MHN There is no explicit error model (the simulation process described in p. 244 of Schill *et al.*, 2020 uses a scheme as the one in CBN, Gerstung *et al.*, 2009, explained below, but that is not part of the MHN model itself).

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). The observation error is a Bernoulli process, in which a mutation is falsely observed with probability ϵ, which is assumed to be the same and independent across all sites (see

[2] As a different example, see the set of transitions for a four-gene example in Schill *et al.*, 2020, Fig. 2.
[3] This λ is different from the λs of CBN and H-ESBCN

also Sakoparnig and Beerenwinkel, 2012, p. 2319); in other words, for all events of all subjects in the sample, if the true observation is a 0, it has a probability of being observed as a 1 of ϵ, and similarly for an observation that is truly a 0.

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

MC-CBN With MC-CBN the model is a mixture between the CBN model and a noise component model, such as the independence model provided by a DAG where all mutations are direct descendants of the root (the empty poset; see details in 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`), as for CBN above.

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). (This ϵ, therefore, has a different meaning from the ϵ of CBN and H-ESBCN).

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, 2008). 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` (ϵ_+), and false negative, `eneg` (ϵ_-), 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` (ϵ_-), and `epos` (ϵ_+), whereas true deviations from the model are only reflected in `epos` (ϵ_+). 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).

So, when obtaining predicted frequencies under the model, for CBN, H-ESBCN, and MHN, we assume perfect compliance with the model; symmetric noise (e.g., genotyping noise) is added only when obtaining finite samples from the model. For OT and OncoBN the predicted frequencies from the model already include deviations from the fitted model.

2.3.7 CPMs: output

All methods provide directly, as output, estimates of the key constituents of their models, in particular:

OT Tree of restrictions, edge weights (πs), errors (ϵ_+, ϵ_-).

OncoBN DAG of restrictions, event θ, spontaneous activation probability or error (ϵ). (The type of model, conjunctive or disjunctive, is not estimated, but set by the user).

CBN DAG of restrictions, λs, error rate.

H-ESBCN DAG of restrictions, including type of restriction (AND, OR, XOR), λs, error rate.

MHN Θ matrix (or its equivalent θ or log-Θ matrices).

In addition, directly derived predictions, such as **predicted probabilities of genotypes** are provided by the original code/implementation (e.g., for OT, OncoBN, MHN) or can be obtained for CBN and H-ESBCN from the transition rate matrices (see details in *"Predicted genotype frequencies", section 3*). From the predicted probabilities of genotypes we can obtain **finite sampled genotype counts**, as explained in *"Error models and obtaining finite samples (or sampled genotype counts)" (section 5.2)*.
Transition rate matrices themselves are not part of the immediate output of any of the methods (except MHN[4]) but, as explained in *"Predicted genotype frequencies" (section 3)*, can be obtained from the DAG and the λs, as we do in EvAM-Tools; we have already seen examples of the transition rate matrices for all of CBN (*"Conjuntive Bayesian Networks (CBN)", section 2.3.3*), H-ESBCN (*"Hidden Extended Suppes-Bayes Causal Networks (H-ESBCN)", section 2.3.4*), and MHN (*"Mutual Hazard networks (MHN)", section 2.3.5*).
From the transition rate matrices it is also possible to obtain the **probabilities of evolutionary paths** and the **transition probabilities** (for instance, using competing exponentials; see references and details in section *4, "Probabilities of evolutionary paths and transition probabilities"*).
All of this output is available from EvAM-Tools, and the web app shows most of them using both figures and tables. (Probabilities of evolutionary paths, even if asked to be computed, are not explicitly available from the web app, as they can be unwieldy to display; they are provided in the output one can download and are, of course, implicit from the transition probabilities between genotypes, and transition probabilities are displayed in the web app.)

2.3.8 CPMs: summary

The following table provides a summary of the main features of each method.

[4] And we saw an example in *"Mutual Hazard networks (MHN)" (section 2.3.5)*

Method	Timed/ untimed	Number and type of dependencies	Restrictions	Representation and output
OT	Untimed	Single	Deterministic	Tree with edge weights (πs)
OncoBN	Untimed	AND (CBN version), OR (DBN version); but a given model can only contain either AND xor OR, not both	Deterministic	DAG with event thetas (θs)
CBN	Timed	AND	Deterministic	DAG with event rates (λs)
H-ESBCN	Timed	AND, OR, XOR	Deterministic	DAG with event rates (λs)
MHN	Timed	Promoting and inhibiting (but only pairwise interactions)	Stochastic dependencies	Θ matrix (diagonal entries: baseline hazards; off-diagonal: multiplicative effects).

3 Predicted genotype frequencies

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

Briefly, 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); we haven seen examples for all these methods in section *"Cancer Progression Models (CPMs): details" (section 2.3)*. 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)[5].

In more detail, 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 *7, "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):

$$\mathbf{p} = \int_0^\infty dt\ e^{-t}\ e^{tQ}\ \mathbf{p}_0 = [I - Q]^{-1}\mathbf{p}_0 \tag{5}$$

where $\mathbf{p}_0$ is the initial distribution (i.e., 1 for WT and 0 for the rest of the genotypes), t is the time of observation (again, assumed to be exponentially distributed with parameter 1), and Q is the transition rate matrix (beware: written here, as in Schill *et al.*, 2020, with Q_{ij} meaning the transition rate from j to i, in contrast to our expressions for transition rate matrices in equation 2 or the transition rate matrix in section 2.3.4). This is implemented in

[5]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.

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`).

3.2 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; these expressions incorporate, when predicting the genotypes, the estimated errors (ϵ_+, ϵ_- for OT, ϵ for OncoBN; see section *"CPMs: Error models", section 2.3.6*). 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`[6].
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 details in section *5.2, "Error models and obtaining finite samples (or sampled genotype counts)"*.

4 Probabilities of evolutionary paths and transition probabilities

How to obtain probabilities of evolutionary paths for CBN and OT 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 how to obtain 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).
The procedures to obtain transition probabilities and probabilities of evolutionary paths for H-ESBCN and MHN are similar to CBN: in all these methods we obtain probabilities of paths from the transition matrix, which is itself obtained from the transition rate matrix. The procedure with OncoBN is analogous to the one used with OT, both being untimed models (and, in both cases, obtaining probabilities of paths, as discussed in Diaz-Uriarte and Vasallo, 2019 is an abuse of the untimed model).

5 Generating random CPM/EvAM models, obtaining finite samples from them, and error models

5.1 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.

[6]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`

- 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 3*).
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 `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 `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; see *"Predicted genotype frequencies" (section 3)*.

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

   As the output from `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`.

5.2 Error models and obtaining finite samples (or sampled genotype counts)

When obtaining a finite sample from a model, in all cases except OT, we have always followed the same procedure: we have first generated the predicted genotype frequencies under the model and, if requested, then added observational (e.g., genotyping) noise to the finite samples obtained from the predicted frequencies[7]. Recall that, for OncoBN, the fitted model (and, thus, the predicted frequencies under the model) already include deviations from the model, as measured by ϵ; see section *2.3.6, "CPMs: Error models"* and section *2.3.2, "OncoBN"*.

For OT, since `epos` (ϵ_+) reflects both observation error and true deviations from the model, the above procedure is not possible. We need to introduce a difference between sampling from a model specified from scratch, such as a random model returned from function `random_evam`, and sampling from the predictions of a fitted model.

The fitted model for OT, when fitting a true data set, includes both `epos` (ϵ_+) and `eneg` (ϵ_-). Predicted genotype frequencies 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''`.[8]

When sampling from a model specified from scratch, such as a random model returned from function `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 (as for the CBN model —section *2.3.6, "CPMs: Error models"*). 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, it is sensible to split errors into two distinct pieces, which is also coherent with

[7] Obtaining a finite sample of size N given a vector of relative frequencies is done in R using the function `sample`

[8] 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 `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` $= y$.

what we do with the rest of the methods: deviations from the model, and noise.

6 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.

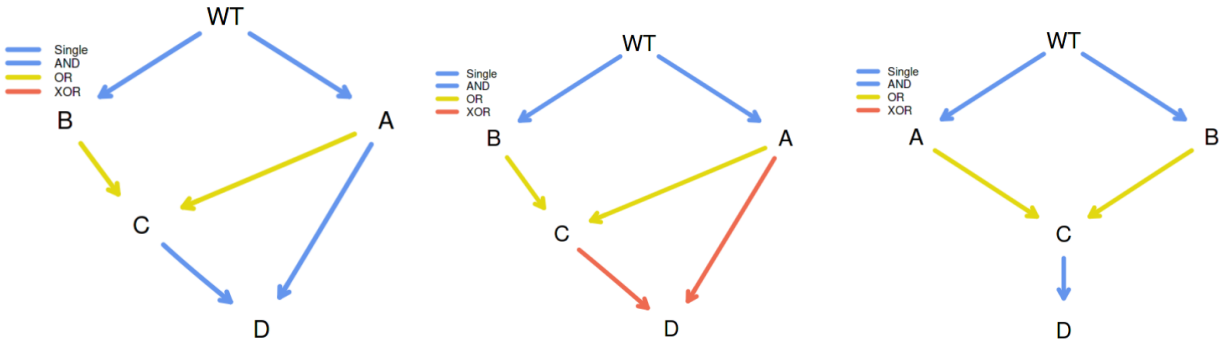


Figure 1: Non-transitively reduced DAG, OR and AND (left), non-transitively reduced DAG, OR and XOR (center), transitively reduced DAG.

- 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.

7 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 turned this into a specific section so as not to break the flow of the former sections.

7.1 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.

7.2 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.

7.3 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 *3.1, "Predicted genotype frequencies for CBN, MCCBN, MHN, H-ESBCN"*.

7.4 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".

(This example is also shown in section *2.3.4, "Hidden Extended Suppes-Bayes Causal Networks (H-ESBCN)"*).

	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																					

7.5 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 set to these tiny values 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:

        * $WT \to A \to AD$
        * $WT \to B \to BD$
        * 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:
        * $WT \to A \to AB \to ABD$
        * $WT \to B \to AB \to ABD$
        * 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:
        * $WT \to A \to AB \to ABD$
        * $WT \to B \to AB \to ABD$
        * $WT \to A \to AD \to ABD$
        * $WT \to B \to BD \to ABD$
        * 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:
    - $WT \to A \to AC$
    - $WT \to B \to BC$
    - 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.
    - $WT \to A \to AC \to ACD$
    - $WT \to B \to BC \to BCD$
    - $WT \to A \to AD \to ACD$
    - $WT \to B \to BD \to BCD$
    - 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.

7.6 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.

8 FAQ

8.1 Web app, figures

8.1.1 In the figures, some times I get the error "Figure margins too large"

Solution: try to reduce the length of the gene names.
Longer explanation: It is impossible to accommodate, automatically, all possible use cases in terms of length of genotypes (e.g., you analyzed a data set with 10 genes, and some have names that are many characters long). We try to catch mistakes, but we might have missed some.

8.1.2 In the figures, some times genotype names are truncated

This problem is related to the previous one: with very large genotype or gene names, sometimes the only way to prevent the "Figure margins too large" error is to make figure margins smaller, which can result in truncation.

8.1.3 In the figures, some times the histograms are too tiny

Similar to previous problems: genotype or gene names are probably too large, so to accommodate them we need to make the rest of the plot smaller.

8.2 Web app, saved output

8.2.1 When data are saved, genes without mutations are excluded

When creating user data (for instance, when adding new genotypes), any gene that has no mutations is automatically excluded from the saved 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).

8.3 Web app: could we reduce the number of required clicks?

This issue was raised by one reviewer: Editing values inside the app typically must be confirmed by an additional button press or key combination. It would be more convenient if values updated automatically after pressing Enter or switching the input field.
We have taken the liberty of adding it to the FAQ because it clarifies our design decisions and provides additional information about the behavior of the GUI.
We have tried to minimize additional button presses. Below we provide a description of the current behavior, with detailed explanations of the reasons for the behavior. There are few remaining cases where additional button presses or key combinations could be avoided.

- "Set the number of genes": moving the slider has an immediate effect.
  - For "Enter genotype frequencies manually", new gene names are immediately added to "Mutations" in the "Add genotypes" box.
  - For DAG, new gene names are immediately added to the "To" and "From" lists under 1. Define DAG, "New edge".
  - For MHN, it immediately resizes the log-Θ matrix; if data have already been generated, and to prevent the data and the log-Θ matrix from being in

inconsistent states, on every change of number of genes a new data set is generated.

  - (This is not applicable to "Upload file": setting the number of genes is not available here.)

- "Use different gene names" requires clicking on "Use these gene names" (on the popup box that is opened on clicking on "Use different gene names"). This is on purpose; first, forcing the change of gene names on switching input fields could lead to disconcerting behavior especially because the renaming is often an operation of renaming the complete set of genes, not just one of them; moreover, we try to convey that using this option carelessly will lead to confusion (several warnings are provided to minimize this careless use: on the box itself and on the tooltip). If users decide they do not want to use different names after all, they can abort the operation by clicking on "Dismiss".

  Since this option will be used sparingly and consciously, we think forcing explicit clicks and not renaming on switching input field is the appropriate behavior.

- When uploading a file (under "Upload file"), there is no need to click on additional buttons after entering a name in "Name for data". The user enters a string for the name, and then clicks on "Load data"; the entered string will become the name of the data when the upload is finished (and that data, with that name, will be shown on the left side, under "Examples and user's data").

- For DAG modification/creation, it is necessary to click on "Add edge" or "Remove edge" after selecting the "From" and "To" nodes. The alternative (adding a non-existent edge or removing an existing edge as soon as two nodes are selected), even if it removes one click, gives rise to non-obvious behavior that can be hard to understand. "Add/Remove edge" require an extra click but this extra click makes the behavior clear and explicit, and allow for correcting the From/To nodes before changing the DAG. Moreover, because we have separate buttons for "Add edge" and "Remove edge", user errors such as trying to remove an edge that does not exist, or trying to add an edge that already exists, are much easier for users to understand: when the error message pops-up, the clicked button ("Add edge", "Remove edge") is still colored gray.

- For DAGs, when changing entries in the DAG table, as soon as "Ctrl + Enter" is pressed, new values of the data are generated according to the new parameters, without any need to press "Generate data from DAG model".

  (Several parameters and/or relations can be changed without clicking "Ctrl + Enter" until all changes have been made: we move between entries of the table with Tab and "Shift + Tab", and click "Ctrl + Enter" only at the end).

- For MHN, if a data set has been generated previously, when changing entries in the MHN table, as soon as "Ctrl + Enter" is pressed, new values of the data are generated according to the new parameters, without any need to press "Generate data from MHN model". (As above, several entries of the table can be changed without clicking "Ctrl + Enter" until all changes have been made; we can move between cells of the table with Tab, and click "Ctrl + Enter" only at the end).

  Why generate new data for MHN on log-Θ matrix modification only if data had been previously generated, but generate it immediately for DAGs even if data had not been modified?

  - We, and the users in our lab, seem to build DAG models step by step, and seem to appreciate the immediate feedback from adding/removing edges, changing the type of relationship, or modifying λs or conditional probabilities.
  - We, and the user in our lab, seem to build MHNs by first thinking about a pattern of relationships that involves more than one entry of the log-Θ matrix. Once the initial model is specified, "Generate data" is clicked. After that has happened, and to avoid the data and the log-Θ matrix from being in inconsistent states, we always force a resampling of data when an entry of the matrix is changed.
  - The difference in behavior would, at most, involve one extra click with MHN.

- For DAGs, when changing the "Type of model" (OT, OncoBN, CBN/H-ESBCN), if a data set has been generated previously, new data are immediately generated, without any need to press "Generate data from MHN model". This both saves one click and prevents the model and the data from possibly being in inconsistent states.

  If no data have been generated we do not generate data. Why? For reasons similar to above. In our experience, if there is no data present, users are likely to change the model and then start modifying the DAG (adding/removing edges; changing type of relationships; changing parameters). It seems reasonable to delay the sampling until the sampling becomes necessary to prevent any possibly ambiguities or inconsistencies.

- For both DAG and MHN, switching from the input fields of "Number of genotypes to sample", "Observational noise", and "epos, ϵ" (only for DAG) will not lead to generating new data unless "Generate data from DAG/MHN" is clicked. In our experience, these three parameters are often modified together; updating the data immediately after switching from the input field leads to intermediate data updates that get in the way of "modify this set of parameters, and then update according to the new set". Moreover, the operation could be slightly expensive, computationally, if "Number of genotypes to sample" is a very large number.

  Note, however, that we have changed the behavior of the app, so that changes to settings of "Number of genotypes to sample" and "Observational noise" are now preserved and they are common to MHN and DAG. We think this will minimize needing to repeatedly change them to the desired settings (as we think it is reasonable that if a user is, say, using a sample size of 10000, this setting will be desired for both DAGs and MHN); this also prevents having to set them again to the desired value after, say, moving to "Upload file" and back.

- When genotype data are modified (under "Change genotype's counts", and with the same behavior in "Upload file", "Enter genotype frequencies manually", "DAG", "MHN"), as soon as the user enters "Ctrl + Enter", the histogram displaying genotype frequencies is updated.

  Note, therefore, that a user can choose to modify the histogram with every change of a genotype, or modify several genotypes without clicking "Ctrl + Enter" until the end: modify the number, move with Tab to the next, modify, move with Tab to the next, etc, and only update the histogram when all modifications have been done with a single "Ctrl + Enter".

- The "Rename the data" box requires entering a name and then clicking on the button "Rename the data". This is also on purpose: we think renaming the data should be a very conscious action, and users will hopefully notice that, as soon as the "Rename the data" button is clicked, that name appears on the left side, with a blue button

denoting it is the current, selected one on which modifications are being made. This should also prevent unwanted proliferation of data names that the user is not fully aware of having explicitly created. Asking for a click here, instead of renaming immediately after switching away from the input field, therefore, seems reasonable.

- Options under "Advanced options": options are set just by changing them, without any additional clicks. For example, the "Number of MCMC iterations" (under "H-ESBCN options") can be changed from the default 200000 to, say, 500000 just by deleting the 2 and putting a 5 without additional clicks. Likewise, changing the "Model" under "OncoBN options" requires clicking on the down arrow of the pull down menu and clicking on "Conjunctive", without additional clicks.

- In the "Output" tab, virtually all operations do not require any additional clicks and are executed immediately. "CPMs to show" does not require extra clicks, but has a small lag of about 0.9 seconds from the first click: redrawing is a potentially expensive operation, and we do not start it until giving some reasonable time for the user to click/unclick methods to show. "Download CPM results and analyzed data" opens the standard popup box that allows to change the name and then asks for clicking on "Save".

In summary, thus, in most cases, few or none additional button presses are needed. We think that those that remain fall into the following cases:

- The additional button or key press is unavoidable becaus of the way Shiny works.

- Not requiring this additional button or key press could lead to surprising and hard to understand behavior.

- Potentially expensive operations that we only want to execute when the user is done changing values.

8.4 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.

8.5 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. There is an example of this in the additional examples.

8.6 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.

8.7 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 *"CPMs: Error models" (section 2.3.6)*.
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 obtaining finite samples (or sampled genotype counts)" (section 5.2)*.

8.8 Docker and setting up your own Shiny app

8.8.1 I want to setup my own Shiny app 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.

8.8.2 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.

8.8.3 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.

8.8.4 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 8.8.2) 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.

8.8.5 I want to build 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/).

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.

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.

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

10 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.

Attolini, C., Cheng, Y., Beroukhim, R., Getz, G., Abdel-Wahab, O., Levine, R. L., Mellinghoff, I. K., and Michor, F. (2010). A mathematical framework to determine the temporal sequence of somatic genetic events in cancer. *Proceedings of the National Academy of Sciences*, **107**(41), 17604–17609.

Beerenwinkel, N., Schwarz, R. F., Gerstung, M., and Markowetz, F. (2015). Cancer evolution: Mathematical models and computational inference. *Systematic Biology*, **64**(1), e1–e25.

Beerenwinkel, N., Greenman, C. D., and Lagergren, J. (2016). Computational cancer biology: An evolutionary perspective. *PLoS Comput. Biol.*, **12**(2), e1004717.

Desper, R., Jiang, F., Kallioniemi, O.-P., Moch, H., Papadimitriou, C. H., and Schäffer, A. A. (1999). Inferring tree models for oncogenesis from comparative genome hybridization data. *Journal of computational biology*, **6**(1), 37–51.

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.

Gerstung, M., Eriksson, N., Lin, J., Vogelstein, B., and Beerenwinkel, N. (2011). The temporal order of genetic and pathway alterations in tumorigenesis. *PloS one*, **6**(11), e27136.

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.

Radmacher, M. D., Simon, R., Desper, R., Taetle, R., Schaffer, A. A., and Nelson, M. A. (2001). Graph Models of Oncogenesis with an Application to Melanoma. *Journal of Theoretical Biology*, **212**(4), 535–548.

Sakoparnig, T. and Beerenwinkel, N. (2012). Efficient sampling for Bayesian inference of conjunctive Bayesian networks. *Bioinformatics (Oxford, England)*, **28**(18), 2318–24.

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

Simon, R., Desper, R., Papadimitriou, C. H., Peng, A., Alberts, D. S., Taetle, R., Trent, J. M., and Schäffer, A. A. (2000). Chromosome abnormalities in ovarian adenocarcinoma: III. Using breakpoint data to infer and test mathematical models for oncogenesis. *Genes, Chromosomes and Cancer*, **28**(1), 106–120.

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.

Szabo, A. and Pappas, L. (2022). *Oncotree: Estimating Oncogenetic Trees*. R package version 0.3.4.

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

EvAM-Tools: examples

Ramon Diaz-Uriarte*

2022-10-01
Version a206824

Contents

*Department of Biochemistry, Universidad Autónoma de Madrid, Instituto de Investigaciones Biomédicas "Alberto Sols" (UAM-CSIC), Madrid, Spain. Author for correspondence: r.diaz@uam.es

1 Introduction

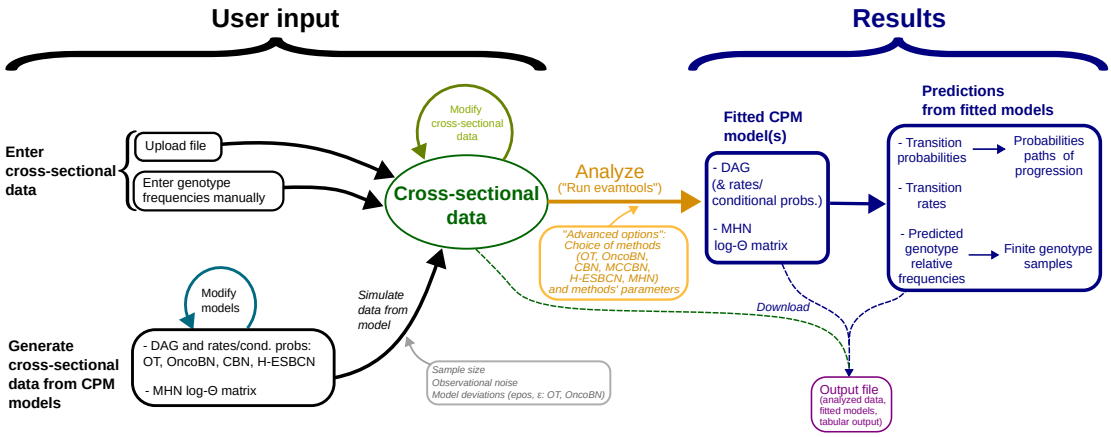
Here we present examples, with both real and simulated data, that illustrate the use and utility of EvAM-Tools. We will start with the analysis of one cancer data set, trying to understand the differences in the output of the different methods, and to do that we will also make use of flexibly modifying the genotype counts to run additional analyses. We will then show the analyses of a different cancer data set, which departs from the previous example in the patterns of differences and similarities between methods. Next, we use two short examples where we simulate data under a given model, and examine how different methods perform (whether or not they can recover the true signal). All the previous examples can be run in the web app, and that is what we use here. In the final section we discuss simulating random models, using the R package.

The objective of this document is not to provide complete analyses of any of the data sets, or address and of the questions mentioned above in full (that would require full papers). The objective is to illustrate the use of EvAM-Tools, especially its web app, and also to include some examples of output that, although not necessarily very common, are not unusual and can some surprising at first (e.g., the variability among fitted H-ESBCN models, in *"Is it just sample size?", section 2.2*, or output with DAGs that are not transitively reduced in *"A model with AND", section 4.3*).

1.1 Web app: overview of workflow and use cases, and relationship to these examples

On the web app landing page, under "About EvAM-tools" (https://www.iib.uam.es/evamtools/#overview) we provide an overview of the workflow and use cases. For completeness we repeat that material here, indicating, in bold, how the examples in this document relate to the major functionalities and workflows discussed there.

The figure below provides an overview of the major workflows with the web app:



The web app encompasses, thus, different major functionalities and use cases, mainly:

1. Inference of CPMs from user data uploaded from a file. **Examples *"Analyzing the BRCA data set" (section 2.1)* and *"Analyzing the ovarian CGH data" (section 2.3)*.**

2. Exploration of the inferences that different CPM methods yield from manually constructed synthetic data. **Example *"Is it just sample size?" (section 2.2)* uses a modification of uploaded data to explore changes in sample size; example *"Analyzing manually constructed synthetic data" (section 3)* constructs synthetic data to examine the effect of aliasing of events.**

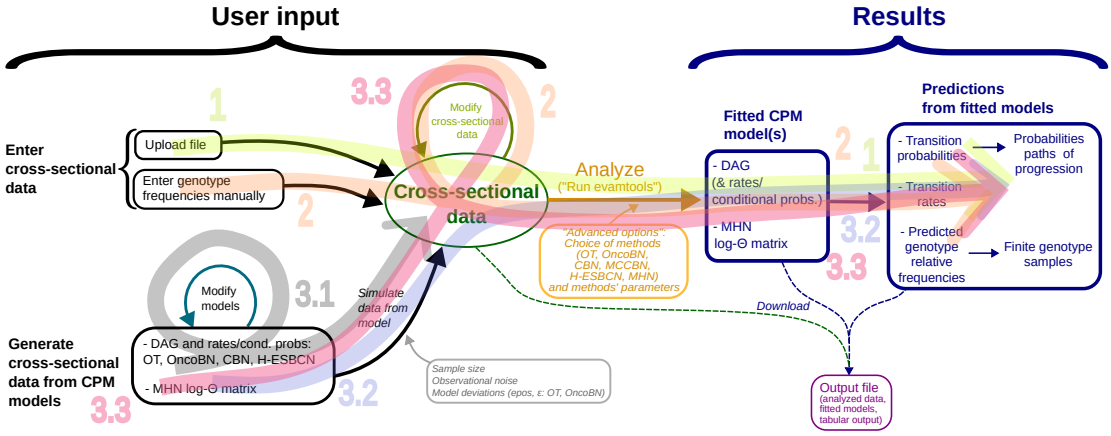
3. Construction of CPM models (DAGs with their rates/probabilities and MHN models) and simulation of synthetic data from them.

    3.1. Examination of the consequences of different CPM models and their parameters on the simulated data. **Examples** ***"What happens if we increase ϵ for OncoBN?" (section 4.1.1)***, ***"A simple exploration of MHN" (section 4.1.2)***.

    3.2. Analysis of the data simulated under one model with methods that have different models (e.g., data simulated from CBN analyzed with OT and OncoBN). **Examples** ***"A model with AND, XOR, OR" (section 4.2)*** **and** ***"A model with AND" (section 4.3)***.

    3.3. Analysis of the data simulated under one model after manual modification of specific genotype frequencies (e.g., data simulated under CBN but where, prior to analysis, we remove all observations with the WT genotype and the genotype with all loci mutated). **Example** ***"Modifying data generated from a CPM model before analysis" (section 4.4)***.

The figure below highlights the different major functionalities and workflows, as numbered above, over-imposed on the previous figure:



Furthermore, note that in all cases, when data are analyzed, in addition to returning the fitted models, the web app also returns the analysis of the CPMs in terms of their predictions such as predicted genotype frequencies and transition probabilities between genotypes. Most of the examples below illustrate this, showing, for example, the predicted genotype frequencies and transition probabilities.

1.2 Additional documentation

See additional documentation in https://rdiaz02.github.io/EvAM-Tools. In particular, additional technical documentation, with details about the models implemented, error models, predicted genotype frequencies, etc, is available from https://rdiaz02.github.io/EvAM-Tools/pdfs/Additional_tech_doc.pdf.

2 Analysis of cross-sectional data

We will analyze two cancer data sets, the ovarian cancer CGH data included in the Oncotree package (Szabo and Pappas, 2022), and the BRCA data set for basal-like subtypes (from Cerami *et al.*, 2012; Gao *et al.*, 2013, originally from Cancer Genome Atlas Research Network, 2012; see

Supplementary File S5_Text, https://doi.org/10.1371/journal.pcbi.1007246.s007 of Diaz-Uriarte and Vasallo, 2019 for full details about data origins and preprocessing).

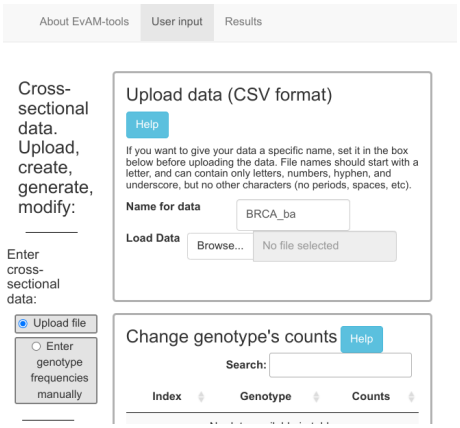
So that you can use these data sets directly, we provide them in the repository (https://github.com/rdiaz02/EvAM-Tools/tree/main/examples_for_upload). The direct links are:

- `BRCA_ba_s.csv`: https://raw.githubusercontent.com/rdiaz02/EvAM-Tools/main/examples_for_upload/BRCA_ba_s.csv
- `ov2.csv`: https://raw.githubusercontent.com/rdiaz02/EvAM-Tools/main/examples_for_upload/ov2.csv

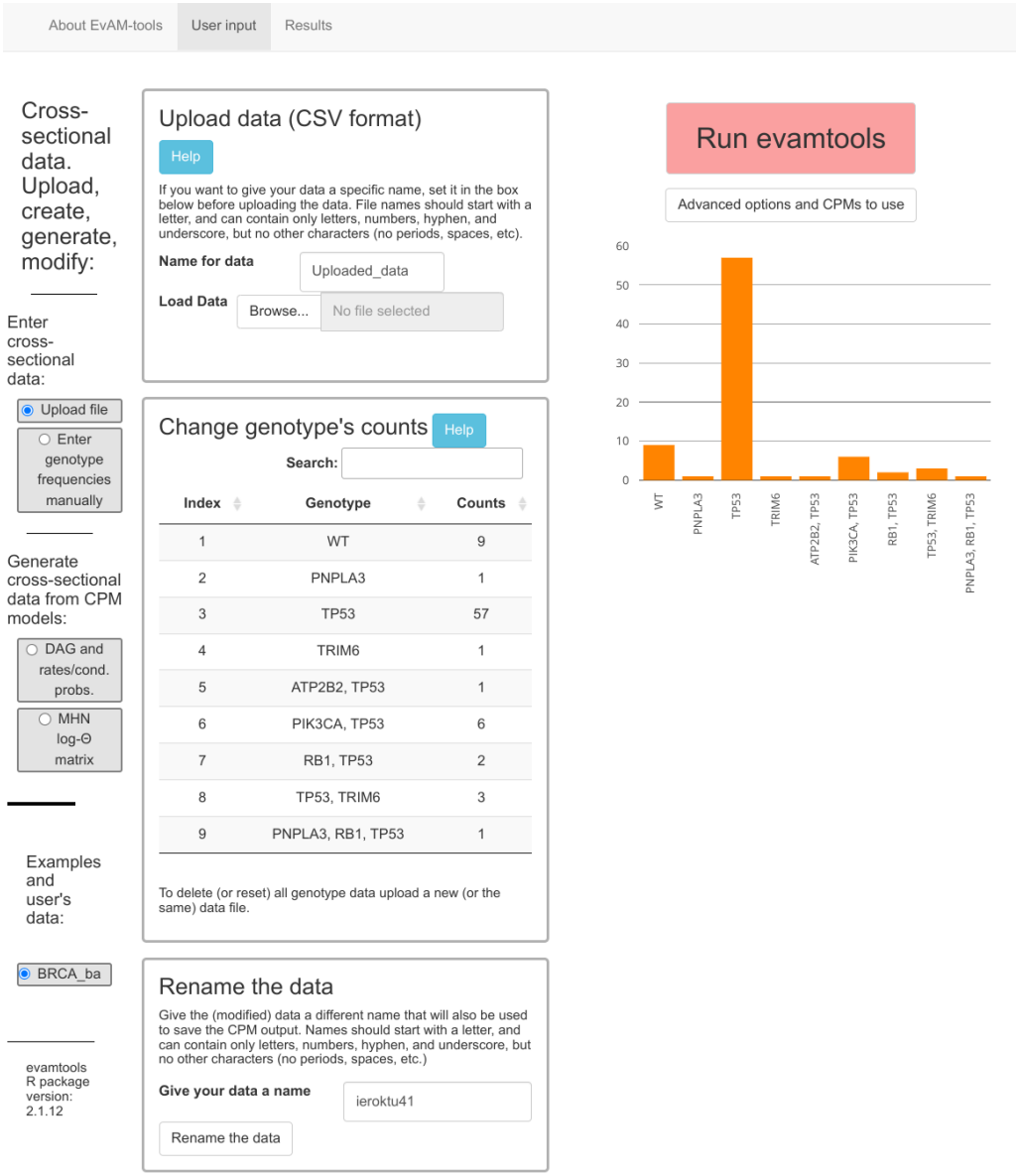
In section *"Appendix: getting the BRCA and Ov data sets from the R console" (section 6)* we show how to obtain the data from the R console.

2.1 Analyzing the BRCA data set

We now import the BRCA csv data set, `BRCA_ba_s.csv`, into the web app, https://iib.uam.es/evamtools. We go to the "User input" tab, and click, on the left, on "Upload file"; we set the "Name for data" as "BRCA_ba"):



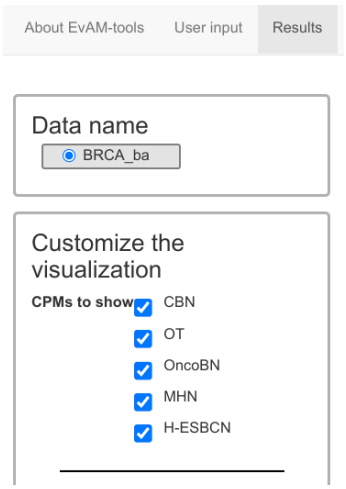
On "Load Data" we click on "Browse" and select the file from our file system; the data is uploaded, and the genotypes' frequencies are shown in the histogram on the right and the table at the bottom (where, if we wanted, we could modify the genotype counts):



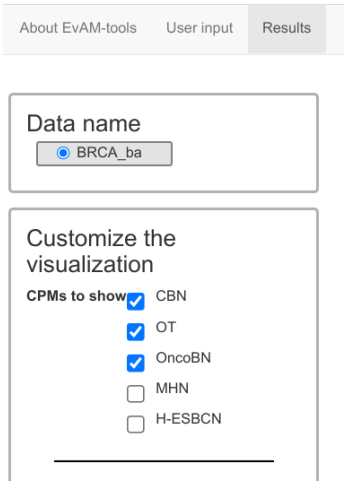
Before running the analysis, we select the unselected H-ESBCN as one of the methods to run (under "Advanced options and CPMs to use"). We also set the number of MCMC iterations to 500000, instead of 200000, for increased stability of results; this increase in iterations will of course result in longer running times.

We click "Run evamtools" and the output is shown in about 30 to 50 seconds[1]. For easier display of the figures in this document, we select first three of the methods to show, by clicking, on the left menu, under "Customize the visualization":

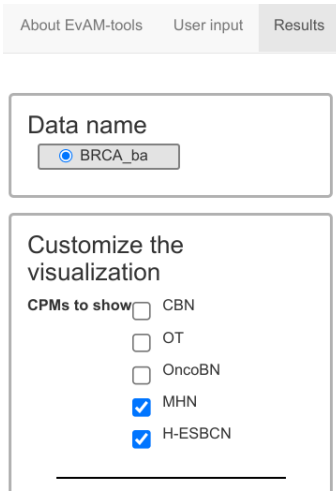
[1] In this example, changing the setting for the number of MCMC iterations of H-ESBCN from 100000 to 500000 is responsible for increasing the total time from about 30 to about 50 seconds.



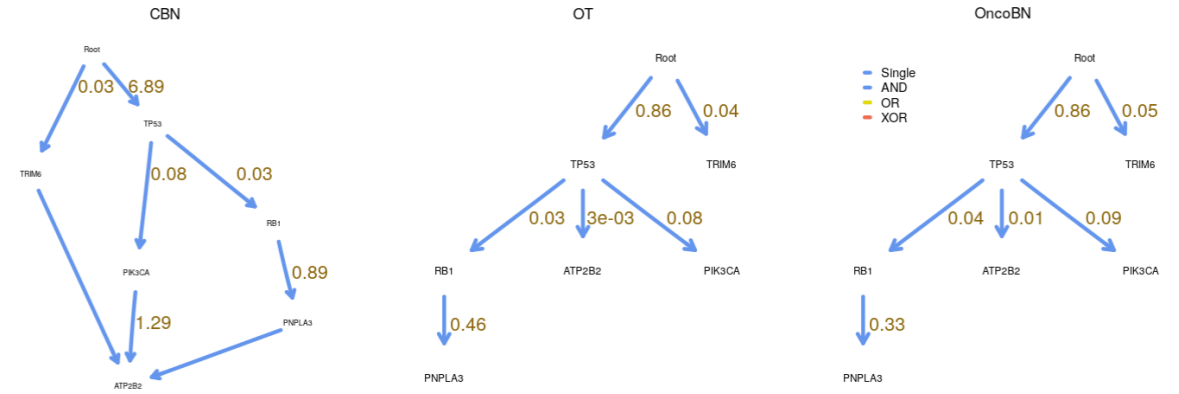
First on the first three (CBN, OT, OncoBN)



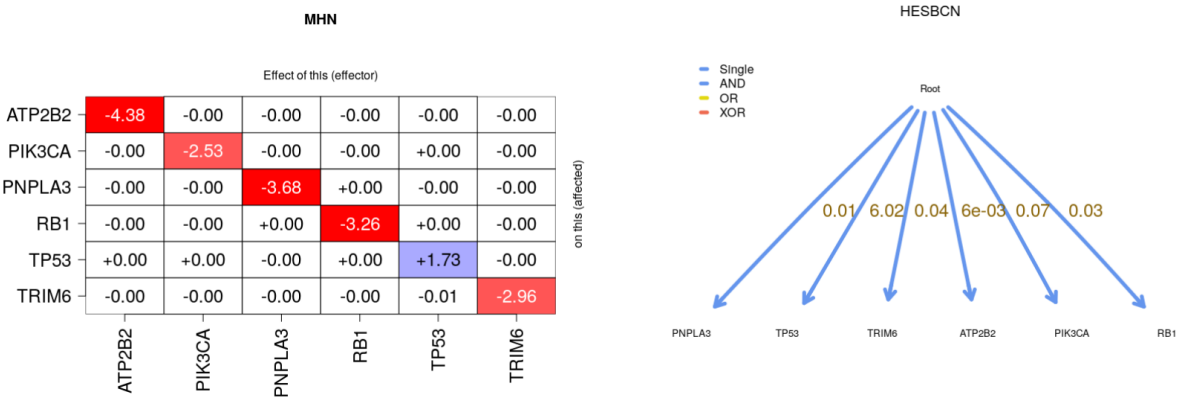
Then on the next two (MHN, H-ESBCN)



This is the output from the first three methods (CBN, OT, OncoBN):



And this from the next two methods (MHN, H-ESBCN):

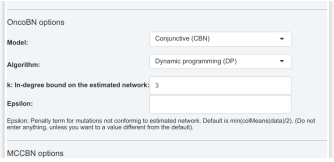


(The above screen-captures only show the DAGs/MHN matrix; we are not showing the figures of the predictions of the fitted models, such as transition probabilities or predicted genotype relative frequencies).

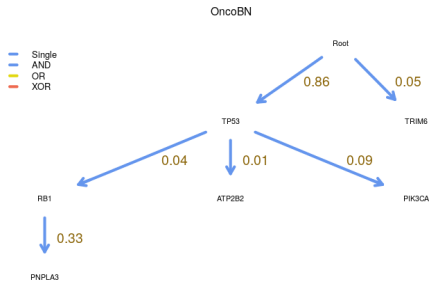
EvAM-Tools makes it immediate to see that:

- The output from OncoBn and OT is essentially identical.
- CBN and H-ESBCN give very different DAGs.
- OT and OncoBN differ from both CBN and H-ESBCN.

That OT and OncoBN give identical results is not surprising since OncoBN has not found any disjunctive pattern and OncoBN is using the disjunctive (OR relationships) model. We can run OncoBN using the conjunctive model. Go back to "User input" and click on "Advanced options and CPMs to run" and set, for "OncoBN options", the "Model" to "Conjunctive":

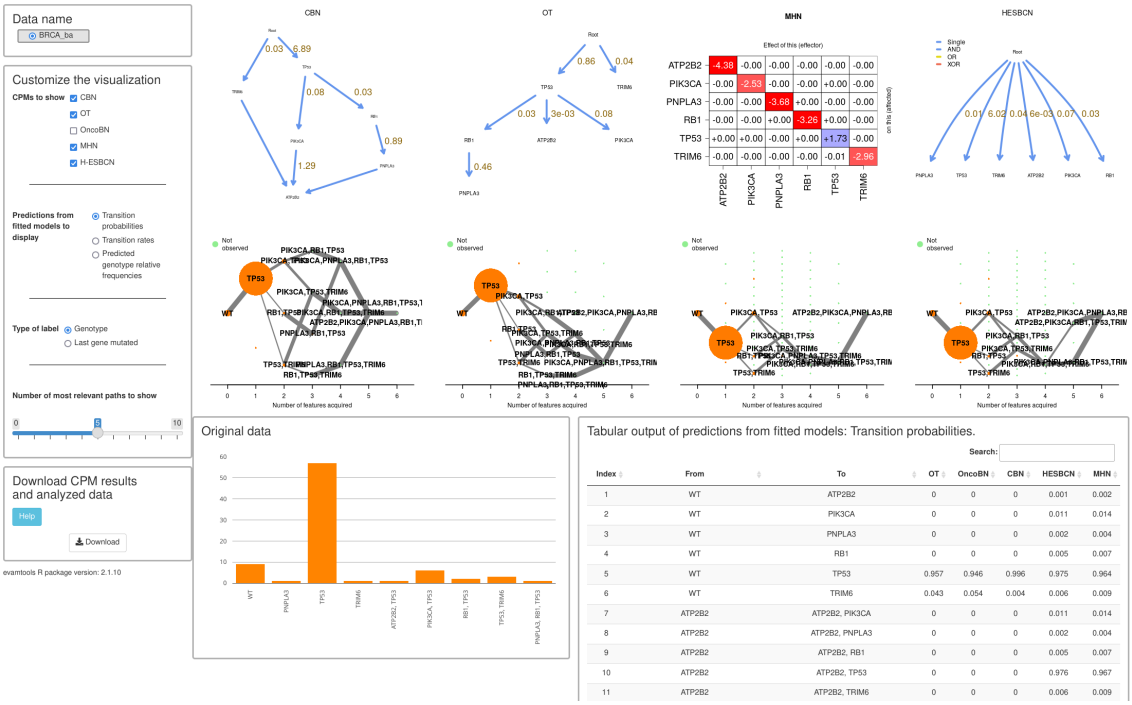


Click on "Run evamtools" to obtain the new fit (since we are only interested in rerunning OncoBN, we might want to unclick the other methods, so as to make the run as fast as possible). Interestingly, if we run OncoBN with conjunctive pattern, it does not show any conjunctions either (the result is the same as shown above):

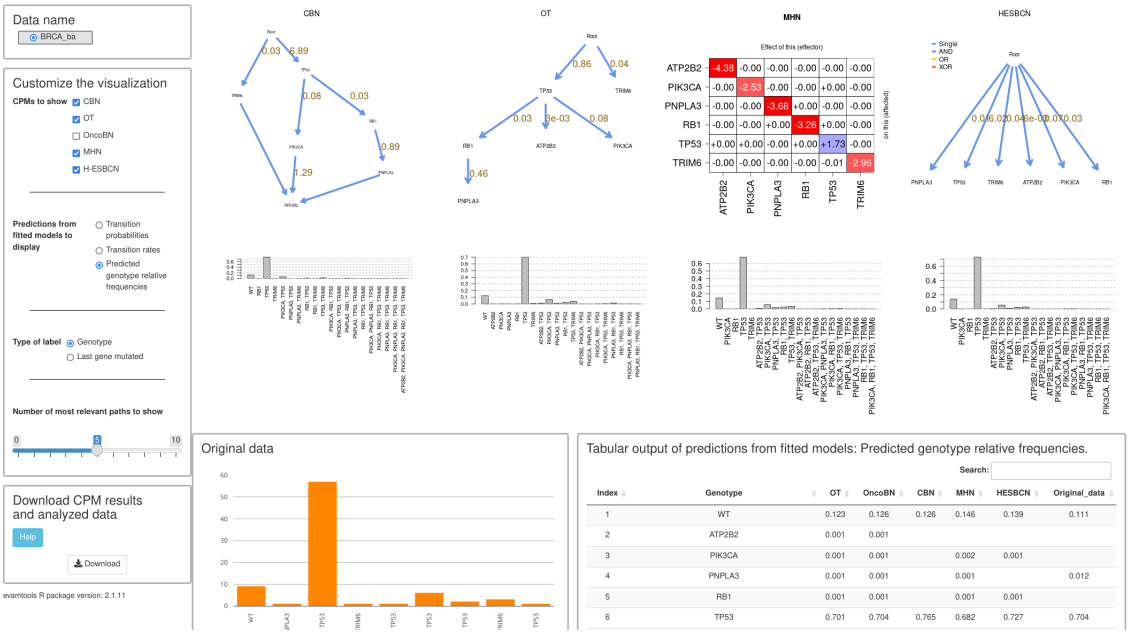


Thus, CBN seems to find support in the data for conjunctive dependencies that neither OncoBN (run using the "Conjunctive" model) nor H-ESBCN find.

EvAM-Tools's output also displays, in the Results tab, the original data as well as transition probabilities, transition rates, and predicted genotype relative frequencies. We show the three differing DAGs, MHN's output, and the data, when we select "Transition probabilities" (tabular output truncated in the screen capture):



and when we select “Predicted genotype relative frequencies”:



(This, incidentally, shows that we probably would have wanted to use shorter genes names as the histogram labels are too long).

From the above display we can conclude:

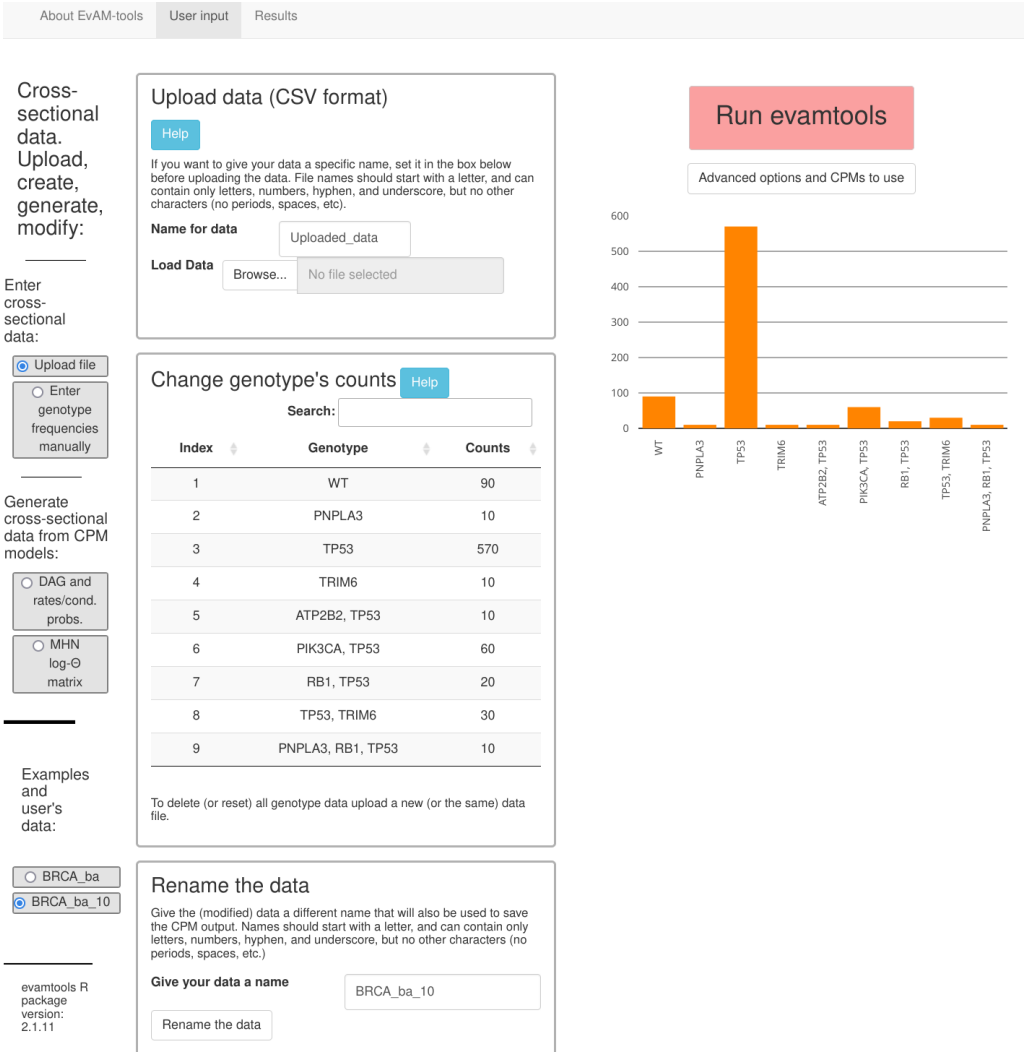
- The data contain very few cases where there are joint occurrences of two or more genes: most joint occurrences appear only once.
- The conditional probabilities from OncoBN (or OT) indicate that the really likely event is gaining TP53; the conditional probability of the remaining ones is very small.
- CBN leads to the same conclusion: the only large λ is that for TP53.
- MHN’s output points in the same direction: except for TP53, the diagonal entries of the matrix are all negative and large in absolute value, and the off-diagonal entries are all essentially 0. Thus MHN’s model is saying that we can fit the data reasonably well without modeling inhibiting or facilitating relations between genes.

Therefore, we can conclude that the apparently different results are caused by differences in the weighting of evidence: H-ESBCN, given the very small frequencies of most genotypes with more than one mutation, is choosing not to take those as evidence of dependencies, and is instead returning a simpler model.

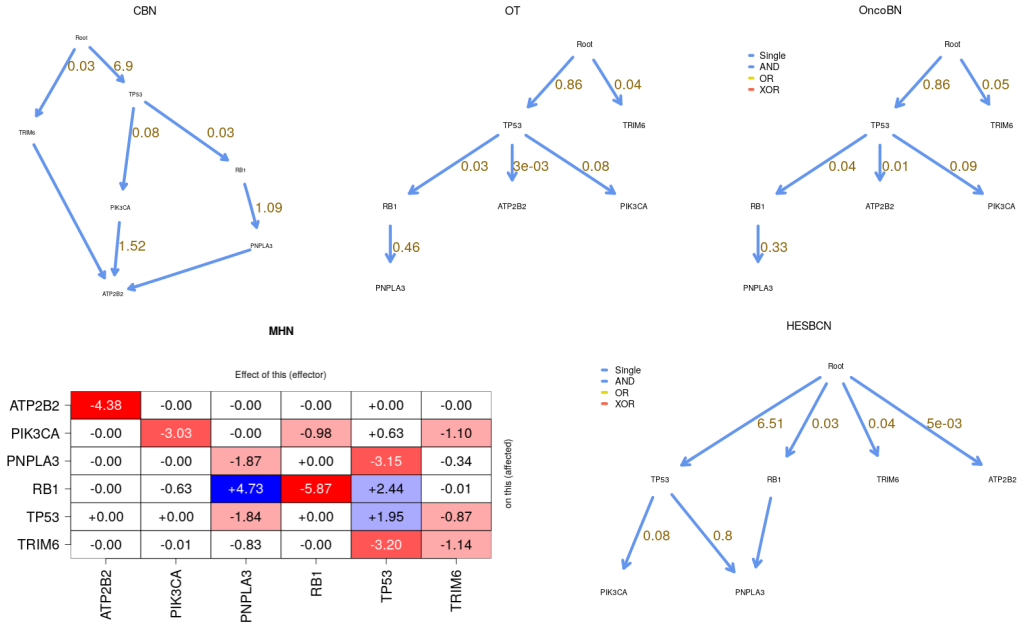
2.2 Is it just sample size?

We can examine more carefully the conjecture above: would H-ESBCN return a different DAG if sample size were much larger but relative proportions were the same? We can do that easily with EvAM-Tools. We simply go to the "User input" tab and multiply the genotype counts, for example by 10 (which is trivially done by clicking on each cell and adding a 0).

We rename the data first, and then increase the sample size. This is what it looks like:

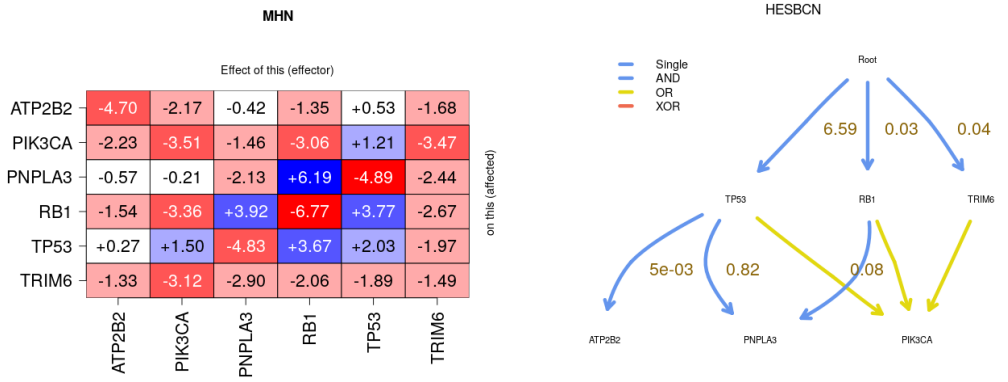


Now, we rerun the analyses:



The DAGs and weights (lambdas, probabilities) are the same for OT, OncoBN and CBN. But the models inferred by MHN and H-ESBCN have both changed, and the second now includes dependencies between some of the genes. Some of these dependencies are similar to the ones in the CBN output (PNPLA3 depends on RB1 and TP53; PIK3CA depends on TP53).

Interestingly, increasing the sample size another 10 times results in additional changes in the MHN and H-ESBCN models (OT, OncoBN, and CBN only show minor changes, not shown below):



Note that in some runs, the output returned by H-ESBCN is different from the one above; with H-ESBCN different runs can sometimes lead to different results. You can fix the random number seed in "Advanced options" to prevent this, though this is probably not advisable, since fixing the seed would precisely prevent us from seeing the instability of the fitted models. For example, if you set the number of MCMC iterations (under "Advanced options") to 100000 and the seed to 19, you will obtain a model with an XOR[2]. In the web app we use, by default, 200000 MCMC iterations,

[2]Note, though, that the output with the XOR also contains an edge with an extreme weight, which makes this model suspect; more detailed exploration would use a range of seeds, and possibly change also the number of MCMC iterations to run

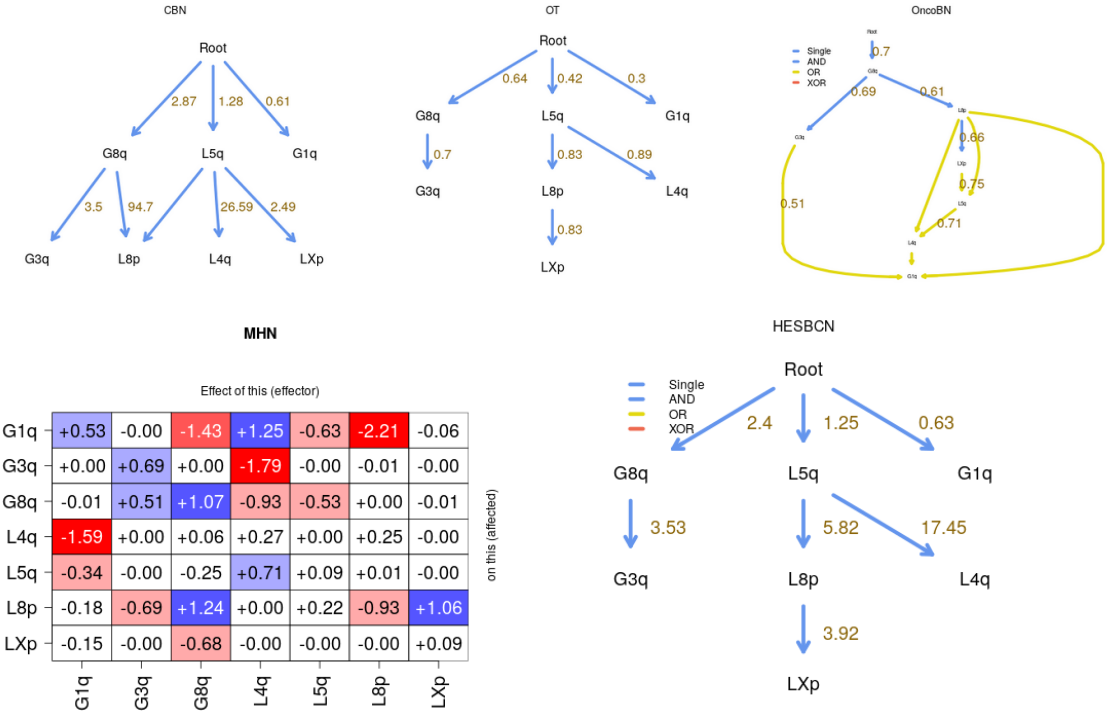
a number larger than the default in https://github.com/danro9685/HESBCN, precisely to minimize this instability). In the examples in this document we use an even larger number of 500000 MCMC iterations to obtain more robust results and because none of the examples shown take longer than about 1 minute to run. Variability of results from different runs can also be observed with CBN sometimes (though, in our experience, it is less common than with H-ESBCN with default parameters).

These results lead us to conclude tentatively that, compared to OT, CBN, and OncoBN, the penalties used in H-ESBCN and MHN seem to have a larger effect on models fitted to modest sample sizes.

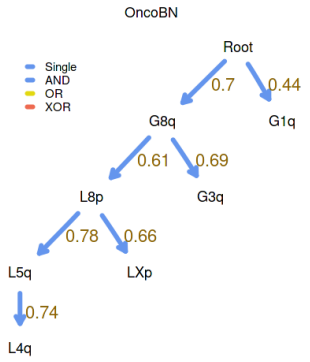
2.3 Analyzing the ovarian CGH data

Lest readers think that the above (coincidence between OT and OncoBN, and H-ESBCN returning star models with moderate sample sizes) are general patterns, we now show, for a different example, the analyses of the ovarian CGH data. We upload the data `ov2.csv` and, as before, include H-ESBCN in the methods and set the number of MCMC iterations of H-ESBCN to 500000.

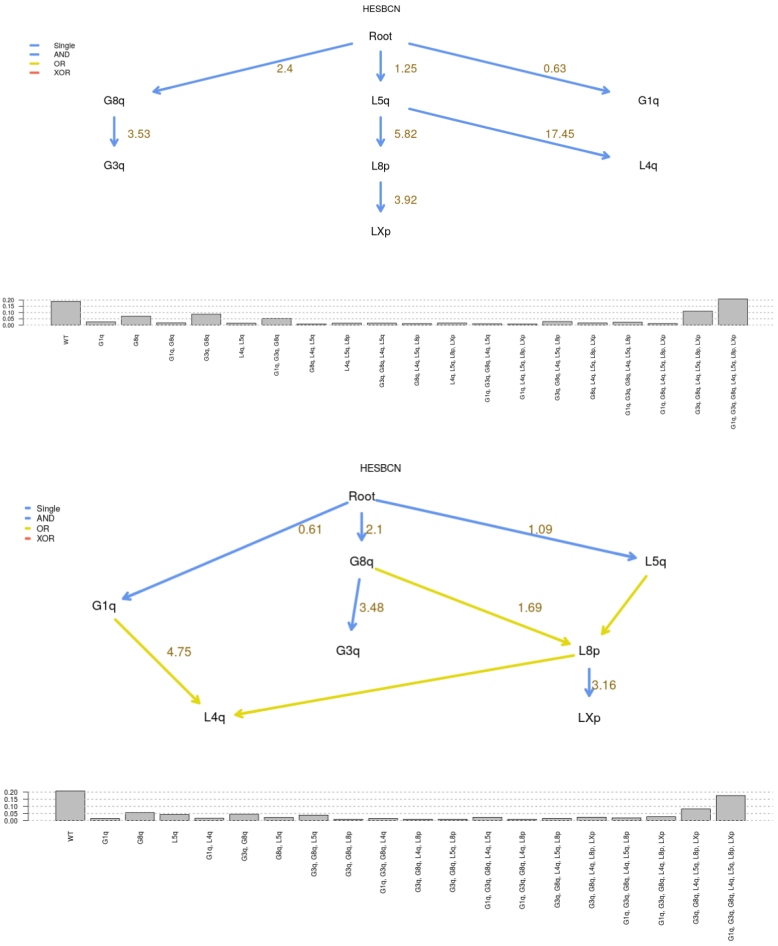
This is the output from the web app (run takes a little bit over one minute):



And we re-run the OncoBN model using the "Conjunctive" options (instead of the default disjunctive —we did this before in section *"Analyzing the BRCA data set", section 2.1*, and it involves going back to "User input", clicking on "Advanced options and CPMs to run" and setting, for "OncoBN options", the "Model" to "Conjunctive"). This is the output:

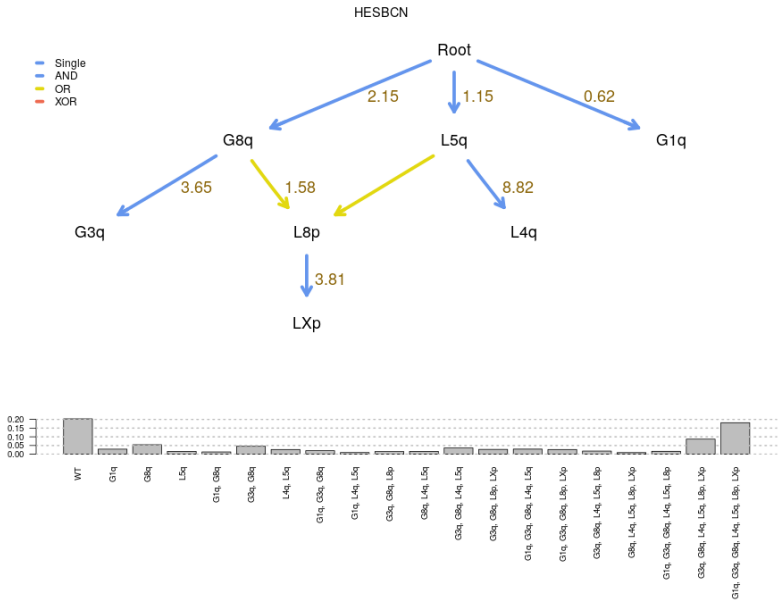


Interestingly, in the above run, H-ESBCN gives an identical structure to that of OT (parameters are, of course, different: OT's weights are conditional probabilities and H-ESBCN λs are rates). Different runs of H-ESBCN can give different results, such as the following ones, where we also ask the web app to display the predicted genotype frequencies:



We can increase the number of MCMC iterations. When using 1000000 most runs tend to give

this model:



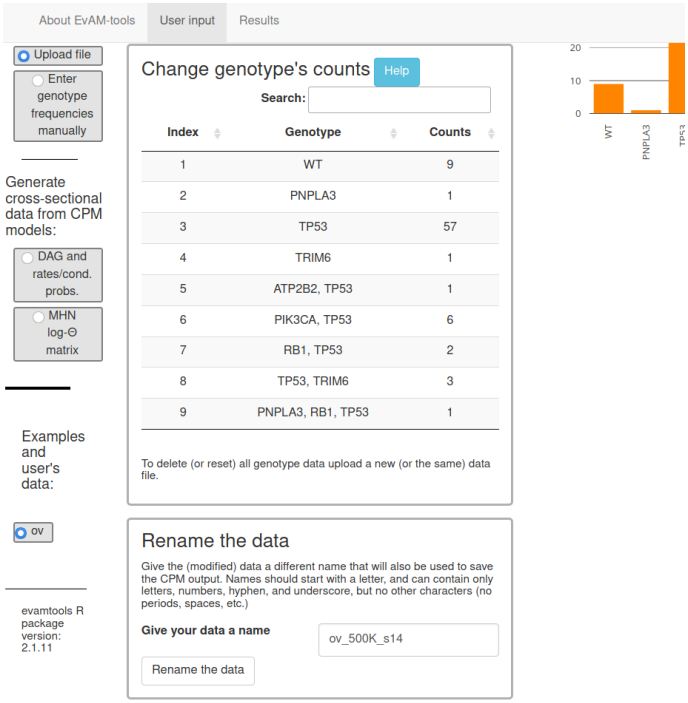
The variability between fitted H-ESBCN models might deserve a more careful exploration and, in "for real" analyses, additional runs with 1000000 iterations or even more.

Now there are large similarities are between OT and CBN (though, of course, there can be no conjunctions in OT). H-ESBCN finds identical restrictions for G3q (3q+) and G8q (8q+) as OT and CBN; it also finds similar patters to OT for L5q and LXp; note that CBN shows L5q → L8p and L5q → LXp and that H-ESBCN shows an OR for the dependency of L8p on G8q and L5q, whereas CBN, which can only model ANDs, places an AND. OncoBN using the default disjunctive relationship (OR, but not XOR) seems to suggest quite a different model (note that H-ESBCN can also fit OR relationships). Interestingly, the conjunctive model for OncoBN is similar, but not identical, to the ones from OT and CBN.

2.4 Using the web app for small computational experiments

We have shown the output of repeated runs of H-ESBCN, changing the number of MCMC iterations and possibly setting different random number seeds. GUIs and web apps are not the most appropriate tools for a systematic exploration; instead, properly documented code as an R script would be the preferred procedure. For small experiments, however, the web app is fine. A simple way to keep track of what is done is as follows:

1. Upload the data, giving it a meaningful name (under "Name for data"); for example, `ov`.
2. Go to the "Rename the data" box, and add the settings you will use to the name of the data; for example, for 500000 MCMC iterations with seed 14 for H-ESBCN enter `ov_500K_s14` in "Give your data a name", and **click on "Rename the data"**.



3. Under "Advanced options and CPMs to use" set the seed to 14 and the number of MCMC iterations to 500000, possible also setting H-ESBCN as the only method to run.
4. Click on "Run evamtools".
5. Repeat steps 2 to 4 as needed.

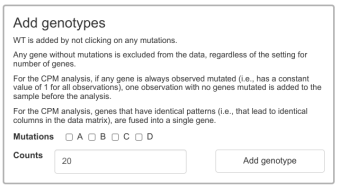
The "Results" tab will contain the output of the different runs, properly labeled so we can examine, at will, outputs from runs with different settings.

3 Analyzing manually constructed synthetic data

We will create some synthetic data to show the consequences of analyzing data where two events are indistinguishable, because they are completely aliased, i.e., indistinguishable, because they have identical patterns —identical columns in the data matrix—. Of course, manually constructed synthetic data can be used to explore or examine many other patterns, unrelated to aliased events.

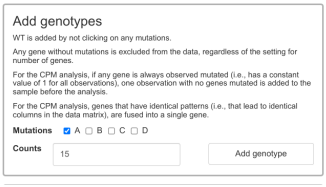
From the "User input" tab we select the "Enter genotype frequencies manually":

Now, we enter some WT, for example, 20 WT observations. We select no mutations, and type a "20" in "Counts":



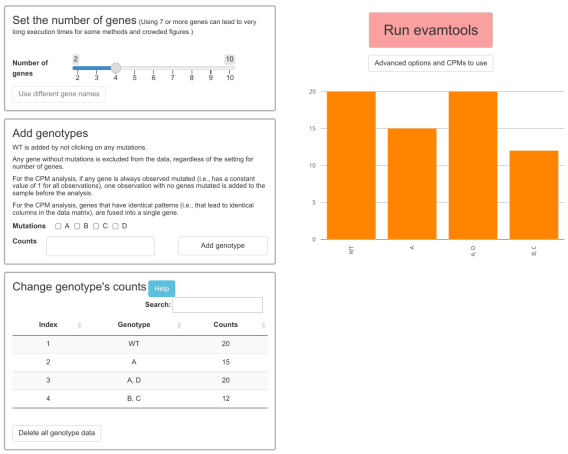
And when we click on "Add genotype" we see the histogram with 20 WT and the genotype table with the 20 WT:

We now add 15 observations with only A mutated (i.e., 15 individuals of genotype A), and 12 with both B and C (i.e., 12 individuals with genotype "B, C"); we first click on "A" on "Mutations" putting a 15 in "Counts"

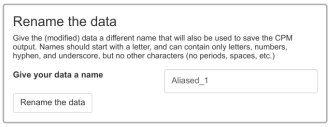


and then on "Add genotype". We next click on B and C in "Mutations" putting a 12 in Counts. We finally add 20 individuals with the "A, D" genotype (steps as before: click on A and D for mutations, and put a 20 in counts). After these steps, we can see the genotype composition as both

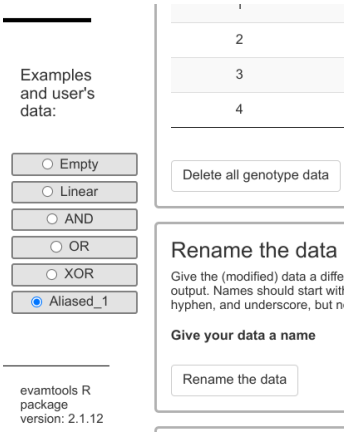
a histogram and table:



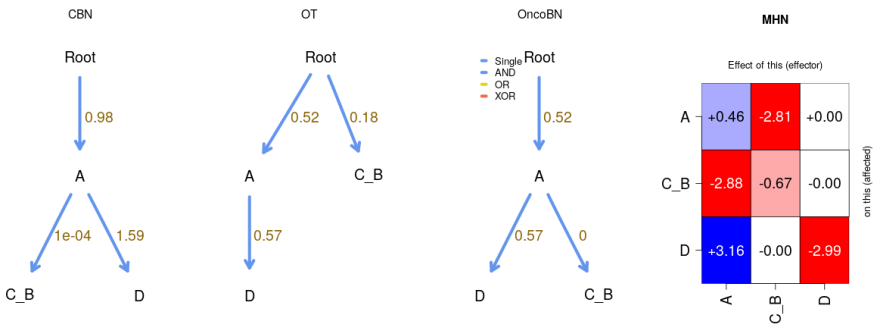
So that it is easier to go back to these data, we give this data set a name, for example "Aliased_1" under "Rename the data".



Click the "Rename the data" button, so the name is used and you will see it listed on the left, under "Examples and user's data:"



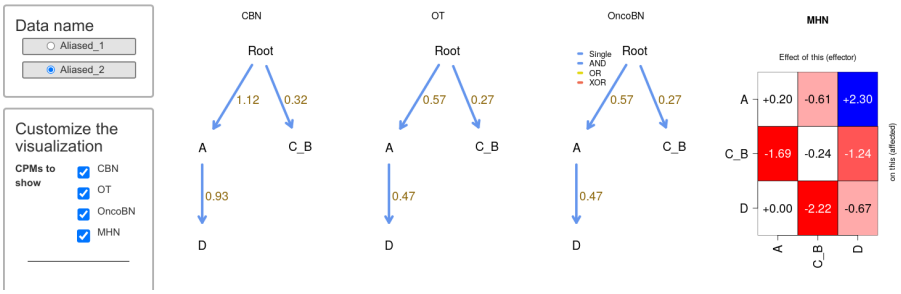
Now we click on "Run evamtools". These are the fitted models (we did not use H-ESBCN here):



In the DAGs and the MHN log Θ matrix, as well as the transition probabilities, an event

labelled “C_B”. “C_B” is the name of the event created automatically by EvAM-Tools by fusing the “C” and “B” events that are not distinguishable.

Note also that the models fitted by the DAGs, for example OncoBN or CBN, do not seem right when we look at the parameters of the “C_B” event. That is because there are no “A, B, C” events, and we would expect to see some if A on the one hand, and B and C on the other, are occurring independently. We will add some observations (eight, for example) with all of A, B, C. We go back to the “User input” tab, we “Rename the data” to “Aliased_2” (so that the new modifications we are about to make do not affect “Aliased_1”), and we add genoytpe “A, B, C” with a count of 8. And we click on “Run evamtools”. This is the output



which is more sensible. Notice, however, that we still have “C_B” as an event because, in fact, they remain completely aliased, indistinguishable. This aliasing would be broken by adding just a single “C” or a single “B”, or a single “A, C”, or “A, B”, or “B, D” or “D, C” or any “A, B, D” or “A, C, D”.

4 Generating data from known models

The discussion in sections *“Analyzing the BRCA data set” (section 2.1)* and *“Analyzing the ovarian CGH data” (section 2.3)* has used CPMs on two cancer data sets for which the truth is unknown. To understand the differences between models, and the performance characteristics of different methods, we can simulate data under a known model and examine if the true pattern can be recovered. This is very easy to do with EvAM-Tools and addresses two commonly asked questions:

- Can we recover the true structure?
- How do different methods perform when data has been generated under the assumptions of another method?

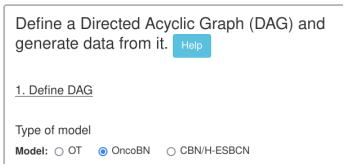
This is what we will do below in examples *“A model with AND, XOR, OR” (section 4.2)* and *“A model with AND” (section 4.3)*. But EvAM-Tools is also useful to understand what different models imply in terms of the data we would observe, even without considering what each method would fit to a given observed data set; this is what we show in *“CPM models: what type of data they imply?” (section 4.1)*.

4.1 CPM models: what type of data they imply?

4.1.1 What happens if we increase ϵ for OncoBN?

We go to “User input” and, by default, the option “DAG and rates/cond. probs” is selected under “Generate cross-sectional data from CPM models”. We can leave the default DAG in the selected

"DAG_Fork_4". In "Type of model" under "Define DAG" we click on "OncoBN":



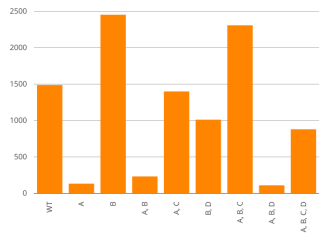
and you can see that the last column of the DAG table is now called "theta":

DAG table

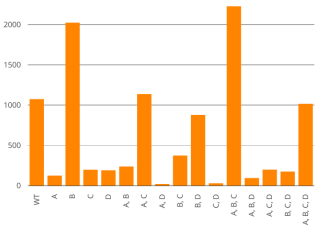
Remember to hit Ctrl-Enter when you are done editing the DAG table for changes to take effect.

From	To	Relation	theta
Root	A	Single	0.5
Root	B	Single	0.7
A	C	Single	0.9
B	D	Single	0.3

Now, click on "Generate data from DAG"; to make patterns easier to observe, set the "Number of genotypes to sample" to a large number, such as 10000. If we look at the histogram we will see something similar to this one:



In particular, notice how the following genotypes are not observed: "A,D", "B, C", "B, C, D", "A, C, D", as they are not possible if the restrictions are completely respected. Now, increase "epos, ϵ" to, say, 0.15, and click again on "Generate data from DAG"; now we will observe at least a few cases of all or most of the above four genotypes, as the predicted genotypes now incorporate deviations from the model.

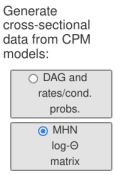


We could continue increasing the value of ϵ; this will result in increasing frequencies of the above four genotypes, and a decrease in WT. Very large increases in ϵ will lead to a blurring of the signature of this DAG.

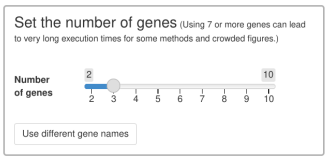
A more advanced exploration of the role of deviations from the model in OT and OncoBN compared to the role of observational noise would alter "epos, ϵ" with and without simultaneously changing the value of "Observational noise".

4.1.2 A simple exploration of MHN

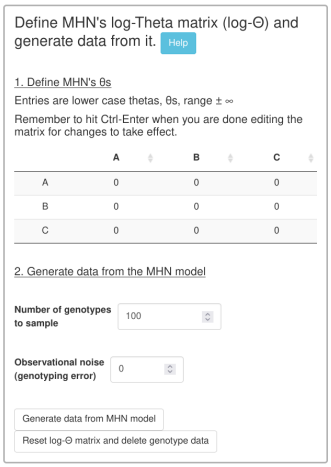
To try to gain a quick intuitive understanding of the multiplicative hazards model of MHN we go to “User input” and then click on “MHN log-Θ matrix”



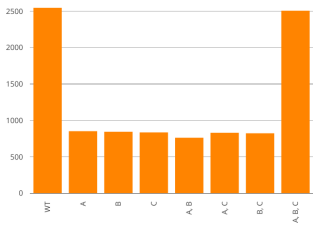
To start with a simple, yet not completely trivial, model, we set the number of genes to three:

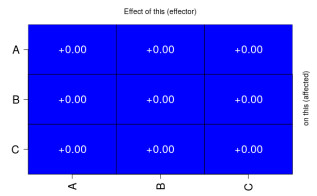


Now, we click on “Generate data from MHN model”:



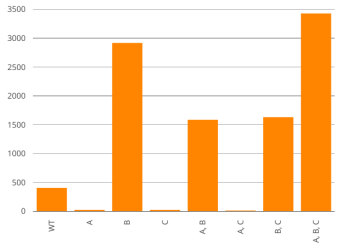
In this model, there are not multiplicative effects between genes. This is what we obtain (again, it might help to increase the Number of genotypes to sample to 10000 to decrease the role of random sampling noise and focus on the predicted genotype frequencies):

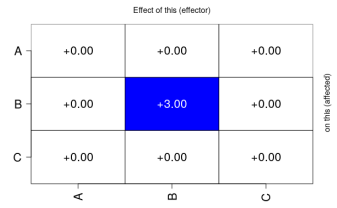




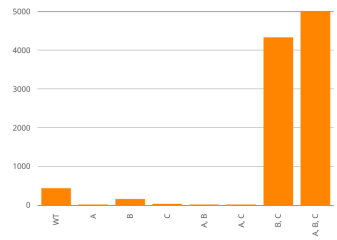
Let us now increase the baseline hazard of event B. For example, set log-$\Theta_{2,2} = 3$. (Modify the

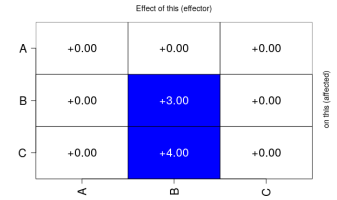
entry, and click on Ctrl-Enter). The figure changes dramatically and all genotypes that have “B” have increased their frequency:





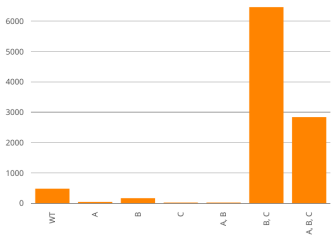
Let us now create a very large promoting effect of B on C; for that, we enter, for example, a 4 on the matrix entry (3, 2): log-$\Theta_{3,2} = 4$:

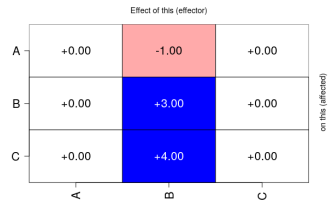




Notice how genotype B is more common than either A or C, but now whenever there is B it frequently in combination with C (very larger frequencies of genotypes B,C and A,B,C).

But what if B also an inhibiting effect on A? Set log-$\Theta_{1,2} = -2$:

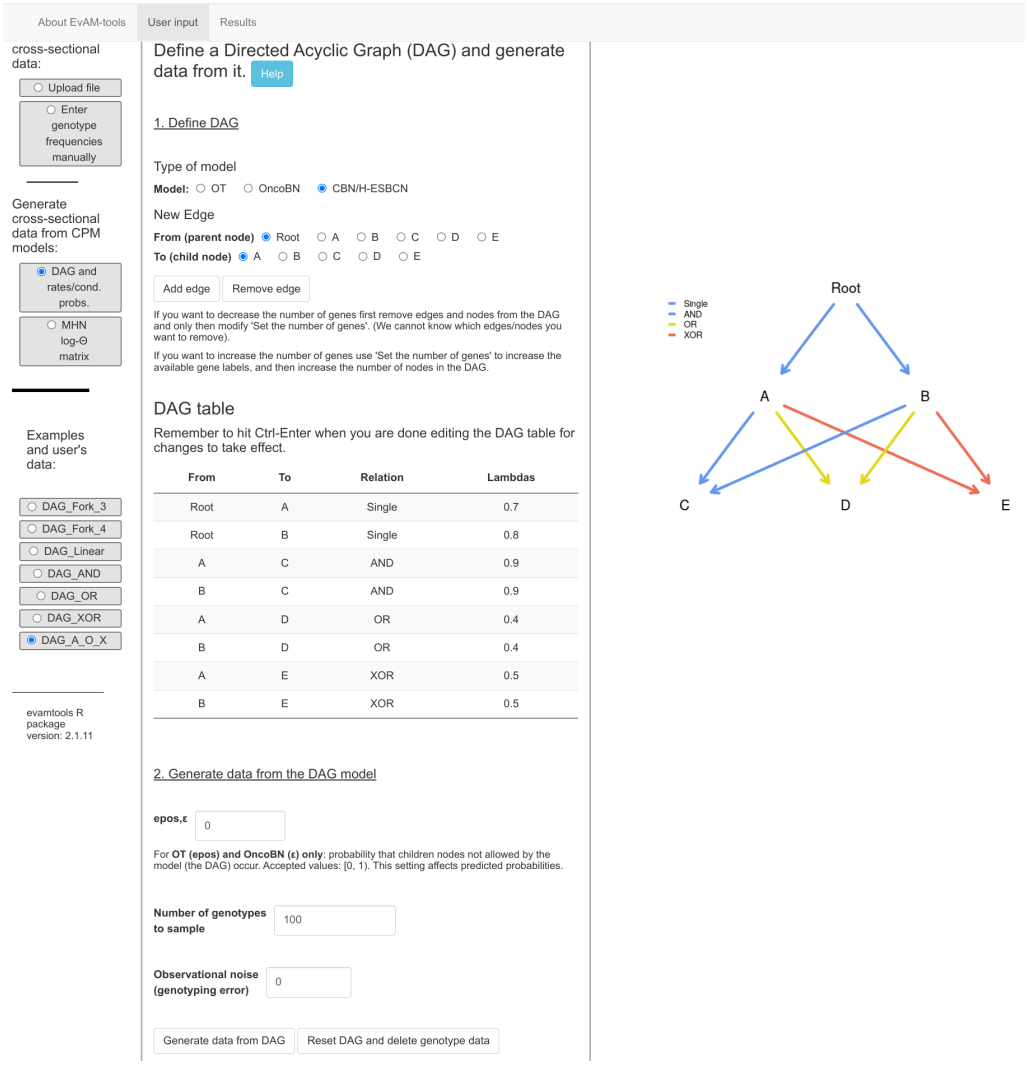




Notice how the frequency of genotype A,B,C has gone down (and also, though it is harder to appreciate, that of A,B).

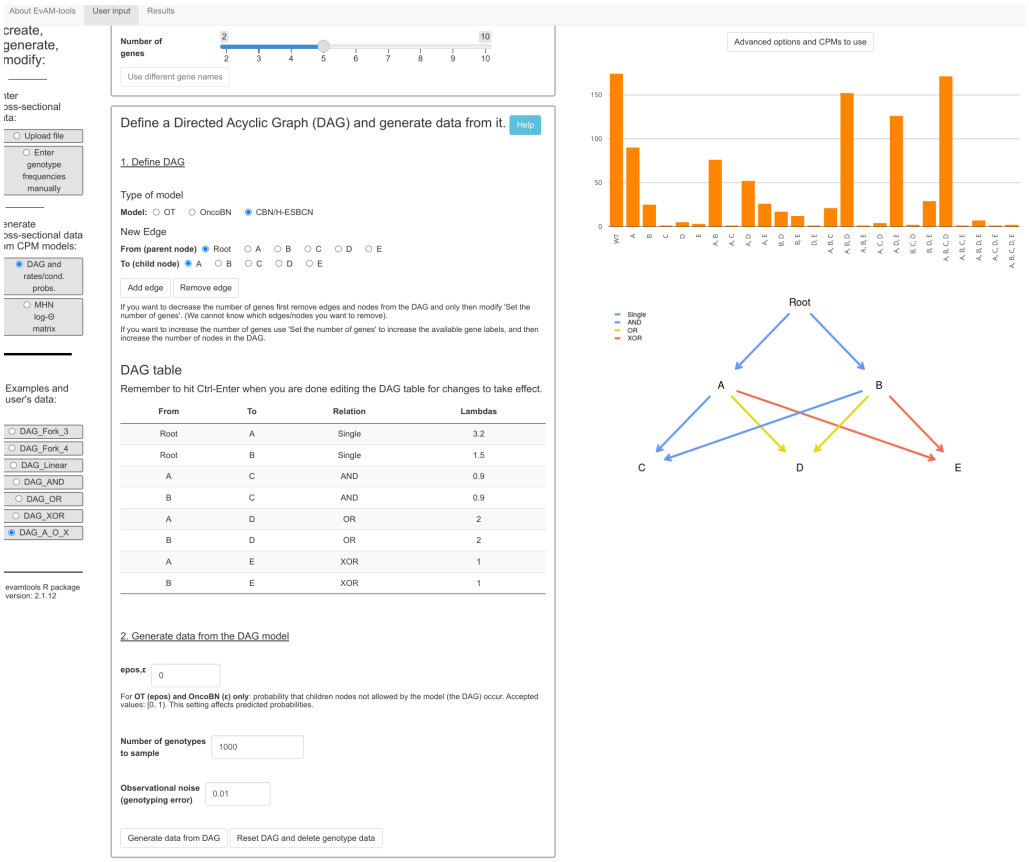
4.2 A model with AND, XOR, OR

Here, we will simulate data under a model that includes both AND, OR, and XOR relationships (e.g., H-ESBCN). EvAM-Tools, in its web app, already includes such a model for five genes:



If we want, we can change values (rates, relationships, noise, etc). We will change the rates of A, B, D, and E, setting them to 3.2, 1.5, 2, and 1, respectively; we set the number of genotypes to sample to 1000 and we will add 1% of Observation noise (i.e., we will type 0.01 in the "Observational nose (genotyping error)" box). Then, we click on "Generate data from DAG",

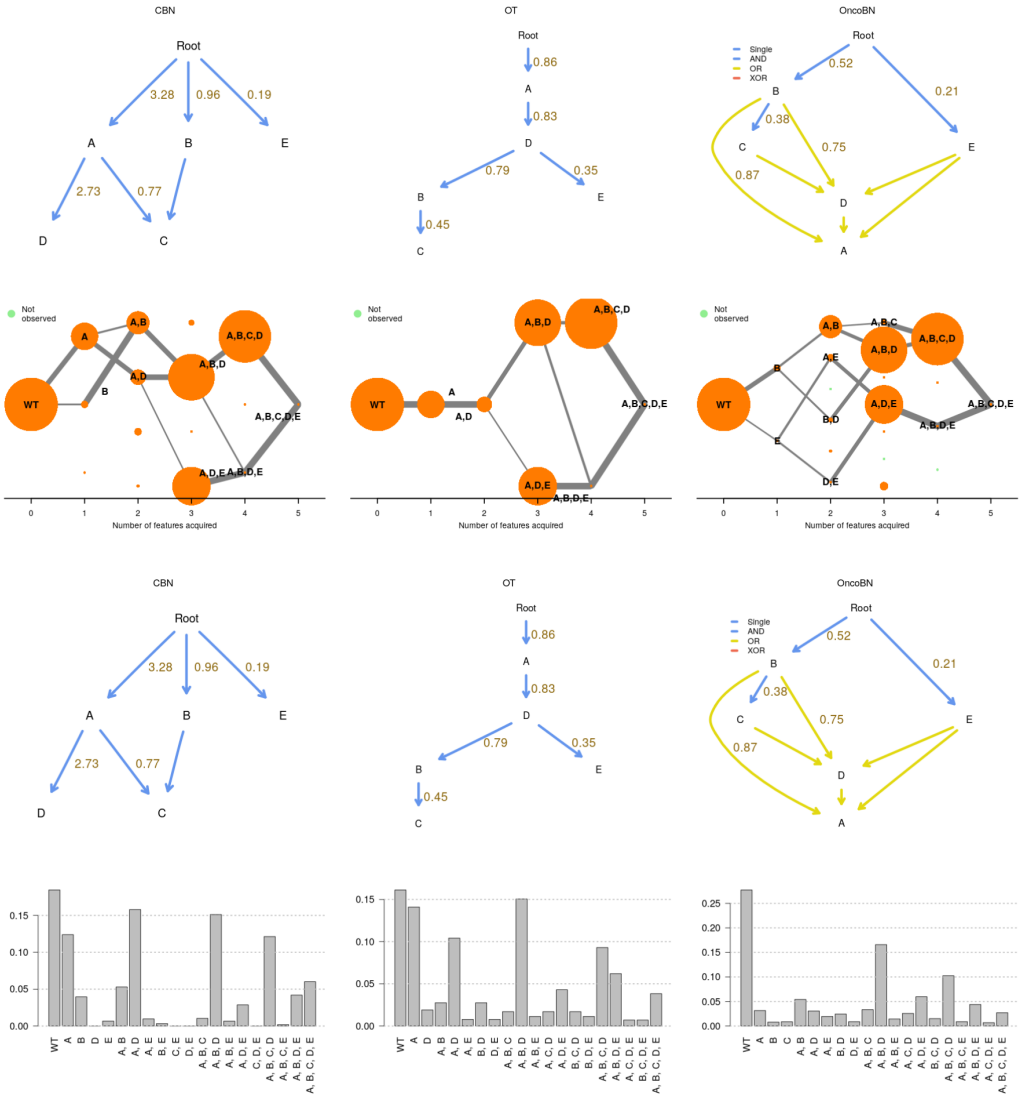
and obtain data simulated under that model:



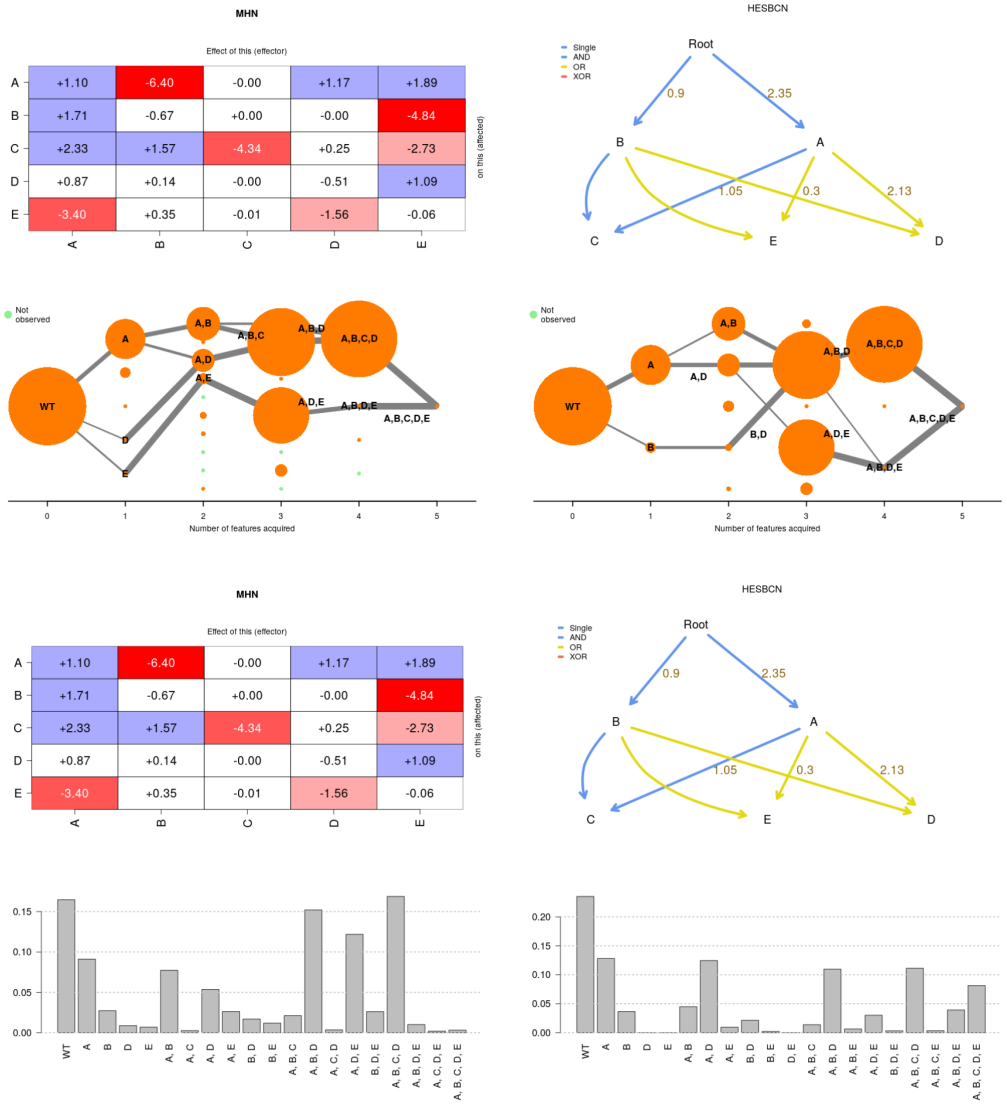
(Of course, the actual simulated data you are likely to obtain will vary differ from this one).

Now, we click “Run evamtools”, after adding H-ESBCN to the set of methods and setting its number of MCMC iterations to 500000 (again, this is done under “Advanced options and CPMs to use”). After about 30 seconds we obtain the output. In the plots below, and as we did before, we split it into three and two methods so it is easier to see. We show the models with two of the predictions: transition probabilities and predicted genotype relative frequencies (these predictions are also shown in the table, which we do not show below).

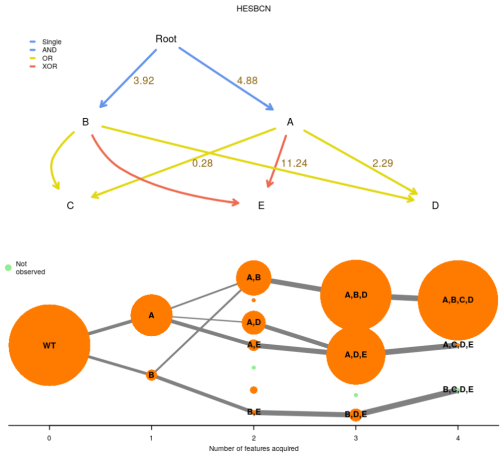
Note that the histograms of predicted genotype frequencies display, at most, the 20 most frequent genotypes (because of reasons of limited plotting space); all predicted genotypes are shown in the table.


CBN
Root
3.28
0.96
0.19
A
B
E
2.73
0.77
D
C
OT
Root
0.86
A
0.83
D
0.79
0.35
B
E
0.45
C
OncoBN
Root
Single
AND
OR
XOR
0.52
0.21
B
0.38
C
0.75
E
0.87
D
A
Not observed
WT
A
A,B
B
A,D
A,B,D
A,B,C,D
A,B,C,D,E
A,D,E
A,B,D,E
Number of features acquired
A,B,C,D
A,B,D
A
A,D
A,B,C,D,E
A,D,E
A,B,D,E
Number of features acquired
A,B,C
A,E
B,D
E
D,E
Number of features acquired

CBN
Root
3.28
0.96
0.19
A
B
E
2.73
0.77
D
C
OT
Root
0.86
A
0.83
D
0.79
0.35
B
E
0.45
C
OncoBN
Root
Single
AND
OR
XOR
0.52
0.21
B
0.38
C
0.75
E
0.87
D
A
0.15
0.10
0.05
0.00
WT
A
B
D
E
A, B
A, D
A, E
B, E
C, E
D, E
A, B, C
A, B, D
A, B, E
A, D, E
C, D, E
A, B, C, D
A, B, C, E
A, B, D, E
A, B, C, D, E
0.15
0.10
0.05
0.00
WT
A
D
A, B
A, D
A, E
B, D
D, E
A, B, C
A, B, D
A, B, E
A, C, D
A, D, E
B, C, D
B, D, E
A, B, C, D
A, B, D, E
A, C, D, E
B, C, D, E
A, B, C, D, E
0.25
0.20
0.15
0.10
0.05
0.00
WT
A
B
C
A, B
A, D
A, E
B, D
D, E
A, B, C
A, B, D
A, B, E
A, C, D
A, D, E
B, C, D
A, B, C, D
A, B, C, E
A, B, D, E
A, C, D, E
A, B, C, D, E



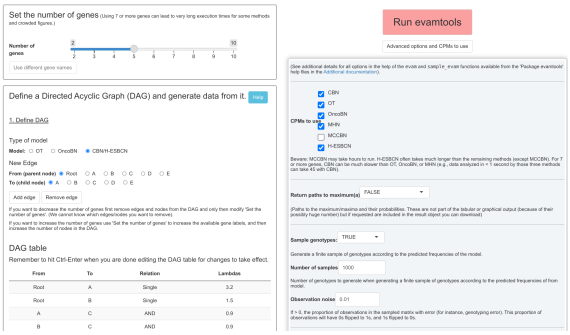
And, as has been the case before, repeated runs of H-ESBCN can lead to different results, for example:



None of OT, CBN, or OncoBN can capture XOR relationships. But in this case, H-ESBCN incorrectly infers an XOR for D (it is really an OR) and an OR for C (it is really an AND).

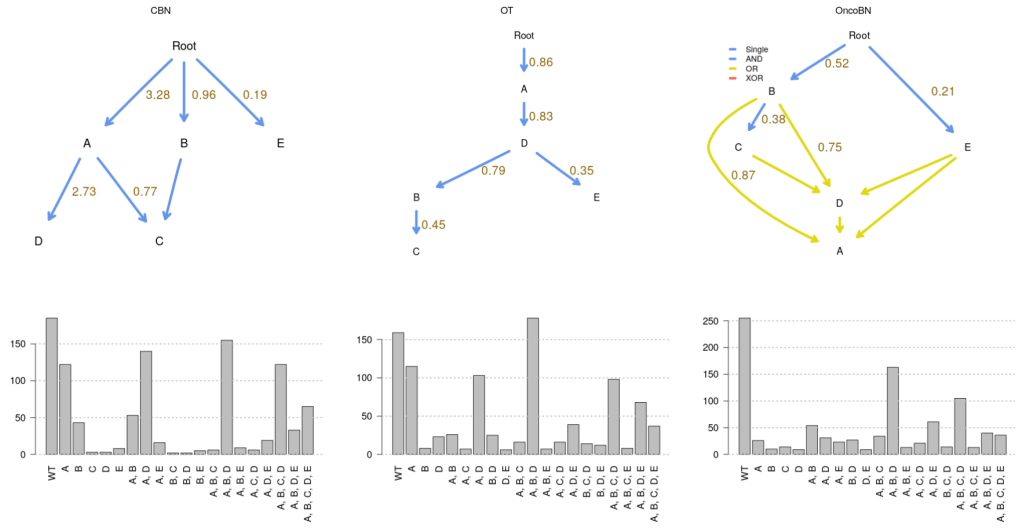
We could increase the sample size by 10 by just setting “Number of genotypes to sample” to 10000 or add different levels of noise, etc.

We could also ask to get, as return, not only the predicted genotype relative frequencies, but sampled genotype counts with, possibly, some noise added. Even without noise added, the relative frequencies of the sampled genotype counts would differ from the predicted genotype relative frequencies just because of sampling noise. We can do that by going back to the “User input” tab, clicking on “Advanced options and CPMs to use” and setting “Sample genotypes” to TRUE and selecting the number of samples, which here we set equal to the sample size of original data set (i.e., 1000); we also set the observation noise to 0.01.



We hit on “Run evamtools” and, as before, we get the output but we now have one extra possible “Predictions from fitted models to display”:

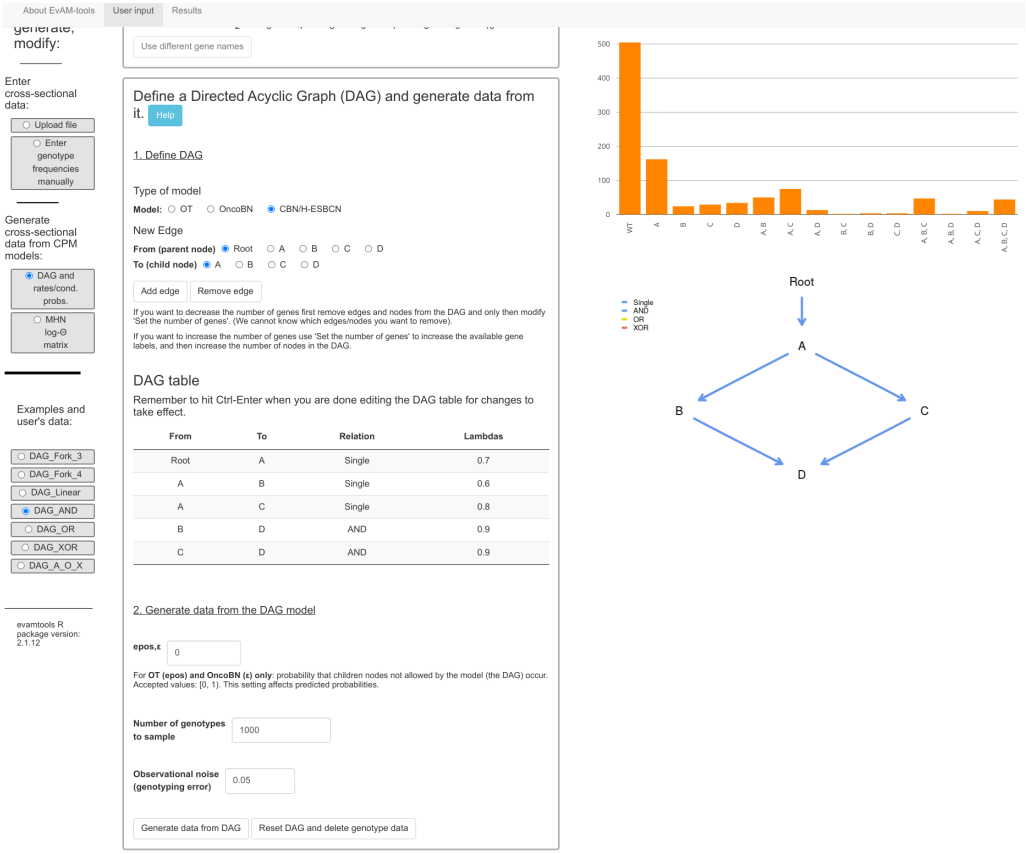
And this is the output for three of the models; notice the added variability (i.e., how the relative heights and even the actual genotypes present are not the same as in the predicted genotype counts):



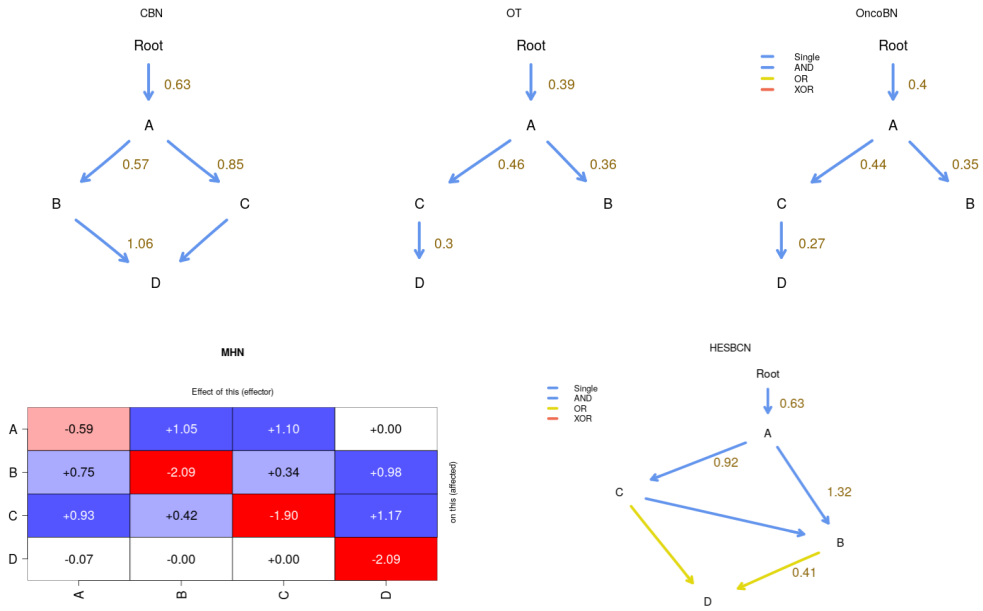
Of course, you can switch from displaying “Sampled genotype counts” to displaying “Predicted genotype counts” just by clicking on the button on the left.

4.3 A model with AND

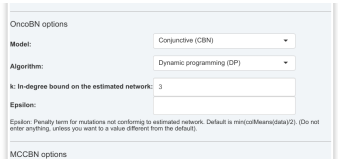
Let us use now a model with AND; we use the preselected “DAG_AND”, with 1000 observations and a 5% (0.05) genotyping error. Remember to click on “Generate data from DAG” after changing the noise level:



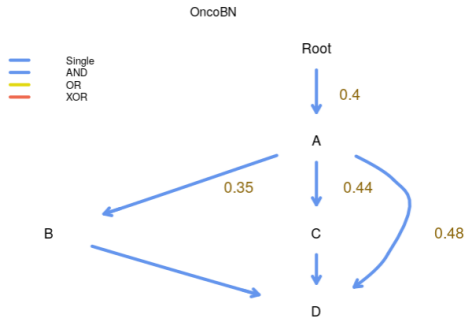
These are the fitted models:



H-ESBCN incorrectly infers the AND as an OR. CBN correctly infers the underlying model and provides estimates of the parameters that are very close to the true ones. OT and OncoBN cannot infer the correct true dependencies: OT because it cannot fit DAGs, but only trees (i.e., each node has only one parent) and OncoBN because it was run in disjunctive mode (OR relationships). We can run OncoBN using the conjunctive model; we have done this before (sections *"Analyzing the BRCA data set", section 2.1* and *"Analyzing the ovarian CGH data", section 2.3*) and it involves going to "Advanced options and CPMs to run" and setting, for "OncoBN options", the "Model" to "Conjunctive":



This is the output we get:



This correctly identifies the joint dependency of D on both B and C, but there is also and edge between A and D that we would never observe with CBN (as CBN always returns the transitively reduced DAGs); this is a technical issue beyond the scope of this document, but one that we have

discussed in the OncoBN repo[3].

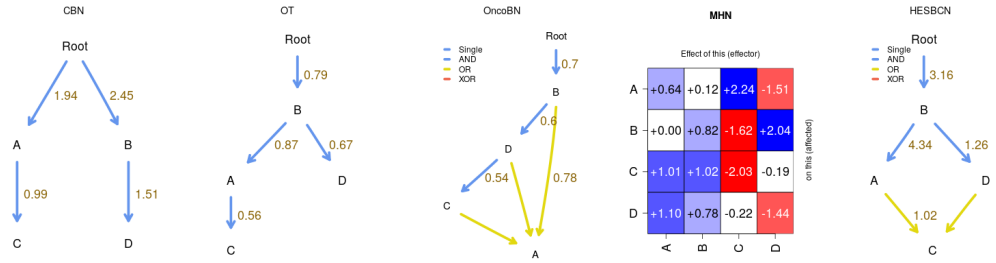
For H-ESBCN the returned model also includes an edge, the one between A and B, that is not necessary: since B depends on C and C depends on A, having B depend with an AND on both A and C is not needed (i.e., the transitive reduction of that DAG would remove the edge from A -> B). Note also that removing the A -> B arrow does not affect the relationships with D. Of course, for the relationships between C, B, and D we should not return the transitive reduction (as that would break the OR of D on either C or B). Thus, a model without the $A \rightarrow B$ would generate the exact same predictions as the model returned by H-ESBCN. (Why does this happen with H-ESBCN? It is a consequence of the heuristic search over DAG structures, which can occasionally return these topologies).

4.4 Modifying data generated from a CPM model before analysis

What if the data came from a given model but some additional process had altered the genotype data? For example, suppose data really came from a CBN model, but the frequency of WT genotypes is too small because the data have been filtered to contain only genotypes with at least one driver mutation, or the data contain some contaminated samples that would never had become tumores and have an excess of WT. We can explore this by generating data from a CPM model and, then, modifying the genotype composition, as we did in section *"Is it just sample size?" (section 2.2)* when we modified uploaded data.

As an example, go to "User input", "DAG and rates/cond. probs", and use the default selected "DAG_Fork_4". Change lambdas to 2, 2.5, 1, 1.5, for A, B, C, D, respectively. Add 1% of observational noise. And set the "Number of genotypes to sample" to 5000, to ensure small sample sizes are not the culprit of different inferences. Click on "Generate data from DAG" and, for simplicity, then analyze the data with OT, OncoBN, CBN, MHN, and H-ESBCN.

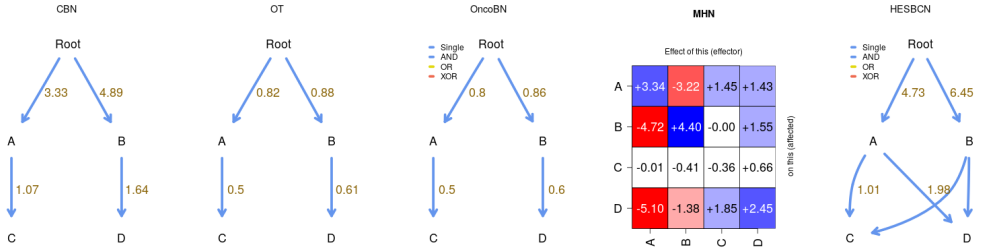
This is the output:



CBN correctly infers the DAG and the estimates of the λs are close to the true ones. The other methods are not able to infer this model correctly.

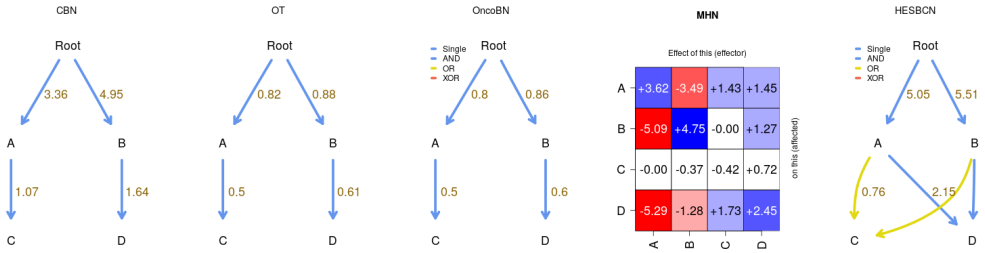
Now, go back to the "User input". Before modifying the data, and to keep a copy of the originally generated one, on the "Rename data" type a name (e.g., "data_original") and click on "Rename data". Now, enter a new name in "Rename data", for example "data_few_wt", click on "Rename data", and modify the WT frequency; I change it from its original value of 912 to 9. And then, analyze the data clicking on "Run evamtools". This is the output:

[3] https://github.com/phillipnicol/OncoBN/issues/5



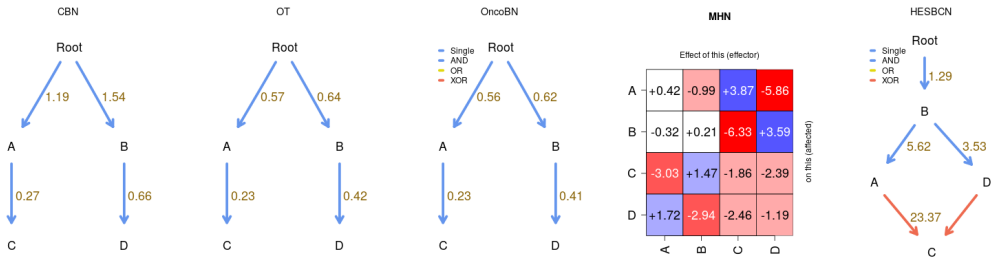
OT and OncoBN get the structure right, and for CBN the main consequence is altering the rates of A and B, increasing them (which is what we would expect). MHN also has increased estimates of log-$\Theta_{A,A}$ and log-$\Theta_{B,B}$ when we reduced the frequency of WT.

What if we removed the WT completely? We go back (and, if it is not the selected one, click, on the left, under "Examples and user's data", on "data_original"), and set WT to 0 (possibly after creating a new data set, "data_no_wt") and analyze them:



The effect is minor, which is not surprising, since the large cut in WT had been going from 912 to 9.

We could, instead, eliminate all the observations with the four mutations (e.g., maybe they are too lethal to ever be observed?). We go to "User input", select, on the left, "data_original", rename it ("no_all_mut"), and set to 0 the A,B,C,D genotype.



The DAG structures for CBN, OT, OncoBN are preserved, but now H-ESBCN has modeled an XOR; the XOR models is not actually present, but unless there are XOR or similar phenomena, models with AND and OR cannot model the absence of A,B,C,D, given the relatively high frequencies of the rest of the genotypes (the CBN model, for example, has decreased the estimates of all the λs).

We could also, as mentioned at the beginning of this section, increase the number of WTs. Etc, etc. We will not pursue this any further here. The key message from this section is that EvAM-Tools makes it very simple to examine targeted, specific, deviations in genotype composition from the genotype composition generated by a CPM model.

5 Simulating random CPMs/evams

If we were interested in systematically examining the performance of the different methods under different models, simulating random CPM (or evam) models is crucial. This type of work (generating and analyzing large numbers of simulations) is not suited for a web app, but it can be easily done with the R package. The key function here is `random_evam`.

```
## Load the package
library(evamtools)
## For reproductibility
set.seed(3)
he_r1 <- random_evam(ngenes = 5, model = "HESBCN")
he_r1$HESBCN_model

##   From To      edge   Lambdas Relation
## 1 Root  A Root -> A 0.6656892   Single
## 2 Root  B Root -> B 1.1189358   Single
## 3    A  C    A -> C 1.8736265   Single
## 4    A  D    A -> D 2.0159447      XOR
## 5    A  E    A -> E 1.6987091   Single
## 6    B  D    B -> D 2.0159447      XOR

## Now, simulate a data set of size 200 from that model
## with 5% genotyping error

he_s1 <- sample_evam(he_r1, N = 200, obs_noise = 0.05)

## Analyze this data with all the methods except MCCBN (for speed)

he_s1_anal <- evam(he_s1$HESBCN_sampled_genotype_counts_as_data,
                   methods = c("CBN", "OT", "OncoBN",
                               "HESBCN", "MHN"))

## Show the fitted DAGs
he_s1_anal[grepl("_model", names(he_s1_anal))]

## $OT_model
##   From To      edge OT_edgeBootFreq OT_edgeWeight OT_obsMarginal
## A Root  A Root -> A              NA     0.3148883          0.345
## B Root  B Root -> B              NA     0.4069873          0.430
## D Root  D Root -> D              NA     0.3040531          0.335
## E    A  E    A -> E              NA     0.5851968          0.225
## C    E  C    E -> C              NA     0.9163967          0.210
##   OT_predMarginal
## A       0.3450000
## B       0.4300000
## D       0.3350000
## E       0.2244513
## C       0.2102331
##
```

```
## $CBN_model
##   From To      edge init_lambda final_lambda rerun_lambda CBN_edgeBootFreq
## A Root  A Root -> A    0.433470     0.433470     0.433470                NA
## B Root  B Root -> B    0.678177     0.678177     0.678177                NA
## C    E  C    E -> C   15.918836    15.918836    15.918836                NA
## D Root  D Root -> D    0.454241     0.454241     0.454241                NA
## E    A  E    A -> E    2.020337     2.020337     2.020337                NA
##
## $MCCBN_model
## [1] NA
##
## $OncoBN_model
##   From To      edge     theta Relation
## 2 Root  B Root -> B 0.4300000   Single
## 3 Root  C Root -> C 0.2100000   Single
## 1    C  A    C -> A 0.9285714   Single
## 4    B  D    B -> D 0.5045045       OR
## 5    C  D    C -> D 0.5045045       OR
## 6    C  E    C -> E 0.7857143   Single
##
## $OncoBN_fitted_model
## [1] "DBN"
##
## $HESBCN_model
##   From To      Edge    Lambdas Relation
## 1 Root  A Root -> A   1.028580   Single
## 2 Root  B Root -> B   1.097880   Single
## 3    A  C    A -> C 122.697000   Single
## 4    A  D    A -> D   7.867050      XOR
## 5    B  D    B -> D   7.867050      XOR
## 6    C  E    C -> E   0.564528       OR
## 7    D  E    D -> E   0.564528       OR
```

```
## Show MHN
he_s1_anal["MHN_theta"]
```

```
## $MHN_theta
##       A     B     C     D     E
## A -0.52  0.61  0.00 -2.48  0.00
## B -2.30 -0.23 -2.33  1.47 -0.01
## C  3.42 -0.31 -3.04 -0.24  0.00
## D  0.19 -3.53  1.23 -0.44  0.00
## E  1.31 -0.44  2.19 -0.32 -2.13
```

This is just one example; serious simulation studies would examine exhaustively a range of scenarios. And we could, of course, compare other output, such as the predicted genotype frequencies, or the probabilities of paths to the maximum (use argument `paths_max = TRUE` when calling `evam`), etc. But this should be enough to show you how EvAM-Tools can be used to systematically compare the performance of different methods in different scenarios.

6 Appendix: getting the BRCA and Ov data sets from the R console

Here we show how to obtain, from the R console, the two data sets used in section *"Analysis of cross-sectional data" (section 2)*. We simply export those data in a CSV format that can be uploaded to the web app.

```
## Load the package to access the BRCA data
library(evamtools)
data(every_which_way_data)

## You can check the names here, which are the same
## as in Suppl File S5 Text of Diaz-Uriarte & Vasallo, 2019
##  names(every_which_way_data)

write.csv(every_which_way_data[["BRCA_ba_s"]],
          file = "BRCA_ba_s.csv", row.names = FALSE,
          quote = FALSE)


## Now export the ovarian cancer CGH data
library(Oncotree)

## Loading required package:  boot

data(ov.cgh)

## Rename column names: they start with a number and
## finish on a "+" (gain) or "-" (loss), so automatically
## reading these column names removes the +/- and adds an
## "X" as the first character. Let us have columns start
## with L or G for loss/gain.

new_cn <- stringi::stri_sub(colnames(ov.cgh), 1L, 2L)
new_cn <- paste(ifelse(grepl("+", colnames(ov.cgh), fixed = TRUE),
                       "G", "L"), new_cn, sep = "")
ov2 <- ov.cgh
colnames(ov2) <- new_cn
write.csv(ov2,
          file = "ov2.csv", row.names = FALSE, quote = FALSE)
```

7 License and copyright

8 References

Cancer Genome Atlas Research Network (2012). Comprehensive molecular characterization of human colon and rectal cancer. *Nature*, **487**(7407), 330–337.

Cerami, E., Gao, J., Dogrusoz, U., Gross, B. E., Sumer, S. O., Aksoy, B. A., Jacobsen, A., Byrne, C. J., Heuer, M. L., Larsson, E., Antipin, Y., Reva, B., Goldberg, A. P., Sander, C., and Schultz, N. (2012). The cBio Cancer Genomics Portal: An Open Platform for Exploring Multidimensional Cancer Genomics Data: Figure 1. *Cancer Discovery*, **2**(5), 401–404.

Diaz-Uriarte, R. and Vasallo, C. (2019). Every which way? On predicting tumor evolution using cancer progression models. *PLOS Computational Biology*, **15**(8), e1007246.

Gao, J., Aksoy, B. A., Dogrusoz, U., Dresdner, G., Gross, B., Sumer, S. O., Sun, Y., Jacobsen, A., Sinha, R., Larsson, E., Cerami, E., Sander, C., and Schultz, N. (2013). Integrative analysis of complex cancer genomics and clinical profiles using the cBioPortal. *Science Signaling*, **6**(269), pl1.

Szabo, A. and Pappas, L. (2022). Oncotree: Estimating oncogenetic trees. R package version 0.3.4.

Package 'evamtools'

October 1, 2022

Type Package

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

Version 2.1.13

Date 2022-09-26

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 , plotly , magrittr , dplyr , tippy

Suggests testthat (>= 3.0.0)

Config/testthat/parallel true

Config/testthat/edition 3

NeedsCompilation yes

R topics documented:

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

Description

Executes all CPMS given a cross sectional data set.

Usage

```
evam(x,
     methods = c("CBN", "OT", "HESBCN", "MHN", "OncoBN", "MCCBN"),
     max_cols = 15,
     cores = detectCores(),
     paths_max = FALSE,
     mhn_opts = list(lambda = 1/nrow(x),
                     omp_threads = ifelse(cores > 1, 1, detectCores())),
     ot_opts = list(with_errors_dist_ot = TRUE),
     cbn_opts = list(
                     omp_threads = 1,
                     init_poset = "OT"
                ),
     hesbcn_opts = list(
                     MCMC_iter = 100000,
                     seed = NULL,
                     reg = c("bic", "aic", "loglik"),
                     silent = TRUE
                ),
     oncobn_opts = list(
                     model = "DBN",
                     algorithm = "DP",
                     k = 3,
                     epsilon = min(colMeans(x)/2),
                     silent = TRUE
                ),
     mccbn_opts = list(
                     model = "OT-CBN",
                     tmp_dir = NULL,
```

```
                addname = NULL,
                silent = TRUE,
                L = 100,
                sampling = c("forward", "add-remove",
                             "backward", "bernoulli", "pool"),
                max.iter = 100L,
                update.step.size = 20L,
                tol = 0.001,
                max.lambda.val = 1e+06,
                T0 = 50,
                adap.rate = 0.3,
                acceptance.rate = NULL,
                step.size = NULL,
                max.iter.asa = 10000L,
                neighborhood.dist = 1L,
                adaptive = TRUE,
                thrds = 1L,
                verbose = FALSE,
                seed = NULL)
)
```

Arguments

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

more than 1 thread will speed things much or what is the best number of threads to use; in fact, sometimes it can even slow things down, in particular if you run multiple evams in parallel.

- cbn_init_poset: Initial poset for CBN; one of "linear" or "OT" (default).

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

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

Details

Probabilities of evolutionary paths or paths of tumor progression

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

Value

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

- OT_model: Data frame with parent and descendant edges, edge weight, and observed and predicted frequencies of genes.
- OT_f_graph: The fitness graph, as a sparse matrix, with weights obtained from the edge weights (this is not a transition rate matrix). See full documentation for details.
- OT_trans_mat: Transition matrix between genotypes. This is really an abuse of what an untimed OT provides. See full documentation for details.

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

- HESBCN_model: As for CBN.
- HESBCN_parent_set: As for CBN.
- HESBCN_trans_rate_mat: As for CBN.
- HESBCN_trans_mat: As for CBN.
- HESBCN_td_trans_mat: As for CBN.
- HESBCN_predicted_genotype_freqs: As for CBN.
- HESBCN_paths_max: As for OT.
- original_data: The original data.
- analyzed_data: The data that were actually analyzed. Can differ from the original data because of the data pre-processing steps.
- genotype_id_ordered: A named vector, from 1:number of genotypes, with names the genotypes. This can be useful for sorting; WT has value 1, and genotypes are ordered by increasing number of mutations and, withing number of mutations, alphanumerically.
- all_options: All of the options used, as a list of lists.

Note

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

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

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

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

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

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

References

OT

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

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

CBN and MCCBN

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

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

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

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

MHN

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

HESBCN (PMCE)

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

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

OncoBN (DBN)

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

Conditional prediction of genotypes and probabilities of paths from CPMs

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

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

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

Reference in details

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

See Also

sample_evam, plot_evam

Examples

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

dag dag, from models with DAGs.

matrix matrix, from MHN.

plot_evam *Plot results from EvAMs (CPMs)*

Description

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

Usage

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

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

```
  fixed_vertex_size = FALSE,
  top_paths = NULL
)
```

Arguments

cpm_output	Output from the cpm
samples	Output from a call to sample_evam. Necessary if you request plot type transitions.
orientation	String. If it is not "vertical" it will be displayed with an horizontal layout. Optional.
methods	Vector of strings with the names of methods that we want to plot. If NULL, all methods with output in cpm_output. The list of available methods is OT, OncoBN, CBN, MCCBN, MHN, HESBCN.
plot_type	One of: • trans_mat: Transition matrix between genotypes (see supporting information for OT and OncoBN). This plots the object of name trans_mat from the output of evam. • trans_rate_mat: Transition rate matrix between genotypes; unavailable for OT and OncoBN. This plots the object of name trans_rate_mat from the output of evam. • obs_genotype_transitions: Observed transitions during the simulation of the sampling process. This plots the object called obs_genotype_transitions from the output of sample_evam.
label_type	Type of label to show. One of: • genotype: Displays all genes mutated • acquisition: Only displays the last gente mutated
fixed_vertex_size	Boolean. If TRUE, all nodes with have the same size; otherwise, scale them proportional to frequencies of observed data.
top_paths	Number of most relevant paths to plot. Default NULL will plot all paths. See below, Description, for details about relevance. With many genes, and particularly for MHN (or other methods, when there are no restrictions in the order of accumulation of mutations), using NULL can lead to a very long time to plot.

Value

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

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

In the top plots, in the DAGs, when a node has two or more incoming edges, color depends on the type of relationship. (With a single incoming edge, there is no difference in model behavior with

type of edge and, for consistency with OT and CBN, nodes with a Single parent have edges colored the same way as nodes with two or more parents and AND relationship).

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

Note

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

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

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

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

plot_CPMs has been deprecated. Use plot_evam.

References

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

Examples

```
dB_c1 <- matrix(
 c(
     rep(c(1, 0, 0, 0, 0), 30) #A
   , rep(c(0, 0, 1, 0, 0), 30) #C
   , rep(c(1, 1, 0, 0, 0), 20) #AB
   , rep(c(0, 0, 1, 1, 0), 20) #CD
   , rep(c(1, 1, 1, 0, 0), 10) #ABC
   , rep(c(1, 0, 1, 1, 0), 10) #ACD
   , rep(c(1, 1, 0, 0, 1), 10) #ABE
   , rep(c(0, 0, 1, 1, 1), 10) #CDE
   , rep(c(1, 1, 1, 0, 1), 10) #ABCE
   , rep(c(1, 0, 1, 1, 1), 10) #ACDE
   , rep(c(1, 1, 1, 1, 0), 5) # ABCD
   , rep(c(0, 0, 0, 0, 0), 1) # WT
 ), ncol = 5, byrow = TRUE
)
colnames(dB_c1) <- LETTERS[1:5]

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). Note that the option to request obs_genotype_transitions was removed from the web app, as it was rarely used, but lead to confusion and could increase without good reason running times. So, from the web app, we sample without using obs_genotype_transitions.

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

Usage

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

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

Arguments

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

Value

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

- sampled_genotype_counts: Counts, or absolute genotype frequencies, obtained by sampling from the predicted frequencies. See also Description, below.
- obs_genotype_transitions: Number of observed transitions between genotypes (as a sparse matrix).
- state_counts: Number of times each genotype was visited during the transitions. Column sums of observed genotype transitions are equal to state counts.
- sampled_genotype_counts_as_data: The genotypes in a matrix of 0/1. This can directly be passed as an argument for `evam`, as the input data.

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

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

Note

sample_CPMs has been deprecated. Use sample_evam.

See Also

random_evam

Examples

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

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

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


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

outS1 <- sample_evam(out, N = 1000)

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

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

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


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

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


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

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

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

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

SHINY_DEFAULTS *Defaults options for running the shiny web app*

Description

Defaults of the web app. This file was generated by running the script /inst/shiny-examples/evamtools/DEFAULTS.R

You will want to rerun it (so that the RData file is created again) whenever you make changes to it.

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

Usage

```
data(SHINY_DEFAULTS)
```

Format

Defaults values of the shiny app

max_genes Maximun number of genes allowed

min_genes Minimum number of genes allowed

ngenes Integer of default number of genes to use when building

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

all_cpms All CPMs in evamtools

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

template_data One of:

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

Index

Using OncoSimulR to get accessible genotypes and transition matrices

Ramon Diaz-Uriarte[1,2,†]

[1]Dept. 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

2022-10-01
Version a206824

Contents

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.

(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.)

[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.

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, not linked to any specific cancer progression model. In other words, given a fitness landscape (a mapping from genotypes to fitness) find the accessible genotypes and the transition matrices (not transition rate matrices) between genotypes.
  - For example, this procedure does not assume that mutations that do not kill a genotype always increase fitness or at least do not decrease it. A mutation might increase fitness in some contexts (with some other mutations) and decrease in other contexts, such as with sign and reciprocal sign epistasis.
  - This procedure **assumes SSWM (strong selection, weak mutation)**. Moreover, **we assume evolution can only move uphill in fitness**. For example, a genotype is considered not accessible if its fitness is less than, or equal to (note the "or equal to") that of its immediate ancestor, and we cannot cross fitness valleys[2].
  - 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 for CBN 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 in p. 7 of the supplementary material of 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), let us define the selective coefficient of a mutation i as the relative fitness difference that it causes along the mutational pathway.

[2]This excludes, for example, the scenarios studied in Weinreich, D. M., & Chao, L. (2005). Rapid evolutionary escape by large populations from local fitness peaks is likely in nature. Evolution; international journal of organic evolution, 59(6), 1175–1182. http://dx.doi.org/10.1111/j.0014-3820.2005.tb01769.x .

- $W_{AB} = W_A\ (1 + s_B)$ or $s_B = \frac{W_{AB} - W_A}{W_A}$.
- Using our previous example, $Pr(A \to AB) = \frac{W_{AB} - W_A}{(W_{AB} - W_A) + (W_{AC} - W_A)}$, where W_x is fitness of genotype x.
- Thus, we get from the above $Pr(A \to AB) = \frac{s_B}{s_B + s_C}$.
- (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"[3].

- 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.

- Note that this is similar to what is done 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. The difference is that in Hosseini et al. the s_i is defined as the fitness difference, not the relative fitness difference (and in Hosseini et al there is a normalizing constant, as given by eq. 8).

- 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, with additional references given.

- So when using OncoSimulR 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), scale all fitnesses by the appropriate factor (that also ensures that WT is kept at one; see, for instance, function `scale_fitness_2` in file `access_genots_from_oncosimul.R`.

- 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 $p_a(1 - p_b)$).

- It is important to emphasize that we are not claiming λ_i should be taken as equal to s_i. We are using this procedure to obtain accessible genotypes and transition probabilities

[3] 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

between genotypes, but not transition rate matrices. For example, as mentioned above, multiplying all s_i by the same constant leaves these transition probabilities unchanged[4]. But even if multiples of the λ_i result in the same transition probabilities, the transition rate matrices are different (the evolutionary process is faster or slower).

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 not observe them become fixated in the population (Gerstung et al., 2009, p. 2810: "(. . .) mutations that need to be present before mutation i can fixate.". Gerstung and Beerenwinkel, 2010, Waiting time models of cancer progression. Mathematical Population Studies, 17, 115–135; p. 126: "with steps including both mutation and clonal expansion occurring at effective rates λ_j". Beerenwinkel, N., & Sullivant, S. (2009). Markov models for accumulating mutations. Biometrika, 96(3), 645, p. 659: "In an evolutionary process, this waiting time includes the generation of the mutation plus the time it takes for the allele to reach fixation in the population" and p. 660 "The parameters λ correspond to the rate of evolution, i.e. the product of population size, mutation rate and fixation probability").

- 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: we said above that a genotype is considered not accessible if its fitness is less than, or equal to (note the "or equal to") that of its immediate ancestor.

- 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 Fitness specification with OncoSimulR: DAGs vs. 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.**"

[4] We can scale all fitness with a function like $W^* = 1 + (W - 1)\ \alpha$.
$W_A = 1 + s_A$. $W_A^* = 1 + s_A^*$. Thus $s_A^* = (W_A - 1)\ \alpha = s_A\ \alpha$.

- 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[5]). You 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 Another example about the relationship between s, λ, s_h

- 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.

[5] 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

  - 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[6], 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

[6] 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`.