`Multi-DICE` Manual

R package for constructing hierarchical co-demographic models and simulating multi-taxa summary statistic vectors

Alexander T. Xue

Department of Biology
Subprogram in Ecology, Evolutionary Biology, and Behavior
City College and Graduate Center of City University of New York
160 Convent Avenue, MR 526
New York, NY 10031

Manual version 1.0.1, January 2017

Table of Contents

Changes from Last Version

Software

Fixed critical typos: 1) temporary intermediate file `dice.sims.fsc2template` previously had placeholder term `tao` instead of `tau`, causing a non-detectable dysfunctional error in `dice.sims` function; 2) attempting to incorporate mutation rates previously had resulted in all polymorphic sites due to a regular-expression search term on the `dice.sims.fsc2template` file based on an older beta version

Updated to accommodate newest versions of `fastsimcoal2`, which: 1) require an `obs` file for simulations under the `FREQ` model, thus previously causing incompatibility; 2) output an error message when called without running, thus previously causing a nuisance error message during the check for a valid `fastsimcoal2` executable

Fixed small bug for outputted matrices `draws.zeta` and `draws.pulse.values` in `roll.dice` when only 0 is included in a `psi.prior` argument, which previously caused failure in `dice.sims` function

Manual

This section has now been added

First example command in “Demographic Syndrome Test – Extended” section previously had an argument starting with `tao`, now fixed to `tau`

Quick Start Guide

This “quick” start guide is an abbreviated instructional manual that details the procedure of conducting multi-taxa comparative inference under unified hierarchical co-demographic models with the `R` package `Multi-DICE` (`Multiple Taxa Demographic Inference of Congruency in Events`). This section is broken down into several steps, including data preparation prior and analyses downstream to utilizing `Multi-DICE`. The aim here is to be an expedited guide through the software package that allows quicker usage and should be sufficient for the simplest of cases, with no intention of being comprehensive. For more detailed information, see the following “Extended `R` Manual” as well as “Demographic Syndrome Test – Extended” sections. The last sections of “Improving ψ Inference through Fixing ζ_s Hyperprior”, “Data Partitioning to Accommodate Mixture of Demographic Syndromes”, and “Implementing Two-Event Demographic Syndromes” describe simulation studies justifying key applications of `Multi-DICE`.

0) Installation. With the `Multi-DICE-X.X.R` file – Within the `R` environment, the command below can be used to load the functions from `Multi-DICE`:

```
source(‘Multi-DICE-X.X.R’)
```

This assumes the `.R` file is in the current working directory. With the `MultiDICE_X.X.tar.gz` package – Within the `R` environment, if the library `devtools` is loaded, the command below can be used to install the package to `R`:

```
install(‘MultiDICE_X.X.tar.gz’)
```

This assumes the package is in the current working directory. From a `bash` terminal, the command below can be used to install the package to `R`:

```
R CMD INSTALL MultiDICE_X.X.tar.gz
```

This assumes the package is in the same directory.

1) Multi-taxa comparative dataset converted to per-taxon summary statistics. For genomic-scale data, the SFS is the appropriate summary statistic vector, and must be in the format: each taxon SFS is within a separate tab-delimited file, each column represents an SFS allele frequency class bin (in the order described by the manual for `fastsimcoal2` or δaδi), and no headers or labels. Values may either be in number of SNPs (double values permitted) or proportions out of the total of polymorphic bins. The SFS may be derived using the python program δaδi; keep in mind output from δaδi has extraneous lines and is space-delimited. Importantly, SNPs should be called from within the single

populations for each taxon; SNPs called across multiple populations may largely be monomorphic for single populations, and if SNPs were pruned to one per locus, to avoid linkage and attain putatively independent sites, prior to reducing multiple populations to one population, then among multiple SNPs for a locus, a SNP monomorphic within a population may be selected over a polymorphic site. Notably, the aSFS assumes independence among single-taxon SFS, thus sites used to construct each taxon SFS do not need to be homologous. For single-sequence data, the appropriate summary statistic vector is comprised of number of haplotypes, haplotype diversity, nucleotide diversity, and Tajima's *D*, in that order, and each taxon summary statistic vector must be within a separate tab-delimited file with no headers or labels. Single-sequence data may also be in sequence format using the argument `convert.sequences=T`; see "Extended `R` Manual" section for more information.

2) Demographic syndrome test. Prior to conducting multi-taxa co-demographic inference, it is prudent to assign taxa *a priori* to demographic syndromes, as well as explore prior distributions, to better inform the hierarchical co-demographic model. This can be done efficiently with `Multi-DICE` across multiple taxa and various demographic syndromes, sampling levels, and prior distributions. See "Demographic Syndrome Test – Extended" section for more information.

3) Multi-taxa coalescent simulation of per-taxon summary statistics under a unified hierarchical co-demographic model.

3a. Example commands. `Multi-DICE` can deploy many hierarchical co-demographic model variations, and this can be accomplished with a single command. For a model similar to that presented in Xue and Hickerson (2015), as well as Chan *et al.* (2014), such that the proportion of co-expanding taxa within a single synchronous pulse ζ_T varies while the remaining co-expanding taxa are temporally idiosyncratic and ζ_T as well as timing of the synchronous pulse event $\tau_{S,1}$ are values of interest, the command below can be used for simplest execution:

```
output=dice.sims(num.sims=100000, num.taxa=10, num.haploid.samples=10,
num.ind.sites=2000, tau.psi.prior=c(1), tau.zeta.prior=c(1:10)/10,
tau.shared.prior=c(1000:1000000), epsilon.idio.prior=c(1000:10000)/100000,
NE.idio.prior=c(1000:100000), fsc2path='fsc25211', output.directory='.')
```

For a model similar to that utilized in `msBayes`, such that co-expanding taxa belong to one of Ψ events, which may or may not be synchronous (*i.e.* containging $\geq$ 2 taxa), yielding the so-called "Chinese restaurant process", the command below can be used for simplest execution:

```
output=dice.sims(num.sims=100000, num.taxa=10, num.haploid.samples=10,
num.ind.sites=2000, tau.psi.prior=c(1:10), tau.zeta.prior=c(1:10)/10,
tau.shared.prior=c(1000:1000000), epsilon.idio.prior=c(1000:10000)/100000,
NE.idio.prior=c(1000:100000), idiosyncratic=F, fsc2path='fsc25211',
output.directory='.')
```

For a model such that the number of synchronous pulses ψ varies while the proportions of co-expanding taxa within these pulses ζ_s are fixed (see "Improving ψ Inference through Fixing ζ_s Hyperprior" section for more information), the command below can be used for simplest execution:

```
output=dice.sims(num.sims=100000, num.taxa=10, num.haploid.samples=10,
num.ind.sites=2000, tau.psi.prior=c(0:3), tau.zeta.prior=c(3:5,10)/10,
tau.shared.prior=c(1000:1000000), epsilon.idio.prior=c(1000:10000)/100000,
NE.idio.prior=c(1000:100000), idiosyncratic=F, fsc2path='fsc25211',
output.directory='.')
```

3b. Explanation of commands. For all of these models, there are 100,000 folded aSFS simulations of 10 taxa with 10 haploid samples and 2,000 SNPs each with prior distributions τ ~ *U*{1,000, 1,000,000}, ε ~ *U*(0.01, 0.10), and *N* ~ {1,000, 100,000}. The aSFS assumes equivalent sampling across all taxa; for haploid samples, this requires SFS down-projection (performed in `δaδi`) to the same level across all taxa, whereas for number of SNPs, simulations can assume the average (or more conservatively, the minimum) across all taxa. Since `Multi-DICE` distributions must be discretized, the ε prior distributions here are discretized by 100,000 uniform intervals across *U*(0.01, 0.10). Any discretized distribution (*e.g.* gamma, beta, log-uniform) is allowed for any `Multi-DICE` specified distribution. For example,

```
tau.shared.prior=exp(c((log(1000)*100000):(log(1000000)*100000))/100000),
epsilon.idio.prior=c(1000:10000)/100000, NE.idio.prior=c(1000:100000))
```

indicates a log-uniform distribution with the same bounds as `tau.shared.prior` above.

3c. Highlighted additional arguments/features. Taxa may be partitioned into user-specified groupings to allow differential data and model specifications, including sampling size of individuals, generation times, demographic syndrome, and taxon-specific nuisance parameter prior distributions; see "Data Partitioning to Accommodate Mixture of Demographic Syndromes" section for more information. A co-contraction model can be easily specified with an ε prior distribution that contains positive values < 1, with the inverse of an ε prior specifying the same distribution for the alternative demographic syndrome (*e.g.* the inverse of the above ε prior, 1/*U*(0.01, 0.10), is essentially the same distribution for co-contraction). Two-event size change models may also be specified; see "Implementing Two-Event Demographic Syndromes" section for more information. A buffer β on the τ prior distribution can be deployed using the arguments `tau.buffer` and `tau.idiosyncratic.buffer`; the former buffers the events specified by `tau.psi.prior` and the latter buffers idiosyncratic events that are activated by the idiosyncratic setting. Idiosyncratic taxa, as activated by the idiosyncratic setting, can be forced to experience size change either more recently or anciently than taxa within events specified by tau.psi.prior with the idiosyncratic.rule setting. Informative parameter summaries, such as $E(\tau)$ and $\Omega(\tau)$, may be outputted (*e.g.* mean.tau=T, disp.index.tau=T).

3d. Simulation execution. For dice.sims, bash commands are called upon, thus it can run only within a bash terminal environment (*e.g.* Mac, Linux), though its preceding functions, which are embedded but may be run separately, can be run in any R environment. Given that dice.sims is a wrapper function for the coalescent simulation command-line program fastsimcoal2, fastsimcoal2 must be separately user-installed. The above commands assume the fastsimcoal2 executable as well as desired directory for output simulation files are in the R current working directory; this can be modified with the arguments fsc2path and output.directory, respectively. Output may also be specified with the arguments append.sims, keep.taxa.draws, output.hyper.draws, output.taxa.draws, and keep.fsc2.files.

3e. Parallelizing. Given the nature of Multi-DICE to produce independent simulations, assuming access to multiple CPUs, it is more efficient to parallelize the workload across separately running jobs. In particular, when keep.taxa.draws=T, constructing the matrices of parameter draws within the R environment can be particularly memory-intensive for high numbers of simulations, further increasing efficiency if parallelizing. To parallelize, a Multi-DICE command, like one of the examples above, can be written to an R script, which can then be executed by simultaneous bash jobs with the command R –s <R_SCRIPT >/dev/null. The output simulation files from the parallelized jobs, which need to be directed to an individual directory per job and can be accomplished through duplicating the R script across multiple directories or calling the R script from multiple directories, can either be concatenated afterward, for example with cat within bash, or collectively inputted into R with dice.aSFS or dice.sumstats (see below for more information).

3f. Iteratively informing models. Modeling within Multi-DICE can be greatly improved in power and resolution by iteratively informing models through successively modified runs, as can be said with any modeling exercise. Specifically, exploration of partitioning (see "Data Partitioning…" section for more information), sampling projections (see "Demographic Syndrome Test…"), demographic syndromes (see "Demographic Syndrome Test…", "Implementing Two-Event…"), hyperprior distributions for Ψ/ψ and $\zeta/\zeta_S/\zeta_T$ (see "Improving ψ Inference…"), prior distributions for shared pulse parameter summaries (*e.g.* τ_S), β values, and idiosyncratic taxa behavior (*e.g.* β_i, idiosyncratic and idiosyncratic.rule settings, etc.) could potentially improve successive models significantly, specifically with the aid of power/cross-validation analyses (see below), such as through fixing specific hyper/parameter values that imply high confidence (see "Improving ψ Inference…"). For example, if a cross-validation analysis suggests high accuracy in estimating τ_S, then the estimated τ_S for that run could be used to fix that value for a successive run to increase accuracy in estimating ζ_S. For this approach, it is possible that additional simulations may want to be added to previous runs, perhaps for tesing an expanded model selection, in which case the argument `append.sims` can be helpful; similarly, if a single `Multi-DICE` command cannot accommodate the different models desired to be specified, such as different ψ values each with an exclusive set of ζ_S values, then multiple commands can be run in a loop with output directed to the same simulation files if `append.sims=T` (alternatively, may be separately run with multiple sets of output simulation files then collectively inputted into `R` with `dice.aSFS` or `dice.sumstats`; see below for more information).

3g. Genomic-scale versus single-sequence data. `Multi-DICE` lends itself nicely to comparing genomic-scale data (*i.e.* aSFS) with single-sequence data that are both collected from the same system, since both data types can be analyzed, either consecutively or simultaneously, under the same hierarchical co-demographic model with the same level of complexity and flexibility. This can be conveniently done in `Multi-DICE` as data type is irrelevant in all functions until `dice.sims`, for which the data type is easily specified in a single argument (`num.ind.sites` or `num.SNPs` versus `length.seq`) and there is no disparity in output format, and `dice.aSFS` and `dice.sumstats` operate analogously and have near identical arguments, resulting in equivalent procedures for both data types with negligible difference.

3h. See "Extended `R` Manual" section for more information, especially on data specification (including accommodating time-series/ancient samples, generation times, and single-sequence data), simulation model, hyperparameterization scheme, and specification of prior distributions.

4) Per-taxon summary statistics converted to multi-taxa summary statistic vector. Either `dice.aSFS` or `dice.sumstats` is subsequently deployed to convert, respectively, the SFS or single-sequence summary statistics, across the simulated independent taxa within output simulation files with the prefix "`dice.simulations`" and numerically indexed from 1 to *n*, into the aSFS or multi-taxa single-sequence summary statistic vector, respectively. For the aSFS, the command below can be used for simplest execution:

```
dice.aSFS(num.sims=100000, num.taxa=10, num.haploid.samples=10)
```

For the multi-taxa single-sequence summary statistic vector, the command below (note identical arguments) can be used for simplest execution:

```
dice.sumstats(num.sims=100000, num.taxa=10, num.haploid.samples=10)
```

Both of these commands are compatible with each of the example `dice.sims` commands above, and the coupling of either of these commands with any of the `dice.sims` commands demonstrates a complete execution of `Multi-DICE`, illustrating that a hierarchical co-demographic model can be easily constructed and exploited to generate multi-taxa summary statistic vector simulations within `Multi-DICE` using only two commands. For `dice.sumstats`, if `convert.sequences=T`, then similar to `dice.sims`, `bash` commands are called upon and it can run only within a `bash` terminal environment (*e.g.* Mac, Linux). The default for the above commands assumes the directory containing the output simulation files are in the `R` current working directory; this can be modified with the argument `output.directory` or `input.directory`. If there are output simulation files across multiple directories, such as from parallelization or related successive runs, then `input.directory` can be used to read in these files. Importantly, given that these functions are independent from `dice.sims` and preceding functions, `num.partitions` may be reduced in value such that heterogeneity across taxa in model specifications (*e.g.* demographic syndrome and nuisance prior distributions) were considered during simulation but ignored during multi-taxa summary statistic vector construction (see "Data Partitioning to Accommodate Mixture of Demographic Syndromes" section for more information). See "Extended `R` Manual" section for more information, especially on data specification.

5) Multi-taxa co-demographic inference. Here, `Multi-DICE` operation is complete and its output can be directed to an inferential software package, such as the `abc` or `randomForest` `R` libraries. In supplement, the multi-taxa summary statistic vector, particularly the aSFS, may also be transformed to reduce dimensionality prior to inference, such as can be performed with the `pls` `R` library, though this may have minimal or even negative effect on performance (Xue and Hickerson 2017). To accomplish inference within the `R` environment, the reference table must be constructed and the observed multi-taxa empirical dataset must be read in. For the reference table, the simulated data are already accounted for via the preceding step with `dice.aSFS`/`dice.sumstats`, thus only the draws for the values to be estimated are remaining. These draws will be within output simulation files with the prefix `"dice.sims.hyper.draws."` and suffix corresponding to the value of interest (*e.g.* `"psi.tau"`, `"zeta.tau.1"`, `"pulse.values.tau"`, `"disp.index.tau"`, etc.), and can be read into the `R` environment with `read.big.matrix` (in `bigmemory` library; recommended) or

`read.table`; matrices read in with `read.big.matrix` operate similarly to other `R` matrices, with a notable exception that matrices may only be accessed if indexed, thus to view in the entirety or to `cbind`/`rbind`, `[]` must be appended to the object name (*e.g.* `example[]`). For the observed data, `dice.aSFS`/`dice.sumstats` can be executed to convert single-taxon data to an empirical multi-taxa summary statistic vector (see “Extended `R` Manual” section for more information). Now, inference may be accomplished within the `R` environment. For hABC model selection, which may also be performed across discrete hyperparameter values, the command below (from the `abc` library) can be used for simplest execution:

```
postpr.object=postpr(target=target, index=index, sumstat=sumstat,
tol=.015, method=‘rejection’)
```

Here, `target` corresponds to the observed data, `index` corresponds to the model index (which may be a column of the matrix for values to be estimated, or user-created if order of simulations with respect to generating model is known), and `sumstat` corresponds to the simulated data (*i.e.* output from preceding step with `dice.aSFS`/`dice.sumstats`); the value for the `tol` argument results in 1,500 retained simulations to construct the posterior given the example commands above; the value for the `method` argument results in the ABC simple rejection algorithm being performed. If `summary(postpr.object)` is performed, then the model posterior distribution and Bayes factors are outputted. For hABC hyperparameter and parameter summary estimation, the command below (from the `abc` library) can be used for simplest execution:

```
abc.object=abc(target=target, param=param, sumstat=sumstat, tol=.015,
method=‘rejection’)
```

Here, the argument values correspond to the previous example, except `param` corresponds to the matrix for values to be estimated (or some subset of columns). If `summary(abc.object)` is performed, then statistics (*i.e.* minimum, 2.5% interval, median, mean, mode, 97.5% interval, maximum) of the posterior distributions for each inferred hyperparameter/parameter summary are outputted. For hRF, which is best performed across a limited number of discrete or categorical values and thus may be seen as analogous/complementary to hABC model selection, the command below (from the `randomForest` library) can be used for simplest execution:

```
rf.object=randomForest(sumstat, index, ntree=10, proximity=T)
```

Here, the argument values `sumstat` and `index` correspond to the previous examples and the value for the `ntree` argument results in 10 decision trees to construct the hRF scheme. It is advisable to cyclically subsample simulations from `sumstat` and construct sets of decision trees per cycle, which can be combined with the command below:

```
rf.combined.object=combine(rj.object[[1]],rf.object[[2]],rf.object[[3]])
```

Here, `rf.object` is a list of length = 3 with each list element an independent cycle of decision trees. This `rf.combined.object` object can then be exploited for hRF prediction with the command below:

```
rf.prediction=predict(rj.combined.object,target)
```

Here, the argument value `target` corresponds to the previous example.

6) Plotting hyperprior and posterior distributions. To plot the Ψ_τ/ψ_τ hyperprior distribution as specified by `tau.psi.prior` from the `build.dice` output, the short script below can be used:

```
hyperprior=NULL
for(i in unique(build.object$tau$draws[[1]][,1])){
 hyperprior.temp=0
 for(j in which(build.object$tau$draws[[1]][,1]==i)){
  hyperprior.temp=hyperprior.temp+sum(build.object$tau$hyperprior==j)
 }
 hyperprior=c(hyperprior,hyperprior.temp)
}
hyperprior.dist=NULL
for(i in length(hyperprior)){
 hyperprior.dist=c(hyperprior.dist,
rep(unique(build.object$tau$draws[[1]][,1])[i], hyperprior[i]))
hist(i)
```

Here, the object `build.object` contains the `build.dice` output. To plot the drawn prior from the `play.dice`/`dice.sims` output, the short script below can be used:

```
hyperprior=NULL
for(i in sort(unique(play.object$roll.object$draws.psi$tau))){
 hyperprior=c(hyperprior,sum(play.object$roll.object$draws.psi$tau==i))
}
hyperprior.dist=NULL
for(i in length(hyperprior)){
 hyperprior.dist=c(hyperprior.dist,
rep(sort(unique(play.object$roll.object$draws.psi$tau))[i],
hyperprior[i]))
hist(i)
```

Here, the object `play.object` contains the `play.dice`/`dice.sims` output; this same script can be used on the `roll.dice` output by replacing `play.object$roll.object` with `roll.object`, assuming the object `roll.object` contains the `roll.dice` output. To plot the $\zeta_{\tau,j}$/$\zeta_{\tau,s,j}$ hyperprior distribution for a given pulse and value of Ψ_τ/ψ_τ from the `build.dice` output, the short script below can be used:

```
hyperprior.total=NULL
for(i in which(build.object$tau$draws[[1]][,1]==psi)){
 hyperprior.total=c(hyperprior.total,0)
 for(j in 1:length(build.object$tau$draws)){
  hyperprior.total[length(hyperprior.total)] =
hyperprior.total[length(hyperprior.total)] +
build.object$tau$combos[[j]][[paste('pulse',psi,sep='')]][build.object$tau
$draws[[j]][i,2],pulse]
 }
}
hyperprior=NULL
for(i in sort(unique(hyperprior.total))){
 hyperprior.temp=0
 for(j in
which(build.object$tau$draws[[1]][,1]==psi)[hyperprior.total==i]){
  hyperprior.temp=hyperprior.temp+sum(build.object$tau$hyperprior==j)
 }
 hyperprior=c(hyperprior,hyperprior.temp)
}
hyperprior.dist=NULL
for(i in length(hyperprior)){
 hyperprior.dist=c(hyperprior.dist,
rep(sort(unique(hyperprior.total))[i]/total.taxa, hyperprior[i]))
hist(i)
```

Here, the objects correspond to the previous examples, and the objects psi=Ψ_τ/ψ_τ, pulse=pulse, and total.taxa=*n*. To plot the drawn prior from the play.dice/dice.sims output, the short script below can be used:

```
hyperprior.total=NULL
for(i in which(play.object$roll.object$draws.psi$tau==psi)){
 hyperprior.total=c(hyperprior.total,0)
 for(j in 1:length(play.object$roll.object$draws.zeta$tau)){

hyperprior.total[length(hyperprior.total)]=hyperprior.total[length(hyperpr
ior.total)] + play.object$roll.object$draws.zeta$tau[[j]][i,pulse]
 }
}
hyperprior=NULL
for(i in sort(unique(hyperprior.total))){
 hyperprior=c(hyperprior,sum(hyperprior.total==i))
}
hyperprior.dist=NULL
for(i in length(hyperprior)){
 hyperprior.dist=c(hyperprior.dist,
rep(sort(unique(hyperprior.total))[i]/total.taxa, hyperprior[i]))
hist(i)
```

Here, the objects correspond to the previous examples. For all of these examples, the hyperprior distributions of $\tau 2$, $\varepsilon 1$, $\varepsilon 2$, and N_E could instead be plotted by replacing `tau` with `tau2`, `epsilon`, `epsilon2`, and `NE`, respectively. To plot any posterior distribution from the `abc` output for hABC hyperparameter and parameter summary estimation, the command below can be used:

```
hist(abc.object[[1]][,C])
```

Here, the object `abc.object` contains the `abc` output, and the object `C` is a column index number that corresponds to the estimated value/posterior distribution of interest with respect to the original column order in the `param` matrix. For any of these plotting examples, the distribution can be plotted as a density plot instead of a histogram by replacing the function `hist` with the functions `plot(density)` (*e.g.* `plot(density((abc.object[[1]][,C])))`). A PCA posterior check to assess the distance/fit between the observed dataset and retained simulations, and thus determine how accurately the specficed model can reproduce the empirical data, can be accomplished using the command below:

```
PCA=princomp(abc.object[[2]],scores=T)
plot(0, 0, ylim=range(c(PCA$scores[,2],sum((target-
PCA$center)*PCA$loadings[,2]))), xlim=range(c(PCA$scores[,1],sum((target-
PCA$center)*PCA$loadings[,1]))), col='white')
points(PCA$scores[,1], PCA$scores[,2], pch=1, col='black')
points(sum((target-PCA$center)*PCA$loadings[,1]), sum((target-
PCA$center)*PCA$loadings[,2]), pch=2, col='green')
dev.off()
```

Here, the object `target` contains the observed dataset.

7) <u>Leave-one-out cross-validation.</u> In brief, “leave-one-out” cross-validation involves iteratively treating a single randomly selected simulation from the reference table as a pseudo-observed dataset (POD) and conducting inference using the remaining simulations to assess accuracy and bias given the specified model. For leave-one-out cross-validation of hABC model selection within the `R` environment, the command below (from the `abc` library) can be used for simplest execution:

```
cv4postpr.object=cv4postpr(index=index, sumstat=sumstat, nval=20,
tol=.015, method=‘rejection’)
```

Here, the argument values for `index`, `sumstat`, `tol`, and `method` correspond to the previous examples, and the value for the `nval` argument results in 20 PODs per unique model value in `index`. If `summary(cv4postpr.object)[[1]]` is performed, then the confusion matrix is outputted, and if `summary(cv4postpr.object)[[2]]` is performed, then the mean posterior matrix is outputted. For leave-one-out cross-validation of hABC hyperparameter and parameter summary estimation within the R environment, the command below (from the abc library) can be used for simplest execution:

```
cv4abc.object=cv4abc(param=param, sumstat=sumstat, nval=50, tol=.015,
method='rejection', statistic='mean')
```

Here, the argument values for param, sumstat, tol, and method correspond to the previous examples, the value for the nval argument results in 50 total PODs, and the value for the statistic argument indicates which statistical average is used for point estimations given the posterior distributions across PODs. To calculate correlation and root mean squared error between the known "true" and estimated values of PODs, the command below can be used:

```
correlation=cor(cv4abc.object[[4]][,C],cv4abc.object[[5]][[1]][,C])
rmse=((mean((abs(cv4abc.object[[4]][,C]-
cv4abc.object[[5]][[1]][,C]))^2))^0.5)
```

Here, the objects for cv4abc.object and C correspond to the previous examples. To obtain the POD indexing used for cv4postpr or cv4abc, assuming the output is in the object cv4postpr.object or cv4abc.object, the index cv4postpr.object[[2]] or cv4abc.object[[2]], respectively, can be used. This may be useful to conduct leave-one-out cross-validation of hRF using the PODs. In such case, PODs can be extracted one at a time and applied to the randomForest output with predict to obtain estimates (see above for more information). The commands above for calculating correlation and root mean squared error can then be similarly applied to these estimates. Additionally, a confusion matrix can be constructed using the short script below:

```
confusion.rf=matrix(rep(0,M^2),ncol=M)
for(i in 1:M){
 for(j in 1:M){
  confusion.rf[i,j]=sum(cv4rf.object[(((i-1)*P)+1):(i*P)]==model[j])
 }
}
```

Here, the object M is the number of unique model/hyperparameter values inputted into randomForest, the object cv4rf.object is a vector containing the POD estimates grouped by the true model/hyperparameter values, the object P is the number of PODs per unique model/hyperparameter value, and the object model is a vector containing the unique model/hyperparameter values in the order they are grouped in cv4rf.object.

build.dice {Multi-DICE} R Documentation

Constructing hyperprior for hierarchical co-demographic model

Description

`build.dice` builds a hyperprior across Ψ/ψ and ζ/ζ_S according to: 1) uniform distribution for Ψ/ψ, for ζ_T within each discrete Ψ/ψ value, and across all combinations of the vector ζ/ζ_S within each discrete ζ_T value; 2) Dirichlet-process that weighs all allowable combinations of Ψ/ψ and ζ/ζ_S according to possible combinations of taxa assignment; 3) customized distribution(s).

Usage

```
build.dice(num.taxa, num.partitions=1, tau.psi.prior=NULL,
epsilon.psi.prior=NULL, NE.psi.prior=NULL, tau.zeta.prior=NULL,
tau2.zeta.prior=NULL, epsilon.zeta.prior=NULL, epsilon2.zeta.prior=NULL,
NE.zeta.prior=NULL, tau.zeta.total.prior=NULL, tau2.zeta.total.prior=NULL,
epsilon.zeta.total.prior=NULL, epsilon2.zeta.total.prior=NULL,
NE.zeta.total.prior=NULL, dirichlet.process=F, idiosyncratic=T,
min.net.tau.zeta.total=NULL, min.net.tau2.zeta.total=NULL,
min.net.epsilon.zeta.total=NULL, min.net.epsilon2.zeta.total=NULL,
min.net.NE.zeta.total=NULL, max.net.tau.zeta.total=NULL,
max.net.tau2.zeta.total=NULL, max.net.epsilon.zeta.total=NULL,
max.net.epsilon2.zeta.total=NULL, max.net.NE.zeta.total=NULL,
min.net.tau.zeta.per.pulse=NULL, min.net.tau2.zeta.per.pulse=NULL,
min.net.epsilon.zeta.per.pulse=NULL, min.net.epsilon2.zeta.per.pulse=NULL,
min.net.NE.zeta.per.pulse=NULL, max.net.tau.zeta.per.pulse=NULL,
max.net.tau2.zeta.per.pulse=NULL, max.net.epsilon.zeta.per.pulse=NULL,
max.net.epsilon2.zeta.per.pulse=NULL, max.net.NE.zeta.per.pulse=NULL)
```

Arguments

DATA

`num.taxa`

List, vector of length = `num.partitions`, or positive integer. Number of taxa per partition. Total sum across partitions equals the total number of taxa *n* in dataset. See also `Details`. Required.

`num.partitions`

Positive integer. Number of partitions for taxa in dataset. Allows differential data and model specifications across user-specified taxa groupings, including sampling size of individuals, generation times, demographic syndrome, and taxon-specific nuisance parameter prior distributions. See also `Details`.

PRIORS

`tau.psi.prior, epsilon.psi.prior, NE.psi.prior`

List, vector, or non-negative integer. Hyperprior distribution for Ψ/ψ of τ, ε, and N_E, respectively. For τ and ε, if list of length = 2, then the first list element applies to the first more recent size change event (*e.g.* $\tau1$, $\varepsilon1$) and the second list element applies to the second more ancient size change event (*e.g.* $\tau2$, $\varepsilon2$), per taxon. The arguments(s) specified here and their according list lengths activate which taxon-specific demographic parameters are to be hyperparameterized via Ψ/ψ as well as $\zeta/\zeta_S/\zeta_T$ downstream. See also `Details`. At least one is required.

`tau.zeta.prior, tau2.zeta.prior,`
`epsilon.zeta.prior,`
`epsilon2.zeta.prior,`
`NE.zeta.prior`

List of length = `num.partitions` or 1, vector, or non-negative proportion (*i.e.* $\leq$ 1 and $\geq$ 0). Hyperprior distribution for ζ_j of $\tau1$, $\tau2$, $\varepsilon1$, $\varepsilon2$, and N_E, respectively, for each *j*th pulse from 1 to Ψ/ψ, as specified by the corresponding `psi.prior`, and per partition. See also `Details`. Required for each corresponding `psi.prior` specified, unless the maximum value in the corresponding `psi.prior` = 0.

`tau.zeta.total.prior,`
`tau2.zeta.total.prior,`
`epsilon.zeta.total.prior,`
`epsilon2.zeta.total.prior,`
`NE.zeta.total.prior`

List of length = 1, vector, or non-negative proportion (*i.e.* $\leq$ 1 and $\geq$ 0). Hyperprior distribution for ζ_T of $\tau1$, $\tau2$, $\varepsilon1$, $\varepsilon2$, and N_E, respectively. Activates a uniform hyperprior such that each discrete Ψ/ψ value, as specified by the corresponding `psi.prior`, is first weighted with equal hyperprior probability, then all discrete ζ_T values are weighted equally per Ψ/ψ value, and finally every possible associated vector ζ/ζ_S is weighted equally per ζ_T value. See also `Details`.

MODEL SPECIFICATIONS

`dirichlet.process`

Logical value. Activates a Dirichlet-process hyperprior that weighs all allowable combinations of Ψ/ψ and ζ/ζ_S according to possible combinations of taxa assignment. See also `Details`.

`idiosyncratic`

Logical value. Allows idiosyncratic taxa that freely vary *i.e.* are ungrouped from any of the pulses, as specified by the `psi.prior` arguments. See also `Details`.

`min.net.tau.zeta.total,`
`min.net.tau2.zeta.total,`
`min.net.epsilon.zeta.total,`
`min.net.epsilon2.zeta.total,`
`min.net.NE.zeta.total,`
`max.net.tau.zeta.total,`
`max.net.tau2.zeta.total,`
`max.net.epsilon.zeta.total,`
`max.net.epsilon2.zeta.total,`
`max.net.NE.zeta.total`

Non-negative proportion (*i.e.* $\leq$ 1 and $\geq$ 0). Rule for the minimum/maximum ζ_T value, across all pulses (as specified by the corresponding `psi.prior`) and partitions, for $\tau1$, $\tau2$, $\varepsilon1$, $\varepsilon2$, and N_E, respectively. See also `Details`.

`min.net.tau.zeta.per.pulse,`
`min.net.tau2.zeta.per.pulse,`
`min.net.epsilon.zeta.per.pulse,`
`min.net.epsilon2.zeta.per.pulse,`
`min.net.NE.zeta.per.pulse,`
`max.net.tau.zeta.per.pulse,`

List, vector of length = maximum value in corresponding `psi.prior`, or non-negative proportion (*i.e.* $\leq$ 1 and $\geq$ 0). Rule for the minimum/maximum ζ_j value of $\tau1$, $\tau2$, $\varepsilon1$, $\varepsilon2$, and N_E, respectively, for each *j*th pulse from 1 to Ψ/ψ (as specified by

max.net.tau2.zeta.per.pulse,
max.net.epsilon.zeta.per.pulse,
max.net.epsilon2.zeta.per.pulse,
max.net.NE.zeta.per.pulse

the corresponding psi.prior) across all partitions. See also Details.

Arguments from other Multi-DICE functions may be included here and are ignored if not applicable.

Details

Multi-DICE cannot currently accommodate models with more than one population per taxon, events aside from population size change, and more than two size change events.

For τ1 and τ2, units are in numbers of generations, and thus may only be positive integers. For ε1 and ε2, units are in ratio of size change from the ancestral effective population size to current effective population size, such that expansions are < 1 and contractions are > 1, and thus may only be positive values. For N_E, unit is in number of effective haploid individuals, and thus may only be positive integers.

For num.taxa, tau.psi.prior, epsilon.psi.prior, and NE.psi.prior, non-integer values are converted to integer values via as.integer. Similarly, after being multiplied by *n*, tau.zeta.prior, tau2.zeta.prior, epsilon.zeta.prior, epsilon2.zeta.prior, NE.zeta.prior, tau.zeta.total.prior, tau2.zeta.total.prior, epsilon.zeta.total.prior, epsilon2.zeta.total.prior, NE.zeta.total.prior, min.net.tau.zeta.total, min.net.tau2.zeta.total, min.net.epsilon.zeta.total, min.net.epsilon2.zeta.total, min.net.NE.zeta.total, max.net.tau.zeta.total, max.net.tau2.zeta.total, max.net.epsilon.zeta.total, max.net.epsilon2.zeta.total, max.net.NE.zeta.total, min.net.tau.zeta.per.pulse, min.net.tau2.zeta.per.pulse, min.net.epsilon.zeta.per.pulse, min.net.epsilon2.zeta.per.pulse, min.net.NE.zeta.per.pulse, max.net.tau.zeta.per.pulse, max.net.tau2.zeta.per.pulse, max.net.epsilon.zeta.per.pulse, max.net.epsilon2.zeta.per.pulse, and max.net.NE.zeta.per.pulse are converted to integer values via as.integer to represent *S* and S_T.

For num.taxa, min.net.tau.zeta.per.pulse, min.net.tau2.zeta.per.pulse, min.net.epsilon.zeta.per.pulse, min.net.epsilon2.zeta.per.pulse, min.net.NE.zeta.per.pulse, max.net.tau.zeta.per.pulse, max.net.tau2.zeta.per.pulse, max.net.epsilon.zeta.per.pulse, max.net.epsilon2.zeta.per.pulse, and max.net.NE.zeta.per.pulse, if list, then all list elements are concatenated to form a single vector, with the ordering within list elements and then between list elements preserved (*e.g.* for list of length = 2, with first list element of length = 2 and second list element of length = 1, the order from first to last is: 1) first vector element in first list element; 2) second vector element in first list element; 3) sole vector element in second list element).

For tau.psi.prior, epsilon.psi.prior, NE.psi.prior, tau.zeta.prior, tau2.zeta.prior, epsilon.zeta.prior, epsilon2.zeta.prior, NE.zeta.prior, tau.zeta.total.prior, tau2.zeta.total.prior, epsilon.zeta.total.prior, epsilon2.zeta.total.prior, and NE.zeta.total.prior, each list element contains an entire individual discrete distribution; if vector, then converted to list of length = 1 with all vector elements comprising the entirety of a single discrete distribution. Per list element, vector elements within (*i.e.* the discrete distribution) do not need to be in any particular order. Relatedly, each vector element is treated as an independent value, thus weighted distributions (*i.e.* not uniform) may be employed by duplicating values (*e.g.* a distribution of `c(0,1,1,1)` signifies 75% probability of drawing "1" and 25% probability of drawing "0"), allowing the specification of any discretized distribution (*e.g.* gamma, beta, log-uniform). Accordingly, a uniform distribution with no gaps for integer values would be of length = range of distribution.

For `num.taxa`, the order of vector elements corresponds to the order of partitions, and for `tau.zeta.prior`, `tau2.zeta.prior`, `epsilon.zeta.prior`, `epsilon2.zeta.prior`, and `NE.zeta.prior`, the order of list elements corresponds to the order of partitions. Additionally, if length = 1 and `num.partitions` > 1, then the sole element is used for all partitions. Similarly, if length < `num.partitions`, then the first element is used for all partitions while ignoring any remaining elements, and if length > `num.partitions`, then the remaining elements beyond length = `num.partitions` are ignored; a caution is provided when the length does not equal 1 or `num.partitions`.

For `min.net.tau.zeta.per.pulse`, `min.net.tau2.zeta.per.pulse`, `min.net.epsilon.zeta.per.pulse`, `min.net.epsilon2.zeta.per.pulse`, `min.net.NE.zeta.per.pulse`, `max.net.tau.zeta.per.pulse`, `max.net.tau2.zeta.per.pulse`, `max.net.epsilon.zeta.per.pulse`, `max.net.epsilon2.zeta.per.pulse`, and `max.net.NE.zeta.per.pulse`, the order of vector elements corresponds to the temporal order, from most recent to most ancient, of pulses (as specified by the corresponding `psi.prior`). Additionally, if length = 1 and maximum value in corresponding `psi.prior` > 1, then the sole element is used for all pulses. Similarly, if length < maximum value in corresponding `psi.prior`, then the first element is used for all pulses while ignoring any remaining elements, and if length > maximum value in corresponding `psi.prior`, then the remaining elements beyond length = maximum value in corresponding `psi.prior` are ignored; a caution is provided when the length does not equal 1 or maximum value in corresponding `psi.prior`.

If `num.partitions=`*n*, then rearrangement of bins across taxa within allele frequency classes based on descending order of the relative SNP proportions is not performed to construct the aSFS and taxon-specific inference of demographic parameters is possible. However, in general, more partitions results in more parameter space with respect to taxa samples that must be explored due to a decrease in order-independence and assumed exchangeability, thus multiple-fold more simulations must be conducted to achieve comparable accuracy in hyperparameter estimation as without partitioning.

For `tau.psi.prior`, `epsilon.psi.prior`, and `NE.psi.prior`, distinguishing between Ψ and ψ is accomplished via the corresponding `zeta.prior`, `zeta.total.prior`, `idiosyncratic` setting, and/or `min`/`max.net` `zeta.total`/`per.pulse`, except for values of 0, which are explicitly for ψ = 0 and thus indicate full idiosyncrasy. If list length > 2 for τ and ε or list length > 1 for N_E, then a caution is provided and the remaining elements beyond length = 2 for τ and ε and length = 1 for N_E are ignored. Applies across all partitions, such that it is regardless of partitioning.

For `tau.zeta.prior`, `tau2.zeta.prior`, `epsilon.zeta.prior`, `epsilon2.zeta.prior`, and `NE.zeta.prior`, if `num.partitions` > 1, may be necessary to include "0.0" as a value, but can control ζ_S and ζ_T across partitions via corresponding `zeta.total.prior` and/or `min`/`max.net zeta.total`/`per.pulse`. Attributes to each partition individually, but proportion values are out of the entirety of taxa dataset, thus if `num.partitions` > 1, then the upper bound for each partition should be the number of taxa within that partition divided by *n*. When ψ = {0, 1}, equivalent to hyperprior distribution for ζ_T. If identical across all partitions, it is more computationally efficient to specify only one *i.e.* list of length = 1, or a vector.

For `tau.zeta.total.prior`, `tau2.zeta.total.prior`, `epsilon.zeta.total.prior`, `epsilon2.zeta.total.prior`, and `NE.zeta.total.prior`, if length > 1, then remaining elements beyond the first are ignored and a caution is provided.

To build a hyperprior, `Multi-DICE` first looks if the corresponding `zeta.total.prior` is specified, then if `dirichlet.process=T`, and if neither is such case, then all possible combinations of corresponding `psi.prior` and `zeta.prior` draws are equally weighted. Therefore, if `num.partitions=1`, corresponding `psi.prior=1`, and `dirichlet.process=F`, then `tau.zeta.total.prior`, `tau2.zeta.total.prior`, `epsilon.zeta.total.prior`, `epsilon2.zeta.total.prior`, and `NE.zeta.total.prior` equivalent to corresponding `zeta.prior` and thus unnecessary to specify.

If `idiosyncratic=F`, then `min.net.tau.zeta.total`, `min.net.tau2.zeta.total`, `min.net.epsilon.zeta.total`, `min.net.epsilon2.zeta.total`, `min.net.NE.zeta.total`, `max.net.tau.zeta.total`, `max.net.tau2.zeta.total`, `max.net.epsilon.zeta.total`, `max.net.epsilon2.zeta.total`, and `max.net.NE.zeta.total`, if specified, may only equal 1.0.

When ψ = 0 *i.e.* full idiosyncrasy, the arguments `idiosyncratic`, `min.net.tau.zeta.total`, `min.net.tau2.zeta.total`, `min.net.epsilon.zeta.total`, `min.net.epsilon2.zeta.total`, `min.net.NE.zeta.total`, `min.net.tau.zeta.per.pulse`, `min.net.tau2.zeta.per.pulse`, `min.net.epsilon.zeta.per.pulse`, `min.net.epsilon2.zeta.per.pulse`, and `min.net.NE.zeta.per.pulse` are ignored.

`Multi-DICE` proceeds only if a valid draw is allowed given corresponding `num.taxa`, `psi.prior`, `zeta.prior` (*e.g.* minimum value in `psi.prior` * minimum value in `zeta.prior` $\leq$ minimum value in `num.taxa`), `zeta.total.prior`, `idiosyncratic` setting, and `min`/`max.net zeta.total`/`per.pulse`. Importantly, there is no check on if all values in `psi.prior` have a valid draw, only if there is a valid draw among any of the values.

Value

Returned value is a list object with each element attributed to a hyperparameterized demographic parameter and accordingly named (*i.e.* `tau`, `tau2`, `epsilon`, `epsilon2`, `NE`). Each of these list elements contain the following components:

`draws` List of matrices. The order of list elements/matrices corresponds to the order of partitions. Per matrix, each row represents a valid draw, with rows across matrices corresponding to each other (*i.e.* the same row number across matrices refers to the same individual draw). The first column is the Ψ/ψ drawn value as specified by the corresponding `psi.prior`. The second column is a row index number corresponding to the matrices in the `combos` component, which contains information about the ζ/ζ_S drawn value(s).

`combos` List of list of matrices. The order of list elements corresponds to the order of partitions. Each of these list elements contain another list of matrices, with these list elements/matrices corresponding to the different unique Ψ/ψ values in the corresponding `psi.prior`, in ascending order of Ψ/ψ values, and accordingly named with the prefix “pulse” concatenated with Ψ/ψ value as a suffix (*e.g.* “pulse1”, “pulse2”). Per matrix, each row represents a possible per-partition draw, each cell is a *S* drawn value, and the order of columns corresponds to the temporal order, from most recent to most ancient, of pulses (as specified by the corresponding `psi.prior`).

`hyperprior` Vector. Contains constructed hyperprior distribution, with each element an indexed draw referring to the row numbers in `draws`.

Author(s)

Alexander T. Xue

References

Chan YL, Schanzenbach D, Hickerson MJ (2014) Detecting concerted demographic response across community assemblages using hierarchical approximate Bayesian computation. *Molecular Biology and Evolution*, **31**, 2501–2515.

Hickerson MJ, Stahl E, Takebayashi N (2007) msBayes: Pipeline for testing comparative phylogeographic histories using hierarchical approximate Bayesian computation. *BMC bioinformatics*, **8**, 268.

Huang W, Takebayashi N, Qi Y, Hickerson MJ (2011) MTML-msBayes: approximate Bayesian comparative phylogeographic inference from multiple taxa and multiple loci with rate heterogeneity. *BMC bioinformatics*, **12**, 1.

Xue AT (2017) Multi-DICE Manual.

Xue AT, Hickerson MJ (2015) The aggregate site frequency spectrum for comparative population genomic inference. *Molecular Ecology*, **24**, 6223–6240.

Xue AT, Hickerson MJ (*submitted*) Multi-DICE: R package for comparative population genomic inference under multi-taxa hierarchical co-demographic models.

See Also

`roll.dice`, `play.dice`, `dice.sims`, `dice.aSFS`, `dice.sumstats`

Examples

```
#simplest execution akin to approach in Xue and Hickerson (2015)
build.dice(num.taxa=10, tau.psi.prior=c(1), tau.zeta.prior=c(1:10)/10)

#simplest execution akin to approach in software package msBayes
```

```
build.dice(num.taxa=10, tau.psi.prior=c(1:10), tau.zeta.prior=c(1:10)/10,
idiosyncratic=F)
```

roll.dice {Multi-DICE} R Documentation

Hyperprior and parameter summary prior draws for hierarchical co-demographic model

Description

`roll.dice` conducts random draws from the hyperprior distribution constructed in `build.dice`, as well as from user-specified prior distributions for shared pulse values. These shared pulse values (*e.g.* $\tau 1_s$, $\tau 2_s$, $\varepsilon 1_s$, $\varepsilon 2_s$, N_s) are parameter summaries of the taxon-specific demographic parameter values for the shared pulses, as specified by the `psi.prior` arguments. `build.dice` is embedded here and is automatically deployed if `build.object` is not specified.

Usage

```
roll.dice(num.sims, num.taxa, num.partitions=1, tau.psi.prior=NULL,
epsilon.psi.prior=NULL, NE.psi.prior=NULL, tau.zeta.prior=NULL,
tau2.zeta.prior=NULL, epsilon.zeta.prior=NULL, epsilon2.zeta.prior=NULL,
NE.zeta.prior=NULL, tau.zeta.total.prior=NULL, tau2.zeta.total.prior=NULL,
epsilon.zeta.total.prior=NULL, epsilon2.zeta.total.prior=NULL,
NE.zeta.total.prior=NULL, tau.shared.prior=NULL, tau2.shared.prior=NULL,
epsilon.shared.prior=NULL, epsilon2.shared.prior=NULL,
NE.shared.prior=NULL, dirichlet.process=F, idiosyncratic=T,
min.net.tau.zeta.total=NULL, min.net.tau2.zeta.total=NULL,
min.net.epsilon.zeta.total=NULL, min.net.epsilon2.zeta.total=NULL,
min.net.NE.zeta.total=NULL, max.net.tau.zeta.total=NULL,
max.net.tau2.zeta.total=NULL, max.net.epsilon.zeta.total=NULL,
max.net.epsilon2.zeta.total=NULL, max.net.NE.zeta.total=NULL,
min.net.tau.zeta.per.pulse=NULL, min.net.tau2.zeta.per.pulse=NULL,
min.net.epsilon.zeta.per.pulse=NULL, min.net.epsilon2.zeta.per.pulse=NULL,
min.net.NE.zeta.per.pulse=NULL, max.net.tau.zeta.per.pulse=NULL,
max.net.tau2.zeta.per.pulse=NULL, max.net.epsilon.zeta.per.pulse=NULL,
max.net.epsilon2.zeta.per.pulse=NULL, max.net.NE.zeta.per.pulse=NULL,
tau.buffer=0, tau2.buffer=0, epsilon.buffer=0, epsilon2.buffer=0,
NE.buffer=0, build.object=NULL)
```

Arguments

`num.sims` Positive integer. Number of simulations. Required.

<u>DATA</u>

`num.taxa` List, vector of length = `num.partitions`, or positive integer. Number of taxa per partition. Total sum across partitions equals the total number of taxa *n* in dataset. See also `Details`. Required.

`num.partitions` Positive integer. Number of partitions for taxa in dataset. Allows differential data and model specifications across user-specified taxa groupings, including sampling size of individuals, generation times, demographic syndrome, and taxon-specific nuisance parameter prior distributions. See also `Details`.

PRIORS

`tau.psi.prior, epsilon.psi.prior, NE.psi.prior`

List, vector, or non-negative integer. Hyperprior distribution for Ψ/ψ of τ, ε, and N_E, respectively. For τ and ε, if list of length = 2, then the first list element applies to the first more recent size change event (*e.g.* $\tau 1$, $\varepsilon 1$) and the second list element applies to the second more ancient size change event (*e.g.* $\tau 2$, $\varepsilon 2$), per taxon. The arguments(s) specified here and their according list lengths activate which taxon-specific demographic parameters are to be hyperparameterized via Ψ/ψ as well as $\zeta/\zeta_S/\zeta_T$ downstream. See also `Details`. At least one is required.

`tau.zeta.prior, tau2.zeta.prior, epsilon.zeta.prior, epsilon2.zeta.prior, NE.zeta.prior`

List of length = `num.partitions` or 1, vector, or non-negative proportion (*i.e.* ≤ 1 and ≥ 0). Hyperprior distribution for ζ_j of $\tau 1$, $\tau 2$, $\varepsilon 1$, $\varepsilon 2$, and N_E, respectively, for each *j*th pulse from 1 to Ψ/ψ, as specified by the corresponding `psi.prior`, and per partition. See also `Details`. Required for each corresponding `psi.prior` specified, unless the maximum value in the corresponding `psi.prior` = 0.

`tau.zeta.total.prior, tau2.zeta.total.prior, epsilon.zeta.total.prior, epsilon2.zeta.total.prior, NE.zeta.total.prior`

List of length = 1, vector, or non-negative proportion (*i.e.* ≤ 1 and ≥ 0). Hyperprior distribution for ζ_T of $\tau 1$, $\tau 2$, $\varepsilon 1$, $\varepsilon 2$, and N_E, respectively. Activates a uniform hyperprior such that each discrete Ψ/ψ value, as specified by the corresponding `psi.prior`, is first weighted with equal hyperprior probability, then all discrete ζ_T values are weighted equally per Ψ/ψ value, and finally every possible associated vector ζ/ζ_S is weighted equally per ζ_T value. See also `Details`.

`tau.shared.prior, tau2.shared.prior, epsilon.shared.prior, epsilon2.shared.prior, NE.shared.prior`

List, vector, or positive value. Prior distribution for the demographic parameter summaries $\tau 1_S$, $\tau 2_S$, $\varepsilon 1_S$, $\varepsilon 2_S$, and N_S, respectively (or $\tau 1$, $\tau 2$, $\varepsilon 1$, $\varepsilon 2$, and N, respectively, if corresponding `psi.prior` specifies Ψ). See also `Details`. Required for each corresponding `psi.prior` specified, unless the maximum value in the corresponding `psi.prior` = 0.

MODEL SPECIFICATIONS

`dirichlet.process`

Logical value. Activates a Dirichlet-process hyperprior that weighs all allowable combinations of Ψ/ψ and ζ/ζ_S according to possible combinations of taxa assignment. See also `Details`.

`idiosyncratic`

Logical value. Allows idiosyncratic taxa that freely vary *i.e.* are ungrouped from any of the pulses, as specified by the psi.prior arguments. See also Details.

min.net.tau.zeta.total,
min.net.tau2.zeta.total,
min.net.epsilon.zeta.total,
min.net.epsilon2.zeta.total,
min.net.NE.zeta.total,
max.net.tau.zeta.total,
max.net.tau2.zeta.total,
max.net.epsilon.zeta.total,
max.net.epsilon2.zeta.total,
max.net.NE.zeta.total

Non-negative proportion (*i.e.* ≤ 1 and ≥ 0). Rule for the minimum/maximum ζ_T value, across all pulses (as specified by the corresponding psi.prior) and partitions, for τ1, τ2, ε1, ε2, and N_E, respectively. See also Details.

min.net.tau.zeta.per.pulse,
min.net.tau2.zeta.per.pulse,
min.net.epsilon.zeta.per.pulse,
min.net.epsilon2.zeta.per.pulse,
min.net.NE.zeta.per.pulse,
max.net.tau.zeta.per.pulse,
max.net.tau2.zeta.per.pulse,
max.net.epsilon.zeta.per.pulse,
max.net.epsilon2.zeta.per.pulse,
max.net.NE.zeta.per.pulse

List, vector of length = maximum value in corresponding psi.prior, or non-negative proportion (*i.e.* ≤ 1 and ≥ 0). Rule for the minimum/maximum ζ_j value of τ1, τ2, ε1, ε2, and N_E, respectively, for each *j*th pulse from 1 to Ψ/ψ (as specified by the corresponding psi.prior) across all partitions. See also Details.

tau.buffer, tau2.buffer,
epsilon.buffer, epsilon2.buffer,
NE.buffer

Non-negative value or function. Pulse buffer β of the demographic parameter summaries $\tau 1_S$, $\tau 2_S$, $\varepsilon 1_S$, $\varepsilon 2_S$, and N_S, respectively. See also Details.

OBJECTS FROM PRECEDING FUNCTIONS

build.object

Output from function build.dice. See also Details.

Arguments from other Multi-DICE functions may be included here and are ignored if not applicable.

Details

Multi-DICE cannot currently accommodate models with more than one population per taxon, events aside from population size change, and more than two size change events.

For τ1 and τ2, units are in numbers of generations, and thus may only be positive integers. For ε1 and ε2, units are in ratio of size change from the ancestral effective population size to current effective population size, such that expansions are < 1 and contractions are > 1, and thus may only be positive values. For N_E, unit is in number of effective haploid individuals, and thus may only be positive integers.

For num.taxa, tau.psi.prior, epsilon.psi.prior, NE.psi.prior, tau.shared.prior, tau2.shared.prior, NE.shared.prior, tau.buffer, tau2.buffer, and NE.buffer, non-integer values are converted to integer values via as.integer. Similarly, after being multiplied by *n*, tau.zeta.prior, tau2.zeta.prior, epsilon.zeta.prior, epsilon2.zeta.prior, NE.zeta.prior, tau.zeta.total.prior, tau2.zeta.total.prior, epsilon.zeta.total.prior, epsilon2.zeta.total.prior, NE.zeta.total.prior, min.net.tau.zeta.total, min.net.tau2.zeta.total, min.net.epsilon.zeta.total, `min.net.epsilon2.zeta.total`, `min.net.NE.zeta.total`, `max.net.tau.zeta.total`, `max.net.tau2.zeta.total`, `max.net.epsilon.zeta.total`, `max.net.epsilon2.zeta.total`, `max.net.NE.zeta.total`, `min.net.tau.zeta.per.pulse`, `min.net.tau2.zeta.per.pulse`, `min.net.epsilon.zeta.per.pulse`, `min.net.epsilon2.zeta.per.pulse`, `min.net.NE.zeta.per.pulse`, `max.net.tau.zeta.per.pulse`, `max.net.tau2.zeta.per.pulse`, `max.net.epsilon.zeta.per.pulse`, `max.net.epsilon2.zeta.per.pulse`, and `max.net.NE.zeta.per.pulse` are converted to integer values via `as.integer` to represent S and S_T.

For `num.taxa`, `min.net.tau.zeta.per.pulse`, `min.net.tau2.zeta.per.pulse`, `min.net.epsilon.zeta.per.pulse`, `min.net.epsilon2.zeta.per.pulse`, `min.net.NE.zeta.per.pulse`, `max.net.tau.zeta.per.pulse`, `max.net.tau2.zeta.per.pulse`, `max.net.epsilon.zeta.per.pulse`, `max.net.epsilon2.zeta.per.pulse`, and `max.net.NE.zeta.per.pulse`, if list, then all list elements are concatenated to form a single vector, with the ordering within list elements and then between list elements preserved (*e.g.* for list of length = 2, with first list element of length = 2 and second list element of length = 1, the order from first to last is: 1) first vector element in first list element; 2) second vector element in first list element; 3) sole vector element in second list element).

For `tau.psi.prior`, `epsilon.psi.prior`, `NE.psi.prior`, `tau.zeta.prior`, `tau2.zeta.prior`, `epsilon.zeta.prior`, `epsilon2.zeta.prior`, `NE.zeta.prior`, `tau.zeta.total.prior`, `tau2.zeta.total.prior`, `epsilon.zeta.total.prior`, `epsilon2.zeta.total.prior`, `NE.zeta.total.prior`, `tau.shared.prior`, `tau2.shared.prior`, `epsilon.shared.prior`, `epsilon2.shared.prior`, and `NE.shared.prior`, each list element contains an entire individual discrete distribution; if vector, then converted to list of length = 1 with all vector elements comprising the entirety of a single discrete distribution. Per list element, vector elements within (*i.e.* the discrete distribution) do not need to be in any particular order. Relatedly, each vector element is treated as an independent value, thus weighted distributions (*i.e.* not uniform) may be employed by duplicating values (*e.g.* a distribution of `c(0,1,1,1)` signifies 75% probability of drawing “1” and 25% probability of drawing “0”), allowing the specification of any discretized distribution (*e.g.* gamma, beta, log-uniform). Accordingly, a uniform distribution with no gaps for integer values would be of length = range of distribution.

For `num.taxa`, the order of vector elements corresponds to the order of partitions, and for `tau.zeta.prior`, `tau2.zeta.prior`, `epsilon.zeta.prior`, `epsilon2.zeta.prior`, and `NE.zeta.prior`, the order of list elements corresponds to the order of partitions. Additionally, if length = 1 and `num.partitions` > 1, then the sole element is used for all partitions. Similarly, if length < `num.partitions`, then the first element is used for all partitions while ignoring any remaining elements, and if length > `num.partitions`, then the remaining elements beyond length = `num.partitions` are ignored; a caution is provided when the length does not equal 1 or `num.partitions`.

For `min.net.tau.zeta.per.pulse`, `min.net.tau2.zeta.per.pulse`, `min.net.epsilon.zeta.per.pulse`, `min.net.epsilon2.zeta.per.pulse`, `min.net.NE.zeta.per.pulse`, `max.net.tau.zeta.per.pulse`, `max.net.tau2.zeta.per.pulse`, `max.net.epsilon.zeta.per.pulse`, max.net.epsilon2.zeta.per.pulse, and max.net.NE.zeta.per.pulse, the order of vector elements corresponds to the temporal order, from most recent to most ancient, of pulses (as specified by the corresponding psi.prior), and for tau.shared.prior, tau2.shared.prior, epsilon.shared.prior, epsilon2.shared.prior, and NE.shared.prior, the order of list elements corresponds to the temporal order, from most recent to most ancient, of pulses (as specified by the corresponding psi.prior). Additionally, if length = 1 and maximum value in corresponding psi.prior > 1, then the sole element is used for all pulses. Similarly, if length < maximum value in corresponding psi.prior, then the first element is used for all pulses while ignoring any remaining elements, and if length > maximum value in corresponding psi.prior, then the remaining elements beyond length = maximum value in corresponding psi.prior are ignored. A caution is provided when the length does not equal 1 or maximum value in corresponding psi.prior, except for the shared.prior arguments, which provide a caution if length > 1 and length < maximum value in corresponding psi.prior, but may have additional list elements for idiosyncratic distributions (not utilized here; see play.dice).

If num.partitions=*n*, then rearrangement of bins across taxa within allele frequency classes based on descending order of the relative SNP proportions is not performed to construct the aSFS and taxon-specific inference of demographic parameters is possible. However, in general, more partitions results in more parameter space with respect to taxa samples that must be explored due to a decrease in order-independence and assumed exchangeability, thus multiple-fold more simulations must be conducted to achieve comparable accuracy in hyperparameter estimation as without partitioning.

For tau.psi.prior, epsilon.psi.prior, and NE.psi.prior, distinguishing between Ψ and ψ is accomplished via the corresponding zeta.prior, zeta.total.prior, idiosyncratic setting, and/or min/max.net zeta.total/per.pulse, except for values of 0, which are explicitly for $\psi = 0$ and thus indicate full idiosyncrasy. If list length > 2 for τ and ε or list length > 1 for N_E, then a caution is provided and the remaining elements beyond length = 2 for τ and ε and length = 1 for N_E are ignored. Applies across all partitions, such that it is regardless of partitioning.

For tau.zeta.prior, tau2.zeta.prior, epsilon.zeta.prior, epsilon2.zeta.prior, and NE.zeta.prior, if num.partitions > 1, may be necessary to include “0.0” as a value, but can control ζ_S and ζ_T across partitions via corresponding zeta.total.prior and/or min/max.net zeta.total/per.pulse. Attributes to each partition individually, but proportion values are out of the entirety of taxa dataset, thus if num.partitions > 1, then the upper bound for each partition should be the number of taxa within that partition divided by *n*. When $\psi = \{0, 1\}$, equivalent to hyperprior distribution for ζ_T. If identical across all partitions, it is more computationally efficient to specify only one *i.e.* list of length = 1, or a vector.

For tau.zeta.total.prior, tau2.zeta.total.prior, epsilon.zeta.total.prior, epsilon2.zeta.total.prior, and NE.zeta.total.prior, if length > 1, then remaining elements beyond the first are ignored and a caution is provided.

To build a hyperprior, Multi-DICE first looks if the corresponding zeta.total.prior is specified, then if dirichlet.process=T, and if neither is such case, then all possible combinations of corresponding psi.prior and zeta.prior draws are equally weighted. Therefore, if num.partitions=1, corresponding psi.prior=1, and dirichlet.process=F, then tau.zeta.total.prior, tau2.zeta.total.prior, epsilon.zeta.total.prior, epsilon2.zeta.total.prior, and NE.zeta.total.prior equivalent to corresponding zeta.prior and thus unnecessary to specify.

For tau.shared.prior, tau2.shared.prior, epsilon.shared.prior, epsilon2.shared.prior, and NE.shared.prior, each successive list element/distribution must have a greater minimum and maximum value than its preceding list elements/distributions. If multiple distributions are utilized for every potential pulse and these distributions overlap in their bounds, running time may slow since, in this case, draws are made from all the distributions independently and then checked if abiding by ordering and buffering, with re-draws if not.

If idiosyncratic=F, then min.net.tau.zeta.total, min.net.tau2.zeta.total, min.net.epsilon.zeta.total, min.net.epsilon2.zeta.total, min.net.NE.zeta.total, max.net.tau.zeta.total, max.net.tau2.zeta.total, max.net.epsilon.zeta.total, max.net.epsilon2.zeta.total, and max.net.NE.zeta.total, if specified, may only equal 1.0.

For tau.buffer, tau2.buffer, epsilon.buffer, epsilon2.buffer, and NE.buffer, if writing a function, there can be only one argument, which is for the value of a particular draw from the corresponding shared.prior, and the output must be a vector of discrete values that are buffered out of the corresponding shared.prior given that particular draw. For epsilon.buffer and epsilon2.buffer, given that these are not integers, it is imperative that the output values are on the same scale/interval/significant figures as the corresponding shared.prior since values are buffered out only if they are exactly equal. There is no check on functionality for buffer functions, thus it is highly recommended that any function is thoroughly user-tested.

If build.object is provided, then the arguments num.taxa, tau.zeta.prior, tau2.zeta.prior, epsilon.zeta.prior, epsilon2.zeta.prior, NE.zeta.prior, tau.zeta.total.prior, tau2.zeta.total.prior, epsilon.zeta.total.prior, epsilon2.zeta.total.prior, NE.zeta.total.prior, dirichlet.process, idiosyncratic, min.net.tau.zeta.total, min.net.tau2.zeta.total, min.net.epsilon.zeta.total, min.net.epsilon2.zeta.total, min.net.NE.zeta.total, max.net.tau.zeta.total, max.net.tau2.zeta.total, max.net.epsilon.zeta.total, max.net.epsilon2.zeta.total, max.net.NE.zeta.total, min.net.tau.zeta.per.pulse, min.net.tau2.zeta.per.pulse, min.net.epsilon.zeta.per.pulse, min.net.epsilon2.zeta.per.pulse, min.net.NE.zeta.per.pulse, max.net.tau.zeta.per.pulse, max.net.tau2.zeta.per.pulse, max.net.epsilon.zeta.per.pulse, max.net.epsilon2.zeta.per.pulse, and max.net.NE.zeta.per.pulse are ignored here.

When $\psi = 0$ *i.e.* full idiosyncrasy, the arguments idiosyncratic, min.net.tau.zeta.total, min.net.tau2.zeta.total, min.net.epsilon.zeta.total, min.net.epsilon2.zeta.total, min.net.NE.zeta.total, min.net.tau.zeta.per.pulse, min.net.tau2.zeta.per.pulse, min.net.epsilon.zeta.per.pulse, min.net.epsilon2.zeta.per.pulse, and min.net.NE.zeta.per.pulse are ignored.

`Multi-DICE` proceeds only if a valid draw is allowed given corresponding `num.taxa`, `psi.prior`, `zeta.prior` (*e.g.* minimum value in `psi.prior` * minimum value in `zeta.prior` $\leq$ minimum value in `num.taxa`), `zeta.total.prior`, `shared.prior`, `idiosyncratic` setting, `min`/`max.net` `zeta.total`/`per.pulse`, and `buffer`. Importantly, there is no check on if all values in `psi.prior` have a valid draw, only if there is a valid draw among any of the values. Additionally, the range of a `shared.prior` argument must be greater than its corresponding $(\Psi/\psi - 1)(2\beta + 1)$. For `tau2.shared.prior`, consideration must also be given to any overlap with `tau.shared.prior`.

Value

Returned value is a list object with the following components:

`draws.psi` List of matrices. Each list element/matrix is attributed to a hyperparameterized demographic parameter and accordingly named (*i.e.* `tau`, `tau2`, `epsilon`, `epsilon2`, `NE`). Per matrix, there is a single column and each cell is the Ψ/ψ value drawn from the corresponding `psi.prior` for that simulation.

`draws.zeta` List of list of matrices. Each list element is attributed to a hyperparameterized demographic parameter and accordingly named (*i.e.* `tau`, `tau2`, `epsilon`, `epsilon2`, `NE`). Each of these list elements contain another list of matrices, with the order of these list elements/matrices corresponding to the order of partitions. Per matrix, each cell is the *S* value drawn from the corresponding `zeta.prior` for that simulation.

`draws.pulse.values` List of matrices. Each list element/matrix is attributed to a hyperparameterized demographic parameter and accordingly named (*i.e.* `tau`, `tau2`, `epsilon`, `epsilon2`, `NE`). Per matrix, each cell is the shared pulse value drawn from the corresponding `shared.prior` for that simulation.

Each row across matrices in `draws.psi`, `draws.zeta`, and `draws.pulse.values` represents an individual simulation and these rows correspond to each other *i.e.* the same row number across matrices refers to the same simulation. For matrices in `draws.zeta` and `draws.pulse.values`, the number of columns is the maximum value in the corresponding `psi.prior` that allows a valid draw, the order of columns corresponds to the temporal order, from most recent to most ancient, of pulses (as specified by the corresponding `psi.prior`), and non-applicable cells (*i.e.* columns beyond the corresponding `draws.psi` value) contain the value 0.

Author(s)

Alexander T. Xue

References

Chan YL, Schanzenbach D, Hickerson MJ (2014) Detecting concerted demographic response across community assemblages using hierarchical approximate Bayesian computation. *Molecular Biology and Evolution*, **31**, 2501–2515.

Hickerson MJ, Stahl E, Takebayashi N (2007) msBayes: Pipeline for testing comparative

phylogeographic histories using hierarchical approximate Bayesian computation. *BMC bioinformatics*, **8**, 268.
Huang W, Takebayashi N, Qi Y, Hickerson MJ (2011) MTML-msBayes: approximate Bayesian comparative phylogeographic inference from multiple taxa and multiple loci with rate heterogeneity. *BMC bioinformatics*, **12**, 1.
Xue AT (2017) Multi-DICE Manual.
Xue AT, Hickerson MJ (2015) The aggregate site frequency spectrum for comparative population genomic inference. *Molecular Ecology*, **24**, 6223–6240.
Xue AT, Hickerson MJ (*submitted*) Multi-DICE: R package for comparative population genomic inference under multi-taxa hierarchical co-demographic models.

See Also

`build.dice`, `play.dice`, `dice.sims`, `dice.aSFS`, `dice.sumstats`

Examples

```
#simplest execution akin to approach in Xue and Hickerson (2015)
roll.dice(num.sims=5, num.taxa=10, tau.psi.prior=c(1),
tau.zeta.prior=c(1:10)/10, tau.shared.prior=c(1000:1000000))

#simplest execution akin to approach in Xue and Hickerson (2015); ln U
distribution applied on tau.shared.prior, with 100,000 intervals
discretized uniformly across ln(tau.shared.prior)
roll.dice(num.sims=5, num.taxa=10, tau.psi.prior=c(1),
tau.zeta.prior=c(1:10)/10,
tau.shared.prior=exp(c((log(1000)*100000):(log(1000000)*100000))/100000))

#simplest execution akin to approach in Xue and Hickerson (2015); assuming
build.dice was previously performed and the output was directed to object
build.object
roll.dice(num.sims=5, tau.psi.prior=c(1),
tau.shared.prior=c(1000:1000000), build.object=build.object)

#simplest execution akin to approach in software package msBayes
roll.dice(num.sims=5, num.taxa=10, tau.psi.prior=c(1:10),
tau.zeta.prior=c(1:10)/10, tau.shared.prior=c(1000:1000000),
idiosyncratic=F)
```

play.dice {Multi-DICE} R Documentation

Idiosyncratic/nuisance prior draws for hierarchical co-demographic model

Description

`play.dice` conducts random draws from user-specified prior distributions for idiosyncratic and nuisance values, as well as determine parameter summary values. Nuisance values are for taxon-specific demographic parameters that are not hyperparameterized (*i.e.* not specified in the corresponding `psi.prior` argument and therefore values are not grouped into shared pulses) and are therefore drawn independently across taxa, similar to idiosyncratic draws. `build.dice` and `roll.dice` are embedded here and are automatically deployed if `build.object`/`roll.object` and `roll.object`, respectively, are not specified.

Usage

```
play.dice(num.sims, num.taxa, num.partitions=1, tau.psi.prior=NULL,
epsilon.psi.prior=NULL, NE.psi.prior=NULL, tau.zeta.prior=NULL,
tau2.zeta.prior=NULL, epsilon.zeta.prior=NULL, epsilon2.zeta.prior=NULL,
NE.zeta.prior=NULL, tau.zeta.total.prior=NULL, tau2.zeta.total.prior=NULL,
epsilon.zeta.total.prior=NULL, epsilon2.zeta.total.prior=NULL,
NE.zeta.total.prior=NULL, tau.shared.prior=NULL, tau2.shared.prior=NULL,
epsilon.shared.prior=NULL, epsilon2.shared.prior=NULL,
NE.shared.prior=NULL, tau.idio.prior=NULL, tau2.idio.prior=NULL,
epsilon.idio.prior=NULL, epsilon2.idio.prior=NULL, NE.idio.prior=NULL,
linked.param=NULL, attached.hyper=NULL, linked.param.partition=NULL,
attached.hyper.pulse=NULL, linked.param.prior=NULL,
linked.param.fixed=NULL, anchor.prior=NULL, change.prior=NULL,
exponential.growth.rate.prior=NULL, exponential.growth.rate.prior2=NULL,
dirichlet.process=F, idiosyncratic=T, min.net.tau.zeta.total=NULL,
min.net.tau2.zeta.total=NULL, min.net.epsilon.zeta.total=NULL,
min.net.epsilon2.zeta.total=NULL, min.net.NE.zeta.total=NULL,
max.net.tau.zeta.total=NULL, max.net.tau2.zeta.total=NULL,
max.net.epsilon.zeta.total=NULL, max.net.epsilon2.zeta.total=NULL,
max.net.NE.zeta.total=NULL, min.net.tau.zeta.per.pulse=NULL,
min.net.tau2.zeta.per.pulse=NULL, min.net.epsilon.zeta.per.pulse=NULL,
min.net.epsilon2.zeta.per.pulse=NULL, min.net.NE.zeta.per.pulse=NULL,
max.net.tau.zeta.per.pulse=NULL, max.net.tau2.zeta.per.pulse=NULL,
max.net.epsilon.zeta.per.pulse=NULL, max.net.epsilon2.zeta.per.pulse=NULL,
max.net.NE.zeta.per.pulse=NULL, tau.buffer=0, tau2.buffer=0,
epsilon.buffer=0, epsilon2.buffer=0, NE.buffer=0,
tau.idiosyncratic.buffer=NULL, tau2.idiosyncratic.buffer=NULL,
epsilon.idiosyncratic.buffer=NULL, epsilon2.idiosyncratic.buffer=NULL,
NE.idiosyncratic.buffer=NULL, idiosyncratic.rule='none', num.changes=1,
flip=F, net.zeta.total=F, net.zeta.per.pulse=F, mean.tau.shared=F,
mean.tau2.shared=F, mean.epsilon.shared=F, mean.epsilon2.shared=F,
mean.NE.shared=F, mean.tau=F, mean.tau2=F, mean.epsilon=F,
mean.epsilon2=F, mean.NE=F, disp.index.tau.shared=F,
disp.index.tau2.shared=F, disp.index.epsilon.shared=F,
disp.index.epsilon2.shared=F, disp.index.NE.shared=F, disp.index.tau=F,
disp.index.tau2=F, disp.index.epsilon=F, disp.index.epsilon2=F,
disp.index.NE=F, build.object=NULL, roll.object=NULL)
```

Arguments

`num.sims`

Positive integer. Number of simulations. Required.

DATA

`num.taxa`

List, vector of length = `num.partitions`, or positive integer. Number of taxa per partition. Total sum across partitions equals the total number of taxa *n* in dataset. See also `Details`. Required.

`num.partitions`

Positive integer. Number of partitions for taxa in dataset. Allows differential data and model specifications across user-specified taxa groupings, including sampling size of individuals, generation times, demographic syndrome, and taxon-specific nuisance parameter prior distributions. See also `Details`.

PRIORS

`tau.psi.prior, epsilon.psi.prior, NE.psi.prior`

List, vector, or non-negative integer. Hyperprior distribution for Ψ/ψ of τ, ε, and N_E, respectively. For τ and ε, if list of length = 2, then the first list element applies to the first more recent size change event (*e.g.* $\tau1$, $\varepsilon1$) and the second list element applies to the second more ancient size change event (*e.g.* $\tau2$, $\varepsilon2$), per taxon. The arguments(s) specified here and their according list lengths activate which taxon-specific demographic parameters are to be hyperparameterized via Ψ/ψ as well as $\zeta/\zeta_S/\zeta_T$ downstream. See also `Details`. At least one is required.

`tau.zeta.prior, tau2.zeta.prior, epsilon.zeta.prior, epsilon2.zeta.prior, NE.zeta.prior`

List of length = `num.partitions` or 1, vector, or non-negative proportion (*i.e.* $\le$ 1 and $\ge$ 0). Hyperprior distribution for ζ_j of $\tau1$, $\tau2$, $\varepsilon1$, $\varepsilon2$, and N_E, respectively, for each *j*th pulse from 1 to Ψ/ψ, as specified by the corresponding `psi.prior`, and per partition. See also `Details`. Required for each corresponding `psi.prior` specified, unless the maximum value in the corresponding `psi.prior` = 0.

`tau.zeta.total.prior, tau2.zeta.total.prior, epsilon.zeta.total.prior, epsilon2.zeta.total.prior, NE.zeta.total.prior`

List of length = 1, vector, or non-negative proportion (*i.e.* $\le$ 1 and $\ge$ 0). Hyperprior distribution for ζ_T of $\tau1$, $\tau2$, $\varepsilon1$, $\varepsilon2$, and N_E, respectively. Activates a uniform hyperprior such that each discrete Ψ/ψ value, as specified by the corresponding `psi.prior`, is first weighted with equal hyperprior probability, then all discrete ζ_T values are weighted equally per Ψ/ψ value, and finally every possible associated vector ζ/ζ_S is weighted equally per ζ_T value. See also `Details`.

`tau.shared.prior, tau2.shared.prior, epsilon.shared.prior, epsilon2.shared.prior, NE.shared.prior`

List, vector, or positive value. Prior distribution for the demographic parameter summaries $\tau1_S$, $\tau2_S$, $\varepsilon1_S$, $\varepsilon2_S$, and N_S, respectively (or $\tau1$, $\tau2$, $\varepsilon1$, $\varepsilon2$, and N, respectively, if corresponding `psi.prior` specifies Ψ). See also `Details`. Required for each corresponding `psi.prior` specified, unless the maximum value in the corresponding `psi.prior` = 0.

`tau.idio.prior, tau2.idio.prior, epsilon.idio.prior, epsilon2.idio.prior, NE.idio.prior`

List of length = `num.partitions` or 1, vector, or positive value. Prior distribution for the taxon-specific demographic parameters $\tau1_i$, $\tau2_i$, $\varepsilon1_i$, $\varepsilon2_i$, and N_i, respectively for idiosyncratic values, and $\tau1$, $\tau2$, $\varepsilon1$, $\varepsilon2$, and N, respectively for nuisance values. See also `Details`.

`linked.param, attached.hyper`

List, vector, or character string, with possible values being the names of the demographic parameters (*i.e.* "`tau`", "`tau2`", "`epsilon`", "`epsilon2`", "`NE`"). Activates nuisance parameters in `linked.param` to have prior distributions be linked to hyperparameterized demographic parameters in `attached.hyper`, such that prior distributions may differentiate across pulses and idiosyncratic taxa with respect to the hyperparameterized demographic parameter in `attached.hyper`. See also `Details`.

`linked.param.partition, attached.hyper.pulse`

List of length = length of `linked.param` or 1, vector, or positive integer. The partitions in the linked nuisance parameter, and the pulses in the attached hyperparameterized demographic parameter, for which each element in `linked.param` and `attached.hyper`, respectively, applies. Each list element may contain multiple partitions/pulses, respectively. See also `Details`.

`linked.param.prior`

List of length = length of `linked.param` or 1, vector, or positive value. Prior distribution for the nuisance demographic parameter in each element of `linked.param`. See also `Details`.

`linked.param.fixed`

List, vector of length = length of `linked.param`, or logical value. Activates a fixed nuisance demographic parameter value for all taxa to which the corresponding elements in `linked.param.partition` and `attached.hyper.pulse` apply. See also `Details`.

`anchor.prior, change.prior`

List, vector, or positive integer. Prior distribution for $\tau2$ based on its difference δ with $\tau1$. This difference value can be assigned to synchronous/shared pulses in $\tau1_S$, as specified by `tau.psi.prior`, and accordingly inferred as a parameter summary vector δ_S (`anchor.prior`), or applied independently across taxa as an idiosyncratic or nuisance value (`change.prior`). See also `Details`.

```
exponential.growth.rate.prior,
exponential.growth.rate.prior2
```

List of length = `num.partitions` or 1, vector, or double value. Prior distribution for the nuisance taxon-specific parameters *r*1 and *r*2, respectively. Activates exponential growth model $N_\tau = N_0 * e^{(r * \tau)}$ for the first and second event, respectively, instead of instantaneous growth. Negative values indicate expansion and positive values indicate contraction. See also `Details`.

MODEL SPECIFICATIONS

```
dirichlet.process
```

Logical value. Activates a Dirichlet-process hyperprior that weighs all allowable combinations of Ψ/ψ and ζ/ζ_S according to possible combinations of taxa assignment. See also `Details`.

```
idiosyncratic
```

Logical value. Allows idiosyncratic taxa that freely vary *i.e.* are ungrouped from any of the pulses, as specified by the `psi.prior` arguments. See also `Details`.

```
min.net.tau.zeta.total,
min.net.tau2.zeta.total,
min.net.epsilon.zeta.total,
min.net.epsilon2.zeta.total,
min.net.NE.zeta.total,
max.net.tau.zeta.total,
max.net.tau2.zeta.total,
max.net.epsilon.zeta.total,
max.net.epsilon2.zeta.total,
max.net.NE.zeta.total
```

Non-negative proportion (*i.e.* $\leq$ 1 and $\geq$ 0). Rule for the minimum/maximum ζ_T value, across all pulses (as specified by the corresponding `psi.prior`) and partitions, for τ1, τ2, ε1, ε2, and N_E, respectively. See also `Details`.

```
min.net.tau.zeta.per.pulse,
min.net.tau2.zeta.per.pulse,
min.net.epsilon.zeta.per.pulse,
min.net.epsilon2.zeta.per.pulse,
min.net.NE.zeta.per.pulse,
max.net.tau.zeta.per.pulse,
max.net.tau2.zeta.per.pulse,
max.net.epsilon.zeta.per.pulse,
max.net.epsilon2.zeta.per.pulse,
max.net.NE.zeta.per.pulse
```

List, vector of length = maximum value in corresponding `psi.prior`, or non-negative proportion (*i.e.* $\leq$ 1 and $\geq$ 0). Rule for the minimum/maximum ζ_j value of τ1, τ2, ε1, ε2, and N_E, respectively, for each *j*th pulse from 1 to Ψ/ψ (as specified by the corresponding `psi.prior`) across all partitions. See also `Details`.

```
tau.buffer, tau2.buffer,
epsilon.buffer, epsilon2.buffer,
NE.buffer
```

Non-negative value or function. Pulse buffer β of the demographic parameter summaries $\tau 1_S$, $\tau 2_S$, $\varepsilon 1_S$, $\varepsilon 2_S$, and N_S, respectively. See also `Details`.

```
tau.idiosyncratic.buffer,
tau2.idiosyncratic.buffer,
epsilon.idiosyncratic.buffer,
epsilon2.idiosyncratic.buffer,
NE.idiosyncratic.buffer
```

Non-negative value or function. Idiosyncratic buffer β_i of the idiosyncratic taxon-specific demographic parameters $\tau 1_i$, $\tau 2_i$, $\varepsilon 1_i$, $\varepsilon 2_i$, and N_i, respectively. See also `Details`.

```
idiosyncratic.rule
```

Character string with possible values `“recent”` and `“ancient”`. Activates rule forcing all idiosyncratic taxa to have values less than the first shared pulse, or values greater than the last shared pulse, respectively. Any other values results in no such rules being placed on idiosyncratic taxa. See also `Details`.

`num.changes`

List, vector of length = `num.partitions`, or value of 1 or 2. Number of demographic change events per taxon. See also `Details`.

`flip`

List, vector of length = `num.partitions`, or logical value. Activates $\tau 2$ to be more recent than $\tau 1$. See also `Details`.

PARAMETER SUMMARIES

`net.zeta.total,`
`net.zeta.per.pulse`

Logical value. Activates output of ζ_T and the vector ζ/ζ_S across partitions, respectively, as a list element/matrix in the `roll.object` list element of the final output, for downstream estimation. Rows of the matrix correspond to individual simulations, which correspond to rows of other matrices in the `roll.object` list element of the final output, and each cell is the aforementioned value.

`mean.tau.shared,`
`mean.tau2.shared,`
`mean.epsilon.shared,`
`mean.epsilon2.shared,`
`mean.NE.shared`

Logical value. Activates output of $E(\tau 1_S)$, $E(\tau 2_S)$, $E(\varepsilon 1_S)$, $E(\varepsilon 2_S)$, and $E(N_S)$ weighted by the vector ζ/ζ_S, respectively, as a list element/matrix in the `roll.object` list element of the final output, for downstream estimation. Rows of the matrix correspond to individual simulations, which correspond to rows of other matrices in the `roll.object` list element of the final output, and each cell is the aforementioned value. See also `Details`.

`mean.tau, mean.tau2,`
`mean.epsilon, mean.epsilon2,`
`mean.NE`

Logical value. Activates output of $E(\tau 1)$, $E(\tau 2)$, $E(\varepsilon 1)$, $E(\varepsilon 2)$, and $E(N)$, respectively, as a list element/matrix in the `roll.object` list element of the final output, for downstream estimation. Rows of the matrix correspond to individual simulations, which correspond to rows of other matrices in the `roll.object` list element of the final output, and each cell is the aforementioned value.

`disp.index.tau.shared,`
`disp.index.tau2.shared,`
`disp.index.epsilon.shared,`
`disp.index.epsilon2.shared,`
`disp.index.NE.shared`

Logical value. Activates output of $\Omega(\tau 1_S)$, $\Omega(\tau 2_S)$, $\Omega(\varepsilon 1_S)$, $\Omega(\varepsilon 2_S)$, and $\Omega(N_S)$ weighted by the vector ζ/ζ_S, respectively, as a list element/matrix in the `roll.object` list element of the final output, for downstream estimation. Rows of the matrix correspond to individual simulations, which correspond to rows of other matrices in the `roll.object` list element of the final output, and each cell is the aforementioned value. See also `Details`.

`disp.index.tau, disp.index.tau2, disp.index.epsilon, disp.index.epsilon2, disp.index.NE`

Logical value. Activates output of $\Omega(\tau1)$, $\Omega(\tau2)$, $\Omega(\varepsilon1)$, $\Omega(\varepsilon2)$, and $\Omega(N)$, respectively, as a list element/matrix in the `roll.object` list element of the final output, for downstream estimation. Rows of the matrix correspond to individual simulations, which correspond to rows of other matrices in the `roll.object` list element of the final output, and each cell is the aforementioned value. See also `Details`.

OBJECTS FROM PRECEDING FUNCTIONS

`build.object`

Output from function `build.dice`. See also `Details`.

`roll.object`

Output from function `roll.dice`. See also `Details`.

Arguments from other `Multi-DICE` functions may be included here and are ignored if not applicable.

Details

`Multi-DICE` cannot currently accommodate models with more than one population per taxon, events aside from population size change, and more than two size change events.

For $\tau1$ and $\tau2$, units are in numbers of generations, and thus may only be positive integers. For $\varepsilon1$ and $\varepsilon2$, units are in ratio of size change from the ancestral effective population size to current effective population size, such that expansions are < 1 and contractions are > 1, and thus may only be positive values. For N_E, unit is in number of effective haploid individuals, and thus may only be positive integers.

For `num.taxa`, `tau.psi.prior`, `epsilon.psi.prior`, `NE.psi.prior`, `tau.shared.prior`, `tau2.shared.prior`, `NE.shared.prior`, `tau.idio.prior`, `tau2.idio.prior`, `NE.idio.prior`, `linked.param.prior` (except when corresponding `linked.param` = `"epsilon"` or `"epsilon2"`), `anchor.prior`, `change.prior`, `tau.buffer`, `tau2.buffer`, `NE.buffer`, `tau.idiosyncratic.buffer`, `tau2.idiosyncratic.buffer`, `NE.idiosyncratic.buffer`, and `num.changes`, non-integer values are converted to integer values via `as.integer`. Similarly, after being multiplied by *n*, `tau.zeta.prior`, `tau2.zeta.prior`, `epsilon.zeta.prior`, `epsilon2.zeta.prior`, `NE.zeta.prior`, `tau.zeta.total.prior`, `tau2.zeta.total.prior`, `epsilon.zeta.total.prior`, `epsilon2.zeta.total.prior`, `NE.zeta.total.prior`, `min.net.tau.zeta.total`, `min.net.tau2.zeta.total`, `min.net.epsilon.zeta.total`, `min.net.epsilon2.zeta.total`, `min.net.NE.zeta.total`, `max.net.tau.zeta.total`, `max.net.tau2.zeta.total`, `max.net.epsilon.zeta.total`, `max.net.epsilon2.zeta.total`, `max.net.NE.zeta.total`, `min.net.tau.zeta.per.pulse`, `min.net.tau2.zeta.per.pulse`, `min.net.epsilon.zeta.per.pulse`, `min.net.epsilon2.zeta.per.pulse`, `min.net.NE.zeta.per.pulse`, `max.net.tau.zeta.per.pulse`, `max.net.tau2.zeta.per.pulse`, `max.net.epsilon.zeta.per.pulse`, `max.net.epsilon2.zeta.per.pulse`, and `max.net.NE.zeta.per.pulse` are converted to integer values via `as.integer` to represent *S* and S_T.

For num.taxa, linked.param, attached.hyper, linked.param.fixed, min.net.tau.zeta.per.pulse, min.net.tau2.zeta.per.pulse, min.net.epsilon.zeta.per.pulse, min.net.epsilon2.zeta.per.pulse, min.net.NE.zeta.per.pulse, max.net.tau.zeta.per.pulse, max.net.tau2.zeta.per.pulse, max.net.epsilon.zeta.per.pulse, max.net.epsilon2.zeta.per.pulse, max.net.NE.zeta.per.pulse, num.changes, and flip, if list, then all list elements are concatenated to form a single vector, with the ordering within list elements and then between list elements preserved (*e.g.* for list of length = 2, with first list element of length = 2 and second list element of length = 1, the order from first to last is: 1) first vector element in first list element; 2) second vector element in first list element; 3) sole vector element in second list element).

For tau.psi.prior, epsilon.psi.prior, NE.psi.prior, tau.zeta.prior, tau2.zeta.prior, epsilon.zeta.prior, epsilon2.zeta.prior, NE.zeta.prior, tau.zeta.total.prior, tau2.zeta.total.prior, epsilon.zeta.total.prior, epsilon2.zeta.total.prior, NE.zeta.total.prior, tau.shared.prior, tau2.shared.prior, epsilon.shared.prior, epsilon2.shared.prior, NE.shared.prior, tau.idio.prior, tau2.idio.prior, epsilon.idio.prior, epsilon2.idio.prior, NE.idio.prior, linked.param.prior, anchor.prior, change.prior, exponential.growth.rate.prior, and exponential.growth.rate.prior2, each list element contains an entire individual discrete distribution; if vector, then converted to list of length = 1 with all vector elements comprising the entirety of a single discrete distribution. Per list element, vector elements within (*i.e.* the discrete distribution) do not need to be in any particular order. Relatedly, each vector element is treated as an independent value, thus weighted distributions (*i.e.* not uniform) may be employed by duplicating values (*e.g.* a distribution of c(0,1,1,1) signifies 75% probability of drawing “1” and 25% probability of drawing “0”), allowing the specification of any discretized distribution (*e.g.* gamma, beta, log-uniform). Accordingly, a uniform distribution with no gaps for integer values would be of length = range of distribution.

For num.taxa, num.changes, and flip, the order of vector elements corresponds to the order of partitions, and for tau.zeta.prior, tau2.zeta.prior, epsilon.zeta.prior, epsilon2.zeta.prior, NE.zeta.prior, tau.idio.prior, tau2.idio.prior, epsilon.idio.prior, epsilon2.idio.prior, NE.idio.prior, change.prior, exponential.growth.rate.prior, and exponential.growth.rate.prior2, the order of list elements corresponds to the order of partitions. Additionally, if length = 1 and num.partitions > 1, then the sole element is used for all partitions. Similarly, if length < num.partitions, then the first element is used for all partitions while ignoring any remaining elements, and if length > num.partitions, then the remaining elements beyond length = num.partitions are ignored; a caution is provided when the length does not equal 1 or num.partitions.

For min.net.tau.zeta.per.pulse, min.net.tau2.zeta.per.pulse, min.net.epsilon.zeta.per.pulse, min.net.epsilon2.zeta.per.pulse, min.net.NE.zeta.per.pulse, max.net.tau.zeta.per.pulse, max.net.tau2.zeta.per.pulse, max.net.epsilon.zeta.per.pulse, max.net.epsilon2.zeta.per.pulse, and max.net.NE.zeta.per.pulse, the order of vector elements corresponds to the temporal order, from most recent to most ancient, of pulses (as specified by the corresponding `psi.prior`), and for `tau.shared.prior`, `tau2.shared.prior`, `epsilon.shared.prior`, `epsilon2.shared.prior`, `NE.shared.prior`, and `anchor.prior`, the order of list elements corresponds to the temporal order, from most recent to most ancient, of pulses (as specified by the corresponding `psi.prior`). Additionally, if length = 1 and maximum value in corresponding `psi.prior` > 1, then the sole element is used for all pulses. Similarly, if length < maximum value in corresponding `psi.prior`, then the first element is used for all pulses while ignoring any remaining elements. For the `zeta.per.pulse` arguments, if length > maximum value in corresponding `psi.prior`, then the remaining elements beyond length = maximum value in corresponding `psi.prior` are ignored; a caution is provided when the length does not equal 1 or maximum value in corresponding `psi.prior`. For the `shared.prior` arguments and `anchor.prior`, there may be additional list elements for idiosyncratic distributions, such that any total length = 1, maximum value in corresponding `psi.prior`, maximum value in corresponding `psi.prior` + 1, or maximum value in corresponding `psi.prior` + `num.partitions`, are allowed; for any other lengths, a caution is provided and excess elements beyond the highest acceptable length are ignored. See below for more information about adding idiosyncratic distributions to these arguments.

If `num.partitions=`*n*, then rearrangement of bins across taxa within allele frequency classes based on descending order of the relative SNP proportions is not performed to construct the aSFS and taxon-specific inference of demographic parameters is possible. However, in general, more partitions results in more parameter space with respect to taxa samples that must be explored due to a decrease in order-independence and assumed exchangeability, thus multiple-fold more simulations must be conducted to achieve comparable accuracy in hyperparameter estimation as without partitioning.

For `tau.psi.prior`, `epsilon.psi.prior`, and `NE.psi.prior`, distinguishing between Ψ and ψ is accomplished via the corresponding `zeta.prior`, `zeta.total.prior`, `idiosyncratic` setting, and/or `min`/`max.net zeta.total`/`per.pulse`, except for values of 0, which are explicitly for ψ = 0 and thus indicate full idiosyncrasy. If list length > 2 for τ and ε or list length > 1 for N_E, then a caution is provided and the remaining elements beyond length = 2 for τ and ε and length = 1 for N_E are ignored. Applies across all partitions, such that it is regardless of partitioning.

For `tau.zeta.prior`, `tau2.zeta.prior`, `epsilon.zeta.prior`, `epsilon2.zeta.prior`, and `NE.zeta.prior`, if `num.partitions` > 1, may be necessary to include “0.0” as a value, but can control ζ_S and ζ_T across partitions via corresponding `zeta.total.prior` and/or `min`/`max.net zeta.total`/`per.pulse`. Attributes to each partition individually, but proportion values are out of the entirety of taxa dataset, thus if `num.partitions` > 1, then the upper bound for each partition should be the number of taxa within that partition divided by *n*. When ψ = {0, 1}, equivalent to hyperprior distribution for ζ_T. If identical across all partitions, it is more computationally efficient to specify only one *i.e.* list of length = 1, or a vector.

For `tau.zeta.total.prior`, `tau2.zeta.total.prior`, `epsilon.zeta.total.prior`, `epsilon2.zeta.total.prior`, and `NE.zeta.total.prior`, if length > 1, then remaining elements beyond the first are ignored and a caution is provided.

To build a hyperprior, `Multi-DICE` first looks if the corresponding `zeta.total.prior` is specified, then if `dirichlet.process=T`, and if neither is such case, then all possible combinations of corresponding `psi.prior` and `zeta.prior` draws are equally weighted. Therefore, if `num.partitions=1`, corresponding `psi.prior=1`, and `dirichlet.process=F`, then `tau.zeta.total.prior`, `tau2.zeta.total.prior`, `epsilon.zeta.total.prior`, `epsilon2.zeta.total.prior`, and `NE.zeta.total.prior` equivalent to corresponding `zeta.prior` and thus unnecessary to specify.

For `tau.shared.prior`, `tau2.shared.prior`, `epsilon.shared.prior`, `epsilon2.shared.prior`, and `NE.shared.prior`, each successive list element/distribution must have a greater minimum and maximum value than its preceding list elements/distributions. If multiple distributions are utilized for every potential pulse and these distributions overlap in their bounds, running time may slow since, in this case, draws are made from all the distributions independently and then checked if abiding by ordering and buffering, with re-draws if not.

For each of `tau`, `tau2`, `epsilon`, `epsilon2`, and `NE`, if hyperparameterized via its corresponding `psi.prior` while `idiosyncratic=T` or a 0 value is in said `psi.prior`, then a corresponding idiosyncratic prior is required. `Multi-DICE` first looks in `change.prior` (for τ2 only), then the corresponding `idio.prior`, and finally the corresponding `shared.prior`. If a `shared.prior` argument is utilized, the additional list elements beyond length = maximum value in corresponding `psi.prior` are considered; if the number of list elements is ≤ maximum value in corresponding `psi.prior`, then only the first list element is considered. These list elements undergo the same specifications as aforementioned for the corresponding `idio.prior`.

For each of `tau`, `tau2`, `epsilon`, `epsilon2`, and `NE`, if not hyperparameterized yet part of the specified model (*i.e.* `tau`, `epsilon`, and `NE` are always part of the model, and `tau2` and `epsilon2` are part of the model when `num.changes` = 2), then a corresponding nuisance prior is required. A nuisance prior differs from an idiosyncratic prior in that variation in the respective nuisance parameter is being considered while not of interest with respect to hyperparameterization (*i.e.* variability in values across taxa governed hierarchically), whereas an idiosyncratic prior is still governed by hyperparameters in coordination with the corresponding shared prior. `Multi-DICE` first looks in `anchor.prior` (for τ2 only), next `change.prior` (for τ2 only), afterward `linked.param.prior` (if applicable; see below for more information), then the corresponding `idio.prior`, and finally the corresponding `shared.prior`. If `linked.param.prior` is utilized for a particular nuisance parameter, either the corresponding `idio.prior` or `shared.prior` must still be specified even if all taxa are always covered by `linked.param.prior` across all simulations. If a `shared.prior` argument is utilized, it undergoes the same specifications as aforementioned for the corresponding `idio.prior`.

If a two-event model is specified, there is an interplay between τ1 and τ2 with respect to synchronous/shared, idiosyncratic, and nuisance draws. If both τ1 and τ2 are hyperparameterized, draws from the $\tau1_S$ synchronous/shared prior and $\tau2_S$ synchronous/shared prior are made independently, but if for a given simulation there are no valid combinations of these draws with respect to the vectors $\zeta_{\tau1}/\zeta_{\tau1,S}$ and $\zeta_{\tau2}/\zeta_{\tau2,S}$ such that τ1 < τ2 is not violated, then the $\tau2_S$ synchronous/shared prior is re-drawn for this simulation until there is a valid combination. Hence, the $\tau2_S$ synchronous/shared prior is conditional on the $\tau1_S$ synchronous/shared prior. Furthermore, for each $\tau2_S$ draw that is not assigned to a $\tau1_S$ draw (*i.e.* cases where τ1 is idiosyncratic while τ2 is synchronous/shared/in a Ψ/ψ pulse as specified by `tau2.psi.prior`), the according $\tau1_i$ idiosyncratic draw is restricted by that $\tau2_S$ value. Therefore, the $\tau1_i$ idiosyncratic prior is conditional on the $\tau2_S$ synchronous/shared prior (if specified). Lastly, every τ2 idiosyncratic or nuisance draw is confined by its according $\tau1$ drawn value, whether it is synchronous/shared, idiosyncratic, or nuisance, thus both the $\tau2_i$ idiosyncratic prior and the $\tau2$ nuisance prior are conditional on the $\tau1_s$ synchronous/shared prior, $\tau1_i$ idiosyncratic prior, and $\tau1$ nuisance prior (whichever is specified).

Considering the aforementioned dependencies between $\tau1$ and $\tau2$, it is highly recommended that prior distribution bounds are as far apart as possible among the two events, ideally mutually exclusive/non-overlapping. Otherwise, there could be a stop of operation due to an invalid/incompatible draw, computational lag, and/or statistical bias on prior distributions.

For `linked.param`, `attached.hyper`, `linked.param.partition`, `attached.hyper.pulse`, `linked.param.prior`, and `linked.param.fixed`, nuisance demographic parameters can be linked to other hyperparameterized demographic parameters, such that taxa within a pulse for a hyperparameterized parameter are also grouped for parameterization for another nuisance parameter. When activating this feature, all six arguments must be specified. Additionally, all six arguments must be of length = length of `linked.param`, except for `linked.param.partition`, `attached.hyper.pulse`, `linked.param.prior`, and `linked.param.fixed`, which may be of length = 1, with the sole element being duplicated to length of `linked.param`; for `linked.param.partition` and `attached.hyper.pulse`, if vector, then converted to list of length = 1. The order of elements in `linked.param` then corresponds to the order of elements in the other five arguments, such that each of these elements forms a different entry. For `linked.param`, nuisance parameters to be linked are specified. For `attached.hyper`, hyperparameterized demographic parameters to be attached are specified. For `linked.param.partition`, the applicable partitions, indexed numerically in the same order as user-specified, that are linking the nuisance parameter is/are specified. For `attached.hyper.pulse`, the applicable pulses of the hyperparameterized parameter is/are specified. For `linked.param.prior`, the nuisance priors are specified. For `linked.param.fixed`, whether all applicable taxa have a shared value in the nuisance parameter or independently draw (*i.e.* vary in values) from the same nuisance prior is specified. By specifying each pulse separately through `attached.hyper.pulse` and fixing each pulse to have a shared value through `linked.param.fixed`, the nuisance parameter would essentially be hyperparameterized by the same hyperparameters as the hyperparameterized parameter. For example, if `linked.param=c(“epsilon”, “epsilon”, “epsilon”)`, `attached.hyper=c(“tau”, “tau”, “tau”)`, `attached.hyper.pulse=list(1,2,3)`, and `linked.param.fixed=T`, then $\varepsilon1$ and $\tau1$ would be governed identically by $\zeta_{\tau1}/\zeta_{\tau1,S}$. Similarly, a nuisance prior may be specified differentially between taxa within a shared pulse and idiosyncratic taxa relative to another, hyperparameterized parameter. For example, if `linked.param=c(“epsilon”)`, `attached.hyper=c(“tau”)`, `linked.param.prior=c(1000:10000)/100000`, and `epsilon.idio.prior=c(1000:100000)/100000`, then taxa that are in synchronous $\tau1$ pulses have the prior distribution $\varepsilon1 \sim U(0.01, 0.10)$ whereas $\tau1$ temporally idiosyncratic taxa have the prior distribution $\varepsilon1 \sim U(0.01, 1.00)$. If a partition of taxa has a nuisance parameter linked to multiple parameters across overlapping entries (*e.g.* `“epsilon”` and `“1”` are both specified in the first two elements for `linked.param` and `linked.param.partition`, respectively, and `“tau”` and `“tau2”` are the first two elements in `attached.hyper` such that the first partition of taxa has the nuisance $\varepsilon1$ parameter attached to both `“tau”` and `“tau2”`), then latter-specified entries may replace previously-specified entries. If for a given entry, `linked.param=“tau”`, vector length of `linked.param.partitions` > 1, and `linked.param.fixed=T`, then the `flip` value for the first partition specified in `linked.param.partitions` is assumed to be the same across all the vector elements/partitions in `linked.param.partitions`.

For `anchor.prior`, a special case of linking a nuisance demographic parameter to a hyperparameterized demographic parameter occurs where $\tau 2$ is linked to $\tau 1_s$ and the difference between these two values δ is parameterized. Per each synchronous/shared pulse in $\tau 1_s$, as specified by `tau.psi.prior`, a draw from `anchor.prior` determines the $\tau 2$ value for taxa within said pulse. For example, if five taxa are in the first $\tau 1$ synchronous pulse and there is an according draw of 100,000 from `anchor.prior`, then $\tau 2 = \tau 1_{s,1}$ + 100,000 for all five of those taxa. Different distributions can be employed per $\tau 1_s$ synchronous/shared pulse in `anchor.prior` (see above for more information). Moreover, additional distributions specified in `anchor.prior` beyond length = maximum value in `tau.psi.prior` are used for taxa that have $\tau 1_i$ idiosyncratic values. If the number of additional distributions < `num.partitions`, then the first additional distribution is used for all partitions, and if the number of additional distributions $\geq$ `num.partitions`, then the order of the first `num.partitions` additional distributions corresponds to the order of partitions. If there are no additional distributions, then the first distribution in `anchor.prior` is used for $\tau 1_i$ idiosyncratic draws across all partitions. For $\tau 1_i$ idiosyncratic values, independent draws are made from `anchor.prior` among taxa, acting here similarly to `change.prior` (see below for more information). Since `anchor.prior` links $\tau 1$ and $\tau 2$ together across all taxa as essentially one set of hyperparameterization, `flip` would have no effect and thus is ignored here. The parameter summary vector δ_s of anchor values corresponding to the vector $\tau 1_s$ is outputted as a list element/matrix in the `roll.object` list element of the final output, available for downstream estimation. Rows of the matrix correspond to individual simulations, which correspond to rows of other matrices in the `roll.object` list element of the final output, and column order corresponds to the temporal order, from most recent to most ancient, of $\tau 1_s$ synchronous/shared pulses.

For `change.prior`, a similar case to `anchor.prior` is employed in that the difference δ between the two values $\tau 1$ and $\tau 2$ is parameterized, except $\tau 2$ is not linked to the hyperparameterization of $\tau 1$. Thus, taxon-specific independent draws are conducted on `change.prior`, which can apply either as an idiosyncratic or nuisance prior for $\tau 2$. If for a given partition, `change.prior` is employed and `flip=T`, then the maximum value in `change.prior` must be < than the minimum value of `tau.shared.prior` and `tau.idio.prior`; this is not checked and thus it is critical that it is user-confirmed.

For `exponential.growth.rate.prior` and `exponential.growth.rate.prior2`, growth occurs according to an exponential growth model with drawn values $r1$ and $r2$ until the drawn values $\varepsilon 1$ and $\varepsilon 2$ are met, respectively. The more ancient time, *i.e.* beginning of size change forward-in-time, is represented by $\tau 1$ and $\tau 2$, respectively, and thus is the time that is buffered, whereas the more recent time, *i.e.* end size change forward-in-time, is a nuisance parameter. There are no checks of the nuisance end time violating any model specifications (*e.g.* if $\tau 1$ is very recent and/or the duration of first event growth is very long, then the first event beginning time may be a negative value, or if the difference between $\tau 1$ and $\tau 2$ is small and/or the duration of second event growth is very long, then the second event beginning time may be prior to $\tau 1$; both examples assume `flip=F`, see below for more information), thus it is critical that this is thoroughly user-investigated. If $r1$/$r2$ contradicts $\varepsilon 1$/$\varepsilon 2$ regarding demographic syndrome, the demographic syndrome indicated by $r1$/$r2$ and the inverse of $\varepsilon 1$/$\varepsilon 2$ are employed, *e.g.* if $r1$ is negative (expansion) while corresponding $\varepsilon 1 > 1$ (contraction), then expansion occurs until $1/\varepsilon 1$ is met. Both $r1$ and $r2$ only act as nuisance parameters here, thus may differentiate in prior distributions across partitions, but cannot be hyperparameterized.

If `idiosyncratic=F`, then `min.net.tau.zeta.total`, `min.net.tau2.zeta.total`, `min.net.epsilon.zeta.total`, `min.net.epsilon2.zeta.total`, `min.net.NE.zeta.total`, `max.net.tau.zeta.total`, `max.net.tau2.zeta.total`, `max.net.epsilon.zeta.total`, `max.net.epsilon2.zeta.total`, and `max.net.NE.zeta.total`, if specified, may only equal 1.0.

For `tau.buffer`, `tau2.buffer`, `epsilon.buffer`, `epsilon2.buffer`, and `NE.buffer`, β buffers are applied to draws of shared pulse values, thus affecting the corresponding shared prior; the corresponding idiosyncratic prior, from which draws are subsequently made, is accordingly affected by the shared pulse buffers. For `tau.idiosyncratic.buffer`, `tau2.idiosyncratic.buffer`, `epsilon.idiosyncratic.buffer`, `epsilon2.idiosyncratic.buffer`, and `NE.idiosyncratic.buffer`, β_i idiosyncratic buffers are then applied to idiosyncratic draws, which additionally affect the corresponding idiosyncratic prior that had already been initially transformed by the shared pulse buffers. In other words, β buffers idiosyncratic taxa from shared pulse values, and β_i buffers idiosyncratic taxa from each other. For example, if n = 10, $\psi_{\tau 1}$ = 2, $\zeta_{\tau 1,T}$ = 0.8, β = 10,000, β_i = 1,000, $\tau 1$ ~ U(1,000, 100,000), $\tau 1_S$ = {11,000, 12,001}, and $\tau 1_i$ = {99,000} after the first idiosyncratic draw, then given that the synchronous/shared pulse buffers result in invalid draws from ~ U(1,000, 13,001) and the first idiosyncratic draw buffer results in invalid draws from ~ U(98,000, 100,000), the remaining second idiosyncratic draw would be from the resulting transformed prior distribution $\tau 1_i$ ~ U(13,002, 97,999). A caution is provided if an `idiosyncratic.buffer` corresponding to a specified `psi.prior` is not specified. Buffers cannot be deployed for nuisance draws (though see below when ψ = 0).

For `tau.buffer`, `tau2.buffer`, `epsilon.buffer`, `epsilon2.buffer`, `NE.buffer`, `tau.idiosyncratic.buffer`, `tau2.idiosyncratic.buffer`, `epsilon.idiosyncratic.buffer`, `epsilon2.idiosyncratic.buffer`, and `NE.idiosyncratic.buffer`, if writing a function, there can be only one argument, which is for the value of a particular draw from a corresponding prior distribution, and the output must be a vector of discrete values that are buffered out of any corresponding prior distributions given that particular draw. For `epsilon.buffer`, `epsilon2.buffer`, `epsilon.idiosyncratic.buffer`, and `epsilon2.idiosyncratic.buffer`, given that these are not integers, it is imperative that the output values are on the same scale/interval/significant figures as the corresponding shared and idiosyncratic prior distributions since values are buffered out only if they are exactly equal. There is no check on functionality for `buffer` and `idiosyncratic.buffer` functions, thus it is highly recommended that any function is thoroughly user-tested.

For `num.changes`, only necessary if there are nuisance demographic parameters for a second event, since one event is the default and demographic parameters for the second event that are hyperparameterized by the corresponding `psi.prior` are already activated. If < 1, then there is a stop of operation, and if > 2, then it is converted to `num.changes=2` and a caution is provided.

If `flip=T`, $\tau 2$ and $\varepsilon 2$ still refers to the second specified event, which in this case is the more recent one.

For `mean.tau.shared`, `mean.tau2.shared`, `mean.epsilon.shared`, `mean.epsilon2.shared`, `mean.NE.shared`, `disp.index.tau.shared`, `disp.index.tau2.shared`, `disp.index.epsilon.shared`, `disp.index.epsilon2.shared`, and `disp.index.NE.shared`, may be applied only if corresponding `psi.prior` is specified, *i.e.* demographic parameter is hyperparameterized. If corresponding `psi.prior` specifies Ψ, then these arguments are equivalent to their corresponding `mean`/`disp.index` arguments (*i.e.* without `.shared` suffix). When full idiosyncrasy (*i.e.* ψ = 0, ζ_T = 0.0), default to = 0 (except when specified by corresponding `psi.prior=`Ψ).

For `disp.index.tau.shared`, `disp.index.tau2.shared`, `disp.index.epsilon.shared`, `disp.index.epsilon2.shared`, `disp.index.NE.shared`, `disp.index.tau`, `disp.index.tau2`, `disp.index.epsilon`, `disp.index.epsilon2`, and `disp.index.NE`, when the applicable number of taxa = 1, default to = 0.

If `roll.object` is provided, then the arguments `tau.zeta.prior`, `tau2.zeta.prior` (if `build.object` provided), `epsilon.zeta.prior`, `epsilon2.zeta.prior`, `NE.zeta.prior`, `tau.zeta.total.prior`, `tau2.zeta.total.prior` (if `build.object` provided), `epsilon.zeta.total.prior`, `epsilon2.zeta.total.prior`, `NE.zeta.total.prior`, `tau.shared.prior` (if idiosyncratic/nuisance prior not specified here), `epsilon.shared.prior` (if idiosyncratic/nuisance prior not specified here), `epsilon2.shared.prior` (if idiosyncratic/nuisance prior not specified here), `NE.shared.prior` (if idiosyncratic/nuisance prior not specified here), `dirichlet.process` (if τ2 not hyperparameterized or `build.object` provided), `min.net.tau.zeta.total`, `min.net.tau2.zeta.total` (if `build.object` provided), `min.net.epsilon.zeta.total`, `min.net.epsilon2.zeta.total`, `min.net.NE.zeta.total`, `max.net.tau.zeta.total`, `max.net.tau2.zeta.total` (if `build.object` provided), `max.net.epsilon.zeta.total`, `max.net.epsilon2.zeta.total`, `max.net.NE.zeta.total`, `min.net.tau.zeta.per.pulse`, `min.net.tau2.zeta.per.pulse` (if `build.object` provided), `min.net.epsilon.zeta.per.pulse`, `min.net.epsilon2.zeta.per.pulse`, `min.net.NE.zeta.per.pulse`, `max.net.tau.zeta.per.pulse`, `max.net.tau2.zeta.per.pulse` (if `build.object` provided), `max.net.epsilon.zeta.per.pulse`, `max.net.epsilon2.zeta.per.pulse`, and `max.net.NE.zeta.per.pulse` are ignored here.

When ψ = 0 *i.e.* full idiosyncrasy, the arguments `idiosyncratic`, `min.net.tau.zeta.total`, `min.net.tau2.zeta.total`, `min.net.epsilon.zeta.total`, `min.net.epsilon2.zeta.total`, `min.net.NE.zeta.total`, `min.net.tau.zeta.per.pulse`, `min.net.tau2.zeta.per.pulse`, `min.net.epsilon.zeta.per.pulse`, `min.net.epsilon2.zeta.per.pulse`, `min.net.NE.zeta.per.pulse`, and `idiosyncratic.rule` are ignored, and the arguments `tau.idiosyncratic.buffer`, `tau2.idiosyncratic.buffer`, `epsilon.idiosyncratic.buffer`, `epsilon2.idiosyncratic.buffer`, and `NE.idiosyncratic.buffer` are activated (even if `idiosyncratic=F`).

`Multi-DICE` proceeds only if a valid draw is allowed given corresponding `num.taxa`, `psi.prior`, `zeta.prior` (*e.g.* minimum value in `psi.prior` * minimum value in `zeta.prior` ≤ minimum value in `num.taxa`), `zeta.total.prior`, shared prior, idiosyncratic prior, `idiosyncratic` setting, `min`/`max.net` `zeta.total`/`per.pulse`, `buffer`, and `idiosyncratic.buffer`. Importantly, there is no check on if all values in `psi.prior` have a valid draw, only if there is a valid draw among any of the values. Additionally, the range of a shared prior must be greater than its corresponding $(\Psi/\psi - 1)(2\beta + 1)$, and the range of an idiosyncratic prior ought to be greater than its corresponding $(\Psi/\psi)(2\beta + 1) + (\sigma - 1)(2\beta_i + 1)$ across every possible combination of Ψ/ψ and σ and with consideration given to `idiosyncratic.rule`. For `tau2.shared.prior` and `tau2.idio.prior`, consideration must also be given to any overlap with `tau.shared.prior` and `tau.idio.prior`, respectively.

Value

Returned value is a list object with two elements, `roll.object` and `sim.specs`. The list element `roll.object` is similar to the output of the function `roll.dice`, except parameter summaries derived here are added (*e.g.* Ω, E(), δ_S; see above for more information); these represent values of interest for estimation. The list element `sim.specs` contains a list of matrices, with each list element/matrix attributed to a taxon-specific parameter and accordingly named (*i.e.* `tau`, `tau2`, `epsilon`, `epsilon2`, `NE`, `exponential.growth.rate.prior`, `exponential.growth.rate.prior2`). Per matrix, each row represents an individual simulation, each column represents an independent taxon to be simulated, and each cell is the according demographic parameter value to be used for simulation. Rows and columns across matrices correspond to each other *i.e.* across matrices, the same row number refers to the same simulation and the same column number refers to the same simulated independent taxon.

Author(s)

Alexander T. Xue

References

Chan YL, Schanzenbach D, Hickerson MJ (2014) Detecting concerted demographic response across community assemblages using hierarchical approximate Bayesian computation. *Molecular Biology and Evolution*, **31**, 2501–2515.

Hickerson MJ, Stahl E, Takebayashi N (2007) msBayes: Pipeline for testing comparative phylogeographic histories using hierarchical approximate Bayesian computation. *BMC bioinformatics*, **8**, 268.

Huang W, Takebayashi N, Qi Y, Hickerson MJ (2011) MTML-msBayes: approximate Bayesian comparative phylogeographic inference from multiple taxa and multiple loci with rate heterogeneity. *BMC bioinformatics*, **12**, 1.

Xue AT (2017) Multi-DICE Manual.

Xue AT, Hickerson MJ (2015) The aggregate site frequency spectrum for comparative population genomic inference. *Molecular Ecology*, **24**, 6223–6240.

Xue AT, Hickerson MJ (*submitted*) Multi-DICE: R package for comparative population genomic inference under multi-taxa hierarchical co-demographic models.

See Also

`build.dice`, `roll.dice`, `dice.sims`, `dice.aSFS`, `dice.sumstats`

Examples

```
#simplest execution akin to approach in Xue and Hickerson (2015)
play.dice(num.sims=5, num.taxa=10, tau.psi.prior=c(1),
tau.zeta.prior=c(1:10)/10, tau.shared.prior=c(1000:1000000),
epsilon.idio.prior=c(1000:10000)/100000, NE.idio.prior=c(1000:100000))

#simplest execution akin to approach in Xue and Hickerson (2015); ln U
distribution applied on tau.shared.prior, with 100,000 intervals
discretized uniformly across ln(tau.shared.prior)
play.dice(num.sims=5, num.taxa=10, tau.psi.prior=c(1),
tau.zeta.prior=c(1:10)/10,
tau.shared.prior=exp(c((log(1000)*100000):(log(1000000)*100000))/100000),
epsilon.idio.prior=c(1000:10000)/100000, NE.idio.prior=c(1000:100000))

#simplest execution akin to approach in Xue and Hickerson (2015); assuming
roll.dice was previously performed and the output was directed to object
roll.object
play.dice(num.sims=5, num.taxa=10, tau.psi.prior=c(1),
tau.idio.prior=c(1000:1000000), epsilon.idio.prior=c(1000:10000)/100000,
NE.idio.prior=c(1000:100000)), roll.object=roll.object)

#simplest execution akin to approach in software package msBayes
play.dice(num.sims=5, num.taxa=10, tau.psi.prior=c(1:10),
tau.zeta.prior=c(1:10)/10, tau.shared.prior=c(1000:1000000),
epsilon.idio.prior=c(1000:10000)/100000, NE.idio.prior=c(1000:100000),
idiosyncratic=F)
```

Simulating under hierarchical co-demographic model

Description

`dice.sims` is a wrapper function for the command-line program `fastsimcoal2` that performs multi-taxa coalescent simulation of per-taxon summary statistics under a unified hierarchical co-demographic model as specified by `build.dice`, `roll.dice`, and `play.dice`, which are also embedded here and are automatically deployed if `build.object`/`roll.object`/`play.object`, `roll.object`/`play.object`, and `play.object`, respectively, are not specified. `fastsimcoal2` must be separately user-installed. `bash` commands are called upon here, thus `dice.sims` can run only within a `bash` terminal environment (*e.g.* Mac, Linux).

Usage

```
dice.sims(num.sims, num.taxa, num.partitions=1, num.haploid.samples,
num.ind.sites=NULL, num.SNPs=NULL, length.seq=NULL, folded=T,
sampling.times=NULL, gen.times=NULL, tau.psi.prior=NULL,
epsilon.psi.prior=NULL, NE.psi.prior=NULL, tau.zeta.prior=NULL,
tau2.zeta.prior=NULL, epsilon.zeta.prior=NULL, epsilon2.zeta.prior=NULL,
NE.zeta.prior=NULL, tau.zeta.total.prior=NULL, tau2.zeta.total.prior=NULL,
epsilon.zeta.total.prior=NULL, epsilon2.zeta.total.prior=NULL,
NE.zeta.total.prior=NULL, tau.shared.prior=NULL, tau2.shared.prior=NULL,
epsilon.shared.prior=NULL, epsilon2.shared.prior=NULL,
NE.shared.prior=NULL, tau.idio.prior=NULL, tau2.idio.prior=NULL,
epsilon.idio.prior=NULL, epsilon2.idio.prior=NULL, NE.idio.prior=NULL,
linked.param=NULL, attached.hyper=NULL, linked.param.partition=NULL,
attached.hyper.pulse=NULL, linked.param.prior=NULL,
linked.param.fixed=NULL, anchor.prior=NULL, change.prior=NULL,
exponential.growth.rate.prior=NULL, exponential.growth.rate.prior2=NULL,
mut.rate.prior=NULL, dirichlet.process=F, idiosyncratic=T,
min.net.tau.zeta.total=NULL, min.net.tau2.zeta.total=NULL,
min.net.epsilon.zeta.total=NULL, min.net.epsilon2.zeta.total=NULL,
min.net.NE.zeta.total=NULL, max.net.tau.zeta.total=NULL,
max.net.tau2.zeta.total=NULL, max.net.epsilon.zeta.total=NULL,
max.net.epsilon2.zeta.total=NULL, max.net.NE.zeta.total=NULL,
min.net.tau.zeta.per.pulse=NULL, min.net.tau2.zeta.per.pulse=NULL,
min.net.epsilon.zeta.per.pulse=NULL, min.net.epsilon2.zeta.per.pulse=NULL,
min.net.NE.zeta.per.pulse=NULL, max.net.tau.zeta.per.pulse=NULL,
max.net.tau2.zeta.per.pulse=NULL, max.net.epsilon.zeta.per.pulse=NULL,
max.net.epsilon2.zeta.per.pulse=NULL, max.net.NE.zeta.per.pulse=NULL,
tau.buffer=0, tau2.buffer=0, epsilon.buffer=0, epsilon2.buffer=0,
NE.buffer=0, tau.idiosyncratic.buffer=NULL,
tau2.idiosyncratic.buffer=NULL, epsilon.idiosyncratic.buffer=NULL,
epsilon2.idiosyncratic.buffer=NULL, NE.idiosyncratic.buffer=NULL,
idiosyncratic.rule='none', num.changes=1, flip=F, net.zeta.total=F,
net.zeta.per.pulse=F, mean.tau.shared=F, mean.tau2.shared=F,
mean.epsilon.shared=F, mean.epsilon2.shared=F, mean.NE.shared=F,
mean.tau=F, mean.tau2=F, mean.epsilon=F, mean.epsilon2=F, mean.NE=F,
disp.index.tau.shared=F, disp.index.tau2.shared=F,
disp.index.epsilon.shared=F, disp.index.epsilon2.shared=F,
disp.index.NE.shared=F, disp.index.tau=F, disp.index.tau2=F,
disp.index.epsilon=F, disp.index.epsilon2=F, disp.index.NE=F, fsc2path,
messages.sims=NULL, output.directory, append.sims=F, keep.taxa.draws=F,
output.hyper.draws=T, output.taxa.draws=F, keep.fsc2.files=F,
build.object=NULL, roll.object=NULL, play.object=NULL)
```

Arguments

`num.sims`

Positive integer. Number of simulations. Required.

DATA

`num.taxa`

List, vector of length = `num.partitions`, or positive integer. Number of taxa per partition. Total sum across partitions equals the total number of taxa *n* in dataset. See also `Details`. Required.

`num.partitions`

Positive integer. Number of partitions for taxa in dataset. Allows differential data and model specifications across user-specified taxa groupings, including sampling size of individuals, generation times, demographic syndrome, and taxon-specific nuisance parameter prior distributions. See also `Details`.

`num.haploid.samples`

List, vector of length = `num.partitions`, or positive integer. Number of haploid samples per partition. See also `Details`. Required.

`num.ind.sites, num.SNPs, length.seq`

List, vector of length = `num.partitions`, or positive integer. Data sampling level per partition, in number of independent sites/SNPs using the `fastsimcoal2 FREQ` simulation model, in number of independent SNPs using the `fastsimcoal2 SNP` simulation model, and in sequence length using the `fastsimcoal2 SNP` simulation model, respectively. For each taxon, the former two simulate the SFS based on independent sites, while the latter one simulates single-sequence summary statistics. See also `Details`. At least one is required.

`folded`

List, vector of length = `num.partitions`, or logical value. Activates folding of the SFS per partition; ignored if single-sequence summary statistics are simulated. See also `Details`.

`sampling.times`

List, vector of length = `num.partitions`, or non-negative integer. Sampling times per partition. Allows simulation of time-series or ancient data. Unnecessary if all data are collected simultaneously and in present-day, since this is assumed by default. See also `Details`.

`gen.times`

List, vector of length = `num.partitions` or *n*, or positive value. Generation times per partition or per taxon, in units of years per generation. If including multiple generation times within a partition (*i.e.* vector of length = *n*), generation times are randomly assigned to sets of parameter draws, though the order of output simulation files corresponds to the user-specified order (*i.e.* `dice.simulations1` corresponds to the first element, `dice.simulations2` to the second, etc.). See also `Details`.

PRIORS

`tau.psi.prior, epsilon.psi.prior, NE.psi.prior`

List, vector, or non-negative integer. Hyperprior distribution for Ψ/ψ of τ, ε, and N_E, respectively. For τ and ε, if list of length = 2, then the first list element applies to the first more recent size change event (*e.g.* $\tau 1$, $\varepsilon 1$) and the second list element applies to the second more ancient size change event (*e.g.* $\tau 2$, $\varepsilon 2$), per taxon. The arguments(s) specified here and their according list lengths activate which taxon-specific demographic parameters are to be hyperparameterized via Ψ/ψ as well as $\zeta/\zeta_S/\zeta_T$ downstream. See also `Details`. At least one is required.

`tau.zeta.prior, tau2.zeta.prior, epsilon.zeta.prior, epsilon2.zeta.prior, NE.zeta.prior`

List of length = `num.partitions` or 1, vector, or non-negative proportion (*i.e.* ≤ 1 and ≥ 0). Hyperprior distribution for ζ_j of $\tau 1$, $\tau 2$, $\varepsilon 1$, $\varepsilon 2$, and N_E, respectively, for each *j*th pulse from 1 to Ψ/ψ, as specified by the corresponding `psi.prior`, and per partition. See also `Details`. Required for each corresponding `psi.prior` specified, unless the maximum value in the corresponding `psi.prior` = 0.

`tau.zeta.total.prior, tau2.zeta.total.prior, epsilon.zeta.total.prior, epsilon2.zeta.total.prior, NE.zeta.total.prior`

List of length = 1, vector, or non-negative proportion (*i.e.* ≤ 1 and ≥ 0). Hyperprior distribution for ζ_T of $\tau 1$, $\tau 2$, $\varepsilon 1$, $\varepsilon 2$, and N_E, respectively. Activates a uniform hyperprior such that each discrete Ψ/ψ value, as specified by the corresponding `psi.prior`, is first weighted with equal hyperprior probability, then all discrete ζ_T values are weighted equally per Ψ/ψ value, and finally every possible associated vector ζ/ζ_S is weighted equally per ζ_T value. See also `Details`.

`tau.shared.prior, tau2.shared.prior, epsilon.shared.prior, epsilon2.shared.prior, NE.shared.prior`

List, vector, or positive value. Prior distribution for the demographic parameter summaries $\tau 1_S$, $\tau 2_S$, $\varepsilon 1_S$, $\varepsilon 2_S$, and N_S, respectively (or $\tau 1$, $\tau 2$, $\varepsilon 1$, $\varepsilon 2$, and N, respectively, if corresponding `psi.prior` specifies Ψ). See also `Details`. Required for each corresponding `psi.prior` specified, unless the maximum value in the corresponding `psi.prior` = 0.

`tau.idio.prior, tau2.idio.prior, epsilon.idio.prior, epsilon2.idio.prior, NE.idio.prior`

List of length = `num.partitions` or 1, vector, or positive value. Prior distribution for the taxon-specific demographic parameters $\tau1_i$, $\tau2_i$, $\varepsilon1_i$, $\varepsilon2_i$, and N_i, respectively for idiosyncratic values, and $\tau1$, $\tau2$, $\varepsilon1$, $\varepsilon2$, and N, respectively for nuisance values. See also `Details`.

`linked.param, attached.hyper`

List, vector, or character string, with possible values being the names of the demographic parameters (*i.e.* `“tau”`, `“tau2”`, `“epsilon”`, `“epsilon2”`, `“NE”`). Activates nuisance parameters in `linked.param` to have prior distributions be linked to hyperparameterized demographic parameters in `attached.hyper`, such that prior distributions may differentiate across pulses and idiosyncratic taxa with respect to the hyperparameterized demographic parameter in `attached.hyper`. See also `Details`.

`linked.param.partition, attached.hyper.pulse`

List of length = length of `linked.param` or 1, vector, or positive integer. The partitions in the linked nuisance parameter, and the pulses in the attached hyperparameterized demographic parameter, for which each element in `linked.param` and `attached.hyper`, respectively, applies. Each list element may contain multiple partitions/pulses, respectively. See also `Details`.

`linked.param.prior`

List of length = length of `linked.param` or 1, vector, or positive value. Prior distribution for the nuisance demographic parameter in each element of `linked.param`. See also `Details`.

`linked.param.fixed`

List, vector of length = length of `linked.param`, or logical value. Activates a fixed nuisance demographic parameter value for all taxa to which the corresponding elements in `linked.param.partition` and `attached.hyper.pulse` apply. See also `Details`.

`anchor.prior, change.prior`

List, vector, or positive integer. Prior distribution for $\tau2$ based on its difference δ with $\tau1$. This difference value can be assigned to synchronous/shared pulses in $\tau1_S$, as specified by `tau.psi.prior`, and accordingly inferred as a parameter summary vector δ_S (`anchor.prior`), or applied independently across taxa as an idiosyncratic or nuisance value (`change.prior`). See also `Details`.

`exponential.growth.rate.prior, exponential.growth.rate.prior2`

List of length = `num.partitions` or 1, vector, or double value. Prior distribution for the nuisance taxon-specific parameters $r1$ and $r2$, respectively. Activates exponential growth model $N_\tau = N_0 * e^{(r * \tau)}$ for the first and second event, respectively, instead of instantaneous growth. Negative values indicate expansion and positive values indicate contraction. See also `Details`.

`mut.rate.prior`

List of length = `num.partitions` or 1, vector, or positive value. Prior distribution for mutation rate μ. See also `Details`. Required if `length.seq` is specified.

MODEL SPECIFICATIONS

`dirichlet.process`

Logical value. Activates a Dirichlet-process hyperprior that weighs all allowable combinations of Ψ/ψ and ζ/ζ_S according to possible combinations of taxa assignment. See also `Details`.

`idiosyncratic`

Logical value. Allows idiosyncratic taxa that freely vary *i.e.* are ungrouped from any of the pulses, as specified by the `psi.prior` arguments. See also `Details`.

`min.net.tau.zeta.total,`
`min.net.tau2.zeta.total,`
`min.net.epsilon.zeta.total,`
`min.net.epsilon2.zeta.total,`
`min.net.NE.zeta.total,`
`max.net.tau.zeta.total,`
`max.net.tau2.zeta.total,`
`max.net.epsilon.zeta.total,`
`max.net.epsilon2.zeta.total,`
`max.net.NE.zeta.total`

Non-negative proportion (*i.e.* $\leq$ 1 and $\geq$ 0). Rule for the minimum/maximum ζ_T value, across all pulses (as specified by the corresponding `psi.prior`) and partitions, for $\tau1$, $\tau2$, $\varepsilon1$, $\varepsilon2$, and N_E, respectively. See also `Details`.

`min.net.tau.zeta.per.pulse,`
`min.net.tau2.zeta.per.pulse,`
`min.net.epsilon.zeta.per.pulse,`
`min.net.epsilon2.zeta.per.pulse,`
`min.net.NE.zeta.per.pulse,`
`max.net.tau.zeta.per.pulse,`
`max.net.tau2.zeta.per.pulse,`
`max.net.epsilon.zeta.per.pulse,`
`max.net.epsilon2.zeta.per.pulse,`
`max.net.NE.zeta.per.pulse`

List, vector of length = maximum value in corresponding `psi.prior`, or non-negative proportion (*i.e.* $\leq$ 1 and $\geq$ 0). Rule for the minimum/maximum ζ_j value of $\tau1$, $\tau2$, $\varepsilon1$, $\varepsilon2$, and N_E, respectively, for each *j*th pulse from 1 to Ψ/ψ (as specified by the corresponding `psi.prior`) across all partitions. See also `Details`.

`tau.buffer, tau2.buffer,`
`epsilon.buffer, epsilon2.buffer,`
`NE.buffer`

Non-negative value or function. Pulse buffer β of the demographic parameter summaries $\tau1_S$, $\tau2_S$, $\varepsilon1_S$, $\varepsilon2_S$, and N_S, respectively. See also `Details`.

`tau.idiosyncratic.buffer,`
`tau2.idiosyncratic.buffer,`
`epsilon.idiosyncratic.buffer,`
`epsilon2.idiosyncratic.buffer,`
`NE.idiosyncratic.buffer`

Non-negative value or function. Idiosyncratic buffer β_i of the idiosyncratic taxon-specific demographic parameters $\tau1_i$, $\tau2_i$, $\varepsilon1_i$, $\varepsilon2_i$, and N_i, respectively. See also `Details`.

`idiosyncratic.rule`

Character string with possible values `“recent”` and `“ancient”`. Activates rule forcing all idiosyncratic taxa to have values less than the first shared pulse, or values greater than the last shared pulse, respectively. Any other values results in no such rules being placed on idiosyncratic taxa. See also `Details`.

`num.changes`

List, vector of length = `num.partitions`, or value of 1 or 2. Number of demographic change events per taxon. See also `Details`.

`flip`

List, vector of length = `num.partitions`, or logical value. Activates $\tau 2$ to be more recent than $\tau 1$. See also `Details`.

PARAMETER SUMMARIES

`net.zeta.total,`
`net.zeta.per.pulse`

Logical value. Activates output of ζ_T and the vector ζ/ζ_S across partitions, respectively, as a list element/matrix in the `roll.object` list element of the final output, for downstream estimation. Rows of the matrix correspond to individual simulations, which correspond to rows of other matrices in the `roll.object` list element of the final output, and each cell is the aforementioned value.

`mean.tau.shared,`
`mean.tau2.shared,`
`mean.epsilon.shared,`
`mean.epsilon2.shared,`
`mean.NE.shared`

Logical value. Activates output of $E(\tau 1_S)$, $E(\tau 2_S)$, $E(\varepsilon 1_S)$, $E(\varepsilon 2_S)$, and $E(N_S)$ weighted by the vector ζ/ζ_S, respectively, as a list element/matrix in the `roll.object` list element of the final output, for downstream estimation. Rows of the matrix correspond to individual simulations, which correspond to rows of other matrices in the `roll.object` list element of the final output, and each cell is the aforementioned value. See also `Details`.

`mean.tau, mean.tau2,`
`mean.epsilon, mean.epsilon2,`
`mean.NE`

Logical value. Activates output of $E(\tau 1)$, $E(\tau 2)$, $E(\varepsilon 1)$, $E(\varepsilon 2)$, and $E(N)$, respectively, as a list element/matrix in the `roll.object` list element of the final output, for downstream estimation. Rows of the matrix correspond to individual simulations, which correspond to rows of other matrices in the `roll.object` list element of the final output, and each cell is the aforementioned value.

`disp.index.tau.shared,`
`disp.index.tau2.shared,`
`disp.index.epsilon.shared,`
`disp.index.epsilon2.shared,`
`disp.index.NE.shared`

Logical value. Activates output of $\Omega(\tau 1_S)$, $\Omega(\tau 2_S)$, $\Omega(\varepsilon 1_S)$, $\Omega(\varepsilon 2_S)$, and $\Omega(N_S)$ weighted by the vector ζ/ζ_S, respectively, as a list element/matrix in the `roll.object` list element of the final output, for downstream estimation. Rows of the matrix correspond to individual simulations, which correspond to rows of other matrices in the `roll.object` list element of the final output, and each cell is the aforementioned value. See also `Details`.

`disp.index.tau, disp.index.tau2,`
`disp.index.epsilon,`
`disp.index.epsilon2,`

Logical value. Activates output of $\Omega(\tau 1)$, $\Omega(\tau 2)$, $\Omega(\varepsilon 1)$, $\Omega(\varepsilon 2)$, and $\Omega(N)$, respectively, as a list element/matrix in the `disp.index.NE`

`roll.object` list element of the final output, for downstream estimation. Rows of the matrix correspond to individual simulations, which correspond to rows of other matrices in the `roll.object` list element of the final output, and each cell is the aforementioned value. See also `Details`.

SIMULATION SPECIFICATIONS

`fsc2path`

Character string of path for `fastsimcoal2` executable file. Must include name of `fastsimcoal2` executable file. May be absolute path (*i.e.* beginning with “/”) or relative path from `output.directory`. Required.

`messages.sims`

Positive integer. Interval length in number of completed simulations for each cycle of messages (*e.g.* if = 10,000, a message is given when 10,000, 20,000, 30,000, etc. simulations are completed). Regardless of value or whether specified, a message is always given when the last simulation is completed.

`output.directory`

Character string of path for directory where output simulation files are deposited, as well as from where `fastsimcoal2` is run (*i.e.* working directory changes to `output.directory` when running `fastsimcoal2`, but returns to current working directory). May or may not end with “/”. May be absolute path (*i.e.* beginning with “/”) or relative path from current working directory. See also `Details`. Required.

`append.sims`

Logical value. Allows output to be added to existing files with the same filename that are located within `output.directory`. See also `Details`.

`keep.taxa.draws`,
`output.hyper.draws`,
`output.taxa.draws`,
`keep.fsc2.files`

Logical value. Activates whether taxon-specific parameter draws (*i.e.* `sim.specs` list elements) are outputted in `R` (not activating this may increase performance speed and decrease memory usage), whether hyperparameter and parameter summary values (*i.e.* `roll.object` list elements) are outputted to a simple text file within `output.directory`, whether taxon-specific parameter draws are outputted to a simple text file within `output.directory`, and whether the `fastsimcoal2` outputs `seed.txt` and `.lhood` are kept in simple text files per simulated independent taxon with respective filename suffixes `.seed` and `.fsc2` within `output.directory`, respectively.

OBJECTS FROM PRECEDING FUNCTIONS

`build.object`

Output from function `build.dice`.

`roll.object` Output from function `roll.dice`.

`play.object` Output from function `play.dice`. See also `Details`.

Arguments from other `Multi-DICE` functions may be included here and are ignored if not applicable.

Details

`Multi-DICE` cannot currently accommodate models with more than one population per taxon, events aside from population size change, and more than two size change events.

For $\tau1$ and $\tau2$, units are in numbers of generations, and thus may only be positive integers. For $\varepsilon1$ and $\varepsilon2$, units are in ratio of size change from the ancestral effective population size to current effective population size, such that expansions are < 1 and contractions are > 1, and thus may only be positive values. For N_E, unit is in number of effective haploid individuals, and thus may only be positive integers.

For `num.taxa`, `tau.psi.prior`, `epsilon.psi.prior`, `NE.psi.prior`, `tau.shared.prior`, `tau2.shared.prior`, `NE.shared.prior`, `tau.idio.prior`, `tau2.idio.prior`, `NE.idio.prior`, `linked.param.prior` (except when corresponding `linked.param` = `“epsilon”` or `“epsilon2”`), `anchor.prior`, `change.prior`, `tau.buffer`, `tau2.buffer`, `NE.buffer`, `tau.idiosyncratic.buffer`, `tau2.idiosyncratic.buffer`, `NE.idiosyncratic.buffer`, and `num.changes`, non-integer values are converted to integer values via `as.integer`. Similarly, after being multiplied by n, `tau.zeta.prior`, `tau2.zeta.prior`, `epsilon.zeta.prior`, `epsilon2.zeta.prior`, `NE.zeta.prior`, `tau.zeta.total.prior`, `tau2.zeta.total.prior`, `epsilon.zeta.total.prior`, `epsilon2.zeta.total.prior`, `NE.zeta.total.prior`, `min.net.tau.zeta.total`, `min.net.tau2.zeta.total`, `min.net.epsilon.zeta.total`, `min.net.epsilon2.zeta.total`, `min.net.NE.zeta.total`, `max.net.tau.zeta.total`, `max.net.tau2.zeta.total`, `max.net.epsilon.zeta.total`, `max.net.epsilon2.zeta.total`, `max.net.NE.zeta.total`, `min.net.tau.zeta.per.pulse`, `min.net.tau2.zeta.per.pulse`, `min.net.epsilon.zeta.per.pulse`, `min.net.epsilon2.zeta.per.pulse`, `min.net.NE.zeta.per.pulse`, `max.net.tau.zeta.per.pulse`, `max.net.tau2.zeta.per.pulse`, `max.net.epsilon.zeta.per.pulse`, `max.net.epsilon2.zeta.per.pulse`, and `max.net.NE.zeta.per.pulse` are converted to integer values via `as.integer` to represent S and S_T.

For `num.taxa`, `num.haploid.samples`, `num.ind.sites`, `num.SNPs`, `length.seq`, `folded`, `sampling.times`, `gen.times`, `linked.param`, `attached.hyper`, `linked.param.fixed`, `min.net.tau.zeta.per.pulse`, `min.net.tau2.zeta.per.pulse`, `min.net.epsilon.zeta.per.pulse`, `min.net.epsilon2.zeta.per.pulse`, `min.net.NE.zeta.per.pulse`, `max.net.tau.zeta.per.pulse`, `max.net.tau2.zeta.per.pulse`, `max.net.epsilon.zeta.per.pulse`, `max.net.epsilon2.zeta.per.pulse`, `max.net.NE.zeta.per.pulse`, `num.changes`, and `flip`, if list, then all list elements are concatenated to form a single vector, with the ordering within list elements and then between list elements preserved (*e.g.* for list of length = 2, with first list element of length = 2 and second list element of length = 1, the order from first to last is: 1) first vector element in first list element; 2) second vector element in first list element; 3) sole vector element in second list element).

For `tau.psi.prior`, `epsilon.psi.prior`, `NE.psi.prior`, `tau.zeta.prior`, `tau2.zeta.prior`, `epsilon.zeta.prior`, `epsilon2.zeta.prior`, `NE.zeta.prior`, `tau.zeta.total.prior`, `tau2.zeta.total.prior`, `epsilon.zeta.total.prior`, `epsilon2.zeta.total.prior`, `NE.zeta.total.prior`, `tau.shared.prior`, `tau2.shared.prior`, `epsilon.shared.prior`, `epsilon2.shared.prior`, `NE.shared.prior`, `tau.idio.prior`, `tau2.idio.prior`, `epsilon.idio.prior`, `epsilon2.idio.prior`, `NE.idio.prior`, `linked.param.prior`, `anchor.prior`, `change.prior`, `exponential.growth.rate.prior`, `exponential.growth.rate.prior2`, and `mut.rate.prior`, each list element contains an entire individual discrete distribution; if vector, then converted to list of length = 1 with all vector elements comprising the entirety of a single discrete distribution. Per list element, vector elements within (*i.e.* the discrete distribution) do not need to be in any particular order. Relatedly, each vector element is treated as an independent value, thus weighted distributions (*i.e.* not uniform) may be employed by duplicating values (*e.g.* a distribution of `c(0,1,1,1)` signifies 75% probability of drawing “1” and 25% probability of drawing “0”), allowing the specification of any discretized distribution (*e.g.* gamma, beta, log-uniform). Accordingly, a uniform distribution with no gaps for integer values would be of length = range of distribution.

For `num.taxa`, `num.haploid.samples`, `num.ind.sites`, `num.SNPs`, `length.seq`, `folded`, `sampling.times`, `gen.times` (except when vector of length = *n*), `num.changes`, and `flip`, the order of vector elements corresponds to the order of partitions, and for `tau.zeta.prior`, `tau2.zeta.prior`, `epsilon.zeta.prior`, `epsilon2.zeta.prior`, `NE.zeta.prior`, `tau.idio.prior`, `tau2.idio.prior`, `epsilon.idio.prior`, `epsilon2.idio.prior`, `NE.idio.prior`, `change.prior`, `exponential.growth.rate.prior`, `exponential.growth.rate.prior2`, and `mut.rate.prior`, the order of list elements corresponds to the order of partitions. Additionally, if length = 1 and `num.partitions` > 1, then the sole element is used for all partitions. Similarly, if length < `num.partitions`, then the first element is used for all partitions while ignoring any remaining elements. Except for `gen.times`, if length > `num.partitions`, then the remaining elements beyond length = `num.partitions` are ignored; a caution is provided when the length does not equal 1 or `num.partitions`. For `gen.times`, if length > `num.partitions` and length < *n*, then the remaining elements beyond length = `num.partitions` are ignored, and if length > *n*, then the remaining elements beyond length = *n* are ignored; a caution is provided when the length does not equal 1, `num.partitions`, or *n*.

For `min.net.tau.zeta.per.pulse`, `min.net.tau2.zeta.per.pulse`, `min.net.epsilon.zeta.per.pulse`, `min.net.epsilon2.zeta.per.pulse`, `min.net.NE.zeta.per.pulse`, `max.net.tau.zeta.per.pulse`, `max.net.tau2.zeta.per.pulse`, `max.net.epsilon.zeta.per.pulse`, `max.net.epsilon2.zeta.per.pulse`, and `max.net.NE.zeta.per.pulse`, the order of vector elements corresponds to the temporal order, from most recent to most ancient, of pulses (as specified by the corresponding `psi.prior`), and for `tau.shared.prior`, `tau2.shared.prior`, `epsilon.shared.prior`, `epsilon2.shared.prior`, `NE.shared.prior`, and `anchor.prior`, the order of list elements corresponds to the temporal order, from most recent to most ancient, of pulses (as specified by the corresponding `psi.prior`). Additionally, if length = 1 and maximum value in corresponding `psi.prior` > 1, then the sole element is used for all pulses. Similarly, if length < maximum value in corresponding `psi.prior`, then the first element is used for all pulses while ignoring any remaining elements. For the `zeta.per.pulse` arguments, if length > maximum value in corresponding `psi.prior`, then the remaining elements beyond length = maximum value in corresponding `psi.prior` are ignored; a caution is provided when the length does not equal 1 or maximum value in corresponding `psi.prior`. For the `shared.prior` arguments and `anchor.prior`, there may be additional list elements for idiosyncratic distributions, such that any total length = 1, maximum value in corresponding `psi.prior`, maximum value in corresponding `psi.prior` + 1, or maximum value in corresponding `psi.prior` + `num.partitions`, are allowed; for any other lengths, a caution is provided and excess elements beyond the highest acceptable length are ignored. See below for more information about adding idiosyncratic distributions to these arguments.

If `num.partitions=`*n*, then rearrangement of bins across taxa within allele frequency classes based on descending order of the relative SNP proportions is not performed to construct the aSFS and taxon-specific inference of demographic parameters is possible. However, in general, more partitions results in more parameter space with respect to taxa samples that must be explored due to a decrease in order-independence and assumed exchangeability, thus multiple-fold more simulations must be conducted to achieve comparable accuracy in hyperparameter estimation as without partitioning.

For `num.ind.sites`, `num.SNPs`, and `length.seq`, `Multi-DICE` first looks in `num.ind.sites`, then `num.SNPs`, and finally `length.seq`. For `num.ind.sites`, specified values equate to the number of genealogies simulated by the `FREQ` simulation model in `fastsimcoal2`. For `num.SNPs`, specified values equate to the number of independent loci (chromosomal structure is not considered) simulated by the `SNP` simulation model in `fastsimcoal2`. For `length.seq`, specified values equate to the number of loci within a linkage block (*i.e.* sites, or length of sequence; recombination is not considered) simulated by the `SNP` simulation model in `fastsimcoal2`; to utilize `length.seq`, `mut.rate.prior` must also be specified. For both the `FREQ` and `SNP` simulation models, infinite sites are assumed by `fastsimcoal2`. Both `num.SNPs` and `length.seq` may be specified to simulate monomorphic sites and linked SNPs for the SFS, *i.e.* derive aSFS from genomic-scale whole-sequence information.

If `gen.times` is specified, output τ values are in units of years. If generation times are equivalent across all *n* taxa, then it is not critical to specify `gen.times`, since output τ values may be multiplied by the generation time scalar to covert units to years if `gen.times` is not specified. Hence, if generation time is 1 year, then output τ values are in units of years as well.

For `tau.psi.prior`, `epsilon.psi.prior`, and `NE.psi.prior`, distinguishing between Ψ and ψ is accomplished via the corresponding `zeta.prior`, `zeta.total.prior`, `idiosyncratic` setting, and/or `min`/`max.net` `zeta.total`/`per.pulse`, except for values of 0, which are explicitly for $\psi = 0$ and thus indicate full idiosyncrasy. If list length > 2 for τ and ε or list length > 1 for N_E, then a caution is provided and the remaining elements beyond length = 2 for τ and ε and length = 1 for N_E are ignored. Applies across all partitions, such that it is regardless of partitioning.

For `tau.zeta.prior`, `tau2.zeta.prior`, `epsilon.zeta.prior`, `epsilon2.zeta.prior`, and `NE.zeta.prior`, if `num.partitions` > 1, may be necessary to include “0.0” as a value, but can control ζ_S and ζ_T across partitions via corresponding `zeta.total.prior` and/or `min`/`max.net` `zeta.total`/`per.pulse`. Attributes to each partition individually, but proportion values are out of the entirety of taxa dataset, thus if `num.partitions` > 1, then the upper bound for each partition should be the number of taxa within that partition divided by *n*. When ψ = {0, 1}, equivalent to hyperprior distribution for ζ_T. If identical across all partitions, it is more computationally efficient to specify only one *i.e.* list of length = 1, or a vector.

For `tau.zeta.total.prior`, `tau2.zeta.total.prior`, `epsilon.zeta.total.prior`, `epsilon2.zeta.total.prior`, and `NE.zeta.total.prior`, if length > 1, then remaining elements beyond the first are ignored and a caution is provided.

To build a hyperprior, `Multi-DICE` first looks if the corresponding `zeta.total.prior` is specified, then if `dirichlet.process=T`, and if neither is such case, then all possible combinations of corresponding `psi.prior` and `zeta.prior` draws are equally weighted. Therefore, if `num.partitions=1`, corresponding `psi.prior=1`, and `dirichlet.process=F`, then `tau.zeta.total.prior`, `tau2.zeta.total.prior`, `epsilon.zeta.total.prior`, `epsilon2.zeta.total.prior`, and `NE.zeta.total.prior` equivalent to corresponding `zeta.prior` and thus unnecessary to specify.

For `tau.shared.prior`, `tau2.shared.prior`, `epsilon.shared.prior`, `epsilon2.shared.prior`, and `NE.shared.prior`, each successive list element/distribution must have a greater minimum and maximum value than its preceding list elements/distributions. If multiple distributions are utilized for every potential pulse and these distributions overlap in their bounds, running time may slow since, in this case, draws are made from all the distributions independently and then checked if abiding by ordering and buffering, with re-draws if not.

For each of `tau`, `tau2`, `epsilon`, `epsilon2`, and `NE`, if hyperparameterized via its corresponding `psi.prior` while `idiosyncratic=T` or a 0 value is in said `psi.prior`, then a corresponding idiosyncratic prior is required. `Multi-DICE` first looks in `change.prior` (for τ2 only), then the corresponding `idio.prior`, and finally the corresponding `shared.prior`. If a `shared.prior` argument is utilized, the additional list elements beyond length = maximum value in corresponding `psi.prior` are considered; if the number of list elements is ≤ maximum value in corresponding `psi.prior`, then only the first list element is considered. These list elements undergo the same specifications as aforementioned for the corresponding `idio.prior`.

For each of `tau`, `tau2`, `epsilon`, `epsilon2`, and `NE`, if not hyperparameterized yet part of the specified model (*i.e.* `tau`, `epsilon`, and `NE` are always part of the model, and `tau2` and `epsilon2` are part of the model when `num.changes` = 2), then a corresponding nuisance prior is required. A nuisance prior differs from an idiosyncratic prior in that variation in the respective nuisance parameter is being considered while not of interest with respect to hyperparameterization (*i.e.* variability in values across taxa governed hierarchically), whereas an idiosyncratic prior is still governed by hyperparameters in coordination with the corresponding shared prior. `Multi-DICE` first looks in `anchor.prior` (for τ2 only), next `change.prior` (for τ2 only), afterward `linked.param.prior` (if applicable; see below for more information), then the corresponding `idio.prior`, and finally the corresponding `shared.prior`. If `linked.param.prior` is utilized for a particular nuisance parameter, either the corresponding `idio.prior` or `shared.prior` must still be specified even if all taxa are always covered by `linked.param.prior` across all simulations. If a `shared.prior` argument is utilized, it undergoes the same specifications as aforementioned for the corresponding `idio.prior`.

If a two-event model is specified, there is an interplay between $\tau1$ and $\tau2$ with respect to synchronous/shared, idiosyncratic, and nuisance draws. If both $\tau1$ and $\tau2$ are hyperparameterized, draws from the $\tau1_s$ synchronous/shared prior and $\tau2_s$ synchronous/shared prior are made independently, but if for a given simulation there are no valid combinations of these draws with respect to the vectors $\zeta_{\tau1}/\zeta_{\tau1,s}$ and $\zeta_{\tau2}/\zeta_{\tau2,s}$ such that $\tau1 < \tau2$ is not violated, then the $\tau2_s$ synchronous/shared prior is re-drawn for this simulation until there is a valid combination. Hence, the $\tau2_s$ synchronous/shared prior is conditional on the $\tau1_s$ synchronous/shared prior. Furthermore, for each $\tau2_s$ draw that is not assigned to a $\tau1_s$ draw (*i.e.* cases where $\tau1$ is idiosyncratic while $\tau2$ is synchronous/shared/in a Ψ/ψ pulse as specified by `tau2.psi.prior`), the according $\tau1_i$ idiosyncratic draw is restricted by that $\tau2_s$ value. Therefore, the $\tau1_i$ idiosyncratic prior is conditional on the $\tau2_s$ synchronous/shared prior (if specified). Lastly, every $\tau2$ idiosyncratic or nuisance draw is confined by its according $\tau1$ drawn value, whether it is synchronous/shared, idiosyncratic, or nuisance, thus both the $\tau2_i$ idiosyncratic prior and the $\tau2$ nuisance prior are conditional on the $\tau1_s$ synchronous/shared prior, $\tau1_i$ idiosyncratic prior, and $\tau1$ nuisance prior (whichever is specified).

Considering the aforementioned dependencies between $\tau1$ and $\tau2$, it is highly recommended that prior distribution bounds are as far apart as possible among the two events, ideally mutually exclusive/non-overlapping. Otherwise, there could be a stop of operation due to an invalid/incompatible draw, computational lag, and/or statistical bias on prior distributions.

For `linked.param`, `attached.hyper`, `linked.param.partition`, `attached.hyper.pulse`, `linked.param.prior`, and `linked.param.fixed`, nuisance demographic parameters can be linked to other hyperparameterized demographic parameters, such that taxa within a pulse for a hyperparameterized parameter are also grouped for parameterization for another nuisance parameter. When activating this feature, all six arguments must be specified. Additionally, all six arguments must be of length = length of `linked.param`, except for `linked.param.partition`, `attached.hyper.pulse`, `linked.param.prior`, and `linked.param.fixed`, which may be of length = 1, with the sole element being duplicated to length of `linked.param`; for `linked.param.partition` and `attached.hyper.pulse`, if vector, then converted to list of length = 1. The order of elements in `linked.param` then corresponds to the order of elements in the other five arguments, such that each of these elements forms a different entry. For `linked.param`, nuisance parameters to be linked are specified. For `attached.hyper`, hyperparameterized demographic parameters to be attached are specified. For `linked.param.partition`, the applicable partitions, indexed numerically in the same order as user-specified, that are linking the nuisance parameter is/are specified. For `attached.hyper.pulse`, the applicable pulses of the hyperparameterized parameter is/are specified. For `linked.param.prior`, the nuisance priors are specified. For `linked.param.fixed`, whether all applicable taxa have a shared value in the nuisance parameter or independently draw (*i.e.* vary in values) from the same nuisance prior is specified. By specifying each pulse separately through `attached.hyper.pulse` and fixing each pulse to have a shared value through `linked.param.fixed`, the nuisance parameter would essentially be hyperparameterized by the same hyperparameters as the hyperparameterized parameter. For example, if `linked.param=c("epsilon", "epsilon", "epsilon")`, `attached.hyper=c("tau", "tau", "tau")`, `attached.hyper.pulse=list(1,2,3)`, and `linked.param.fixed=T`, then $\varepsilon1$ and $\tau1$ would be governed identically by $\zeta_{\tau1}/\zeta_{\tau1,s}$. Similarly, a nuisance prior may be specified differentially between taxa within a shared pulse and idiosyncratic taxa relative to another, hyperparameterized parameter. For example, if `linked.param=c("epsilon"),attached.hyper=c("tau"), linked.param.prior=c(1000:10000)/100000`, and `epsilon.idio.prior=c(1000:100000)/100000`, then taxa that are in synchronous $\tau1$ pulses have the prior distribution $\varepsilon1 \sim U(0.01, 0.10)$ whereas $\tau1$ temporally idiosyncratic taxa have the prior distribution $\varepsilon1 \sim U(0.01, 1.00)$. If a partition of taxa has a nuisance parameter linked to multiple parameters across overlapping entries (*e.g.* `"epsilon"` and `"1"` are both specified in the first two elements for `linked.param` and `linked.param.partition`, respectively, and `"tau"` and `"tau2"` are the first two elements in `attached.hyper` such that the first partition of taxa has the nuisance $\varepsilon1$ parameter attached to both `"tau"` and `"tau2"`), then latter-specified entries may replace previously-specified entries. If for a given entry, `linked.param="tau"`, vector length of `linked.param.partitions` > 1, and `linked.param.fixed=T`, then the `flip` value for the first partition specified in `linked.param.partitions` is assumed to be the same across all the vector elements/partitions in `linked.param.partitions`.

For `anchor.prior`, a special case of linking a nuisance demographic parameter to a hyperparameterized demographic parameter occurs where $\tau2$ is linked to $\tau1_S$ and the difference between these two values δ is parameterized. Per each synchronous/shared pulse in $\tau1_S$, as specified by `tau.psi.prior`, a draw from `anchor.prior` determines the $\tau2$ value for taxa within said pulse. For example, if five taxa are in the first $\tau1$ synchronous pulse and there is an according draw of 100,000 from `anchor.prior`, then $\tau2 = \tau1_{S,1} + 100{,}000$ for all five of those taxa. Different distributions can be employed per $\tau1_S$ synchronous/shared pulse in `anchor.prior` (see above for more information). Moreover, additional distributions specified in `anchor.prior` beyond length = maximum value in `tau.psi.prior` are used for taxa that have $\tau1_I$ idiosyncratic values. If the number of additional distributions < `num.partitions`, then the first additional distribution is used for all partitions, and if the number of additional distributions ≥ `num.partitions`, then the order of the first `num.partitions` additional distributions corresponds to the order of partitions. If there are no additional distributions, then the first distribution in `anchor.prior` is used for $\tau1_I$ idiosyncratic draws across all partitions. For $\tau1_I$ idiosyncratic values, independent draws are made from `anchor.prior` among taxa, acting here similarly to `change.prior` (see below for more information). Since `anchor.prior` links $\tau1$ and $\tau2$ together across all taxa as essentially one set of hyperparameterization, `flip` would have no effect and thus is ignored here. The parameter summary vector δ_S of anchor values corresponding to the vector $\tau1_S$ is outputted as a list element/matrix in the `roll.object` list element of the final output, available for downstream estimation. Rows of the matrix correspond to individual simulations, which correspond to rows of other matrices in the `roll.object` list element of the final output, and column order corresponds to the temporal order, from most recent to most ancient, of $\tau1_S$ synchronous/shared pulses.

For `change.prior`, a similar case to `anchor.prior` is employed in that the difference δ between the two values $\tau1$ and $\tau2$ is parameterized, except $\tau2$ is not linked to the hyperparameterization of $\tau1$. Thus, taxon-specific independent draws are conducted on `change.prior`, which can apply either as an idiosyncratic or nuisance prior for $\tau2$. If for a given partition, `change.prior` is employed and `flip=T`, then the maximum value in `change.prior` must be < than the minimum value of `tau.shared.prior` and `tau.idio.prior`; this is not checked and thus it is critical that it is user-confirmed.

For `exponential.growth.rate.prior` and `exponential.growth.rate.prior2`, growth occurs according to an exponential growth model with drawn values $r1$ and $r2$ until the drawn values $\varepsilon1$ and $\varepsilon2$ are met, respectively. The more ancient time, *i.e.* beginning of size change forward-in-time, is represented by $\tau1$ and $\tau2$, respectively, and thus is the time that is buffered, whereas the more recent time, *i.e.* end size change forward-in-time, is a nuisance parameter. There are no checks of the nuisance end time violating any model specifications (*e.g.* if $\tau1$ is very recent and/or the duration of first event growth is very long, then the first event beginning time may be a negative value, or if the difference between $\tau1$ and $\tau2$ is small and/or the duration of second event growth is very long, then the second event beginning time may be prior to $\tau1$; both examples assume `flip=F`, see below for more information), thus it is critical that this is thoroughly user-investigated. If $r1/r2$ contradicts $\varepsilon1/\varepsilon2$ regarding demographic syndrome, the demographic syndrome indicated by $r1/r2$ and the inverse of $\varepsilon1/\varepsilon2$ are employed, *e.g.* if $r1$ is negative (expansion) while corresponding $\varepsilon1 > 1$ (contraction), then expansion occurs until $1/\varepsilon1$ is met.

For `mut.rate.prior`, utilized only if `length.seq` is utilized as well. Values of 0 may technically be included, though this results in all polymorphic sites. If `keep.taxa.draws=T`, draws are outputted as a list element/matrix named `mut.rate` in the `sim.specs` list element of the final output. Each row represents an individual simulation, each column represents a simulated independent taxon, and each cell is the according mutation rate value used for simulation. Rows and columns correspond to other matrices *i.e.* the same row number refers to the same simulation and the same column number refers to the same simulated independent taxon.

For `exponential.growth.rate.prior`, `exponential.growth.rate.prior2`, and `mut.rate.prior`, $r1$, $r2$, and μ, respectively, only act as nuisance parameters here, thus may differentiate in prior distributions across partitions, but cannot be hyperparameterized.

If `idiosyncratic=F`, then `min.net.tau.zeta.total`, `min.net.tau2.zeta.total`, `min.net.epsilon.zeta.total`, `min.net.epsilon2.zeta.total`, `min.net.NE.zeta.total`, `max.net.tau.zeta.total`, `max.net.tau2.zeta.total`, `max.net.epsilon.zeta.total`, `max.net.epsilon2.zeta.total`, and `max.net.NE.zeta.total`, if specified, may only equal 1.0.

For `tau.buffer`, `tau2.buffer`, `epsilon.buffer`, `epsilon2.buffer`, and `NE.buffer`, β buffers are applied to draws of shared pulse values, thus affecting the corresponding shared prior; the corresponding idiosyncratic prior, from which draws are subsequently made, is accordingly affected by the shared pulse buffers. For `tau.idiosyncratic.buffer`, `tau2.idiosyncratic.buffer`, `epsilon.idiosyncratic.buffer`, `epsilon2.idiosyncratic.buffer`, and `NE.idiosyncratic.buffer`, β_i idiosyncratic buffers are then applied to idiosyncratic draws, which additionally affect the corresponding idiosyncratic prior that had already been initially transformed by the shared pulse buffers. In other words, β buffers idiosyncratic taxa from shared pulse values, and β_i buffers idiosyncratic taxa from each other. For example, if $n = 10$, $\psi_{\tau1} = 2$, $\zeta_{\tau1,T} = 0.8$, $\beta = 10{,}000$, $\beta_i = 1{,}000$, $\tau1 \sim U(1{,}000, 100{,}000)$, $\tau1_s = \{11{,}000, 12{,}001\}$, and $\tau1_i = \{99{,}000\}$ after the first idiosyncratic draw, then given that the synchronous/shared pulse buffers result in invalid draws from $\sim U(1{,}000, 13{,}001)$ and the first idiosyncratic draw buffer results in invalid draws from $\sim U(98{,}000, 100{,}000)$, the remaining second idiosyncratic draw would be from the resulting transformed prior distribution $\tau1_i \sim U(13{,}002, 97{,}999)$. A caution is provided if an `idiosyncratic.buffer` corresponding to a specified `psi.prior` is not specified. Buffers cannot be deployed for nuisance draws (though see below when $\psi = 0$).

For `tau.buffer`, `tau2.buffer`, `epsilon.buffer`, `epsilon2.buffer`, `NE.buffer`, `tau.idiosyncratic.buffer`, `tau2.idiosyncratic.buffer`, `epsilon.idiosyncratic.buffer`, `epsilon2.idiosyncratic.buffer`, and `NE.idiosyncratic.buffer`, if writing a function, there can be only one argument, which is for the value of a particular draw from a corresponding prior distribution, and the output must be a vector of discrete values that are buffered out of any corresponding prior distributions given that particular draw. For `epsilon.buffer`, `epsilon2.buffer`, `epsilon.idiosyncratic.buffer`, and `epsilon2.idiosyncratic.buffer`, given that these are not integers, it is imperative that the output values are on the same scale/interval/significant figures as the corresponding shared and idiosyncratic prior distributions since values are buffered out only if they are exactly equal. There is no check on functionality for `buffer` and `idiosyncratic.buffer` functions, thus it is highly recommended that any function is thoroughly user-tested.

For `num.changes`, only necessary if there are nuisance demographic parameters for a second event, since one event is the default and demographic parameters for the second event that are hyperparameterized by the corresponding `psi.prior` are already activated. If < 1, then there is a stop of operation, and if > 2, then it is converted to `num.changes=2` and a caution is provided.

If `flip=T`, τ2 and ε2 still refers to the second specified event, which in this case is the more recent one.

For `mean.tau.shared`, `mean.tau2.shared`, `mean.epsilon.shared`, `mean.epsilon2.shared`, `mean.NE.shared`, `disp.index.tau.shared`, `disp.index.tau2.shared`, `disp.index.epsilon.shared`, `disp.index.epsilon2.shared`, and `disp.index.NE.shared`, may be applied only if corresponding `psi.prior` is specified, *i.e.* demographic parameter is hyperparameterized. If corresponding `psi.prior` specifies Ψ, then these arguments are equivalent to their corresponding `mean`/`disp.index` arguments (*i.e.* without `.shared` suffix). When full idiosyncrasy (*i.e.* $\psi = 0$, $\zeta_T = 0.0$), default to = 0 (except when specified by corresponding `psi.prior=`Ψ).

For `disp.index.tau.shared`, `disp.index.tau2.shared`, `disp.index.epsilon.shared`, `disp.index.epsilon2.shared`, `disp.index.NE.shared`, `disp.index.tau`, `disp.index.tau2`, `disp.index.epsilon`, `disp.index.epsilon2`, and `disp.index.NE`, when the applicable number of taxa = 1, default to = 0.

There cannot be a filename of “dice.sims.fsc2.template” within `output.directory`; `output.directory=‘.’` to specify current working directory.

If `append.sims=F`, then cannot have a filename of “seed.txt”, “MRCAs.txt”, “dice.log”, “dice.simulations” with a numerical suffix of “1” through *n*, or anything with the prefix “dice.sims” within `output.directory`. If `append.sims=T`, temporary intermediate files may cause files that have the prefix “dice.sims” within `output.directory` to be deleted.

If `play.object` is provided, then the arguments `tau.psi.prior`, `epsilon.psi.prior`, `NE.psi.prior`, `tau.zeta.prior`, `tau2.zeta.prior`, `epsilon.zeta.prior`, `epsilon2.zeta.prior`, `NE.zeta.prior`, `tau.zeta.total.prior`, `tau2.zeta.total.prior`, `epsilon.zeta.total.prior`, `epsilon2.zeta.total.prior`, NE.zeta.total.prior, tau.shared.prior, epsilon.shared.prior, epsilon2.shared.prior, NE.shared.prior, tau.idio.prior, tau2.idio.prior, epsilon.idio.prior, epsilon2.idio.prior, NE.idio.prior, linked.param, attached.hyper, linked.param.partition, attached.hyper.pulse, linked.param.prior, linked.param.fixed, anchor.prior, change.prior, exponential.growth.rate.prior, exponential.growth.rate.prior2, dirichlet.process, idiosyncratic, min.net.tau.zeta.total, min.net.tau2.zeta.total, min.net.epsilon.zeta.total, min.net.epsilon2.zeta.total, min.net.NE.zeta.total, max.net.tau.zeta.total, max.net.tau2.zeta.total, max.net.epsilon.zeta.total, max.net.epsilon2.zeta.total, max.net.NE.zeta.total, min.net.tau.zeta.per.pulse, min.net.tau2.zeta.per.pulse, min.net.epsilon.zeta.per.pulse, min.net.epsilon2.zeta.per.pulse, min.net.NE.zeta.per.pulse, max.net.tau.zeta.per.pulse, max.net.tau2.zeta.per.pulse, max.net.epsilon.zeta.per.pulse, max.net.epsilon2.zeta.per.pulse, max.net.NE.zeta.per.pulse, tau.buffer, tau2.buffer, epsilon.buffer, epsilon2.buffer, NE.buffer, tau.idiosyncratic.buffer, tau2.idiosyncratic.buffer, epsilon.idiosyncratic.buffer, epsilon2.idiosyncratic.buffer, NE.idiosyncratic.buffer, and idiosyncratic.rule are ignored here.

When $\psi = 0$ *i.e.* full idiosyncrasy, the arguments idiosyncratic, min.net.tau.zeta.total, min.net.tau2.zeta.total, min.net.epsilon.zeta.total, min.net.epsilon2.zeta.total, min.net.NE.zeta.total, min.net.tau.zeta.per.pulse, min.net.tau2.zeta.per.pulse, min.net.epsilon.zeta.per.pulse, min.net.epsilon2.zeta.per.pulse, min.net.NE.zeta.per.pulse, and idiosyncratic.rule are ignored, and the arguments tau.idiosyncratic.buffer, tau2.idiosyncratic.buffer, epsilon.idiosyncratic.buffer, epsilon2.idiosyncratic.buffer, and NE.idiosyncratic.buffer are activated (even if idiosyncratic=F).

Multi-DICE proceeds only if a valid draw is allowed given corresponding num.taxa, psi.prior, zeta.prior (*e.g.* minimum value in psi.prior * minimum value in zeta.prior $\leq$ minimum value in num.taxa), zeta.total.prior, shared prior, idiosyncratic prior, idiosyncratic setting, min/max.net zeta.total/per.pulse, buffer, and idiosyncratic.buffer. Importantly, there is no check on if all values in psi.prior have a valid draw, only if there is a valid draw among any of the values. Additionally, the range of a shared prior must be greater than its corresponding $(\Psi/\psi - 1)(2\beta + 1)$, and the range of an idiosyncratic prior ought to be greater than its corresponding $(\Psi/\psi)(2\beta + 1) + (\sigma - 1)(2\beta_i + 1)$ across every possible combination of Ψ/ψ and σ and with consideration given to idiosyncratic.rule. For tau2.shared.prior and tau2.idio.prior, consideration must also be given to any overlap with tau.shared.prior and tau.idio.prior, respectively.

Value

Returned value is identical to that of play.dice, except if keep.taxa.draws=F, in which case the list element sim.specs is omitted, or if keep.taxa.draws=T and both length.seq and mut.rate.prior are utilized, in which case μ draws are also added to sim.specs (see above for more information). Additionally, output simulation files in simple text format are deposited into `output.directory`: the filenames of “`dice.simulations`” with a numerical suffix of `1` through *n* (*i.e.* “`dice.simulations1`”, “`dice.simulations2`”, etc.) contain per-taxon summary statistics, tab delimited; the filenames of “`dice.sims.hyper.draws.`” with a suffix corresponding to the names of hyperparameters and parameter summaries (*i.e.* names of `roll.object` list elements *e.g.* “`psi.tau`” for $\Psi_{\tau1}/\psi_{\tau1}$ as specified by `tau.psi.prior`, “`zeta.tau.1`” for $\zeta_{\tau1}/\zeta_{\tau1,s}$ of the first partition $\zeta_{\tau1}.1/\zeta_{\tau1,s}.1$, “`net.zeta.total.tau`” for $\zeta_{\tau1,T}$, “`net.zeta.per.pulse.tau`” for $\zeta_{\tau1}/\zeta_{\tau1,s}$, “`pulse.values.tau`” for $\tau1_s$ (or $\tau1$ if `tau.psi.prior` specifies $\Psi_{\tau1}$), “`disp.index.tau`” for $\Omega(\tau1)$, etc.) contain the according values of interest for estimation, space delimited; the filenames of “`dice.sims.taxa.draws.`” with a suffix corresponding to the names of taxon-specific parameters (*i.e.* names of `sim.specs` list elements: `tau`, `tau2`, `epsilon`, `epsilon2`, `NE`, `exponential.growth.rate.prior`, `exponential.growth.rate.prior2`, `mut.rate`) contain the according nuisance values used for simulation across taxa, space delimited; if `keep.fsc2.files=T`, the filenames of “`dice.sims`” with a suffix of `1` through *n* attached to either `.seed` and `.fsc2` (*i.e.* “`dice.sims1.seed`”, “`dice.sims1.fsc2`”, “`dice.sims2.seed`”, “`dice.sims2.fsc2`”, etc.) contain, per simulated independent taxon, the `fastsimcoal2` outputs `seed.txt` and `.lhood`, respectively. Each row per file represents an individual simulation, and rows across files correspond to each other *i.e.* the same row number refers to the same simulation across files. For “`dice.sims.taxa.draws.`” files, each column per file represents each simulated independent taxon, and column order across these files and the numeric order implied by the names of the “`dice.simulations`”, “`.seed`”, and “`.fsc2`” files correspond to each other *i.e.* the same column number and filename number refer to the same simulated independent taxon. For “`dice.simulations`” files, each column per file represents an SFS allele frequency class bin (in the order described by the manual for `fastsimcoal2`) or single-sequence summary statistic (in the order of number of haplotypes, haplotype diversity, nucleotide diversity, and Tajima’s *D*), depending on data type simulated, for that respective simulated independent taxon.

Author(s)

Alexander T. Xue

References

Chan YL, Schanzenbach D, Hickerson MJ (2014) Detecting concerted demographic response across community assemblages using hierarchical approximate Bayesian computation. *Molecular Biology and Evolution*, **31**, 2501–2515.

Excoffier L, Dupanloup I, Huerta-Sánchez E, Sousa VC, Foll M (2013) Robust demographic inference from genomic and SNP data. *PLoS genetics*, **9**, e1003905.

Excoffier L (2014) fastsimcoal2 manual.

Hickerson MJ, Stahl E, Takebayashi N (2007) msBayes: Pipeline for testing comparative phylogeographic histories using hierarchical approximate Bayesian computation. *BMC bioinformatics*, **8**, 268.

Huang W, Takebayashi N, Qi Y, Hickerson MJ (2011) MTML-msBayes: approximate Bayesian comparative phylogeographic inference from multiple taxa and multiple loci with rate heterogeneity. *BMC bioinformatics*, **12**, 1.

Xue AT (2017) Multi-DICE Manual.

Xue AT, Hickerson MJ (2015) The aggregate site frequency spectrum for comparative population genomic inference. *Molecular Ecology*, **24**, 6223–6240.

Xue AT, Hickerson MJ (*submitted*) Multi-DICE: R package for comparative population genomic inference under multi-taxa hierarchical co-demographic models.

See Also

`build.dice`, `roll.dice`, `play.dice`, `dice.aSFS`, `dice.sumstats`

Examples

```
#simplest execution akin to approach in Xue and Hickerson (2015)
dice.sims(num.sims=5, num.taxa=10, num.haploid.samples=10,
num.ind.sites=2000, tau.psi.prior=c(1), tau.zeta.prior=c(1:10)/10,
tau.shared.prior=c(1000:1000000), epsilon.idio.prior=c(1000:10000)/100000,
NE.idio.prior=c(1000:100000), fsc2path='example', output.directory='.')

#simplest execution akin to approach in Xue and Hickerson (2015); ln U
distribution applied on tau.shared.prior, with 100,000 intervals
discretized uniformly across ln(tau.shared.prior)
dice.sims(num.sims=5, num.taxa=10, num.haploid.samples=10,
num.ind.sites=2000, tau.psi.prior=c(1), tau.zeta.prior=c(1:10)/10,
tau.shared.prior=exp(c((log(1000)*100000):(log(1000000)*100000))/100000),
epsilon.idio.prior=c(1000:10000)/100000, NE.idio.prior=c(1000:100000),
fsc2path='example', output.directory='.')

#simplest execution akin to approach in Xue and Hickerson (2015); assuming
play.dice was previously performed and the output was directed to object
play.object
dice.sims(num.sims=5, num.taxa=10, num.haploid.samples=10,
num.ind.sites=2000, fsc2path='example', output.directory='.',
play.object=play.object)

#simplest execution akin to approach in software package msBayes
dice.sims(num.sims=5, num.taxa=10, num.haploid.samples=10,
num.ind.sites=2000, tau.psi.prior=c(1:10), tau.zeta.prior=c(1:10)/10,
tau.shared.prior=c(1000:1000000), epsilon.idio.prior=c(1000:10000)/100000,
NE.idio.prior=c(1000:100000), idiosyncratic=F, fsc2path='example',
output.directory='.')
```

dice.aSFS {Multi-DICE} R Documentation

Constructing aSFS from per-taxon SFS simulations/data

Description

`dice.aSFS` transforms per-taxon SFS, either simulated by `dice.sims` or empirically produced (*i.e.* observed/collected data), into the aSFS, following the procedure described in Xue and Hickerson (2015).

Usage

```
dice.aSFS(num.sims, num.taxa, num.partitions=1, num.haploid.samples,
folded=T, remove.afclasses=NULL, output.directory='.',
input.directory=NULL, input.base=NULL, input.files=NULL)
```

Arguments

`num.sims`

Positive integer. Number of simulations. Required. See also `Details`.

DATA

`num.taxa`

List, vector of length = `num.partitions`, or positive integer. Number of taxa per partition. Total sum across partitions equals the total number of taxa *n* in dataset. See also `Details`. Required.

`num.partitions`

Positive integer. Number of partitions for taxa in dataset. Allows differential data and model specifications across user-specified taxa groupings, including sampling size of individuals, generation times, demographic syndrome, and taxon-specific nuisance parameter prior distributions. See also `Details`.

`num.haploid.samples`

List, vector of length = `num.partitions`, or positive integer. Number of haploid samples per partition. See also `Details`. Required.

`folded`

List, vector of length = `num.partitions`, or logical value. Activates folding of the SFS per partition. See also `Details`.

`remove.afclasses`

List of length = `num.partitions` or 1, vector, or non-negative integer. Allele frequency classes to be removed from the SFS per partition prior to aSFS construction, with 0 referring to monomorphic bins, 1 referring to singleton bins, 2 referring to doubleton bins, etc. By default, monomorphic bins are removed. See also `Details`.

SIMULATION SPECIFICATIONS

output.directory, input.directory

Character string of path(s) for directory where SFS files are located. May or may not end with "/". May be absolute path (*i.e.* beginning with "/") or relative path from current working directory. If output.directory is specified, then it is assumed that there is only one directory with output SFS filenames as produced by dice.sims. To specify multiple directories or different filenames, input.directory must be specified, which may be a list or vector. See also Details.

input.base

Character string of prefix for SFS filenames. Assumed all SFS files within input.directory begin with prefix and end with a numerical suffix of 1 through *n*. See also Details.

input.files

List, vector = *n*, or character string of SFS filenames within input.directory. See also Details.

Arguments from other Multi-DICE functions may be included here and are ignored if not applicable.

Details

Multi-DICE cannot currently accommodate data for more than one population per taxon.

For num.taxa and remove.afclasses, non-integer values are converted to integer values via as.integer.

For num.taxa, num.haploid.samples, folded, input.directory, and input.files, if list, then all list elements are concatenated to form a single vector, with the ordering within list elements and then between list elements preserved (*e.g.* for list of length = 2, with first list element of length = 2 and second list element of length = 1, the order from first to last is: 1) first vector element in first list element; 2) second vector element in first list element; 3) sole vector element in second list element).

For remove.afclasses, if vector, then converted to list of length = 1. Per list element, vector elements within do not need to be in any particular order. Relatedly, only unique vector elements are considered, with duplicated values ignored.

For num.taxa and folded, the order of vector elements corresponds to the order of partitions, and for remove.afclasses, the order of list elements corresponds to the order of partitions. Additionally, if length = 1 and num.partitions > 1, then the sole element is used for all partitions. Similarly, if length < num.partitions, then the first element is used for all partitions while ignoring any remaining elements, and if length > num.partitions, then the remaining elements beyond length = num.partitions are ignored; a caution is provided when the length does not equal 1 or num.partitions.

If empirical data are converted to aSFS format, then num.sims = number of multi-taxa comparative datasets (most likely 1). SFS files must be in similar format to output SFS files produced by dice.sims, such that each taxon SFS is within a separate tab-delimited file, number of rows =

`num.sims`, number of columns = number of SFS allele frequency classes (*i.e.* number of haploid samples + 1 (number of haploid samples – 1 if monomorphic bins are already removed), or if folded, then may be (number of haploid samples/2) + 1, rounded up, with + 1 omitted if monomorphic bins are already removed), and no headers or labels. Values are converted to proportions out of the total of polymorphic bins by default, thus values do not need to be user-converted.

If `num.partitions=`*n*, then rearrangement of bins across taxa within allele frequency classes based on descending order of the relative SNP proportions is not performed to construct the aSFS and taxon-specific inference of demographic parameters is possible. However, in general, more partitions results in more parameter space with respect to taxa samples that must be explored due to a decrease in order-independence and assumed exchangeability, thus multiple-fold more simulations must be conducted to achieve comparable accuracy in hyperparameter estimation as without partitioning.

For `remove.afclasses`, monomorphic bins are not removed by default if specified, thus 0 must be included if desired to be removed. To include the monomorphic frequency class without removing any bins, the integer of `num.haploid.samples` + 1 (or any value(s) > `num.haploid.samples`) must be specified.

To determine input directory, `Multi-DICE` first looks in `input.directory`, then `output.directory`. If `input.directory` is utilized, may be of any length, with order corresponding to arrangement of output aSFS matrix rows. Filenames are assumed to be consistent across `input.directory` directories. To determine filename format, `Multi-DICE` first looks in `input.base`, then `input.files`. If `input.base` is list or vector of length > 1, then only first vector element is considered. For `input.files`, length must be ≥ *n*; if > *n*, then the remaining elements beyond length = *n* are ignored and a caution is provided. If neither `input.files` nor `input.base` is specified, or if `output.directory` is utilized, then filenames assumed to follow the same format as ouput SFS files produced by `dice.sims` (*i.e.* `"dice.simulations1"`, `"dice.simulations2"`, etc.). To accommodate parallelized runs across multiple directories, `input.directory` ought to be utilized.

Value

Returned value is aSFS matrix, with each row representing an individual simulation and in the same order as input files (followed then by the user-specified order of input directories if applicable), and each column representing an aSFS bin.

Author(s)

Alexander T. Xue

References

Xue AT (2017) Multi-DICE Manual.

Xue AT, Hickerson MJ (2015) The aggregate site frequency spectrum for comparative population genomic inference. *Molecular Ecology*, **24**, 6223–6240.

Xue AT, Hickerson MJ (*submitted*) Multi-DICE: R package for comparative population genomic inference under multi-taxa hierarchical co-demographic models.

See Also

`build.dice`, `roll.dice`, `play.dice`, `dice.sims`

Examples

```
#simplest execution
dice.aSFS(num.sims=5, num.taxa=10, num.haploid.samples=10,
output.directory=‘example’)
```

dice.sumstats {Multi-DICE} R Documentation

Constructing multi-taxa single-sequence summary statistic vector from per-taxon summary statistics simulations/data

Description

`dice.sumstats` transforms per-taxon single-sequence summary statistics, either simulated by `dice.sims` or empirically produced (*i.e.* observed/collected data), into the multi-taxa single-sequence summary statistic vector, following the procedure described in Chan *et al.* (2015). Empirical sequence data may also be inputted. If `convert.sequences=T`, then `bash` commands are called upon here and `dice.sumstats` can run only within a `bash` terminal environment (*e.g.* Mac, Linux).

Usage

```
dice.sumstats(num.sims, num.taxa, num.partitions=1, num.haploid.samples,
convert.sequences=F, output.directory='.', input.directory=NULL,
input.base=NULL, input.files=NULL)
```

Arguments

`num.sims`
Positive integer. Number of simulations. Required. See also `Details`.

<u>DATA</u>

`num.taxa`
List, vector of length = `num.partitions`, or positive integer. Number of taxa per partition. Total sum across partitions equals the total number of taxa *n* in dataset. See also `Details`. Required.

`num.partitions`
Positive integer. Number of partitions for taxa in dataset. Allows differential data and model specifications across user-specified taxa groupings, including sampling size of individuals, generation times, demographic syndrome, and taxon-specific nuisance parameter prior distributions. See also `Details`.

`num.haploid.samples`
List, vector of length = `num.partitions`, or positive integer. Number of haploid samples per partition. See also `Details`. Required.

`convert.sequences`
Logical value. Activates sequence data input. See also `Details`.

<u>SIMULATION SPECIFICATIONS</u>

`output.directory`, `input.directory`
Character string of path(s) for directory where SFS files are located. May or may not end with "/". May be absolute path (*i.e.*

beginning with “/”) or relative path from current working directory. If `output.directory` is specified, then it is assumed that there is only one directory with output SFS filenames as produced by `dice.sims`. To specify multiple directories or different filenames, `input.directory` must be specified, which may be a list or vector. See also `Details`.

`input.base` Character string of prefix for SFS filenames. Assumed all SFS files within `input.directory` begin with prefix and end with a numerical suffix of `1` through *n*. See also `Details`.

`input.files` List, vector = *n*, or character string of SFS filenames within `input.directory`. See also `Details`.

Arguments from other `Multi-DICE` functions may be included here and are ignored if not applicable.

Details

`Multi-DICE` cannot currently accommodate data for more than one population per taxon.

For `num.taxa`, non-integer values are converted to integer values via `as.integer`.

For `num.taxa`, `num.haploid.samples`, `input.directory`, and `input.files`, if list, then all list elements are concatenated to form a single vector, with the ordering within list elements and then between list elements preserved (*e.g.* for list of length = 2, with first list element of length = 2 and second list element of length = 1, the order from first to last is: 1) first vector element in first list element; 2) second vector element in first list element; 3) sole vector element in second list element).

For `num.taxa`, the order of vector elements corresponds to the order of partitions. Additionally, if length = 1 and `num.partitions` > 1, then the sole element is used for all partitions. Similarly, if length < `num.partitions`, then the first element is used for all partitions while ignoring any remaining elements, and if length > `num.partitions`, then the remaining elements beyond length = `num.partitions` are ignored; a caution is provided when the length does not equal 1 or `num.partitions`.

If empirical data are converted, then `num.sims` = number of multi-taxa comparative datasets (most likely 1). If `convert.sequences=F`, files must be in similar format to output simulation files produced by `dice.sims`, such that each taxon single-sequence summary statistics are within a separate tab-delimited file, number of rows = `num.sims`, number of columns = 4, no headers or labels, and columns are in order of: number of haplotypes, haplotype diversity, nucleotide diversity, Tajima’s *D*.

If `num.partitions=`*n*, then distribution moments (*i.e.* transforming into multi-taxa single-sequence summary statistic vector) are irrelevant (mean = per-taxon value; variance, skewness, kurtosis set to 0) and taxon-specific inference of demographic parameters is possible. However, in general, more partitions results in more parameter space with respect to taxa samples that must be explored due to a decrease in order-independence and assumed exchangeability, thus multiple-fold more simulations must be conducted to achieve comparable accuracy in hyperparameter estimation as without partitioning.

If `convert.sequences=T`, assumed that: each taxon single-sequence data are within a separate file; sequence format is in 1/0 (with missing data = 9), AGTC (with missing data = N or -), or IUPAC (with missing data = N or -); each row corresponds to a haploid (diploid for IUPAC format) sample such that number of rows = `num.haploid.samples`; no headers or labels; each multi-taxa comparative dataset is within a separate directory such that length of directories = `num.sims` and `num.sims=1` if `output.directory` is utilized; sequence format is consistent across all directories. To calculate number of haplotypes per taxon, missing data in otherwise monomorphic sites are converted to the according monomorphic base and sites that are missing across all samples are removed, then unique haplotypes are discovered (while ignoring missing sites) from the sequences in order from least to most missing data (ties result in original user-specified order). Additionally, there cannot be a filename of “dice.sims.fsc2.template” within any directory.

To determine input directory, `Multi-DICE` first looks in `input.directory`, then `output.directory`. If `input.directory` is utilized, may be of any length, with order corresponding to arrangement of output matrix rows. Filenames are assumed to be consistent across `input.directory` directories. To determine filename format, `Multi-DICE` first looks in `input.base`, then `input.files`. If `input.base` is list or vector of length > 1, then only first vector element is considered. For `input.files`, length must be ≥ *n*; if > *n*, then the remaining elements beyond length = *n* are ignored and a caution is provided. If neither `input.files` nor `input.base` is specified, or if `output.directory` is utilized, then filenames assumed to follow the same format as ouput files produced by `dice.sims` (*i.e.* `“dice.simulations1”`, `“dice.simulations2”`, etc.). To accommodate parallelized runs across multiple directories, `input.directory` ought to be utilized.

Value

Returned value is matrix of multi-taxa single-sequence summary statistic vectors, with each row representing an individual simulation and in the same order as input files (followed then by the user-specified order of input directories if applicable), and each column representing an element of the multi-taxa single-sequence summary statistic vector and in the order of mean, variance, skewness, and kurtosis, per single-taxon summary statistic (following same order as before: number of haplotypes, haplotype diversity, nucleotide diversity, Tajima’s *D*), per partition.

Author(s)

Alexander T. Xue

References

Chan YL, Schanzenbach D, Hickerson MJ (2014) Detecting concerted demographic response across community assemblages using hierarchical approximate Bayesian computation. *Molecular Biology and Evolution*, **31**, 2501–2515.
Xue AT (2017) Multi-DICE Manual.
Xue AT, Hickerson MJ (*submitted*) Multi-DICE: R package for comparative population genomic inference under multi-taxa hierarchical co-demographic models.

See Also

`build.dice`, `roll.dice`, `play.dice`, `dice.sims`

Examples

```
#simplest execution
dice.sumstats(num.sims=5, num.taxa=10, num.haploid.samples=10,
output.directory=‘example’)
```

Demographic Syndrome Test – Extended

Assigning taxa *a priori* to demographic syndromes, as well as exploring prior distributions, to better inform the hierarchical co-demographic model can be done efficiently with `Multi-DICE` across multiple taxa and various demographic syndromes and sampling levels. Provided below is an example command:

```
output=dice.sims(num.sims=500000, num.taxa=1, num.partitions=15,
num.haploid.samples=rep(c(6,8,10,16,20),3),
num.ind.sites=rep(c(500,1000,2500,2000,1500),3), tau.psi.prior=0,
tau.idio.prior=c(10000:1000000),
epsilon.idio.prior=list(c(100:1000)/10000,c(100:1000)/10000,c(100:1000)/10
000,c(100:1000)/10000,c(100:1000)/10000, c(1),c(1),c(1),c(1),c(1),
10000/c(100:1000),10000/c(100:1000),10000/c(100:1000),10000/c(100:1000),10
000/c(100:1000)),
NE.idio.prior=list(c(100000:1000000),c(100000:1000000),c(100000:1000000),c
(100000:1000000),c(100000:1000000),
c(1000:1000000),c(1000:1000000),c(1000:1000000),c(1000:1000000),
c(1000:1000000),
c(1000:100000),c(1000:100000),c(1000:100000),c(1000:100000),c(1000:100000)
), output.hyper.draws=F, output.taxa.draws=T, fsc2path='fsc25211',
output.directory='.')
```

According to this command, 500,000 SFS simulations are conducted per three demographic syndromes and five sampling level combinations, for a total of 15 * 500,000 = 7,500,000 single-taxon simulations. Since `tau.psi.prior=0`, all 15 simulated taxa are completely independent in their prior draws, and since `num.partitions=`*n* (by setting `num.taxa=1` and `num.partitions=15`, there is one taxon per each of 15 partitions), these 15 simulated taxa can also be specified differently with respect to demographic syndrome and sampling level, thus resulting in 15 separate sets of single-taxon simulations that can be independently applied to each single-taxon dataset.

The three demographic syndromes include, in order, instantaneous expansion, constant size, and instantaneous contraction, as indicated by the according ε prior distributions *U*(0.01, 0.10), *U*(1.00, 1.00), and 1/*U*(0.01, 0.10), respectively. Notably, other demographic syndromes involving two size change events may be tested using the `num.changes` argument (see “Implementing Two-Event Demographic Syndromes” section for more information). It is assumed here that the different demographic syndromes of expansion, constant size, and contraction result in different N_E prior distributions *U*{100,000, 1,000,000}, *U*{1,000, 1,000,000}, and *U*{1,000, 100,000}, respectively. Importantly, prior distributions for both ε and N_E are duplicated to correspond to the 15 partitions, ordered such that the five partitions are for expansion, the second five partitions are for constant size, and the last five partitions are for contraction.

The five sampling levels include, in order, 6 haploid samples with 500 SNPs, 8 haploid samples with 1,000 SNPs, 10 haploid samples with 2,500 SNPs, 16 haploid samples with 2,000 SNPs, and 20 haploid samples with 1,500 SNPs. During sampling projection, as can be accomplished in `δaδi`, the relationship here between haploid samples and SNPs is often observed, such that lower numbers of samples result in SNP drop-off due to SNPs assigned as monomorphic and higher numbers of samples result in SNP drop-off due to missing data, therefore there is an intermediate number of samples that result in the highest number of SNPs. However, this intermediate number of samples may be below an optimal threshold while the number of SNPs may be sufficient at other numbers of samples. Hence, testing different sampling levels may be of interest to explore these different sampling projections, especially considering that the aSFS requires all taxa to be of the sampling level. Importantly, the number of haploid samples and number of independent sites are duplicated to correspond to the 15 partitions, ordered as aforementioned within each series of five partitions corresponding to demographic syndrome.

Here, the settings `output.hyper.draws=F` and `output.taxa.draws=T` are also deployed. Since the hyperparameterization built into `Multi-DICE` was used here essentially as a nuisance setting to exploit its multi-taxa simulation capabilities, the `dice.sims.hyper.draws` files are not of interest. Conversely, the `dice.sims.taxa.draws` files actually contain the values of interest that will be used for estimation, and thus must be specified given the default value is `FALSE`. Alternatively or in addition, `keep.taxa.draws=T` could also be specified, which will output these values within the `R` environment (see “Extended `R` Manual” section for more information).

The output matrices of summary statistics and parameter values can then be easily converted to reference tables for downstream ABC estimation (or other statistical inference). To convert the output simulation files from this example to reference tables using `bash` (assuming this is conducted in the same directory as the output simulation files):

```
for i in {1..5}
 do
 let j="${i}-1"
 let k="${j}*3"
 let l="${k}+1"
 let m="${i}*3"
 for n in $(seq $l $m)
  do
  cat dice.simulations${n} >>sfs.${i}
  for o in tau epsilon NE
   do
   cat dice.sims.taxa.draws.${o}|cut -d ' ' -f ${n} >>${o}.${i}
  done
 done
done
```

This will produce four reference table files per the five sampling levels. The four reference table files have the prefixes "`sfs`", "`tau`", "`epsilon`", and "`NE`" for the SFS summary statistic vectors and τ, ε, and N_E prior draws, respectively. For the prior draws, if `keep.taxa.draws=T`, the original `R` output/returned value could alternatively be exploited rather than manipulating the output simulation files. The five sampling levels, in the aforementioned order, have the suffixes "`1`", "`2`", "`3`", "`4`", and "`5`", respectively. Each reference table has 1,500,000 simulations, with the first 500,000 for the expansion demographic syndrome, the next 500,000 for the constant size demographic syndrome, and the last 500,000 for the contraction demographic syndrome. These files, along with the single-taxon datasets, can then be read into the `R` environment with `read.big.matrix` (in `bigmemory` library; recommended for reference table files) or `read.table`, and directed to `postpr`/`abc` and `cv4postpr`/`cv4abc` (in `abc` library) for ABC estimation and leave-one-out cross-validation (see "Quick Start Guide" section for more information); an additional demographic syndrome index vector would also need to be created (*e.g.* `demographic.syndrome=c(rep(1,500000), rep(2,500000), rep(3,500000))`, with `1` = expansion, `2` = constant size, and `3` = contraction). Additionally, a sample of the reference table can be directed to `PCA` for a holistic and visual assessment of power to discriminate demographic syndromes; single-taxon datasets can also be projected onto the PCA to holistically and visually determine demographic syndrome (see "Quick Start Guide" section for more information). Importantly, most of the computational effort is spent on constructing the reference table, whereas estimation per single-taxon dataset requires minimal time, thus allowing great efficiency for exploration across many taxa.

In addition to different demographic syndromes and sampling levels, various priors for any of the demographic parameters τ, ε, and N_E may also be explored to determine an optimal distribution (*e.g.* resulting in a bell-shaped posterior distribution). The example command above could be easily extended to accommodate this. Of course, subsequent commands always may be iteratively employed to explore additional demographic syndromes, sampling levels, and prior distributions, with these subsequent simulations allowed to be appended to original output simulation files with `append.sims=T`.

Importantly, this application is not restricted to *a priori* analyses for the purpose of multi-taxa co-demographic inference, as it can also be appropriate if SFS-based ABC inference of single-population demography for a single taxon (limited to two size change events) is the explicit intention, with the features offered here for efficiently testing multiple demographic syndromes and exploring sampling levels and prior distributions still of convenience.

As a reminder, given that `Multi-DICE` acts as a wrapper for `fastsimcoal2` single-sequence summary statistics as well as SFS simulation, the above also applies to single-sequence datasets.

Improving ψ Inference through Fixing ζ_s Hyperprior

Table 1. Sets of simulations testing different ζ_s priors.

Model	Idiosyncratic Taxa?	ζ_s Hyperprior

Fixed, Equal ζ_j	No	$\zeta_1 \approx \ldots \approx \zeta_\psi$
		i.e. for $\psi = 1$, $\zeta_1 = 1.0$;
		for $\psi = 2$, $\zeta_s = \{0.5, 0.5\}$;
		for $\psi = 3$, $\zeta_s = \{0.3, 0.3, 0.4\}$ (in random order)

Varying ζ_j	No	for $\psi = 1$, $\zeta_1 = 1.0$;
		for $\psi = 2$, $\zeta_s \sim \{0.3, 0.4, 0.5, 0.6, 0.7\}$;
		for $\psi = 3$, $\zeta_s \sim \{0.2, 0.3, 0.4, 0.5\}$

Varying ζ_j
(wider range)
	No	for $\psi = 1$, $\zeta_1 = 1.0$;
		for $\psi = 2$, $\zeta_s \sim \{0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8\}$;
		for $\psi = 3$, $\zeta_s \sim \{0.2, 0.3, 0.4, 0.5, 0.6\}$

Varying ζ_j
(w/ idiosyncratic)
	Yes	for $\psi = 1$, $\zeta_1 \sim \{0.7, 0.8, 0.9, 1.0\}$;
		for $\psi = 2$, $\zeta_s \sim \{0.4, 0.5, 0.6\}$;
		for $\psi = 3$, $\zeta_s \sim \{0.3, 0.4\}$

Varying ζ_j
(wider range &
w/ idiosyncratic)
	Yes	for $\psi = 1$, $\zeta_1 \sim \{0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$;
		for $\psi = 2$, $\zeta_s \sim \{0.3, 0.4, 0.5, 0.6, 0.7\}$;
		for $\psi = 3$, $\zeta_s \sim \{0.3, 0.4\}$

Sets of simulations, each with different $\zeta_{\tau,s}$ (hereby ζ_s) priors as well as allowance of idiosyncratic taxa, were conducted with `Multi-DICE` to explore how restricting ζ_s prior space affects ψ_τ (hereby ψ) inference (Table 1). For each simulation set, a reference table containing 400,000, with 100,000 per value of $\psi \sim U\{0, 3\}$, aSFS simulations of instantaneous co-expansion was produced. Per individual aSFS simulation, 10 folded SFS were generated, each containing only polymorphic bins and proportional SNP frequencies, from 20 haploid samples and 5,000 independent genealogies, and according to the prior distributions $\tau \sim U\{5{,}000, 250{,}000\}$ with $\beta = 30{,}000$ and $\beta_i = 30{,}000$ (except in the special case of $\psi = 0$, $\beta_i = 10{,}000$ due to constraint from the τ prior range and to allow more flexibility in the temporal dispersion for the total idiosyncrasy scenario), $\varepsilon \sim U(0.01, 0.10)$, and $N \sim U\{50{,}000, 250{,}000\}$. Per reference table, leave-one-out cross-validation of hRF prediction of ψ, hABC model selection of ψ, and hABC parameter summary estimation of Ω_τ and $E(\tau)$ were performed. For hRF, using the `R` package `randomForest`, a total of 1,000 decision trees, with the default maximum of 33 variables randomly sampled as candidates at each tree split and from 10 trees per each of 100 cycles of randomly sub-sampling 1,000 simulations per ψ (for a total of 4,000 simulations) with replacement after each cycle, was built to capture variation in ψ and leveraged to predict ψ for each leave-one-out POD using the `predict` function. For hABC, using the functions `cv4postpr` and `cv4abc` from the `R` package `abc`, simple rejection only was executed with an accepted tolerance level of 0.00375, resulting in 1,500 total retained simulations, and the median and mode of the posterior distributions were calculated for point estimates; for the former, there were 20 leave-one-out PODs per ψ value, leading to a total of 80 PODs, which were the same used for hRF cross-validation and thus were all removed prior to constructing the hRF, and for the latter, there were 50 total leave-one-out PODs. To clarify, though each discrete value of ψ was treated as a separate model, the numeric values of ψ were exploited to determine the median of the model posterior distribution; mean was additionally calculated. For each inferential application, Pearson's *r* correlation and root mean squared error (RMSE) were calculated from estimated values against true values.

Table 2. Sets of cross-validations between reference tables with different ζ_S priors.

Cross-validation Set	PODs	Reference Table
1	Varying ζ_j	Fixed, Equal ζ_j
2	Varying ζ_j (wider range)	Fixed, Equal ζ_j
3	Varying ζ_j (w/ idiosyncratic)	Fixed, Equal ζ_j
4	Varying ζ_j (w/ idiosyncratic)	Varying ζ_j
5	Varying ζ_j (w/ idiosyncratic)	Varying ζ_j (wider range)
6	Varying ζ_j (wider range & w/ idiosyncratic)	Fixed, Equal ζ_j
7	Varying ζ_j (wider range & w/ idiosyncratic)	Varying ζ_j
8	Varying ζ_j (wider range & w/ idiosyncratic)	Varying ζ_j (wider range)

Additionally, another form of cross-validation was achieved against each reference table with PODs extracted from other reference tables (Table 2). The intention here was to assess how the more restricted and hence simpler hyperparameterizations performed with data that were produced under more complex scenarios, as may be expected with real empirical data. The same protocols detailed above were applied here, except the functions `postpr` and `abc` from the `R` package `abc` were employed for hABC model selection and hABC parameter summary estimation, respectively, since the PODs tested against each reference table were generated elsewhere. To clarify, PODs extracted from a reference table were the same across all reference tables they were cross-validated against.

Table 3. Results from testing different ζ_s priors.

	Fixed, Equal ζ_j		Varying ζ_j		Varying ζ_j (wider range)		Varying ζ_j (w/ idiosyncratic)		Varying ζ_j (wider range & w/ idiosyncratic)	
	Pearson's r correlation	*Root Mean Squared Error*	*Pearson's r correlation*	*Root Mean Squared Error*	*Pearson's r correlation*	*Root Mean Squared Error*	*Pearson's r correlation*	*Root Mean Squared Error*	*Pearson's r correlation*	*Root Mean Squared Error*
hRF prediction of ψ										
	0.654	0.85	0.595	0.90	0.613	0.89	0.507	0.97	0.560	0.93
hABC model selection of ψ										
Mean	0.615	0.89	0.554	0.94	0.570	0.92	0.485	0.98	0.602	0.90
Median	0.483	1.06	0.493	1.03	0.571	0.99	0.415	1.06	0.568	0.94
Mode	0.539	1.08	0.416	1.20	0.443	1.20	0.346	1.28	0.474	1.19
hABC parameter summary estimation of Ω_τ										
Median	0.934	9320	0.887	12008	0.947	8071	0.920	10670	0.779	12578
Mode	0.848	13278	0.844	15268	0.808	13903	0.892	11463	0.808	14097
hABC parameter summary estimation of $E(\tau)$										
Median	0.963	12635	0.961	13073	0.951	13099	0.934	14938	0.909	14175
Mode	0.961	12985	0.937	15128	0.972	12829	0.928	13062	0.935	13487

Table 4. Results from cross-validations between reference tables with different ζ_s priors.

	1		2		3		4		5		6		7		8	
	Pearson's r correlation	*Root Mean Squared Error*	*Pearson's r correlation*	*Root Mean Squared Error*	*Pearson's r correlation*	*Root Mean Squared Error*	*Pearson's r correlation*	*Root Mean Squared Error*	*Pearson's r correlation*	*Root Mean Squared Error*	*Pearson's r correlation*	*Root Mean Squared Error*	*Pearson's r correlation*	*Root Mean Squared Error*	*Pearson's r correlation*	*Root Mean Squared Error*
hRF prediction of ψ																
	0.381	1.29	0.177	1.49	0.396	1.36	0.708	0.79	0.696	0.81	0.080	1.62	0.414	1.04	0.400	1.05
hABC model selection of ψ																
Mean	0.631	0.87	0.478	0.98	0.677	0.83	0.677	0.83	0.653	0.85	0.477	0.99	0.451	1.00	0.413	1.03
Median	0.540	1.00	0.418	1.11	0.650	0.91	0.652	0.91	0.631	0.95	0.420	1.14	0.482	1.10	0.449	1.16
Mode	0.536	1.07	0.303	1.33	0.584	1.08	0.541	1.16	0.508	1.20	0.546	1.15	0.417	1.31	0.363	1.40
hABC parameter summary estimation of Ω_τ																
Median	0.825	12341	0.788	15694	0.908	8633	0.774	15741	0.920	10221	0.654	10221	0.880	13452	0.876	13350
Mode	0.782	14410	0.774	16908	0.872	11640	0.755	17374	0.913	11152	0.613	22406	0.839	15542	0.859	14731
hABC parameter summary estimation of $E(\tau)$																
Median	0.930	15097	0.931	16223	0.895	17148	0.909	14871	0.929	13442						
Mode	0.934	15173	0.927	16887	0.897	16986	0.908	14770	0.928	13821						

hABC parameter summary estimation of E(τ)							
Median	12380	0.928	14301	0.932	13598	12380	
Mode	12594	0.921	14768	0.923	14470	12594	

Fixed, equal ζ_j values across pulses displayed the overall best performance in the leave-one-out cross-validations (Table 3), as well as often demonstrating to be the best, or at least near the best, reference table despite how the PODs were generated (Tables 3-4). This suggests that simplifying the model such that ζ_j values are fixed to be equal is a robust hyperparameterization scheme, though expectedly deteriorates in effectiveness when pulse membership among taxa in the dataset is greatly disproportional, which is exacerbated with highly idiosyncratic taxa. This perhaps makes iteratively pruning a dataset of outlier taxa a worthwhile consideration. Notably, when the model generating PODs was mismatched with the model underlying the reference table, hRF routinely struggled, implying that hRF may be very sensitive to model misspecification/violation. Thus, although hRF may at times have more inferential power than hABC, it is best used as a complementary tool rather than relying on it solely.

Data Partitioning to Accommodate Mixture of Demographic Syndromes

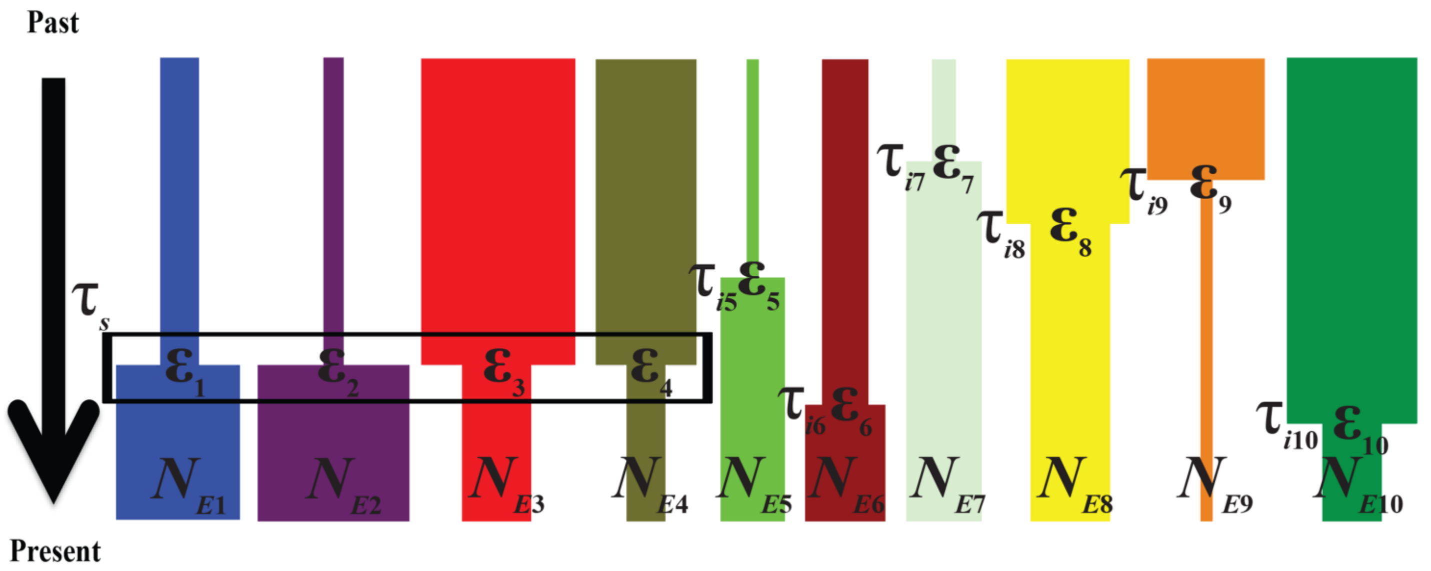
Past
τ_s
ε_1
ε_2
ε_3
ε_4
τ_{i5} ε_5
τ_{i6} ε_6
τ_{i7} ε_7
τ_{i8} ε_8
τ_{i9} ε_9
τ_{i10} ε_{10}
N_{E1}
N_{E2}
N_{E3}
N_{E4}
N_{E5}
N_{E6}
N_{E7}
N_{E8}
N_{E9}
N_{E10}
Present

Figure 1. Example mixed co-demographic model. This mixes instantaneous co-expanders with instantaneous co-contractors at *a priori* known proportions within a single hierarchical model. The model is such that four of the ten taxa, specifically two of the five co-expanders and two of the five co-contractors, are assigned to one synchronous pulse (ψ = 1 and ζ_T = 0.4) occurring at $\tau_{s,1}$. The remaining six taxa are then behaving idiosyncratically in time from all other taxa ($\{\tau_{i,5}, \ldots, \tau_{i,10}\}$) and all taxa are allowed nuisance demographic parameter draws independent from each other (ε and N). To accomplish inference using this model, a partitioning scheme was applied on the aSFS.

To combine instantaneous co-expanders and co-contractors within a single hierarchical model and thus infer upon co-demography jointly across both syndromes (Figure 1), a partitioning scheme may be applied on the aSFS based on the *a priori* known proportions of these demographic syndromes. This partitioning involves simulating a pre-determined number of taxa as co-expanders in one partition and the remaining as co-contractors in another partition, with the according SFS reordered within each partition independently according to the aSFS construction procedure, such that separate aSFS are built for co-expanders versus co-contractors. These two partitioned aSFS are then concatenated to form a single whole conjoined aSFS. This partitioning procedure can be performed on single-sequence data as well, such that the distribution moments are calculated among taxa per partition independently with subsequent concatenation across partitions of the separately produced multi-taxa single-sequence summary statistics, resulting in a total of multi-taxa single-sequence summary statistics = 16 * number of partitions. Importantly, independent data specification and parameterization among partitions is permitted, thus allowing other forms of taxa categorization for more flexible modeling, including accommodating datasets with heterogeneity of number of individuals, generation times, sampling times, and taxon-specific prior distributions. Specifically, by allowing differential parameterization, partitioning can accommodate within a single dataset and analysis multiple guilds that differ in biotic traits such as habitat preference, predator-prey relationships, or taxonomic group. Furthermore, if the assumption of taxa exchangeability is significantly violated, this option can be extended to prevent aSFS rearrangement (or calculation of distribution moments on single-sequence data) by setting the number of partitions equal to the number of taxa while still allowing for hierarchical model-based inference of co-demography. Notably, partitions can be used in preceding `Multi-DICE` functions (*i.e.* `build.dice`, `roll.dice`, `play.dice`, `dice.sims`) to differentially specify taxa and then removed in `dice.aSFS`/`dice.sumstats`, allowing data and/or parameterization heterogeneity within the same partition when constructing the multi-taxa summary statistic vector. However, this may result in assumption violations when re-ordering/pooling data among taxa assumed to be exchangeable, which may or may not be significant.

To test if there is signal within a partitioned aSFS, three reference tables were constructed with `Multi-DICE`, each containing 500,000 aSFS simulations and with different proportions of co-expanders to co-contractors from n = 10 total taxa: scenario 1) 5:5; scenario 2) 7:3; and scenario 3) 3:7. Each individual aSFS simulation followed the same specifications outlined in the previous simulation study, except ψ was fixed to $\psi = \{0,1\}$ with idiosyncratic taxa allowed and $\zeta_T \sim U\{1, 10\}/10$ being estimated following Xue and Hickerson (2015), $\beta = 10{,}000$, $\beta_i = 0$ (*i.e.* no τ_i idiosyncratic buffer), and $\varepsilon \sim 1/U(0.01, 0.10)$ and $N \sim U\{5{,}000, 50{,}000\}$ for co-contractors. Leave-one-out cross-validation per reference table also followed the same specifications outlined in the previous simulation study, except: each cycle of 10 hRF decision trees involved 5,000 sub-sampled simulations to capture variation in ζ_T and leveraged to predict ζ_T; the hABC accepted tolerance level was 0.003; hABC model selection of ψ was changed to ζ_T; there was a total of 200 PODs for hABC model selection given 20 for each discrete value of ζ_T; hABC hyperparameter estimation of ζ_T.1 (partition for co-expanders) and ζ_T.2 (partition for co-contractors) with the function `cv4abc` from the `R` package `abc` was added; hABC parameter summary estimation of E(τ) was changed to $\tau_{s,1}$; means were calculated of the posterior distributions generated from hABC hyperparameter and parameter summary estimation. To clarify, the ratio of ζ_T.1: ζ_T.2 does not necessarily reproduce the *a priori* assigned ratio of co-expanders to co-contractors as any combination of co-expanders and co-contractors is allowed per ζ_T value.

The cross-validation experiments establish that there is sufficient signal for inference of mixed demographic events when the aSFS data is *a priori* partitioned between co-expanders and co-contractors (Table 5). Specifically, co-expansion seemingly is easier to detect than co-contraction as evidenced from the overall worse performance for hABC estimation of ζ_T.2 compared to ζ_T.1 as well as generally decreased accuracy when there were 3 co-expanders: 7 co-contractors in comparison to the other two scenarios. Importantly, estimation capability for $\tau_{s,1}$ was promising and consistent in all three reference tables. In sum, partitioning is validated here to be a fruitful option for accommodating different demographic syndromes within a dataset without severe loss in accuracy.

However, over-partitioning is cautioned against since as the number of partitions increases, orders of magnitude more simulations are required to achieve comparable accuracy. In fact, assuming every taxon has a different set of parameter draws and in turn distinguishable SFS per each simulation, considering the probability that the random taxa assignment to partitions matches that of the empirical dataset, a factor of up to $\binom{n}{n_1 \dots n_p}$ more simulations may be required, converging to n! multiples more simulations for a completely non-reordered, concatenated set of SFS, with $\{n_1, \dots, n_p\}$ referring to the number of taxa within each of *p* total number of partitions. Hence, it is advisable to partition prudently. For example, specifically in the case of differential sampling of individuals among taxa, it is likely preferable to project all taxon-specific SFS to the same sampling, which would reduce bias from variation in sample sizes as well, rather than partition the data. An alternative, or supplement, to partitioning a dataset within a single analysis is to partition a dataset across multiple, independent analyses.

Table 5. Results from partitioning co-expanders from co-contractors.

Co-expanders: Co-contractors	5 : 5		7 : 3		3 : 7	
	Pearson's r correlation	*Root Mean Squared Error*	*Pearson's r correlation*	*Root Mean Squared Error*	*Pearson's r correlation*	*Root Mean Squared Error*
hRF prediction of ζ_T						
	0.724	0.200	0.681	0.211	0.548	0.241
hABC model selection of ζ_T						
Mean	0.691	0.209	0.680	0.211	0.546	0.242
Median	0.684	0.211	0.704	0.206	0.542	0.247
Mode	0.674	0.271	0.644	0.279	0.478	0.342
hABC hyperparameter estimation of ζ_T.1 (partition for co-expanders)						
Mean	0.486	0.142	0.648	0.168	0.499	0.090
Median	0.630	0.120	0.794	0.126	0.730	0.076
Mode	0.612	0.153	0.699	0.173	0.267	0.113
hABC hyperparameter estimation of ζ_T.2 (partition for co-contractors)						
Mean	0.434	0.150	0.406	0.093	0.549	0.163
Median	0.511	0.130	0.600	0.080	0.706	0.144
Mode	0.585	0.159	0.650	0.087	0.401	0.245
hABC parameter summary estimation of Ω_τ						
Mean	0.823	10928	0.746	11209	0.799	13500
Median	0.864	12147	0.848	13682	0.755	16527
Mode	0.851	11581	0.786	15322	0.725	19117
hABC parameter summary estimation of $\tau_{s,1}$						

Mean	0.667	47289	0.763	45669	0.753	47924
Median	0.879	38029	0.700	50156	0.758	48635
Mode	0.697	56392	0.467	71105	0.850	41140

Implementing Two-Event Demographic Syndromes

Table 6. Hierarchical co-demographic models.

Model #	Model Description
1	Instantaneous co-expansion
2	Instantaneous co-contraction
3	Instantaneous co-expansion with nuisance prior (more ancient) bottleneck
4	Instantaneous co-expansion with nuisance subsequent (more recent) bottleneck
5	Instantaneous co-contraction with nuisance prior (more ancient) expansion
6	Instantaneous co-contraction with nuisance subsequent (more recent) expansion

Another simulation experiment was performed to explore a wider set of potentially useful hierarchical co-demographic models, including co-bottleneck-expansion and co-expansion with subsequent contraction (Table 6). To elaborate upon these two-event size change models, either instantaneous co-expansion or co-contraction, which is hyperparameterized via ζ_T, is coupled with either a prior or subsequent opposing nuisance size change event of smaller magnitude (*e.g.* co-expansion with subsequent minor nuisance co-contraction, or a co-bottleneck with previous minor nuisance co-expansion).

For this investigation, `Multi-DICE` was deployed to construct a reference table for each model. These reference tables followed the same specifications outlined in the previous simulation study, except $\tau1$ was hyperparameterized and had the same prior distribution as described previously for τ, and the additional priors were utilized: nuisance subsequent event time $\tau2 \sim \{1,000, 100,000\}$ (in these cases, $\tau1 > \tau2$); "change" parameter, representing difference in time between $\tau1$ and nuisance prior event $\tau2$, $\delta \sim U\{10,000, 100,000\}$ (in these cases, $\tau1 < \tau2$); nuisance expansion magnitude $\varepsilon2 \sim U(0.1, 0.5)$; nuisance bottleneck magnitude $\varepsilon2 \sim 1/U(0.1, 0.5)$. Leave-one-out cross-validation per reference table also followed the same specifications outlined in the previous simulation study, except ζ_T.1 and ζ_T.2 were not estimated and only median and mode were calculated on posterior distributions generated from hABC parameter summary estimation.

Similar to the study described in the "Improving ψ Inference through Fixing ζ_S Hyperprior" section, another form of cross-validation was performed by applying PODs from the two-event models to the reference table for either the single event co-expansion or co-contraction model, whichever was applicable, to assess the effect of model misspecification. The procedure here is identical to above except the functions `postpr` and `abc` from the `R` package `abc` were employed for hABC model selection and hABC parameter summary estimation, respectively, since the PODs tested against each reference table were generated elsewhere.

The cross-validation experiments suggest that co-contraction is much more difficult to infer than co-contraction (Table 7), and that co-expansions largely suppress prior co-contraction signal whereas subsequent co-contractions have a more minor albeit still considerable confounding effect on prior co-expansion signal (Tables 7-8). Importantly, it seems that model misspecification usually has a fairly profound negative effect on estimation, highlighting the importance of properly assigning demographic syndrome *a priori* (Table 8).

Table 7. Results from exploring hierarchical co-demographic models.

Model #:	1		2		3		4		5		6	
	Pearson's r correlation	*Root Mean Squared Error*	*Pearson's r correlation*	*Root Mean Squared Error*	*Pearson's r correlation*	*Root Mean Squared Error*	*Pearson's r correlation*	*Root Mean Squared Error*	*Pearson's r correlation*	*Root Mean Squared Error*	*Pearson's r correlation*	*Root Mean Squared Error*
hRF prediction of ζ_T												
	0.761	0.187	0.534	0.243	0.724	0.199	0.634	0.223	0.567	0.238	0.493	0.251
hABC model selection of ζ_T												
Mean	0.744	0.193	0.518	0.248	0.718	0.201	0.594	0.233	0.573	0.237	0.477	0.254
Median	0.736	0.200	0.493	0.259	0.723	0.203	0.593	0.235	0.562	0.240	0.455	0.259
Mode	0.683	0.259	0.466	0.317	0.667	0.254	0.540	0.310	0.518	0.311	0.403	0.375
hABC parameter summary estimation of Ω_τ												
Median	0.909	9843	0.840	13816	0.898	12041	0.848	19226	0.674	17706	0.355	20876
Mode	0.765	19916	0.728	15722	0.842	10546	0.594	20426	0.747	15286	0.578	25896
hABC parameter summary estimation of $\tau_{S,1}$												
Median	0.733	45869	0.724	47206	0.886	33260	0.838	41850	0.708	48741	0.585	55168
Mode	0.801	48725	0.633	61762	0.786	40207	0.786	48411	0.546	68036	0.429	76011

Table 8. Results from cross-validating misspecified hierarchical co-demographic models.

PODs Model #:	3		4		5		6	
Reference Table Model #:	1		1		2		2	
	Pearson's r correlation	*Root Mean Squared Error*	*Pearson's r correlation*	*Root Mean Squared Error*	*Pearson's r correlation*	*Root Mean Squared Error*	*Pearson's r correlation*	*Root Mean Squared Error*
hRF prediction of ζ_T								
	0.731	0.197	0.411	0.271	0.523	0.267	0.143	0.285
hABC model selection of ζ_T								
Mean	0.729	0.198	0.407	0.263	0.521	0.260	0.150	0.359
Median	0.724	0.201	0.400	0.268	0.504	0.274	0.107	0.401
Mode	0.665	0.257	0.348	0.337	0.401	0.373	-0.056	0.516
hABC parameter summary estimation of Ω_τ								
Median	0.833	12173	0.765	22705	0.570	20844	0.696	32901
Mode	0.841	13051	0.765	22626	0.495	25307	0.465	37224
hABC parameter summary estimation of $\tau_{S,1}$								
Median	0.750	49803	0.681	54780	0.521	60855	0.536	106667
Mode	0.707	55787	0.681	56190	0.509	68591	0.319	127257

References

Chan YL, Schanzenbach D, Hickerson MJ (2014) Detecting concerted demographic response across community assemblages using hierarchical approximate Bayesian computation. *Molecular Biology and Evolution*, **31**, 2501–2515.

Csilléry K, François O, Blum MGB (2012) abc: an R package for approximate Bayesian computation (ABC). *Methods in Ecology and Evolution*, **3**, 475–479.

Excoffier L, Dupanloup I, Huerta-Sánchez E, Sousa VC, Foll M (2013) Robust demographic inference from genomic and SNP data. *PLoS genetics*, **9**, e1003905.

Excoffier L (2014) fastsimcoal2 manual.

Gutenkunst RN, Hernandez RD, Williamson SH, Bustamante CD (2009) Inferring the joint demographic history of multiple populations from multidimensional SNP frequency data. *PLoS genetics*, **5**, e1000695.

Hickerson MJ, Stahl E, Takebayashi N (2007) msBayes: Pipeline for testing comparative phylogeographic histories using hierarchical approximate Bayesian computation. *BMC bioinformatics*, **8**, 268.

Huang W, Takebayashi N, Qi Y, Hickerson MJ (2011) MTML-msBayes: approximate Bayesian comparative phylogeographic inference from multiple taxa and multiple loci with rate heterogeneity. *BMC bioinformatics*, **12**, 1.

Liaw A, Wiener M (2002) Classification and Regression by randomForest. *R news*, **2**, 18–22.

Mevik B-H, Wehrens R (2007) The pls Package: Principal Component and Partial Least Squares Regression in R. *Journal Of Statistical Software*, **18**, 1–24.

Xue AT (2017) Multi-DICE Manual.

Xue AT, Hickerson MJ (2015) The aggregate site frequency spectrum for comparative population genomic inference. *Molecular Ecology*, **24**, 6223–6240.

Xue AT, Hickerson MJ (*submitted*) Multi-DICE: R package for comparative population genomic inference under multi-taxa hierarchical co-demographic models.