Simulations for allopolyploid networks: AlloppDT

Graham Jones

2012-09-01, updated 2012-11-06

1 Introduction

This describes version one (started 2012-08-04) of the simulation program 'AlloppDT' used to make testing data for the allopolyploid models (AlloppNET) in BEAST and analyze the results. AlloppDT is for the network model with an arbitrary number (at least 2) of diploids, tetraploids (at least 1) and an arbitrary number of hybridizations (between 1 and the number of tetraploids). It is an extension of AlloppSim used for the one-hybridization case.

AlloppDT inputs a config file and outputs a bunch of BEAST XML files. It then runs BEAST on these, and then TreeAnnotator on the results. For each set of parameters (number of genes G, number of individuals N, and mutation rate T) it does:

1. Make the MUL-tree
2. Make the gene tree topologies and node times
3. Call Seq-Gen to produce the sequences
4. Constructs the BEAST XML file containing the sequences and 'standard' options for an allopolyploid model in BEAST
5. Runs BEAST on the XML file
6. Runs TreeAnnotator on BEAST output (samples of MUL-trees) and optionally samples of gene trees

There is also some code for analysis.

There is some support for generating BEAST XML files for real data. See `2012-02-14-manual-real-data.pdf` for example of use.

AlloppDT is written in R by me. Seq-Gen is written in C by Andrew Rambaut, slightly modified by me. The Seq-Gen source here should only be used for reproducing my results; otherwise get the latest version from Andrew Rambaut.

1.1 Changes from AlloppSim (v5)

1. The BEAST XML has changed significantly from the one-hybridization case. There are now three population parameters, for tip, rootward, and hybridization populations. The priors for these have been moved from a generic `MixedLikelihood` element into the

`alloppNetworkPriorModel` element, so that the prior can be sampled from when the number of hybridizations change. There are new operators `hybPopSizesScaleOperator`, `changeNumHybridizations`.

2. The configuration supplies the MUL-tree rather than a diploid tree, tetraploid trees, and legs. This is more inconvenient when making the config files, but simpler to process.
3. The MUL-tree model is omitted from this code.
4. New name AlloppDT

See 2012-05-05-simulations.pdf for earlier changes.

2 Installation and Usage

2.1 Installation

(TODO)

2.2 Usage

I run AlloppDT with a command like this on my desktop Windows computer:

```
"C:\Program Files\R\R-2.14.1\bin\R.exe" --slave --vanilla --args VOSTROWIN7
C:\Users\Graham\AAA\Programming\GoteborgSim\2012-09-01\D <
C:\Users\Graham\AAA\Programming\Goteborg\AlloppRcode-nhybs\AlloppDT_main.r
```

First argument (eg `VOSTROWIN7` or `NETBOOKWIN7`) tells `AlloppDT_main.r` where it is, enabling it to find the rest of the R code, `seqgen`, `BEAST`, `java`, and what `classpath` should be.

Second argument (`C:\Users\Graham\AAA\Programming\GoteborgSim\2012-09-01/D`) specifies the directory containing the config file and is where all the results go. Within this directory, all subdirectories and files have 'programmed' names, see 2011-10-11-simulations-plan.pdf. Note that the directory name (a single letter) encodes the scenario.

The code near the start of `AlloppDT_main.r` will need editing to run the simulations on other computers.

3 Configuration files

3.1 Example configuration file: two hybridizations

```
PROGRAMPARAMETERS
beastseed 652989819
beastchainlength 1500000
beastscreen.logevery 10000
beastparams.logevery 1000
beastgtrees.logevery 1000
beastmultree.logevery 1000
beastdbugtune.logevery 10000
treeannburnin 501
SCENARIO
nofGNRT 2
3 1 20 4e-8 8e-8
1 2 10 8e-8
genelength 500
noftetratrees 2
nofdips 2
noftets 2
aA                          hgt 0    hybhgt *    tipp 100000 rootp 100000 hybp *
bA                          hgt 0    hybhgt *    tipp 100000 rootp 100000 hybp *
zA                          hgt 0    hybhgt 0.01 tipp 100000 rootp 100000 hybp 100000
zB                          hgt 0    hybhgt 0.01 tipp 100000 rootp 100000 hybp 100000
yA                          hgt 0    hybhgt 0.02 tipp 100000 rootp 100000 hybp 100000
yB                          hgt 0    hybhgt 0.02 tipp 100000 rootp 100000 hybp 100000
(zA,yA)                     hgt 0.03 hybhgt *    tipp *      rootp 100000 hybp *
(zB,yB)                     hgt 0.03 hybhgt *    tipp *      rootp 100000 hybp *
(aA,(zA,yA))                hgt 0.05 hybhgt *    tipp *      rootp 100000 hybp *
(bA,(zB,yB))                hgt 0.06 hybhgt *    tipp *      rootp 100000 hybp *
((aA,(zA,yA)),(bA,(zB,yB))) hgt 0.07 hybhgt *    tipp *      rootp *      hybp *
```

This is for scenario shown in Figure 1. The tree on the left is the one that is described in the config file. It has two hybridizations, but the topology and coincidence of node heights mean that it is equivalent to the middle tree which only has one hybridization. These cannot be distinguished using genetic data in the AlloppNET model. Further assumptions (eg about populations) might make it possible. The resulting MUL-tree from either is on the right.

The config file is read line by line, and each line consists of space-separated tokens (so don't introduce blank lines or extra spaces). An asterisk * is used to indicate nonexistent values.

```
PROGRAMPARAMETERS
beastseed 652989819
beastchainlength 500000
beastscreen.logevery 10000
beastparams.logevery 1000
beastgtrees.logevery 1000
beastmultree.logevery 1000
```

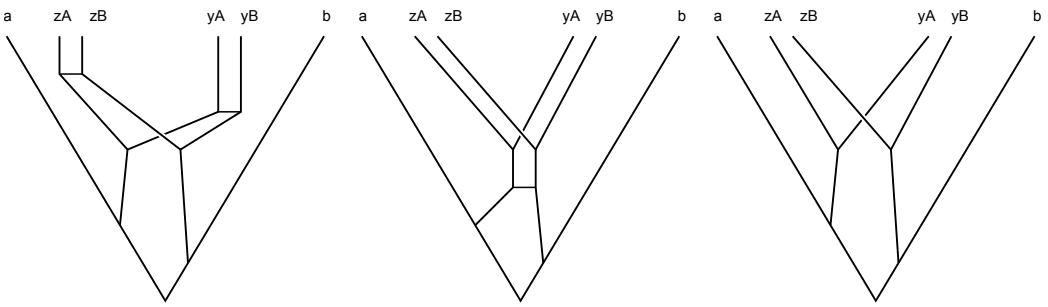

Figure 1: Scenario in config file example

```
beastdbugtune.logevery 10000
treeannburnin 501
SCENARIO
```

This section specifies some program parameters for BEAST and TreeAnnotator which are independent of the scenario and the model. It begins with `PROGRAMPARAMETERS`, then `SCENARIO` starts the next section. If `beastseed` is 0, the seed will be the same as the seed used for the pair (first gene, particular replicate number). If `beastseed` is a positive integer, it is used for all replicates.

```
nofGNRT 2
```

Six sets of G, N, R, T values. (Number of genes G, number of individuals N, number of replicates R, and set of mutation rates T.) For example

```
3 1 20 4e-8 8e-8
```

means 3 genes, one individual per species, 20 replicates, and two mutation rates 4e-8 8e-8 in units of expected substitutions per site per generation. There can be any number of mutation rates, each results in a separate set of (G,N,T) values, hence different simulations. In this example, there will be 6*2*20 BEAST runs.

```
genelength 500
```

Each sequence (for all genes) has 500 nucleotides. (Value is passed to Seq-Gen.)

```
noftetratrees 2
nofdips 2
noftets 2
```

The network contains two allotetraploid subtree (two hybridization events), two diploids, two tetraploids.

Then there is a list of nodes in the MUL-tree. A tip node has a clade specified by a lower-case letter, follwed by an upper case one, and my convention is for diploids to be `a,b,c,...` and tetraploids to be `z,y,x,...`. The seond letter indentifies a sequence. An internal node uses Newick style like `(left child, right child)` for its clade.

Each node has five fields apart from the clade; some are not used (the `*`s). The fields are node height (`hgt`), hybridization height (`hybhgt`), tipward population(`tipp`) rootward population

(`rootp`), population just after hybridization (`hybp`). The node heights are in substitution units. The populations are diploid gene counts.

4 Notes on the BEAST XML

Strict clock for branch rates. HKY model for substitutions. No site rate heterogeneity.

Different mutation rates for different genes are assumed, relative to the first gene which is fixed at 1.0.

Mixed distribution (two gammas) like *BEAST for populaton prior is assumed.

Some comments are put into the XML.

5 Code structure and algorithms

The main R script is `AlloppSDT_main.r` which calls the routines below. The main control flow is

```
Read the config file
for (each (G,N,T)) {
  Make a MUL-tree
  for (each replicate) {
    Make BEAST XML
    }
  }
for (each (G,N,T)) {
  for (each replicate) {
    Run BEAST
    }
  }
for (each (G,N,T)) {
  for (each replicate) {
    Run TreeAnnotator
    }
  }
```

5.1 Reading the config file

`AlloppDT_1readconfig2.r` reads the input, checks it (a bit) and stores the config.

5.2 Making a MUL-tree

`AlloppDT_2makemultree.r` converts the MUL-tree as read from the file (a list of nodes), into a tree structure (with lft, rgt, anc fields). This requires parse the Newick style clade strings.

It also converts this into a Newick representation so plot.phylo() from ape can display/output the MUL tree for checking and documenting.

5.3 Making a set of gene trees

`AlloppDT_3makegtrees.r` Simulates the coalescent process withing the MUL-tree. For each gene it produces a Newick string. It first makes a gene tree as a 'nearly-Newick' string with node heights rather than branch lengths. This is converted to a node list which looks like this. This is converted into a Newick string which looks like this. This can be passed to Seq-Gen.

```
((01bA:0.0302458,01zB:0.0302458):0.0137192,(01aA:0.02711082,01zA:0.02711082):0.01685418);
```

5.4 Making the sequences

`AlloppDT_4makeseqfiles.r`

Seq-Gen is called for each gene tree with a command like

```
seqgen.exe -mHKY -t3.0 -f0.3,0.2,0.2,0.3 -l500
-n1 -z6565174 An2GTg1n1t40r1-1.txt -y An2SQg1n1t40r1-1.txt
```

where the `-l` parameter (500) is taken from input config. In the file names `A` stands for scenario (second last letter of `scenAn`), `n` stands for AlloppNET (last letter of `scenAn`),`GT` stands for gene tree, `SQ` for sequence. The first `2` just helps keeps files in order.

5.5 Making the XML

`AlloppDT_5beastxml_toplevel.r` make the BEAST XML file, calling many low-level routines in `AlloppDT_6beastxml_bits.r` to do so.

V2 to V3 changes. The function `make.beastxml()` renamed as `make.beastxml.from.sim.network()` and refactored to call `make.beastxml.start()`, `make.beastxml.after.patterns()`. A flag `mulTreeVersion` is passed down to choose between AlloppNET and AlloppMUL models.

5.6 Running BEAST, TreeAnnotator

Straightforward once classpath is figured out. Running BEAST takes at least 90% of the time.

5.7 Analysis

`Analysis_lookat_gtrees.r`, `Analysis_lookatparams.r`, `Analysis_netdisplay.r` `Analysis_partition-multrees.r` `Analysis_re_annotate_tree.r` are for analysing output from BEAST and TreeAnnotator. They are run separately, after the BEAST log files have been produced and the concensus mul-trees have been found by TreeAnnotator.