Simulations for allopolyploid networks: AlloppSim (v2)

Graham Jones

2011-10-21

1 Introduction

This describes version two (2011-10-21) of the simulation program 'AlloppSim' used to make testing data for allopolyploid model in BEAST. AlloppSim (v2) 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 the 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

AlloppSim is written in R by me. Seq-Gen is written in C by Andrew Rambaut, slightly modified by me.

1.1 Changes since v1

The main change is that the configuration file describes a complete set of simulations (for varying numbers of genes and individuals, and various mutation rates) instead of just one. However it is more restricted than version one in that numbers of individuals must be the same for all species - so first version may remain useful.

Another change is that random seeds are not specified in the configuration file. See 2011-10-11-simulations-plan.pdf for new method.

2 Installation

The ZIP file contains two directories `simulation_di_tets_multigene` and `SeqGenSource`. Seq-Gen needs to be compiled and copied to `simulation_di_tets_multigene`.

Compile seqgen under windows:

```
gcc -Wall -pedantic -Wextra aamodels.c eigen.c evolve.c gamma.c global.c
model.c nucmodels.c progress.c seq-gen.c treefile.c twister.c -o seqgen.exe
```

Compile seqgen under Linux:

```
gcc -Wall -pedantic -Wextra -lm aamodels.c eigen.c evolve.c gamma.c global.c
model.c nucmodels.c progress.c seq-gen.c treefile.c twister.c -o seqgen
```

`gcc` might need replacing with a path to the executable for the Gnu C compiler. Under Linux, note the extra `-lm` and omission of `.exe`. Put the executable in a directory with the R code:

```
AlloppSim_1readconfig.r
...
AlloppSim_main.r
seqgen.exe
```

3 Usage

Run AlloppSim with a command like this on my desktop Windows computer:

```
"C:\Program Files\R\R-2.11.1\bin\R.exe" --slave --vanilla --args
DELLVISTA scenA < AlloppSim_main.r
```

and this on Albiorix cluster:

```
R --slave --vanilla --args CLUSTER /home/gjones/scenA < AlloppSim_main.r
```

First argument (`DELLVISTA` or `NETBOOKUBUNTU` or `CLUSTER`) tells `AlloppSim_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 (`scenA`, `/home/gjones/scenA`) specifies the directory containing the config file and is where all the results go. Within this direstory, all subdirectories and files have 'programmed' names, see 2011-10-11-simulations-plan.pdf.

3.1 Example config file

Here is a simple example of the input. (See notes on first version 2011-09-08-simulations.pdf for a slightly more complicated network.)

```
nofGNRT 6
1 1 20 4e-8 8e-8
1 3 20 4e-8 8e-8
3 1 20 4e-8 8e-8
3 3 20 4e-8 8e-8
10 1 20 4e-8 8e-8
10 3 20 4e-8 8e-8
noftetratrees 1
genelength 500
ditree ntips 2
a height 0 tippop 100000 prevpop 100000
b height 0 tippop 100000 prevpop 100000
(a,b) height 0.04 tippop * prevpop *
tetratree ntips 1
z height 0 tippop 100000 prevpop *
hyb 0.01 100000 join * * foot1 a .02 100000 foot2 b .03 100000
```

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.

`nofGNRT 6`

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.

`noftetratrees 1`

The network contains one allotetraploid subtree (one hybridization event).

`genelength 500`

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

Then there is a list of homoploid trees, one for the diploids and one or more for the allotetraploids. These are specified as a list of nodes, with an extra line for the 'legs' of each allotetraploid tree. Each node specifies its clade, and the nodes appear in a tips-to-root order. (Both children of a node must appear before the node).

A tip node has a clade specified by one lower-case letter, and my convention is for diploids to be a,b,c,... and allotetraploids to be z,y,x,.... An internal node uses Newick style like (left child, right child) for its clade.

All nodes have a height, which must be 0 for tip nodes.

A tip node then specifies a tip population. All nodes except roots of trees specify a 'previous' population which specifies the population at the rootward end of the branch leading to this node *in the MUL-tree.*

```
ditree ntips 2
a height 0 tippop 100000 prevpop 100000
b height 0 tippop 100000 prevpop 100000
(a,b) height 0.04 tippop * prevpop *
```

The diploid tree with two tips.

```
tetratree ntips 1
z height 0 tippop 100000 prevpop *
hyb 0.01 100000 join * * foot1 a .02 100000 foot2 b .03 100000
```

An allotetraploid tree with one tip. Note in this case the tip is also the root. One leg meets clade `a` and the other meets clade `b`.

The last line specifies the evolutionary history of the allotetraploid before the root. Either the two legs meet the diploid tree separately in which case join is unused (as above), or the two legs join before meeting the diploid tree, in which case foot2 is unused (as below).

```
hyb 0.01 1000 join 0.02 210000 foot1 a .03 220000 foot2 * * *
```

The parameters are specified as follows.

`hyb` [height of the hybridization event] [population just after hybridization]

`join` [height when legs join] [population of both legs between join and hybridization]

`foot1` [diploid clade where first leg meets] [height of foot] [*previous* population of the foot node *in the MUL-tree*]

`foot2` is the same as `foot1`.

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 `AlloppSim_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

`AlloppSim_1readconfig2.r` reads the input, checks it (a bit) and stores the config.

5.2   Making a MUL-tree

`AlloppSim_2makemultree.r` converts the network to a MUL-tree as a list of nodes, and fills the nodes with times, populations, etc. The result looks like this.

```
idx   clade            anc  l  r    hgt  hybhgt nlin  tippop  hybpop  prevpop
 1 aA                    6 -1 -1      0     -1    1    1e+05     -1    1e+05
 2 bA                    7 -1 -1      0     -1    1    1e+05     -1    1e+05
 3 aA,zA,bA,zB          -1  6  7   0.04     -1   -1      -1     -1      -1
 4 zA                    6 -1 -1      0   0.01    1    1e+05  1e+05    1e+05
 5 zB                    7 -1 -1      0   0.01    1    1e+05  1e+05    1e+05
 6 aA,zA                 3  1  4   0.02     -1   -1      -1     -1    1e+05
 7 bA,zB                 3  2  5   0.03     -1   -1      -1     -1    1e+05
```

5.3   Making a set of gene trees

`AlloppSim_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

`AlloppSim_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 A2GTg1n1t40r1-1.txt -y A2SQg1n1t40r1`

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

5.5 Making the XML

`AlloppSim_5beastxml_toplevel.r` make the BEAST XML file, calling many low-level routines in `AlloppSim_6beastxml_bits.r` to do so.

5.6 Running BEAST, TreeAnnotator

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