Simulations for allopolyploid networks

Graham Jones

2011-09-08

1 Introduction

This describes the simulation program 'AlloppSim' used to make testing data for allopolyploid model in BEAST.

AlloppSim inputs a config file and outputs a BEAST XML file. It first makes the gene tree topologies and node times. It then calls Seq-Gen to produce the sequences. Then it constructs the BEAST XML file containing the sequences and 'standard' options for the allopolyploid model in BEAST.

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

2 Installation

Compile seqgen:

```
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
```

`gcc` might need replacing with a path to the executable for the Gnu C compiler. Under Linux, add `-lm` to command line. Put `seqgen.exe` in a directory with the R code:

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

3 Usage

Run with a command like

```
"C:\Program Files\R\R-2.11.1\bin\R.exe" --slave --vanilla --args
config.txt output.xml < AlloppSim_main.r
```

where `config.txt` and `output.xml` specify the input config and output BEAST XML files.

3.1 Example config file

Here is a simple example of the input.

```
noftetratrees 1
nofgenes 5
genelength 500
mutationrate 1e-7
gtree_seed 5400127
sequence_seed 7745719
ditree ntips 2
a indivs 2 tippop 110000 prevpop 120000
b indivs 2 tippop 130000 prevpop 140000
(a,b) height 0.04 tippop * prevpop *
tetratree ntips 1
z indivs 2 tippop 150000 prevpop *
hyb 0.01 1000 join * * foot1 a .02 210000 foot2 b .03 220000
```

It corresponds to this scenario.

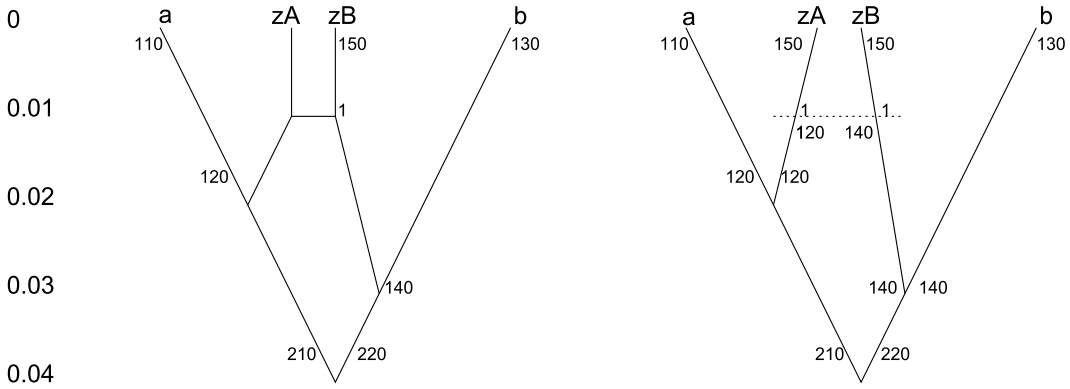

Figure 1: example

The left diagram shows the network as described by the input, and the right diagram is the corresponding MUL-tree. Populations are shown in thousands. Node heights are on the far left. Node heights are in units of expected number of substitutions.

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.

`noftetratrees 1`

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

`nofgenes 5`

Make sequences for five genes.

`genelength 500`

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

`mutationrate 1e-8`

The mutation rate: expected substitutions per site per generation.

`gtree_seed 5400127`

Random seed for the gene trees (topology and coalescent times).

`sequence_seed 7745719`

Random seed for the sequences. (Value used to make one seed per gene, which are then 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,...`. A tip node then specifies the number of individuals sampled and a tip population. A tip node is assumed to have height zero.

An internal node uses Newick style like `(left child, right child)` for its clade and specifies a height (but not a number of individuals).

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 indivs 2 tippop 110000 prevpop 120000
b indivs 2 tippop 130000 prevpop 140000
(a,b) height 0.04 tippop * prevpop *
```

The diploid tree with two tips.

```
tetratree ntips 1
z indivs 2 tippop 150000 prevpop *
hyb 0.01 1000 join * * foot1 a .02 210000 foot2 b .03 220000
```

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 following routines.

5.1 Reading the config file

`AlloppSim_1readconfig.r` reads the input, checks it (a bit) and stores the config.

5.2 Making the 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	2	110000	-1	120000
2	bA	7	-1	-1	0	-1	2	130000	-1	140000
3	aA,zA,bA,zB	-1	6	7	0.04	-1	-1	-1	-1	-1
4	zA	6	-1	-1	0	0.01	2	150000	1000	120000
5	zB	7	-1	-1	0	0.01	2	150000	1000	140000
6	aA,zA	3	1	4	0.02	-1	-1	-1	-1	210000
7	bA,zB	3	2	5	0.03	-1	-1	-1	-1	220000

5.3 Making the 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.

num	name	time	lft	rgt	anc
1	01zA	0.000000	0	0	3
2	02zA	0.000000	0	0	3
3		0.000195	1	2	7
4	01aA	0.000000	0	0	6
5	02aA	0.000000	0	0	6
6		0.000172	4	5	7
7		0.022259	3	6	15
8	02zB	0.000000	0	0	10
9	01zB	0.000000	0	0	10
10		0.003049	8	9	14
11	02bA	0.000000	0	0	13
12	01bA	0.000000	0	0	13
13		0.001078	11	12	14
14		0.030253	10	13	15
15		0.041909	7	14	0

This is converted into a Newick string which looks like this.

```
(((01zA:0.00019532,02zA:0.00019532):0.0220638,
(01aA:0.00017193,02aA:0.00017193):0.02208719):0.01964963,
((02zB:0.00304943,01zB:0.00304943):0.02720394,
(02bA:0.00107758,01bA:0.00107758):0.02917579):0.01165538)
```

This can be passed to Seq-Gen.

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 genetree2.txt -y seqs2.txt
```

with the following parameters taken from input config.

```
-l genelength
-z a random seed derived from sequence_seed
```

and the file names like `genetree2.txt` (gene tree input to Seq-Gen) and `seqs2.txt` (output sequences).

5.5 Making the XML

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