treesim

This document is currently in early stages, if you have questions please email: Email: niallc@gmail.com and I will answer your question and update this document, possibly introducing an FAQ section.

1 Installation

Basic instructions (assuming that you have access to Linux and have some familiarity with it):

Copy all the files into a directory of your choice. I'll call this directory workDir, open a terminal and navigate to workDir: Indented text should be entered directly into the terminal: Run:

```
tar -xf treesim.tar
tar -xf testFiles.tar
```

this should make all the source files appear, and some new directories.
type

```
make
```

Having done this, if there are no problems, treesim should be installed (so a file called treesim should exist in workDir). Now to use the test files copied from the webpage. I'll break the command over multiple lines and then give a one line version at the end. Brief explanations of the parameters are given in the file params.cpp

2 Set up of example data

Make a directory for the output, either:

```
mkdir test_output/
```

or make your own and then use the -outputDir argument to tell the program to put the trees there.

You'll also need to unzip the data files, so type:

```
gunzip testFiles/*
```

which will unzip files ending in .gz and tell you that it's ignoring the others.

3 Input File formats

The file formats required are 'impute format' files, specifically the: haplotype, legend and genetic map files described here:

`http://www.stats.ox.ac.uk/~marchini/software/gwas/file_format.html`

genotype files are not required. If using 1000 genomes data then these can be produced from vcf files using vcf tools (downloadable from http://vcftools.sourceforge.net/)

4 Execution

To run the program you'll need to type the name of the executable (treesim) followed by a series of command line arguments. These are specified by first entering a dash followed by a word eg: -legend and then a space and another word/number. An example command line is below. Note that the \ characters allow the command to be broken over multiple lines for clarity, this can also be written on one line.

```
./treesim                                \
  -haplotypes testFiles/testHaps1.txt    \
  -legend testFiles/testLegend1.txt      \
  -genMap testFiles/testMap1.txt         \
  -writeType genecluster                 \
  -lower 14005000                        \
  -upper 14005200                        \
  -buffer 50
```

The program should then think for a while and then produce a file in the output directory containing the tree information. The format of these trees is described below. If the argument "-writeType genecluster" is omitted then the format "oldStyle" will be used, with this argument (which is useful if you want to plot the tree using the supplied code) then the "genecluster" format is used.

To plot the trees: Download the file treesimHelperCode.r to a directory which I'll refer to as codeDir. Load *R*, then you can then load these functions using the line:

```
source(paste(codeDir, "treesimHelperCode.r", sep=""))
```

Then assuming the results from running treesim are in the directory treesDir you can plot a one of the trees in the test results by running:

```
treeFile = paste(treesDir, "test_Trees_model=SMC_n=120_genecluster.txt", sep="")
plot.tree.file(tree.file=treeFile, focal.posn=13958290)
```

The focal position here is an example from the file, the list of 'focal positions' (at which trees are supplied) is defined when running treesim, and also given again in the output file.

5 Fast Genealogies

When, for example, the user wants to generate a set of genealogies, one best guess genealogy per site, then a plausible command line would be:

```
./treesim                                \
  -haplotypes yourHapsFile               \
  -legend yourLegendFile                 \
  -genMap yourMapFile                    \
  -lower startOfRegion_bp                \
  -upper endOfRegion_bp                  \
  -buffer 0                              \
```

```
  -coal_fast         true                  \
  -first_nonrec     true                  \
  -greedy_mle        true                  \
  -choose_hap_first  true                  \
  -no_runs           1                     \
  -change_rho        0                     \
  -write_likelihoods false                 \
  -write_trees       true
```

6 Likelihoods

When running the program to calculate a likelihood curve on a small data set there will be too many genealogies to record, so a command like:

```
./treesim                                  \
  -haplotypes yourHapsFile                 \
  -legend yourLegendFile                   \
  -genMap yourMapFile                      \
  -lower startOfRegion_bp                  \
  -upper endOfRegion_bp                    \
  -buffer flankingSNPs_kbp                 \
  -coal_fast         false                 \
  -first_nonrec     false                 \
  -greedy_mle        false                 \
  -choose_hap_first  true                  \
  -no_runs           200000                \
  -change_rho        12                    \
  -bridge_resolution 110                   \
  -write_likelihoods true                  \
  -write_trees       false
```

7 Understanding output Files

7.1 tree files

There are multiple possible output formats for tree files. The specific type can be chosen using the argument:

```
-writeType
```

and the options for this are:

```
-writeType genecluster
-writeType ZhanMark1
-writeType allFiles
-writeType allEvents
-writeType oldStyle
```

If the writeType is set to 'genecluster', then the output will have the form:

```
13955700 1; 120; 110 30, 114 92, 79 3  , 10 9, ...
13956425 1; 120; 110 30, 114 92, 79 3  , 10 9, ...
13958176 1; 120; 110 30, 114 92, 79 3  , 10 9, ...
13962089 1; 120; 110 30, 114 92, 82 121, 10 9, ...
13962487 1; 120; 110 30, 114 92, 82 121, 10 9, ...
```

Each line has the same format. The first integer is the genomic position that the tree refers to. The second number is always 1 in the case of treesim and is included to satisfy the formatting requirements of the program *genecluster*. There is then a semicolon and the next number is the number of haplotypes. Following the next semicolon is a collection of $n-1$ pairs of numbers, separated by commas. These indicate the sequence of coalescence events: The haplotypes are numbered from $1-n$ according to the order they appear in the input data file. When two sequences coalesce this generates a new lineage, the first coalescent event creates lineage $n+1$ the second $n+2$ and so on (for example, the number 121 represents the lineage created by the first coalescence event in the example data above).

If the writeType is set to 'oldStyle', then the output will look something like:

```
No sites:                      49
Rho from 1st to last site:     1e-09
Number of sequences:           120
Using SMC? (0=Coal):           1

site      Dscndt1     Dscndt2    Time     Total time below  Active?

0            -1          -1         0           0              0
0            -1          -1         0           0              0
...
0            110          30        0.0227      0.0453         0
0            114          92        0.0327      0.0654         0
0            79           3         0.0005      0.112          0
...
1            -1          -1         0           0              0
1            -1          -1         0           0              0
...
1            110          30        0.0227      0.0453         0
1            114          92        0.0327      0.0654         0
1            79           3         0.0562      0.112          0
...
n            -1          -1         0           0              0
n            -1          -1         0           0              0
...
n            110          30        0.0227      0.0453         0
n            114          92        0.0327      0.0654         0
n            79           3         0.0562      0.112          0
```

The top few lines simply describe the input data, and which model was used. This is followed by a table. The columns in the table represent

- site: The index of the site that this tree describes. As this increments the trees move from left to right across the SNPs.

- Dscndt1: The label of the first (arbitrarily defined) of the two daughter lineages of this node. Lineages are initially numbered from 1 to 120, and lineages created by coalescence are assigned the lowest positive integer not already used. Daughter lineages of leaf nodes do not exist, and are reported as -1.
- Dscndt2: The label of the second (arbitrarily defined) of the two daughter lineages of this node
- Time: The distance, in coalescent time, from the present day, backwards to when this event occurred
- Total time below: The total time, across all branches in the tree, between the point at which this node is created and the present
- Active: Kept for back-compatibility only. Should always read zero, to indicate that the simulation has gone back to the common ancestor at this site

The first n (where n=the number of input haplotypes) lines of each file are redundant, and if "all_nodes" is set to 'false' at the command line then these will be omitted. Brief explanations of various arguments for the program are given below. Email me for more details.

7.2 Times files

By default the program will also output times summarising the tMRCA at each locus. This can be turned off by setting: '-collect_times false'.

Warning: Note that the times produced by treesim may be biased upwards when heuristic tree building is used, especially in regions with complex patterns of recombination. This is because the heuristic will not typically find the most parsimonious (and therefore most likely) patterns of recombination to remove the incompatibilities that may exist. This must be done for the process to complete. This may be especially relevant to those detecting *selection* as it may cause a confounding effect between selection and recombination.

7.3 Likelihood files

At present the likelihood files are more designed for human readability than computer readability, this somewhat undesireable situation is somewhat rectified by the release of R code which can parse, collate and plot the likelihood information.

The main sections in the file are the likelihoods directly calculated from the driving values and then the much greater number of values calculated by using these driving values to infer likelihoods at values not used in the proposal.

8 Useful Arguments

```
-legend             legend file to process
-haplotypes         hapoltypes file to process
-genMap             genetic map file to process
-upper              upper end of interval to analyse
-lower              lower end of interval to analyse
```

```
-buffer                extra SNPs (in kb) to avoid edge effects
-display_width         irrelevant display parameter
-Ne                    effective population size
-rhoScale              scale on which to measure rec rate (rho) - "cMperMb" or "coalescentScale"
-theta                 mutation rate
-rho                   recombination rate
-SMC                   use SMC? (if not coalescent)
-choose_hap_first      Speed up (from Fearnhead and Donnelly) highly recommended (set to 1)
-coal_fast             Speed up, recommended for genealogies, not for likelihoods
-threshold             Controls coal_fast, higher values are slower, threshold=1 equivalent to coal_fast=false
-first_nonrec          Very fast genealogies, but more parsimonious than samples from the posterior
-use_logs              Improves speed when there is no risk of underflow (so useful for likelihoods)
-set_seed              Allows manual setting of the seed, 0 uses the system clock
-using_wattersons      Instead of the supplied mutation rate, use an automatic value. (!) requires discussion.
-no_runs               The number of genealogies to simulate
-change_rho            The number of values of rho to try (only useful for likelihoods I think)
-greedy_mle            Produce a single best-guess genealogy, using a greedy algorithm
-write_likelihoods     Output files with information about the likelihood estimate?
-write_trees           Output the simulated trees (have this set to 'false' for large numbers of runs!)
-writeType             Controls the format of the output trees
-outputDir             The directory in which to place the output files
-collect_times         Write a file giving the estimated tMRCA at leach locus?
-file_tag              A string to be added to each file (helps avoid overwriting)
-DEBUG                 controls level of screen output and when program should stop/pause
-verbose               controls level of screen output, ideally will take over from DEBUG
-debugMaxNumHaps       If set to 'n' then only the first n haplotypes will be used. Mostly for debugging.
```

9 defaults

```
theta             = 1e-9 ;  mutation rate
rho               = 1e-9 ;  recombination rate
SMC               = true ;  should the program use the SMC model (if not, then the full coalescent)
importance_sample = true ;  should the program perform importance sampling? Set to 1 even for intelligent tree simulation.
sim_data          = false;  should the program simulate data (probably not implemented yet)?
using_rec_map     = false;  should the program use an explicit rec_map? Set to FALSE as this is done a different way
all_nodes         = true ;  should the program output the leaf nodes in the genealogies (redundant info, but maybe convenient)
DEBUG             = 0    ;  affects behaviour for minor exceptions and control level of output to the console
verbose           = 1    ;  new barely implemented variable: controls level of console output
sampler           = 2    ;  what approximation to P(H1 | H2, ..., Hn) to be used. 2 is Li & Stephens. Only 2 and 4 available
no_runs           = 1;      how many genealogies to simulate per location?
change_rho        = 0;      Simulate genealogies for this many values of rho, if 0 then use supplied value
bridge_resolution = 110;    used in likelihood calculation. Sets density of bridge sampling based likelihood values
change_theta      = 0;      As change rho, but for the mutation rate
greedy_mle        = true ;  This tries to find a single 'best' ARG. When true program is NOT stochastic, should only have use run.
choose_hap_first  = true ;  Chooses a faster method of generating genealogies - next even hap chosen from prior rates.
coal_fast         = true ;  A 'fast' method of generating genealogies, not great for likelihood calculation
threshold         = 0.2  ;  (0 < Threshold < 1, sets how approximate the genealogies are with coal fast)
choose_nonrec     = true ;  Supposed to be a very fast method of generating genealogies - but makes little difference)
first_nonrec      = false;  A very fast method of generating genealogies)
use_logs          = true ;  Should calculations within the Li&Stephens function use logs (faster, but potential underflow problems)
cheap_coal_upwt   = false;  always set to false
display_events    = false;  This displays every event in each genealogy when No Runs is small.
display_haps      = false;  This displays the haplotypes in each epoch
display_weights   = false;  This displays the weights for each event in each epoch
set_seed          = 1    ;  Put in a positive integer, n, here and the seed will be set to -n (else uses system clock to set seed)
using_wattersons  = true ;  Should the Watterson estimate of 'theta' be used? If not then theta defined above is used
write_likelihoods = false;  should the average likelihoods should be written to file
write_trees       = true ;  should the trees should be written to file
write_full_data   = false;  should every likelihood should be written to file
writeType         = "oldStyle"; //  default - What type of tree format to use at the end (if writing trees)
outputDir         = "./test_output/";  By default output files will be placed in the directory run from.
output_MLE        = false;  Must remain at false - not fully implemented yet.
collect_times     = true ;  should certain patterns of tMRCA across sequences should be stored
record_every_time = true ;  should tMRCA at every site should be recorded on every iteration
collect_densities = false;  This indicates whether to report the density of each ARG.
file_tag          = "test";  a string to be appended to output files. Can be set at the command line for multiple runs of the program
tree_file_prefix  = "use_default";  prefix to put on the front of treeOutput files
dataFormat        = "hapmap"     ; names of the data input format to be used - use 'hapmap' unless you know whhat you're doing.
useHapmap = dataFormat=="hapmap" ; somewhat redundant boolean to record whether hapmaap input data being used.

hapmapParams.legend_file       = "testFiles/testLegend1.txt";       //default filename
hapmapParams.haps_file.fName   = "testFiles/testHaps1.txt"  ;       //default filename
hapmapParams.map_file          = "testFiles/testMap1.txt"   ;       //default filename
hapmapParams.upper             = -1;                                //default upper bound for region to analyse
hapmapParams.lower             = -1;                                //default lower bound for region to analyse
hapmapParams.buffer            = -1;                                //default region around results region to avoiid edge effects
hapmapParams.display_width     = 10;                                //default display setting, unimportant
hapmapParams.Ne                = 1e4;                               //default effective population size
hapmapParams.strand_file.fName = "strand file not implemented yet";
hapmapParams.sample_file.fName = "sample file not implemented yet";
hapmapParams.rhoScale          = "cMperMb"; //alt="coalescentScale" // default scale on which to measure recombination rate

debugMaxNumHaps   = -1       ; // deflt - Just for debugging, can be used to reduce the number of haplotypes
```

10 Helper R code

The main functions are:

1. **runTreesim** - run a single instance of treesim, using args from "prepareTreeSimLine"
2. **combineMultipleTreeSimRuns** - parse and collate output from multiple treesim runs
3. **displayTreesimLikelihoods** - plot graph of treesims likelihoods including bridge sampling curves
4. **parseOutput** - read in the output from a single treesim run
5. **getFileInfo** - edit this so that files and be automaticaly found and created
6. **prepareTreeSimLine** - edit this to adjust command lines to be as you want them