ScisTree: A Program for Maximum Likelihood Cell Tree Inference and Genotype Calling from Noisy Single Cell Data

User Manual

Version 1.2.1

May 28, 2019

Yufeng Wu
CSE Department, University of Connecticut Storrs, CT 06269, U.S.A.
Email: yufeng.wu@uconn.edu

Yufeng Wu, Accurate and Efficient Cell Lineage Tree Inference from Noisy Single Cell Data: the Maximum Likelihood Perfect Phylogeny Approach, manuscript, 2019.

1 Getting Started with ScisTree

1.1 Program availability

ScisTree is written in C++. Executables for popular platforms such as Linux 32 bits or 64 bits and MacOS are downloadable from GitHub:
`https://github.com/yufengwudcs/ScisTree`. Files can be downloaded using "Save Link/Target As..." After downloading the softwares, you may need to change file access permissions (e.g. chmod u+x scistree-linux). In case that you want to compile the code yourself, source code is also available for download at the above URL. To compile the code, first put the gzip file in the directory youd like and unzip it: use gunzip and tar commands such as:
▷ gunzip ⟨scistree-src.tar.gz⟩
▷ tar -xvf ⟨scistree-src.tar⟩
Then type:
▷ make at the prompt. This creates an executable called stells, which can be run by typing
▷ scistree at the prompt. You will need to specify some input options - see below.

1.2 What is ScisTree?

First, where does the name ScisTree come from? It stands for {S}ingle {c}ell {i}nfinite {s}ites {Tree}.

ScisTree is designed to infer cell tree (the evolutionary tree of single cells) and call genotypes from noisy single cell data. ScisTree takes uncertain genotypes in the form of genotype probability. This is because genotypes called from single cell sequence data tend to be very noisy. It is usually difficult to call a fixed genotype at a position. Different from several existing methods for cell tree inference, ScisTree works with uncertain genotypes with **individualized** genotype probabilities. That is, each genotype (at a site and a cell) can have its own probability, which specifies how likely this genotype has a particular genotype state. This can better utilize information contained in single cell data: often some genotypes in the single cell data can be almost fully determined while others have much larger uncertainty.

ScisTree infers cell tree and call genotypes simultaneously. It can deal with single cell technological noises such as doublets. One key advantage of ScisTree over some existing methods is that ScisTree is very efficient: it works for data with hundreds of cells and thousands of single nucleotide variants (SNVs). My tests show that ScisTree can be 100 times or more faster than existing methods such as SCITE.

1.3 How does ScisTree work?

ScisTree assumes the infinite sites model. This allows a very simple algorithm for finding the maximum likelihood estimate (MLE) of the cell tree and genotypes that maximize the likelihood of the data under the infinite sites model. Refer to the paper for more details on the methodology of ScisTree.

2 Functionalities and Usage of ScisTree

2.1 Preparing inputs

To run ScisTree, the user needs to provide the genotype probabilities for the SNVs of the cells under study. The genotype probability is specified in a matrix. Genotypes can be either binary or ternary. Here is a simple example of the input.

```
# This is an example.
HAPLOID 5 4 c1 c2 c3 c4
s1  0.8   0.02  0.8   0.8
s2  0.02  0.02  0.02  0.8
s3  0.8   0.02  0.02  0.8
s4  0.02  0.8   0.8   0.8
s5  0.8   0.02  0.8   0.02
```

ScisTree ignores lines starting with #, which are considered to be comments. The first (non-comments) line should have: $< \mathsf{FORMAT} >< num-sites >< num-cells >< cell-name-1 >< cell-name-2 > \ldots$. Here, "FORMAT" can be either "Haploid" or "Ternary". Haploid format refers to binary genotypes, while ternary format refers to

ternary genotypes. The user needs to specify the number of (SNV) sites and the number of cells. There is a line for the probabilities for genotypes of each site. The line starts with a string for cell's name (in the above example, $s_1, \ldots s_5$). Then, the line contains probability for each cell at this site. For each genotype at a site, one specifies the genotype probability sequentially: for binary genotype, use a single value to specify the probability of genotype 0; for ternary genotype, use two values to specify the probability of genotype 0 and genotype 1. In the above example, at the first site s_1, genotype probabilities given mean that the four cells have probability of 0.8, 0.02, 0.8 and 0.8 of being genotype 0. Note that the probability of being genotype 1 is not specified since the probabilities of being 0 and 1 add up to 1.

2.2 Usage

To run ScisTree, you must provide an input file with the single cell genotype probability (using the format as specified above).

▷ ./scistree-mac <genotype-probability-file>

By default, ScisTree outputs the inferred cell tree (in the Newick format) only. In order to output the called genotypes, you should specify the "-v" option:

▷ ./scistree-mac -v <genotype-probability-file>

ScisTree outputs the imputed genotypes from genotype probability. For reference, it first outputs the maximal probability genotypes (taking the most probable genotype at each position in the matrix). Then ScisTree shows the genotypes that are changed from the maximal probability genotypes. The called genotypes are shown below "Imputed genotypes:". ScisTree infers a cell tree without branch length.

ScisTree has also implemented several additional functionalities. These include doublet imputation. See the following for more details.

2.2.1 How to calculate genotype probabilities from input?

There are two basic scenarios on the preparation of genotype probabilities input.

1. Sometimes, there are called genotypes (say 0/1) where there are some uncertainties about genotypes. Here, one only has an estimate of global error rates: $p_{0\to1}$ (probability of mistaking a 0 to 1) and $p_{1\to0}$ (probability of mistaking a 0 to 1). Then, the probability of genotypes can be computed based on the genotype and the corresponding error rate. That is, if the called genotype is 0, then its genotype probability is $1 - p_{1\to0}$; if the called genotype is 1, then its genotype probability is $p_{0\to1}$. This uniform error rate scenario is similar to those considered by the program SCITE and SiFit.

2. The main motivation of ScisTree is handling non-uniform error rate scenario. The most common situation is that we have sequence reads from single cell DNA sequencing. In this case, probability of individual genotypes is determined by the sequence reads. There are various ways of obtaining genotype probabilities from sequence reads. (i) One may use the program Monovar, which calls single cell genotypes from sequence

reads. Monovar can output the genotype probabilities in the VCF format, and (ii) alternatively, you can use a customized program written by myself that converts the single cell read counts to genotype probabilities. More details on converting sequence data to probabilities are given in the Appendix.

2.3 Command line options

For ease of reference, I now provide the list of (optional) command line options.

1. -t ⟨threshold⟩: only use genotypes with genotype probability clearly favoring a single genotype by a clear margin (specified by the threshold value). This option works for binary genotypes at the moment. In this case, if the genotype probability of 0 is 0.98 and the threshold is 0.9, this genotype is considered to be reliable: the difference between 0.98 and 0.02 (the probability of the alternative state) is 0.96, which is larger than the threshold value (0.9). If the genotype probability of 0 is 0.9 instead then this genotype will be discarded when constructing initial trees. This is because the difference between probabilities of alternative genotypes is 0.8, which is smaller than the threshold value.

   Note: the default setting is that ScisTree will use all genotypes for initial tree construction (i.e., default threshold is 0.0). This is not saying you shouldn't use this option. My simulation shows that when data contains significant noise, choosing a relatively high threshold (e.g. 0.9) can improve the inference accuracy. Thus, I **do** recommend to activate this option when the input data is noisy. I didn't impose some non-zero threshold mainly because I don't know the type of data you are dealing with. In future releases of ScisTree, I may choose some non-zero threshold as default.

2. -v: output more information about the results by ScisTree. It outputs the called genotypes, along with genotypes called by simple single site maximal probability genotypes, difference between the two set of genotypes, and some additional information.

3. -d ⟨number of doublets⟩: specify the number of doublets to impute; the default value is zero (i.e., no doublets). If this option is invoked, ScisTree will impute the resulting genotypes with doublets imputed; and then it will infer a cell lineage tree with doublets.

4. -n: only output the cell tree constructed by simple neighbor joining. In this case, neighbor joining is run with the maximal probable genotypes from single positions. This can be useful when one wants to find a quick cell tree for very large data.

5. -e: output a tree that is called mutation tree. Briefly, this tree may not be binary. Mutation tree is implied by the imputed genotypes. Mutation tree is meant to specify the ancestral relationships among mutations (i.e., site labels). If the mutation tree is very large, you may use "-e0" to output a mutation tree without mutation labels. This may help to visualize the tree better for large trees.

6. -o ⟨file name⟩: output a mutation tree in the GML format. By default, the mutation tree is output as "mutation-tree.gml".

3 Revision History

1. 05/28/2019: Release of v.1.2.1: small changes to correct errors.
2. 05/08/2019: Release of v.1.2.0: this involves change of input format to become more informative. Added instructions on how to compute the genotype probabilities from sequence reads.
3. 03/20/2019: Release of v.1.1.0. Added the option to discard low quality genotypes when constructing initial trees. My simulation shows that this can be important to use when dealing with data with significant noise.
4. 10/01/2018: Release of v.1.0.0. Include basic functionality of cell tree inference and genotype calling from uncertain genotypes.

Appendix: Calculating genotype probabilities from sequence reads

Things you need

First, you need to have a proper reference genome first. You also need the following tools. I will assume you have added the paths to the executable to your environment.

1. samtools
2. Monovar: `https://bitbucket.org/hamimzafar/monovar/src/master/`. Monovar calls the single nucleotide variation (SNV) and also genotypes of individual single cell genotypes at the called SNV sites. For each called genotype, it reports the genotype probabilities.
3. scprob: this small tool is to calculate genotype probabilities from single cell read counts. You need to compile it using g++ (something like: g++ scprob.cpp -o scprob) and place the executable (or its link) to the current working directory.

Sequence reads processing

Single cell sequence reads may come in the form of unaligned (in fastq format) or aligned (in BAM format). If it is unaligned, first use a reads mapping tool (e.g., BWA or Bowtie) to align the reads to obtain BAM files. So from now on, we assume the sequence reads are in the BAM format. You should sort and index BAM files using something like (make sure samtools is installed and the path is added to the shell; assume reads.bam is the BAM file containing the mapped reads):

▷ samtools sort reads.bam reads.sorted.bam

▷ samtools index reads.sorted.bam reads.sorted.bam.bai

Next, you need to use samtools' mpileup and Monovar to generate the called genotypes with genotype probabilities (in VCF format).

▷ samtools mpileup -BQ0 -d10000 -f ref.fq -q 40 -b list-bam-files.txt — monovar.py -p 0.002 -a 0.2 -t 0.05 -m 1 -f ref.fq -b list-bam-files.txt -o output.vcf

There are several things to note here. (i) First, ref.fq is the reference genome in the FastQ format. This reference genome needs to be indexed. (ii) list-bam-files.txt is the list of BAM files. One line is for a single BAM file from a cell. (ii) Monovar choices: -p is the false positive rate (FPR, an error turning wild-type to mutant); -a is the false negative rate (FNR, an error turning wild-type to mutant). You need to pick the proper values based on your data. Usually, FNR is much higher than FPR due to allelic dropouts; -o specifies output file (in VCF format). I suggest you to check the Monovar user manual to get more understanding about how to use Monovar.

Monovar-based genotype probability calculation

Once you have obtained the VCF file, you can directly use the calculated genotype probabilities contained in the VCF file to construct the genotype probability matrix for Monovar. For your convenience, I have provided a simple script (in AWK) that can do the conversion for you.

▷ awk -f convMonovarOutputToHapProb-v1.2.0.awk c=100 output.vcf > output.vcf.gprob

Here, `c` refers to the number of cells.

Alternative genotype probability calculation

Alternatively, you can use my own tool (called scprob) for calculating the genotype probabilities. In my current simulations, it appears this can be somewhat more accurate than directly using the genotype probabilities by Monovar. The scprob tool calculates the genotype probabilities based on the read counts (of each cell at a SNV site) that are contained in the VCF file. To use scprob, you first extract reads counts from the VCF file and dump to a read counts file as follows:

▷ awk -f dumpCellReadCounts.awk c=100 output.vcf > output.vcf.cnt

Then, you run scprob on the read counts file:

▷ ./scprob output.vcf.cnt > output.vcf.gprob

I have provided a small sample data in the repository (with step-by-step instruction) on how to perform the above tasks.