Generating and Validating Cancer Mutation Trees
By Christian Choe and Min Cheol Kim

## Introduction

Single cell exome sequencing of cancer cells can give us information regarding their progression. Specifically, we can infer the order in which specific genes get mutated for normal cells to transform into cancer cells. We present a Bayesian technique to infer the mutation orders and a validation procedure for our algorithm. The objective of the project was to implement a version of Kim and Simon's algorithm [1], construct a mutation tree for essential thrombocythemia (ET) cancer similar to theirs, and design a framework to validate and simulate the algorithm under different conditions.

## Data

As mentioned previously, we will implement the mutation tree portion of Kim and Simon's paper and test it on Hou et al's ET data. The ET data comes from a single-cell exome sequencing performed on a patient with ET. The details of this procedure can be found in Hou et al.'s paper on ET tumor evolution.[2]

The example format of the data is shown in Table 1. For each mutation site and for each cell, a 1 indicates that a mutation has occurred and a 0 indicates that no mutation occurred (wild-type). For example, when there are 10 cells sampled and we are tracking 3 mutation sites, we would have 30 1's and 0's indicating whether a specific sample cell has the mutation or not.

| Gene\Sample ID | 1 | 2 | 3 | 5 | 6 | 7 | 8 | 9 | 12 | 16 | 18 | 19 | 20 | 22 | 24 | 25 | 26 | 29 | 30 | 31 | 36 | 37 | 40 | 41 | 43 | 44 | 45 | 47 | 48 | 49 | 50 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PDE4DIP(A->G) | - | - | - | 1 | 1 | 0 | - | 0 | - | - | - | 0 | - | - | - | - | - | - | 1 | - | - | - | 0 | - | 0 | - | 1 | 0 | - | - | - |

Table 1. Example of genotype data for mutation site PDE4DIP

## Methodology

*Determining pairwise mutation order*

We use a version of Kim and Simon's approach to infer the pairwise mutation orders, *x* and *y* (referring to a mutation in gene X and a mutation in gene Y). There are three possible orderings between *x* and *y*: : x→y (mutation site x is upstream of y), x←y (mutation y is upstream of x), and x≠y (mutation x and y occurs in completely separate subtrees).

Let L(x~y) be the likelihood of the x~y relationship (where x~y can be any of the tree relationships above). This likelihood can be written as [1]:

(1)
$$L\left(x \sim y\right) = \prod_{k=1}^{n} \Pr\left(\left(i_k, j_k\right) \,\middle|\, x \sim y\right)$$

Using this likelihood, we can write the probability of a pairwise relationship using the Bayes Theorem [1]:

$$\Pr(x \sim y \mid D) \propto L(x \sim y) \Pr(x \sim y)$$

where D represents our data from our cells. Pr(x ~ y) represents the prior probability of the specific relationships, without our data.

For computation of Pr(x ~ y | D), we need first compute values for Pr(x ~ y) and Pr( (i,j) ) | x ~ y). In other words, we need to compute 1) how often does relationship x ~ y occur (prior) and 2) the probability that a specific genotype pair (for two mutation sites) results given x ~ y. For example, for for the x→y case, we would want to know how often x→y occurs (compared to the other two cases) and what the distribution of (0, 0), (1,0), (0,1), (1,1) is for that specific relation. With no errors, Pr( (0, 1) | x→y ) should be zero, since mutation *y* is downstream of *x*.

To compute these values, we simulate the genealogy tree many times. The for the computation of these values is as follows:

1. Generate a random genealogicla binary tree with *n* terminal nodes, such as one in Figure 1.
2. Time intervals between each branching point was generated as exponentially distributed samples. The time from the very first ancestor to the first mutation event was modeled to using $T_1=\alpha(T_2 + T_3 + T_4 + \ldots..)$ . The justification for this approach comes from the Wright-Fisher model and Kim and Simon. The alpha value represents proportion of time from the earliest mutation to the most recent common ancestor of the sampled cells
3. Once the mutation tree scaffold was generated, intendent mutation pairs (x, y) were introduced to the tree, and the leaf genotypes were noted. This introduction and propagation of mutation was performed Bmut= 10000 times. This step was implemented as changing the appropriate edge (where the random mutation is introduced) and propagating down the mutation all the way to its leaves. The leaves from each and every node were pre-cached at the time of tree generation.
4. Steps 1 - 3 were repeated for Btree = 1000 times.

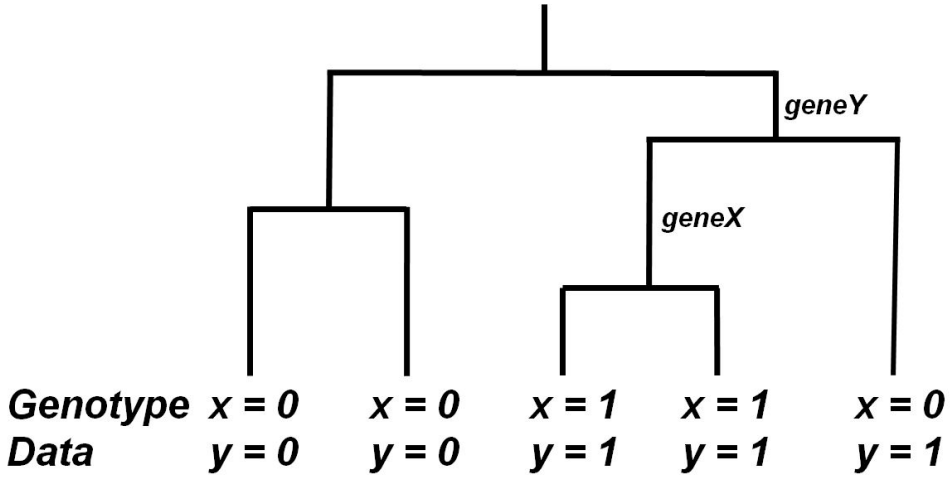


Figure 1. A possible genealogy tree with the given genotype

We gather the required information from the simulations using these formulas:

1. $\Pr(x \to y) = \Pr(\mathrm{x} \leftarrow \mathrm{y}) = \frac{1}{2} * \frac{\#\ of\ mutation\ pairs\ on\ same\ lineage}{B_{tree} * B_{mut}}$
2. $\Pr(x \neq y) = 1 - 2 * \Pr(x \to y)$
3. $\Pr\big((i,j)\big|x \sim y\big) = \frac{\#\ of\ cases\ of\ (i,j) when\ x \sim y}{\#\ of\ cases\ of\ all\ x \sim y}$

From this, we can calculate the above mentioned Pr(x ~ y | D).

*Minimum Spanning Tree*

Once all the probabilities for the different mutation relationships are calculated, a directed graph is constructed using every mutation as a node and edge corresponds to a mutation order. The weight of an edge is equal to the -log[Pr(i→j)] to weight the higher probability relationship lower. If the relation between two sites with maximum posterior probability is that they are not on the same lineage then no edge is drawn between the two nodes since that indicates they are more likely to exist on separate lineage.

After constructing the graph the minimum spanning tree is found to find the optimal mutation tree. We used Edmond's minimal spanning tree algorithm implemented in NetworkX. Since the edges have a lower weight for higher probability, the resulting tree will be the one with the maximum posterior probability.

*Generating Test Data*

Kim and Simon provided no validation of their model, so we decided to invent a framework for testing our algorithm. First we generated random genealogy tree, much like in Step 1 of the algorithm for calculating the priors (Figure 2). Then, we introduce our desired number of mutations to the scaffold tree. Finally, we read off the leaf genotypes to create our "fake"

dataset. As with our calculations of priors, time intervals between each branching point was generated as exponentially distributed samples. In addition, our method for generating test data also yields the “grand truth” mutation tree, since we know exactly in what order the mutations were generated.

*Comparing Trees*

In order to compare the mutation trees generated by our method to the true tree in testing, we needed a technique to measure the similarity between two trees. The techniques we decided to use to measure the similarity between two trees was to examined every node pairs in both trees. When examining an individual pair we considered whether they were in the same lineage and the order they occur if they are in the same lineage.[3] Since every tree we compare will have the same node due to our tree construction method we simply iterated through each pair and added one if they had the same relationship in both trees. After summing the similarities they were divided by the total number of nodes choose 2 to normalize the similarity score from 0 to 1.

In addition, we considered ambiguities in mutation orders when comparing trees. Ambiguities arise when the order of genes can occur in either way. For example if you obtain cell data with genotypes x = 0, y = 0, and x = 1, and y = 1, mutation x can occur before mutation y or vice-versa. Since either answer is correct due to the lack of information, we give some leeway for these cases.

*Simulation and Validation*

After we designed protocols for generating test data and comparing mutation trees, we performed the following steps listed below to validate our algorithm under various conditions (number of cells available, the alpha value, and the number of mutation sites tracked). In addition, we also incorporated error into our model by including a false discovery rate of 0.4309 and a allelic dropout rate of $6.04 \times 10^5$ which were the values used by Kim and Simon.[1] False discovery rate is when a 0 is read as a 1 and allelic dropout rate is when a 1 is read as a 0.

Simulation Protocol:

1. Generate an artificial genotype dataset.
2. Use our algorithm to produce the inferred mutation tree.
    a. Some runs included false discovery rate and allelic dropout rate.
3. Compare the inferred tree to the true tree generated along with the artificial genotype dataset to yield a similarity score.
4. Repeat Steps 1-3 1000 times to find the average similarity score.

**Results**

We first generated the mutation tree using Hou et al.'s dataset, similar to Kim and Simon's. Our trees were not identical, but had many similar subtrees. Perhaps most importantly, the nodes closer to the root (upstream mutations) stayed near the root, and most of the leaves in Kim and Simon's tree were also leaves in our tree. The top part of our tree is shown in Figure 2. Comparing our ET mutation tree to theirs, the two trees have a similarity score of 0.65. Although the similarity score may seem low, our tree has the same parent node as well as all the correct leaf nodes. The colored nodes and arrow show the substructure in our trees that are also found in Kim and Simon's tree.

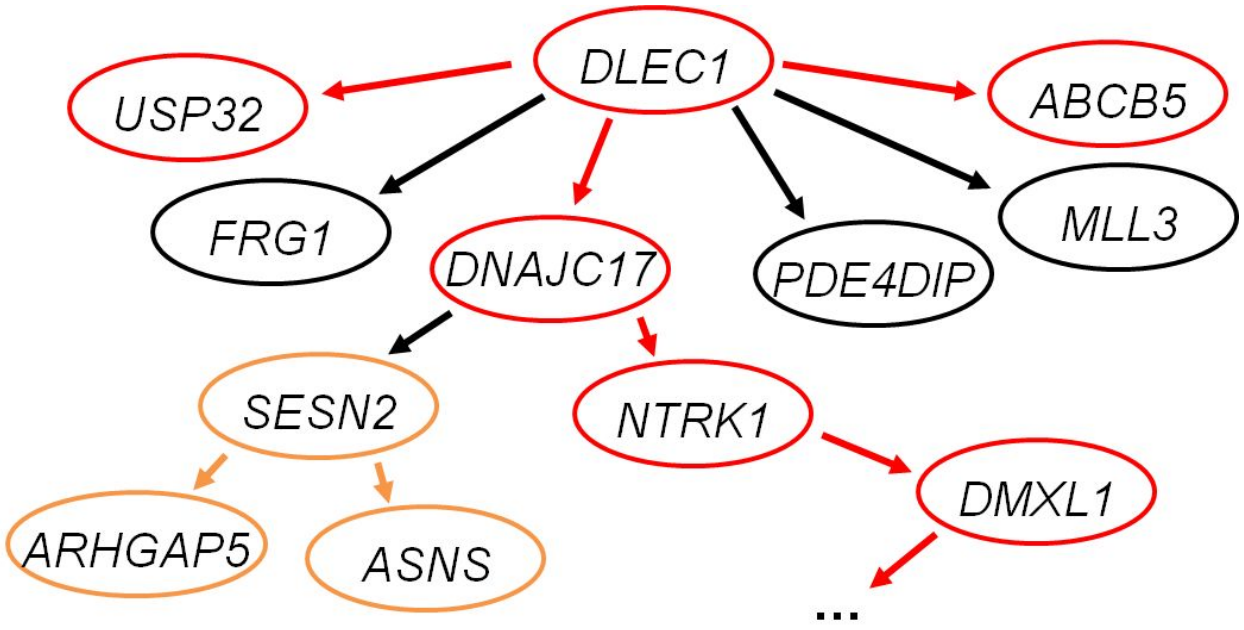


Figure 2. Our inferred mutation tree for ET cancer dataset

We now present our results from validation of the algorithm under various scenarios. We evaluated the algorithm for number of cells = 5, 10, 25, 50, number of tracked mutation sites = 3, 6, 12, 19, and alpha values of 0.3, 0.5, 0.7, and 0.9. The alpha value is biologically determined, and estimating the true alpha value was not part of this project; we incorporated the "true" alpha value in both the simulation and the algorithm. In other words, the algorithm "inferred" the perfect alpha value. Every combination of those values were simulated. In addition, we also simulated with parameters number of cells = 58 and the number of tracked mutation sites = 18, which is the case for the ET cancer dataset. For select simulations, we also performed a baseline simulation where we randomly guess the mutation orders instead of using our algorithm.

Surprisingly, we discovered that with the parameters for the ET cancer dataset (#cell = 58, #mut = 18), the algorithm is unable to stay above the baseline for all alphas (Figure 3).

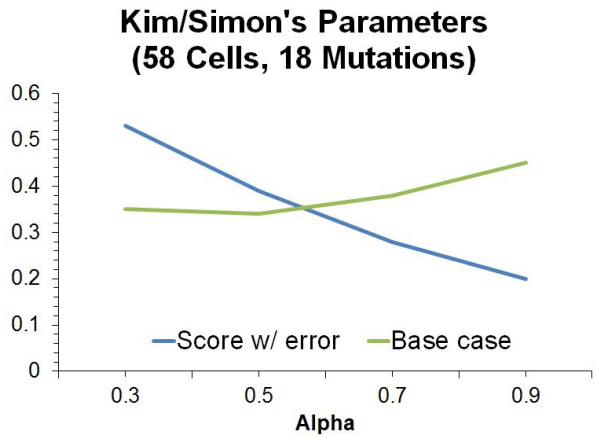


Figure 3. Simulation results with Kim/Simon's parameters

However, we also showed that with large enough number of cells and small number of mutations tracked, we can stay above the baseline for all alpha values. We present the best case scenario and the worst case scenario in the following two graphs Figure 4). This suggests that Kim and Simon's approach is valid only when given enough cells and we are attempting to track relatively few number of mutations. In the worst case, similar to Kim and Simon's case, the algorithm is unable to stay above the baseline.


Figure 4. Best and worst case scenarios.

**Summary/Future Work**

We were able to successfully infer a mutation tree similar to that of Kim and Simon, and also devised a framework to test our algorithm in numerous scenarios. In addition, we discovered that our algorithm is effective when the number of sampled cells are much greater than the number of mutation sites examined; based on our analysis, we question the validity of the

mutation tree generated by Kim and Simon, as they had relatively small number of cells but tracked many mutations.This conclusion is similar to Ross et al.'s conclusion that this algorithm may be effective with large number of cells. [3]

In the future, we may want to try a machine learning approach to predicting tree structure. Another future challenge would be inferring the alpha value from the given data, which was not in the scope of this project. With advances in single cell RNA-seq technologies, this algorithm could be further developed to better infer mutation trees given large amount of single cell information.

**References**

[1] Kim KI, Simon R: Using single cell sequencing data to model the evolutionary history of a tumor. BMC Bioinformatics. 2014, 15:27.

[2] Hou Y, Song L, Zhu P, Zhang B, Tao Y, Xu X, Li F, Wu K, Liang J, Shao D, Wu H, Ye X, Ye C, Wu R, Jian M, Chen Y, Xie W, Zhang R, Chen L, Liu X, Yao X, Zheng H, Yu C, Li Q, Gong Z, Mao M, Yang X, Yang L, Li J, Wang W, et al: Single-cell exome sequencing and monoclonal evolution of a, JAK2-negative myeloproliferative neoplasm. Cell. 2012, 148 (5): 873-885.

[3]Ross Edith M., Markowetz Florian: OncoNEM: inferring tumor evolution from single-cell sequencing data. Genome Biology 17: 69.