Generalized Time-Reversible Model Performance: A Comparison

Michigan State University

Aramis Matos (University of Puerto Rico, Mayaguez Campus)

Kevin Liu PhD (Michigan State University)

Summer 2022

Contents

Abstract

Over the years, many approaches have been developed to estimate the relatedness of organisms. There are several methods as to which relatedness is modeled, the pertinent one is called a phylogeny and relatedness is measured as the genetic distance between two species. Analysis of genetic distance is rather complicated and labor intensive due to sheer number of comparisons necessary among species. As such, computational models have been developed to automate the process and reduce human error. However, there are a variety of models to tackle this conundrum and the earlier models make assumptions about the mutation rates of individual nucleotides. The Generalized Time-Reversible Model (GTR) allows for mutation rates to be tailored to a particular nucleotide and it was created as a response to better align with the reality that some base pairs mutate faster than others. This research aims to measure performance of GTR under varying mutation rates to demonstrate if there is a significant difference in runtime as mutation rates change. The phylogenies that will be used in the runtime analysis are randomly generated. RAxML, a program for estimating ancestral state on large phylogenies, was used to compare a variety of mutation rates and the runtime for each run was measured. We hypothesize an overall increase in runtime as the mutation rates become larger or exceedingly marginal. This is probably due to the fact that as mutation rates become larger or marginal, performance is sacrificed for the sake of accuracy. On this basis, it is recommended that future research measures how much error is tolerable if performance is a serious concern. This research is important since it serves as a reference guide for runtime on the GTR model under various mutation rates. Further research is necessary to gauge if these findings are reproducible on other datasets.

Introduction

The field of computational phylogenetics is an interdisciplinary field in between evolutionary biology and computer science. As the name implies, it involves an evolutionary model that presents relations among species, called a phylogeny. These phylogenies are calculated via models such as the General Time Reversible (GTR) (Tavare, S, 1986) model. What these models achieve is the following: Given a tree, estimate the DNA sequences of all the species in the tree. However, these models are costly in terms of computation time and memory. More specifically, in the time that is required to complete, called runtime. This research aims to see if changing the parameters of the GTR, specifically the substitution rate matrix, has an affect on runtime. This chapter will provide an introduction to the study by first discussing background information and basic definitions, followed by research problem, aims, objective and questions, the research significance and finally concluding with the research limitations.

Basic definitions

Usually, phylogenies are represented as trees. Simply put, trees are a kind of graph that has a parent node connected to two child nodes by a branch, on per each child of a specific length, implying the time since the most recent common ancestor. Each child can be a parent itself. The relevant part is that trees naturally separate into different lineages and one can trace those lineages back to a find common ancestor. Since all life is assumed to have a common ancestor, the root node of a phylogenetic tree represents the common ancestor all of the taxa represented in that tree. An

example of a phylogenetic tree is presented in figure 1:

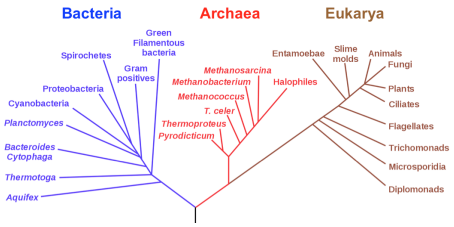

Figure 1: Phylogeny of life
This figure is provided by (*Phylogenetic Trees and Geologic Time | Organismal Biology*, n.d.)

Time is required for changes to accrue, triggered by some separation of two populations of the same species, called a speciation event. As such, the time since that speciation event is called time to most recent common ancestor (Hein, Schierup, & Wiuf, 2005).

Now a question arises. How does one know what species are most closely related to each other? There exists two methods to calculate how related two species are:

1. Comparison of Physical Traits (CPT) (Felsenstein, 2004)

    - Based upon the idea that species can be classified based upon when traits evolved or disappeared from a lineage

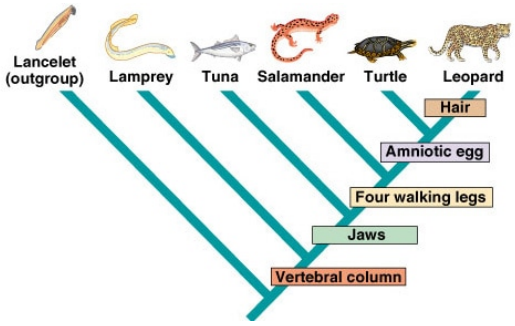

Figure 2: CPT Example
This figure was provided by (*Macroevolution*, n.d.)

    - More intuitive to calculate but more prone to error

2. Genetic Distance (GD) (Felsenstein, 2004)

- Based upon the idea that if one counts the number of changes in and divides that number by the length of the smallest DNA sequence, one can gauge how dissimilar two species are. For example:

  DNA Sequence 1: TGAAGGGATC

  DNA Sequence 2: TGCGTAGTTT

  Number of Differences : 6

  Genetic Distance (Normalized Hamming Distance): 0.6

  The lower the genetic distance, the closely related the two species are.

- Less intuitive to calculate but more accurate

The one most relevant for our research is genetic distance, since it is most easily quantified into a number. Historically, phylogenies had to be constructed manually. It is seemingly trivial to classify a small number of species by hand via CPT, especially if the species in question are very different. However, once the number of species increase or the distinction between species is less clear-cut, taxonomic classification becomes near impossible via GD (Felsenstein, 2004). As a consequence of this, many computational models have been developed in order to automate the process of taxonomic classification, called phylogenetic models. These models take a number of parameters in order to estimate evolutionary change and they require some parameters in order to function. For example, the first and simplest model is (Jukes, Cantor, et al., 1969) model which only requires one parameter, the a single rate as to how often nucleotide bases flip into their corresponding base, i.e. substitution rate. Other models may require different parameters as assumptions that are made in the model's construction are dropped. Among these models is the GTR model. Unlike Jukes and Cantor 1969 (JC69), GTR does not assume equal substitution rate among nucleotide bases, thus necessitating a list of substitution rates for each nucleotide base.

Research Problem

Phylogenetic models such as GTR can be rather computationally intensive. We are defining computational intensity as the time that is necessary for a computer to complete its task, called runtime. Runtime on these models can be considerably large. They are relatively fast on smaller data sets. However, as the amount of time required to complete the task does not scale linearly with the amount of inputs and as a consequence of this, runtime balloons as the number of taxa increase.

However, most research in the computational phylogenetics space focuses primarily with either the models themselves or with programs that package up those models into suites like RAxML (Stamatakis, 2014) or Seq-Gen (Andrew Rambaut, n.d.). This leaves a knowledge gap in the current research as to how particular parameters, specifically the substitution rate parameters, within GTR affect runtime performance.

Due to this gap, academic centers which are notorious for scarce computational resources are affected in a substantial manner. Since the simulations require a large amount of computation time, research may be slowed down by not having access to speedy computation, particularly on exceedingly data sets with a large number of taxa. Thus, this research is import in allowing academic institutions to better allocate time and resources if the characteristics of the phylogenetic job that is about to be processes are known.

Research Aims, Objectives and Questions

Research Aims

Given the lack of research into how different values in the substitution rate matrix for the GTR model affect its runtime, this study will aim to identify if values that are exceedingly marginal, those approaching zero, or exceedingly large values in the substitution rate matrix affect runtime in the aforementioned model.

Research Questions

1. Do exceedingly small values in the substitution matrix for the GTR runtime increase?
2. Do exceedingly large values in the substitution matrix for the GTR runtime increase?
3. Is there a significant difference between the runtimes of exceedingly small and large values?

We hypothesize an overall increase in runtime as the mutation rates become larger or exceedingly marginal.

Significance

This study will contribute to the body of knowledge on computational phylogenetics by examining if certain values in the substitution rate vector, mainly those who are exceedingly small or large, have an effect of runtime in a field that that appreciates any reduction is runtime.

Study Limitations

This research has a number of limitations:

- The findings are only applicable to the GTR model
- The values of the substitution rate vector are a very small subset of all possible set of values
- Floating point error is a very serious problem for very small values
- A lack of knowledge on the intricacies of the GTR model and its particular implementation in the software used in the study

These limitations cause the following:

- Lack of generalizability towards other evolutionary models
- Due to the physical impossibility of calculating all possible combinations of inputs, the sample size is low overall when compared to the size of infinity
- Precise measurements are difficult compute and may skew the results
- The lack of knowledge may lead to misinterpretation of the results

These effects can be mitigated in future work by:

- Creating a frame of reference to ensure interoperability with other evolutionary models
- Parallelization of tests to achieve a better sample space
- The use of binary coded decimal to achieve better accuracy
- Study the models and implementation more

Methods

As mentioned previously, this research aims to see if exceedingly small or exceedingly large values in the substitution rate vector affect runtime in the GTR model when compared to a control set of substitution rates and to each other.

The structure of this chapter is as follows:

1. Data Analysis Techniques
2. Methodological Limitations

The trees were generated utilizing a Monte-Carlo approach.

Data Collection Methods

Tree Generation and Selection

The tree that was utilized for all groups, those being the control group, the Close to Zero Values (CTZ) group and the Very Large Values (VLV) group, was generated via INDELible (Fletcher & Yang, 2009), a program that randomly generates a tree given some parameters. The particular parameters for the tree that was generated were the following:

```
1 [MODEL] m [submodel] JC
2
3 [TREE] t3
```

```
  [rooted] 1000 2.4 1.1 0.2566 0.34
  [treedepth] 1

[PARTITIONS] p3 [t3 m 1000]

[EVOLVE]
 p3 10 p3_out

```

This code was placed into a file called **control.txt** and said file was accompanied by the INDELible binary. The command that was used to run this simulation is **./indelible**.

This generated 10 ultrametric trees, in a file called **trees.txt**, with random branch lengths with 1000 taxa each. Furthermore, the trees were generated under the birth-death model with the following parameters:

- Birth Rate: 2.4
- Death Rate: 1.1
- Sampling Rate: 0.2566
- Mutation Rate: 0.34

A random tree was selected among the 10 that were generated. The specific tree used can be found at the project's GitHub repository (aramis matos, 2022) under the title **example.tree**.

Seq-Gen Automation Script (sgt.bash)

Each run for all groups, the control group, the CTZ group and the VLV group, were run with Seq-Gen. Seq-Gen is a program used to estimate the ancestral state of a phylogeny's taxa and was ran with the following parameters:

- Model: GTR
- Base Frequencies for A, C, G and T Respectively: 0.3,0.2,0.2,0.3
- Length of the Resulting Sequence: 40
- Tree used as input: **example.tree**
- G to T Substitution Rate: 1

As for the remaining 5 values substitution rate vector, A to C, A to G, A to T, C to G and C to T, they are assigned by the Runtime Calculation Script (seq_gen script.hs). The details of the assignment shall be discussed in the next subsection. The reason as to why the G to T substitution rate is set to one is mere because it acts as a scalar on the rest of the values in the vector. Therefore, it was given the value of 1 for the sake of accuracy and simplicity. As for the base frequency and length of the resulting sequence, they were chosen because the example in the documentation for Seq-Gen used them and we infer that this means they are sensible defaults

The aforementioned parameters are implemented in the **sgt.bash** file in the project's repository (aramis matos, 2022).

Runtime Calculation Script (seq_gen_script.hs)

```
import Data.List (intercalate)
import GHC.Float (powerDouble)
import System.Process (readProcess, callCommand)

closeToZero = take 100 $ map(\x -> 1 / powerDouble 10 x)[1.0 ..]

closeToDoubleLimit = [maxBound - 99 .. maxBound] :: [Int]

turnToDouble x = read (filter (/= '\n') x) :: Double

getRuntime val = do
  let str_val = show val
  let str = intercalate "," $ replicate 5 str_val
  let runs = replicate 1000 $ readProcess "./sgt.bash"
  (return str) [] >>= \x -> return $ turnToDouble x
  x <- sequence runs
  let avg = sum x / 1000
  return $ str_val ++ "," ++ show avg ++ "\n"

main = do
  let header = "val,time\n"
  writeFile "control_group.csv" header
  writeFile "close_to_zero.csv" header
  writeFile "close_to_double_limit.csv" header
  ctz <- mapM getRuntime closeToZero
  ctd <- mapM getRuntime closeToDoubleLimit
```

```
27  gc <- getRuntime 0.5
28  let ctz_str = concat ctz
29  let ctd_str = concat ctd
30  appendFile "close_to_zero.csv" ctz_str
31  appendFile "close_to_double_limit.csv" ctd_str
32  appendFile "control_group.csv" gc
33  callCommand "rm temp.txt"
```

As can been seen on line 5, the numbers that are considered CTZ can be expressed as the first 100 elements of the series $\frac{1}{10^i}$ for $\mathbb{Z}_i$ from $[1, \infty)$. These values were chosen for two reasons: First, they are easy to calculate. Second, the series progresses into an ever smaller quantity that in a computer, for all intents and purposes, is 0.

As for the group of VLV, we define large as the largest number a signed 64 bit integer can hold, in other words:

$2^{63} - 1 = 9,223,372,036,854,775,807$

Thus, the series for large values is $2^{63} - 100 + i\ \mathbb{Z}_i$ from $[1, 100)$, as shown in line 7.

The control group's value is simply 0.5.

The *getRuntime* function on lines $11 - 18$ receives a value from from any group and adds returns the average runtime of that value across 1000 runs of Seq-Gen. The value of a 1000 for the number of repetitions was chosen in order to reduce the variance of the sample, thus producing a more accurate result. Important to note that the values for A to C, A to G, A to T, C to G and C to T in the substitution rate vector are the same for each run for a particular group. For example, if the value chosen was 0.1, then the substitution rate vector would look like $0.1, 0.1, 0.1, 0.1, 0.1, 1$ for A to C, A to G, A to T, C to G, C to T and G to T respectively. The reason why this design decision was made is twofold. First, it is logical to assume that if the runtime is affected by a singular extreme value, then it should be affected even more by having all but one value, that being the G

to T substitution rate, be extreme. Second, due to computational and pragmatic limitations, only a small subset of all possible inputs can be tested, given that by the very nature of real numbers, they are infinite.

The generation of the data points across the CTZ series, the VLV series as well as the control set of values is produced in the *main* function. Here, function *getRuntime* is calculated as follows:

1. The runtimes for the CTZ values are calculated and printed into a file called **close_to_zero.csv**
2. The runtimes for the VLV are calculated and printed into a file called **close_to_double_limit.csv**
3. The runtime for the control group value is calculated and printed into a file called **control_group.csv**

Finally, with those data files calculated, data analysis can be performed.

Data Analysis Techniques

The Data Analysis Script (**describe_thingy.py**) code was used to calculate the descriptive statistics for all the groups. The total count, mean, standard deviation, minimum value, maximum value and the 25^{th}, 50^{th} and 75^{th} percentiles. paired samples T-Test in between VLV group and the control, paired samples T-Test in between CTZ group and the control, paired samples T-Test in between VLV group and the CTZ group are to be conducted to assess accuracy. The reason as to which paired T-tests are used is to compare the means of pairs of groups, exactly what is necessary given the research questions.

Methodological Limitations

This research has a couple of key limitations:

1. It is impossible to check the runtime of all possible combinations of the substitution rate vector due to the nature of real numbers
2. It is impossible to check the runtime of all possible combinations of the base frequency rate vector due to the nature of real numbers
3. It is impossible to check the runtime of all possible combinations of the previous items together due to both time, knowledge and budgetary constraints
4. Floating point error and the finite nature of memory make absolute precision near impossible
5. The fairly small sizes of the data sets may be unrepresentative in real research

The first four limitations are simply problems imposed when working with real numbers. The final limitation is brought on by time and computational constraints. However, by having the number of runs per data point at 1000 and then averaged out, we attempted to reduce variance as much as possible given the limitations of the research.

Summary

In summary, INDELible was used to generate 10 trees at random and 1 was selected randomly. That tree was used for the runs in the control set, CTZ set as well as the VLV set. Each value from each set was passed through **sgt.bash** 1000 times by **seq_gen_script.hs**. These values were placed in three separate files: **close_to_zero.csv**, **close_to_double_limit.cs** and **control_group.csv** for the CTZ set, VLV set and the control set respectively. Then these files are passed to **describe_thingy.py** to generate the descriptive statistics for the three files.

Results

Research Question Review

To recapitulate and better guide the key findings in the results, a review of the research questions is necessary:

Research Objectives:

Research Questions:

1. Do exceedingly small values in the substitution matrix for the GTR runtime?
2. Do exceedingly large values in the substitution matrix for the GTR runtime increase?
3. Is there a significant difference between the runtimes of exceedingly small and large values?

The most appropriate statistical analyses to evaluate these is by utilizing the following tests:

1. Paired Samples T-Test in between VLV group and the control
2. Paired Samples T-Test in between CTZ group and the control
3. Paired Samples T-Test in between VLV group and the CTZ group

Introduction

This chapter will commence by presenting the data points in a line graph for the VLV, CTZ and control groups. This is followed by the descriptive statistics of all three groups, followed by the aforementioned paired samples T-Tests, then a box-plot for all three groups, concluded by a summary of key finding.

Data Shape

This table requires an explanation. For the CTZ and VLV lines, 0 on the x-axis represents less extreme values and the ones on the right more extreme values. In other words, for CTZ the values approximate zero the more rightward one goes and for VLV to larger values the more rightward one goes. As for the control group, this rule does not apply since the values in the substitution rate vector do not change, the line is simply running Seq-Gen 100 times on the same settings.

Figure 3: Time taken for Each Group

Statistical Analysis

The key statistical points in the following tables are:

- The mean (as marked in red)

- The low standard deviation

	time
count	100.000000
mean	0.000148
std	0.000003
min	0.000139
25%	0.000146
50%	0.000148
75%	0.000149
max	0.000156

Table 1: VLV Statistics

	time
count	100.000000
mean	0.000145
std	0.000002
min	0.000140
25%	0.000143
50%	0.000145
75%	0.000146
max	0.000153

Table 2: CTZ Values Statistics

	time
count	100.000000
mean	0.000146
std	0.000004
min	0.000141
25%	0.000144
50%	0.000145
75%	0.000148
max	0.000160

Table 3: Control Set Statistics

The paired sample T-Tests in between:

- VLV group and the control
- CTZ group and the control

Paired Samples Test

		Paired Differences					t	df	Significance	
					95% Confidence Interval of the Difference					
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			One-Sided p	Two-Sided p
Pair 1	vlv_time - cg_time	1.3776400E-6	4.2906421E-6	4.2906421E-7	5.2628351E-7	2.2289965E-6	3.211	99	<.001	.002
Pair 2	ctz_time - cg_time	-1.4704300E-6	4.3703290E-6	4.3703290E-7	-2.3375981E-6	-6.0326192E-7	-3.365	99	<.001	.001
Pair 3	vlv_time - ctz_time	2.8480700E-6	3.7674331E-6	3.7674331E-7	2.1005295E-6	3.5956105E-6	7.560	99	<.001	<.001

Figure 4: Paired Sample T-Tests for the Three Groups

- VLV group and the CTZ group

As can be seen by the following box-plot, the variance for each group is fairly low.

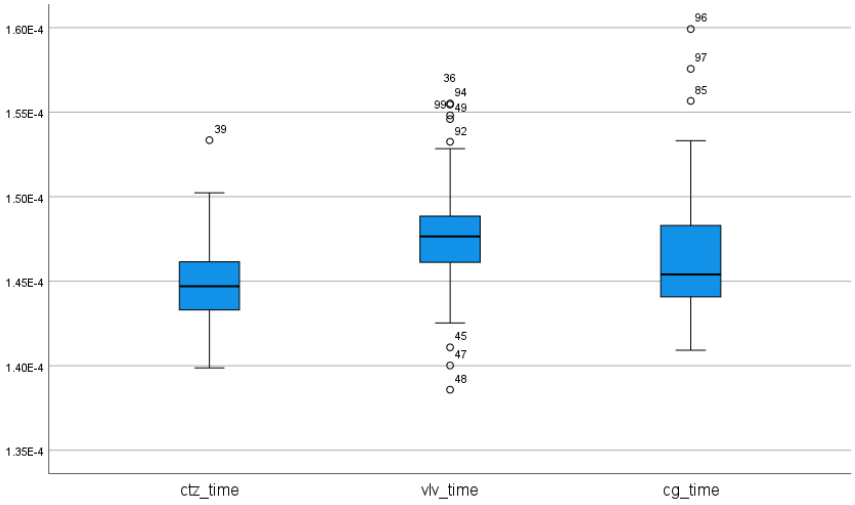

Figure 5: Box Plot for All Three Groups

Sources of Error

There are two main sources of error in this research:

1. Run to run variance due to how and when the operating system decides to run the program
2. Floating point inaccuracy

Point #1 is simply a fact of running programs on modern operating systems. All that can be done and was done is to increase the number of runs per setting to reduce variance as much as possible. Point #2 is also inevitable outside of implementing something like binary coded decimal.

Hypothesis Testing

- H_0 = There is no significant overall increase in runtime as the mutation rates become larger or exceedingly marginal
- H_1 There is a significant overall increase in runtime as the mutation rates become larger or exceedingly marginal

Assuming a 95% confidence interval, as shown in figure 4, the two sided P-values for the VLV group and the control (0.002), CTZ group and the control (0.001) and VLV group and the CTZ (< 0.001) do not exceed 5% and thus the null hypothesis cannot be rejected.

Summary

The key takeaways from the results are the following:

- The VLV (0.000148s), CTZ (0.000145s) and control (0.000146s) groups all have have average runtimes within the margin of error
- The variance within and among the groups is very low

Discussion and Future Work

We will quickly recapitulate the research aims and questions and hypothesis:

Research Aims: Given the lack of research into how different values in the substitution rate matrix for the GTR model affect its runtime, this study will aim to identify if values that are exceedingly marginal, those approaching zero, or exceedingly large values in the substitution rate matrix affect runtime in the aforementioned model.

Research Questions:

1. Do exceedingly small values in the substitution matrix for the GTR runtime?
2. Do exceedingly large values in the substitution matrix for the GTR runtime increase?
3. Is there a significant difference between the runtimes of exceedingly small and large values?

Hypothesis: We hypothesize an overall increase in runtime as the mutation rates become larger or exceedingly marginal.

Key Findings

The data opposes the theory that an overall increase in runtime as mutation rates become large or exceedingly marginal due to:

- The VLV (0.000148 s), CTZ (0.000145 s) and control (0.000146 s) groups all have have average runtimes within the margin of error

- The variance withing and among the groups is very low

As shown in figure 3 suggest, there is not a strong relationship between extremeness of value and runtime relative to the control group. All groups behaved similarly, if not practically identically. This fact is bolstered by the averages in tables 1, 2 and 3. All three groups have very similar averages and variances, demonstrating that the values are consistent. This conclusion is further exemplified by figures 4 and 5. Given a 95% confidence interval, as shown in figure 4, the two sided P-values for the VLV group and the control (0.002), CTZ group and the control (0.001) and VLV group and the CTZ (< 0.001) do not exceed 5% and thus the null hypothesis cannot be rejected.

The reasons for these results are:

- Given that Seq-Gen takes a numerical vector of floating point values, the particular data-type of each value in the vector must be consistent
- The values in the substitution rate matrix simply act as scalars inside the program and are not part of any reposition structure

That is to say, 16 bit float is always processed the same way by the CPU, irrespective of its values. As a consequence of this, runtime should not and does not significantly change if values in the substitution vector change, be they large or small.

These results clearly answer the research questions in this study:

- Exceedingly small values in the substitution vector do not increase runtime overall
- Exceedingly large values in the substitution vector do not increase runtime overall
- There is not significant difference in between exceedingly large and exceedingly small values

Research Limitations

This research has a couple of key limitations:

1. It is impossible to check the runtime of all possible combinations of the substitution rate vector due to the nature of real numbers
2. It is impossible to check the runtime of all possible combinations of the base frequency rate vector due to the nature of real numbers
3. It is impossible to check the runtime of all possible combinations of the previous items together due to both time, knowledge and budgetary constraints
4. Floating point error and the finite nature of memory make absolute precision near impossible
5. The fairly small sizes of the data sets may be unrepresentative in real research

Recommendations

Given that this research has shown that there is no statistical difference in runtime for the performance of the GTR model. Phylogenetic inference can be performed without fear of incurring performance penalties due to values in the substitution rate matrix. This would benefit sectors such as epidemiology because fast phylogenetic inference is crucial in the classification and thus research of diseases such as COVID-19.

Further research can be done into other parameters of the GTR model and even outside of that specific model. Moreover, research into how the lengths of the generated sequences affect runtime may bear more fruitful results.

Summary

The key takeaways from this chapter are:

- There is no significant effect on runtime in between exceedingly large, exceedingly small or average values in the substitution rate matrix

- Due to the small sample size. These results may differ for different combinations of values. However, that seems unlikely due to software and hardware constraints such as how data-types are processed
- These findings can help fields such as epidemiology to require fast phylogenetic inference

Conclusion

We were successfully able to assess whether exceedingly large or exceedingly small values in the substitution rate matrix of the Generalized Time Reversible Model affected runtime performance. Our hypothesis that runtime was going to be affected by such aforementioned values was disproven. We have a couple of suggestions regarding how to improve the experimental design:

- The use of larger sample sizes and of binary coded decimal for more precise measurement is crucial for improving accuracy
- The use of a larger sample space is needed to assess larger overall trends would lend credence to the results in this research

We hope that this research helped bring insight into a minute yet important subject in the field computational phylogenetics.

References

Andrew Rambaut. (n.d.). *rambaut/Seq-Gen: Sequence simulator.* Retrieved 2022-06-28, from https://github.com/rambaut/Seq-Gen

aramis matos. (2022, July). *aramis-matos/seq-gen-script.* Retrieved 2022-07-03, from https://github.com/aramis-matos/seq-gen-script (original-date: 2022-06-30T23:03:57Z)

Felsenstein, J. (2004). *Inferring phylogenies*. Sunderland, Mass: Sinauer Associates.

Fletcher, W., & Yang, Z. (2009, August). INDELible: A Flexible Simulator of Biological Sequence Evolution. *Molecular Biology and Evolution*, *26*(8), 1879–1888. Retrieved 2022-06-14, from https://academic.oup.com/mbe/article-lookup/doi/10.1093/molbev/msp098 doi: 10.1093/molbev/msp098

Hein, J., Schierup, M. H., & Wiuf, C. (2005). *Gene genealogies, variation and evolution: a primer in coalescent theory*. Oxford ; New York: Oxford University Press.

Jukes, T. H., Cantor, C. R., et al. (1969). Evolution of protein molecules. *Mammalian protein metabolism*, *3*, 21–132.

Macroevolution. (n.d.). Retrieved 2022-07-12, from https://theoriginofspeciesbio3u.weebly.com/macroevolution.html

Phylogenetic Trees and Geologic Time | Organismal Biology. (n.d.). Retrieved 2022-07-12, from https://organismalbio.biosci.gatech.edu/biodiversity/phylogenetic-trees

Stamatakis, A. (2014, May). RAxML version 8: a tool for phylogenetic analysis and post-analysis of large phylogenies. *Bioinformatics*, *30*(9), 1312–1313. Retrieved 2022-06-14, from https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioi
doi: 10.1093/bioinformatics/btu033

Tavare, S. (1986). Some probabilistic and statistical problems in the analysis of DNA sequences. *Providence, R.I. American Mathematical Society, c1986.*, 57–86.

List of Tables

Acronyms

describe_thingy.py Data Analysis Script 16, 17

seq_gen_script.hs Runtime Calculation Script 13, 17

sgt.bash Seq-Gen Automation Script 11, 12, 17

CPT Comparison of Physical Traits 5, 6

CTZ Close to Zero Values 10, 11, 14–24, 30

GD Genetic Distance 5, 6

GTR General Time Reversible 4, 6–9, 12, 18, 23, 25

JC69 Jukes and Cantor 1969 6

VLV Very Large Values 10, 11, 14–24, 30