Title
A Study on Correlations between Genes' Functions and Evolutions

Author
Bejraburnin, Natth

Publication Date
2015

Peer reviewed|Thesis/dissertation

A Study on Correlations between Genes' Functions and Evolutions

by

Natth Bejraburnin

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Mathematics

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Lior Pachter, Chair
Professor James W. Pitman
Professor Brent D. Mishler

Fall 2015

A Study on Correlations between Genes' Functions and Evolutions

Abstract

A Study on Correlations between Genes' Functions and Evolutions

by

Natth Bejraburnin

Doctor of Philosophy in Mathematics

University of California, Berkeley

Professor Lior Pachter, Chair

Genes are functional units in organisms' genomes that are believed to play an important role in how organisms develop their inheritable characteristics such as eye and hair colors. One of the fundamental problems in computational biology is to predict gene functions from DNA sequence data. Researchers have developed several methods, both experimental and computational, to tackle the problem and some of those methods rely on an assumption that genes that have similar functions would likely evolve in a correlated fashion and vice versa. I aim to investigate this assumption under a statistical framework.

I define a measure that quantifies the dissimilarity level of the evolutions of any pair of genes. In order to properly define the measure, I use an evolutionary model to represent the evolution of each gene. The model essentially serves as an encoding of the distribution of all possible character sequences that could be observed at the leaf nodes of the model. Then the measure between any two evolutionary models is precisely defined as the Kullback-Leibler divergence between the two distributions, encoded by the models. Since computing the exact measures are not computationally tractable, I instead propose an efficient algorithm for estimating them. Genes' functions are determined based on the Gene Ontology consortium database. In the end, I apply statistical tests for clustering to verify if genes with correlated evolutions tend to have similar functions. I find that the hypothesis is not always true. There are some groups of genes whose functions are not correlated with their evolutions and there are some other groups of genes whose functions and evolutions are correlated well.

In addition, the methods presented in this research lay out a framework for any studies that involve quantitative analysis on genes' or proteins' evolutions. This thesis exhibits one application in this framework, which focuses solely on genes' functions. But the methods can be applied to other type of attributes such as proteins' structures.

To my family.

Contents

List of Figures

List of Tables

Acknowledgments

I would like to first thank my advisor, Professor Lior Pachter, for his continuous support throughout my graduate career at UC Berkeley. It has been a privilege to be part of his group and get to learn about computational biology as well as how to be a good scientist. I especially appreciate his flexibility and encouragement for his students to explore interests and career opportunities.

I am so grateful to my dissertation committee, Professor Jim Pitman and Professor Brent Mishler, who have spent time reading my drafts and providing feedback, and to my qualifying exam committee for getting me prepared for the following steps in my Ph.D. career.

It's been a pleasure getting to learn and grow as a scientist with members in the Pachter lab. The weekly meetings we had, conferences we attended and classes we took together were more lively with great friends around. I appreciate everyone of them who have been part of making these experiences memorable.

My gratitude goes to my friends in Berkeley, who have been companions alongside me through the ups and downs in these past years. I am especially grateful for Bhikkhu Tanachai Acinjano and Summa who have influenced me through inspirational conversations and practices that have changed my thoughts about life forever. Parichart and Thanabordi also deserve my thanks for offering to proofread my dissertation draft; so do #onebreathatatime group members who had made my last few months in Berkeley more meaningful than usual.

I am also thankful to all my students in the sections for which I used to GSI. They have helped me a great deal to become a better teacher and to develop various skills, such as communication and people management, proven useful in many occasions in my life.

Most importantly, words are powerless to express my wholehearted gratitude to my parents, without whom I would not have had a chance to be in this world, to be writing this dissertation, or to stand where I am today. They keep teaching me a life lesson through their own practice that true happiness can be earned from giving to others. They are the ones who are most happy about every of my achievements and the ones who are most saddened by every failure I made. Their unwavering support throughout my time in the US are beyond anyone could imagine. I am forever indebted to them.

Chapter 1

Gene Evolutions and Their Representations

In 1859, Charles Darwin used a tree model to describe his theory of evolution in the book *On the Origin of Species* [12], which originated the idea that species, both extant and extinct, evolve through time in a tree-like fashion. This idea is widely considered as a foundation of modern evolutionary theories and later studies support that it is also applicable to other *operational taxonomic units*. In particular, genes, which are small functional units in organisms' genomes, are believed to go through a tree-like evolutionary process. There are a number of factors that could influence their evolution histories and one of them is arguably their functions. This thesis aims to apply a statistical analysis to investigate whether the genes' evolution histories have some correlation with their functions.

In order to approach the problem in a statistical framework, there is a need for a proper representation of a gene's evolution as well as a well-defined set of gene functions. One requirement for choosing such a representation is that it must allow for having a quantitative measure of the dissimilarity or distance between a pair of genes' evolutions. Using an evolutionary tree model to represent a gene's evolution is a reasonable choice for this purpose. The rest of this chapter will explain concepts relevant to gene evolutions as well as the relevant data and the methods for constructing their representations.

1.1 Genes

Genes are broadly defined as small functional units in organisms' genomes that are believed to play important roles in how organisms develop their inheritable characteristics. When organisms reproduce, their genes get transferred to their offsprings, from one generation to the next and so on. So we, as humans or *Homo sapiens*, are expected to possess some genes that would also appear in another species such as chimpanzees, as a result of inheriting the genes from the two species' common ancestors. This type of genes is an example of *homologs*, which mean they are related by descent from common ancestors, though not necessarily from

different species. In fact, some gene in humans' genomes has homologs in 99 other vertebrae genomes as found in the UCSC data used in this thesis. However, there are some genetic events such as mutations that can alter some genes during their evolution. This results in variations of homologous genes that appear in different genomes, and in some cases, these variations are useful for inferring the evolution history of the genes.

There are two types of homologous genes – paralogs and orthologs. Paralogs are genes related through a gene duplication event within one species while orthologs are genes related through a speciation event within different species [3]. This property makes orthologous genes useful for inferring their evolution history. In the next sections, I will describe the methods used to reconstruct an evolution tree of orthologous genes from multiple sequence alignments, which are assumed to be the alignment of orthologous genes' DNA sequences. Moreover, orthologous genes often have the same functions [3] but I will make the assumption for this experiment that this is always true.

It is worth a clarification that, in the context of this thesis, *a gene* usually refers to a collection of orthologous genes within different species as given by the UCSC's multiple sequence alignment data used in this project. A gene's function refers to a common function of all the orthologous genes in the collection. It is well-defined because of the assumption that orthologous genes always have the same functions. Lastly, the evolution of a gene refers to the tree-like evolution history that describes ancestral relations among the gene's orthologs within different species.

1.2 Phylogenetic trees and evolutionary models

Phylogenetic trees

A phylogenetic tree is a binary tree whose nodes represents *operational taxonomic units* (OTUs), such as species and genes, and whose edges describe their evolutionary relations. The OTUs at the tip or leaf nodes are often extant species whose relevant data can be directly observed. Edge lengths are optionally used to determine some forms of evolutionary distances, such as mutation rates and evolution times. A phylogenetic tree may be rooted, which means one of its nodes is designated as the root, or it may be unrooted. But in a rooted tree, one can infer ancestral relations between any two adjacent nodes, i.e. the one closer to the root is an ancestor of the one farther down. The notion of *the most recent common ancestor* between any two nodes can be defined as the lowest interior node that is an ancestor of both of them. This term will be referred to when I describe methodology later in the thesis.

A phylogenetic tree is an important component in an evolutionary model, which will be used to represent a gene's evolution. In fact, when the Jukes-Cantor model is assumed, an evolutionary model and a phylogenetic tree are interchangeable terms as one can be uniquely derived from the other. The problem of phylogenetic tree reconstruction has still been an active area for research nowadays. There are many different methods as well as different

types of data, including but not limited to genetic and morphological data, available for phylogenetic tree reconstructions. However, there is no single method that is constituted as the standard in the literature. I will need to choose one that is suitable for the scope and requirements of this project and need to assume that it produces reasonably accurate trees. Taking into consideration the details of this project, I decide to approach the problem using the neighbor-joining algorithm and DNA sequence data for the reasons to be discussed in section 1.3.

Evolutionary models

An evolutionary model, or more accurately a nucleotide substitution model, describes a process of DNA substitution at a single locus in a set of orthologous genes of interest. The bases at this locus in the genomes are believed to descend from the same origin nucleotide and go through a series of branching and substitution events during the course of evolution until they become observable in the extant species. The multiple sequence alignment data, in fact, contain DNA sequences from many loci in the same gene, and it is quite common to assume that the locus or the site does not affect the evolution. Therefore the alignments at different loci are treated as independently and identically-distributed samples (from the same distribution defined by the evolutionary model). So, in this research, these i.i.d. samples are used to infer the model by the methods discussed later. It is worth noting that, in order for the model to work, it is important to assume that the multiple sequence alignment data are accurate.

An evolutionary model is a hidden-Markov tree model, comprised of a phylogenetic tree, a set of parameters specifying a mutation rate along each branch of the tree, and a distribution of the bases at the root node (or the stationary distribution of the Markov process). The tree is assumed to be semi-labeled; only the leaf nodes are labeled with extant species (or OTUs), whose DNA could be observed. The interior nodes represent some ancestral taxonomic units, whose DNA could not be directly observed, and are thus considered as hidden in the model. The model parameters generally take the form of a rate matrix $Q = [q_{ij}], i, j \in \Sigma$ where Σ is the set of all possible states of a node, such as {A, C, G, T}, when the model takes on the DNA sequence data. The number q_{ij} determines the rate of change from state i to j and Q needs to satisfy the following properties

$$\begin{aligned} q_{ij} &\geq 0, && \text{for } i \neq j, \\ q_{ii} &< 0 && \text{for all } i \in \Sigma, \\ \sum_{j \in \Sigma} q_{ij} &= 0, && \text{for all } i \in \Sigma. \end{aligned}$$

Note that some models may have a separate rate matrix for each branch while some may use the same matrix for all the branches.

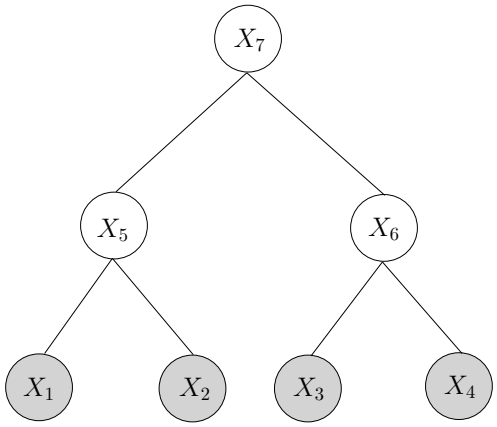

Figure 1.1: A sample tree model whose root node is X_7 and the leaf nodes are shaded in gray, signifying that their values are observed.

From the rate matrix Q, one can compute the transition matrix

$$\theta(t) = e^{Qt} = \sum_{k=0}^{\infty} \frac{1}{k!} Q^k t^k,$$

whose entry $\theta_{ij}(t)$ denotes the probability that state i will change to state j in a time interval of length t. These quantities will be more useful for computations related to evolutionary models later on.

A statistical evolutionary model is mainly used to represent a family of distributions on the set of all possible states of the leaf nodes in the tree. Assuming that the model uses the same rate matrix Q for all the branches, the tree topology is defined as in Figure 1.1, and the root distribution is $\{\pi_{\mathrm{A}}, \pi_{\mathrm{C}}, \pi_{\mathrm{G}}, \pi_{\mathrm{T}}\}$, the probability of observing an outcome at the leaf nodes can be computed using the following identity,

$$P(X_1 = \mathrm{A}, X_2 = \mathrm{G}, X_3 = \mathrm{G}, X_4 = \mathrm{A}) = \sum_{x_5, x_6, x_7 \in \Sigma} P(X_1 = \mathrm{A}, X_2 = \mathrm{G}, X_3 = \mathrm{G}, X_4 = \mathrm{A}, X_5 = x_5, X_6 = x_6, X_7 = x_7).$$

Each of the term in the sum on the right can be computed based on the fully-observed Markov tree model, which is assumed to possess the Markov property. This means that, given the state of an interior node X, all of its children nodes are conditionally independent from each other. In other words, the nucleotide substitution event at a node depends only on the current state of its parent node and nothing else, given that the state of the parent node is known. Therefore, each term can be recursively factorized to

$$P(x_7)P(x_5 \mid x_7)P(x_6 \mid x_7)P(X_1 = \mathrm{A} \mid x_5)P(X_2 = \mathrm{G} \mid x_5)P(X_3 = \mathrm{G} \mid x_6)P(X_4 = \mathrm{A} \mid x_6).$$

Then each of the probability terms above is substituted by an appropriate value from the given root distribution and transition matrix as follows:

$$P(x_7) = \pi_{x_7}, \quad P(X_i = x_i \mid X_j = x_j) = \theta_{x_j x_i}(t_{ij}),$$

where t_{ij} is the branch length between the node i and node j and $\theta_{x_j x_i}(t_{ij})$ is the transition probability from state x_j to x_i in a time interval t_{ij}. After computing all of such probabilities, one sums over all possible states of (x_5, x_6, x_7) to get the marginal probability of the leaf nodes. It is worth noting that there are $|\Sigma|^3 = 64$ terms in the sum and this number in fact grows exponentially with the size of the tree, making the computations non-trivial when the underlying trees are large.

As discussed above, an evolutionary model can be defined by 3 components – a phylogenetic tree, a rate matrix, and a stationary distribution. Apart from the tree itself, the rate matrix for DNA bases has 12 free parameters, while the stationary distribution has 3 free parameters. However, the time-reversibility is generally assumed in a reasonable evolutionary model, in which case only 9 free parameters remain as in the most generalized version of a reversible model defined by Tavaré [28]. On the other extreme, the simplest time-reversible model is the Jukes-Cantor model (JC69) [16], which allows for only 1 free parameter. There are models in between the two extremes that relax some of the assumptions in JC69 to better align with empirical data. For more details on the hierarchy of evolutionary models, please refer to [20, page 153-155]. While it is preferable to use more general models in order to achieve more accurate results, there is an obvious limitation by the computation complexity, which makes it impractical to use some of the more generalized evolutionary models. Therefore, the Jukes-Cantor model is exclusively used in this dissertation, which significantly helps to improve the computation complexity and makes it feasible to handle a large amount of data.

The Jukes-Cantor model

This model, proposed by Jukes and Cantor [16], assumes uniform root distribution (i.e. $\pi_{\mathrm{A}} = \pi_{\mathrm{C}} = \pi_{\mathrm{G}} = \pi_{\mathrm{T}} = 0.25$) and equal transition probabilities among the bases. More precisely, the rate matrix is given by

$$Q = \begin{pmatrix} -3\alpha & \alpha & \alpha & \alpha \\ \alpha & -3\alpha & \alpha & \alpha \\ \alpha & \alpha & -3\alpha & \alpha \\ \alpha & \alpha & \alpha & -3\alpha \end{pmatrix}, \quad \alpha > 0.$$

α is proportional to the overall mutation rate and it can be estimated from the data.

Then one can easily compute the transition or substitution matrix as follows:

$$Q^k = \frac{1}{4}(-4\alpha)^k \begin{pmatrix} 3 & -1 & -1 & -1 \\ -1 & 3 & -1 & -1 \\ -1 & -1 & 3 & -1 \\ -1 & -1 & -1 & 3 \end{pmatrix}, \text{ for } k > 0,$$

and therefore

$$\begin{aligned} \theta(t) = e^{Qt} &= \sum_{k=0}^{\infty} \frac{1}{k!} Q^k t^k = I + \sum_{k=1}^{\infty} \frac{1}{k!} Q^k t^k \\ &= I + \frac{1}{4} \begin{pmatrix} 3 & -1 & -1 & -1 \\ -1 & 3 & -1 & -1 \\ -1 & -1 & 3 & -1 \\ -1 & -1 & -1 & 3 \end{pmatrix} \times \left(\sum_{k=1}^{\infty} \frac{(-4\alpha t)^k}{k!} \right) \\ &= I + \frac{1}{4} \begin{pmatrix} 3 & -1 & -1 & -1 \\ -1 & 3 & -1 & -1 \\ -1 & -1 & 3 & -1 \\ -1 & -1 & -1 & 3 \end{pmatrix} \times \left(e^{-4\alpha t} - 1 \right) \\ &= \frac{1}{4} \begin{pmatrix} 1 + 3e^{-4\alpha t} & 1 - e^{-4\alpha t} & 1 - e^{-4\alpha t} & 1 - e^{-4\alpha t} \\ 1 - e^{-4\alpha t} & 1 + 3e^{-4\alpha t} & 1 - e^{-4\alpha t} & 1 - e^{-4\alpha t} \\ 1 - e^{-4\alpha t} & 1 - e^{-4\alpha t} & 1 + 3e^{-4\alpha t} & 1 - e^{-4\alpha t} \\ 1 - e^{-4\alpha t} & 1 - e^{-4\alpha t} & 1 - e^{-4\alpha t} & 1 + 3e^{-4\alpha t} \end{pmatrix}. \end{aligned}$$

The parameter t denotes the evolutionary time, which may vary from branch to branch in a tree depending on how far apart the two species nodes incident to the edge/branch are. The evolutionary time is in fact proportional to the *branch length*, $3\alpha t$, or the expected number of mutations per site that occur during the evolution from the parent species to the child. When t is very small, the substitution matrix is close to the identity matrix, which means the bases in any two adjacent nodes are likely the same. On the other hand, when t approaches ∞, the matrix becomes close to $\frac{1}{4} \times \mathbf{1}$ where $\mathbf{1}$ is the all-one matrix. This means that any two adjacent nodes are almost independent where the state of one node does not give us any information for the state of its adjacent node.

Moreover, the off-diagonal entries are all the same, given that t is fixed, which means the transition probabilities are equal regardless of the actual transition states. The same is also true for the diagonal entries which represent the transition probabilities with the same original and final states. Moreover, the Chapman-Komolgorov equation [20, page 148] states that

$$\theta(t)\theta(s) = \theta(t+s), \quad t, s > 0. \tag{1.1}$$

This is a useful property as it allows us to compute the transition probabilities between any two nodes by first computing the total branch length l of the unique path between them

and then computing $\theta(l)$. It saves us from having to compute the sum over exponentially many terms corresponding to all possible states of the intermediate nodes on that path. Lastly, one can verify that the marginal probabilities on any single node X_i are also uniform. Let X_r denote the root node, l the length of the path between X_i and X_r. Consider that

$$\begin{aligned} P(X_i = x_i) &= \sum_{x_r \in \Sigma} P(X_i = x_i, X_r = x_r) \\ &= \sum_{x_r \in \Sigma} P(X_i = x_i \mid X_r = x_r) P(X_r = x_r) \\ &= \frac{1}{4} \left(\theta_{Ax_i}(l) + \theta_{Cx_i}(l) + \theta_{Gx_i}(l) + \theta_{Tx_i}(l) \right) = \frac{1}{4}. \end{aligned}$$

1.3 Gene trees

As previously mentioned in section 1.2, the nodes of a phylogenetic tree generally represent operational taxonomic units or OTUs, which could be, for example, species or genes. When the OTUs are genes (species), the corresponding tree is referred to as a *gene tree* (a *species tree*, respectively). In this dissertation, the study focuses exclusively on gene trees and the term *phylogenetic tree* will be used to refer to a gene tree, unless explicitly stated otherwise.

In fact, some gene trees may be discordant with the species tree [27] due to the difference in the types of events that cause branching points in a gene tree and a species tree. In a species tree, a branching point corresponds to a speciation event that produces new biological species. However, in a gene tree, a branching point occurs mainly due to point mutations in DNA sequences, which can be described by an evolutionary model.

In addition, the gene trees themselves, which are inferred from different genes, are likely to differ from each other, even though they are reconstructed from the same set of taxa. This is because point mutations play an important role in the branching patterns of those trees. Additionally, point mutations occur at individual sites in the genomes and they could occur at sites in different genic regions at different times, which in turn cause genes to have their own evolution histories. Therefore, it is reasonable to infer the gene tree for each gene separately.

In this experiment, I use the UCSC Conservation Track dataset, which contains multiple sequence alignments (MSA) of 21520 genes from 100 vertebrae genomes (see Appendix A for more details). The multiple alignments are generated using MultiZ program [6], which basically runs a dynamic programming to find the optimal MSAs from pairwise alignments between each species's genome and the human genome. However, only 21488 of them have sufficient data for gene tree reconstructions by the methods `njs`, `dist.dna` in the R package `ape`. For genes with insufficient data, it could be because they contain some sequences with too many gaps ("-") that some pairs of them do not have bases aligned at all. In that case, the evolutionary distances could not be estimated, and therefore the neighbor-joining algorithm could not reconstruct gene trees due to the missing data. I decide to exclude these 32 genes from the experiment altogether.

1.4 Gene tree reconstruction from multiple sequence alignments

The phylogenetic tree reconstruction problem aims to infer evolutionary relations among a set of OTUs, which, in this case, are homologous genes within different species. The inferred tree will describe the evolutionary history of the gene, which is largely influenced by point mutations. Once all such gene trees are reconstructed, I can quantitatively compare them and see how different they are based on the distance measure I propose in Chapter 2. This problem has been widely studied and there are several methods available with varying assumptions and limitations. They can be classified into three main approaches, which are parsimony, maximum likelihood and distance-based.

The parsimony approach (or character-based) assigns characters to the leaf nodes (or taxa) of a tree where the characters, in this case, would come from a column of the multiple sequence alignment. Then it tries to find the tree topology which can explain the data with fewest mutations possible, when taking all the sites into account. Such a tree is known as *the most parsimonious tree* and the method itself is referred to as the maximum parsimony method. However, this method is known to be statistically inconsistent in some cases [15], which means it would fail to produce the true tree even when given an infinite amount of data.

The second approach uses a statistical framework to find the maximum likelihood tree given the observed multiple sequence alignments. Each column in the alignment is considered as an independent sample from the distribution determined by an underlying evolutionary model as described in section 1.2. However, the computation could be very demanding due to the enormous size of the candidate tree space, which allows variations in both tree topologies and individual branch lengths. Although some alternative approach exists, such as Markov Chain Monte Carlo (MCMC), it is still infeasible for this project which aims to include more than 20,000 genes in the analysis.

Lastly, the distance-based approach reconstructs the tree from a matrix of pairwise evolutionary distances between the taxa, which may be computed from the alignment. Once the matrix is generated, it tries to find the tree whose tree metric is closest to the distance matrix based on some criterion such as the ordinary least square (OLS) and balanced minimum evolution (BME). One of the most popular methods in this category is neighbor-joining, which greedily optimizes the BME criterion [13]. Although it does not have a solid statistical foundation as much as the likelihood approach does, it generally runs much faster and produces good results. So I decide to use this approach in this thesis to handle a large number of phylogenetic tree reconstructions.

To infer the phylogenetic tree of each gene, I use the UCSC multiple sequence alignments of homologs from 100 vertebrate species. The data are taken from the UCSC database [29], which contains multi-alignments of 21520 genes in total. For each gene, there are two steps in inferring the gene tree from the given multi-alignment data. Firstly, I estimate the pairwise evolutionary distances between species for this particular gene and secondly, I use

the neighbor-joining algorithm to reconstruct a tree from such distances. These evolutionary distances will in turn determine the branch lengths, or equivalently, the mutation rate along each branch in the tree.

Computing evolutionary distances between taxa

I assume the evolutions follow the Jukes-Cantor model and to estimate the distance between two species' nodes (X_i, X_j), I use the maximum likelihood method. Let the model be parameterized by α and π, where

$$P(x_i|x_j) = \begin{cases} \alpha & \text{if } x_i = x_j \\ \pi & \text{if } x_i \neq x_j \end{cases}$$

It necessarily follows that $\alpha + 3\pi = 1$. Also, let the observed DNA sequences corresponding to the two species be $\mathbf{a}^i = a_1^i a_2^i \dots a_L^i$ and $\mathbf{a}^j = a_1^j a_2^j \dots a_L^j$, where L is the length of the sequences. Then

$$\begin{aligned} P(\mathbf{a}^i, \mathbf{a}^j \mid \alpha, \pi) &= \prod_{l=1}^{L} P(a_l^i, a_l^j \mid \alpha, \pi) \\ &= \prod_{l=1}^{L} P(x_j = a_l^j \mid \alpha, \pi) P(x_i = a_l^i \mid x_j = a_l^j, \alpha, \pi) \\ &= \prod_{l=1}^{L} \frac{1}{4}\, \alpha^{\mathbb{1}(a_l^i = a_l^j)} \pi^{\mathbb{1}(a_l^i \neq a_l^j)} \\ &= \left(\frac{1}{4}\right)^L (\alpha)^{L-D} (\pi)^D \propto (1-3\pi)^{L-D} (\pi)^D, \end{aligned}$$

where D is the number of sites at which the DNA bases between the two species are distinct. Note that the third equation follows from the second because according to the Jukes-Cantor model, the marginal probabilities of all nodes are uniform. Next, I want to find π which maximizes the above likelihood and some calculus reveals that the optimal

$$\pi^* = \frac{D}{3L}.$$

According to the Jukes-Cantor model, the evolutionary distance between the two nodes is given by $3\alpha t = -(3/4) \cdot \log(1 - 4\pi)$, whose maximum likelihood estimator is

$$-\frac{3}{4} \cdot \log\left(1 - \frac{4}{3}\frac{D}{L}\right). \tag{1.2}$$

I use the R implementation of this algorithm that is available in the `ape` package [22].

The second step is to infer the phylogenetic tree based on the pairwise distances between species. I will use the well-known neighbor-joining algorithm by Saitou and Nei [25], which constructs the tree by recursively merging two leaf clusters/nodes that are believed to be a *cherry* in the tree. For completeness, I briefly state the algorithm here.

Neighbor-Joining algorithm

Given the set of n nodes $\{1, 2, 3, \ldots, n\}$ and the distance matrix $[d(i,j)]_{ij}$, the algorithm will construct a (trivalent) phylogenetic tree whose leaf nodes are exactly the given n nodes. Let V be the set of the current active nodes (some of which represents a cluster of nodes) that need to be included in the tree. Let $S(i) = \sum_{k \neq i} d(i,k)$.

1. Set $V = \{1, 2, 3, \ldots, n\}$.
2. Compute the Q-criterion
$$Q_d(i,j) = (|V| - 2) \cdot d(i,j) - S(i) - S(j), \text{ for all } i, j \in V.$$
3. Find (i^*, j^*) which minimizes $Q_d(i,j)$.
4. Join the two nodes i^*, j^* at a new (internal) node u and set the branch lengths as follow.
   a) $d(i^*, u) = \frac{1}{2} d(i^*, j^*) + \frac{1}{2(|V|-2)} [S(i^*) - S(j^*)]$
   b) $d(j^*, u) = \frac{1}{2} d(i^*, j^*) + \frac{1}{2(|V|-2)} [S(j^*) - S(i^*)]$
5. Set $V := V \cup \{u\} \setminus \{i^*, j^*\}$ and update the distances to the new node u as follow.
$$\text{For } k \in V \setminus \{i^*, j^*\},\ d(k,u) = \frac{1}{2} [d(i^*,k) - d(i^*,u)] + \frac{1}{2} [d(j^*,k) - d(j^*,u)] .$$
6. Repeat step 2 - 5 until $|V| = 1$.

This algorithm runs with time complexity of $O(n^2)$ with a proper choice of implementation. It first needs to compute the Q-criteria for all pairs of n nodes, which takes $O(n^2)$. Then it repeats step 2-5 for $n - 1$ rounds while each round involves finding the minimal Q-criteria among all pairs of the active nodes in V, which could be achieved in $O(n)$ by keeping the pairwise Q-criterion table and updating it in step 4 only on the relevant rows while always keeping track of the minimal value. Therefore, the total running time is $O(n^2)$, where n is the size of the tree. This algorithm makes it feasible for a project that requires gene tree reconstructions of around 20,000 genes, the scale of which rules out the possibility of using an MCMC approach such as `MrBayes`.

One of the potential caveats is that the algorithm is known to possibly produce some negative branch lengths in the output tree, while is generally not allowed in evolutionary models. This would happen during step 4 in the algorithm when

$$\frac{1}{|V|-2}\,|S(i^*) - S(j^*)| > d(i^*, j^*),$$

which means the average of the difference between the distances from every node to i^* and j^* has greater magnitude than the distance between i^* and j^* itself. In any tree topology, this would make sense only when the node i^* is closer to every other node than the node j^* is (or vice versa), which could not be realized when the nodes i^*, j^* are hanging away from another node u. I therefore set the node i^* (if $d(i^*, u) < 0$ or j^* if $d(j^*, u) < 0$) to be the same as u, or equivalently set any negative branch length to 0.

It is also worth noting that the neighbor-joining algorithm, in fact, produces unrooted trees while evolutionary models are usually represented by rooted trees, especially in this project. By the Chapman-Komolgorov equation 1.1, the transition probabilities between any two nodes do not depend on which node is the root. This, in turn, implies that the computations are also independent of the root node. So I can make the convention that sets the last node in V to be the root node when reconstructing each gene tree.

Missing species in gene trees

From the multiple alignment data, it can be observed that some genes do not have homologs from all the 100 species; and those gene trees will thus have less than 100 leaf nodes due to the missing species. These missing homologs are likely a result of some fixed *gene loss* events in the past, which have occurred as a part of the natural selection process. However, in order to compare the gene trees using the proposed measure, the trees need to have the same number of leaf nodes. So for a gene with missing homologs, I decide to add independent nodes that correspond to the missing species to its respective gene tree model. These independent (leaf) nodes can be treated as having infinite evolutionary distances from the remaining nodes in the tree. This is a reasonable option as the missing sequences of some species from the multi-alignment of a gene are not some missing data in the usual sense but rather they are simply non-existent and irrelevant to the gene, and cannot be inferred from the existing data. I should therefore treat them as being independent from the other sequences in the alignment. In other word, this means that knowing the existing observed sequences does not give any information on the missing ones and vice versa.

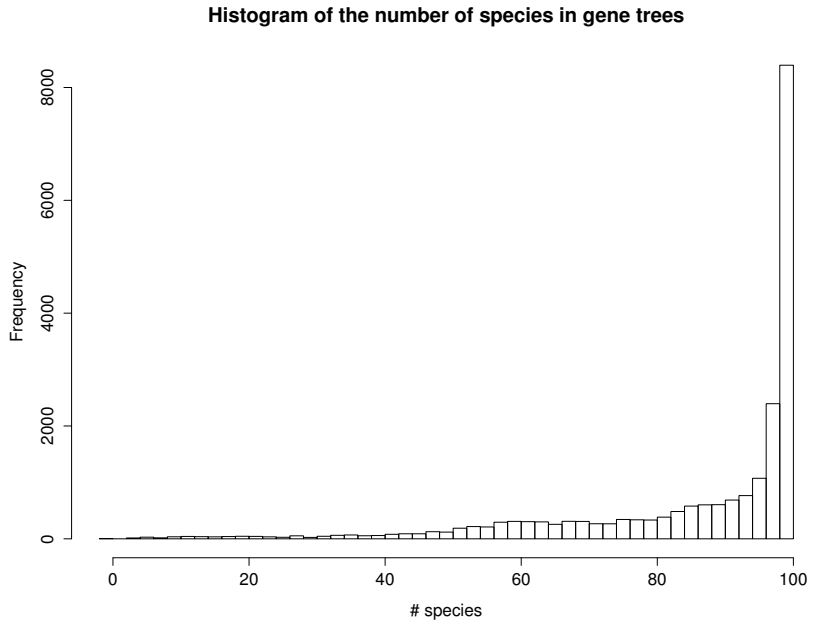

Figure 1.2: The histogram of the number of homologs/species that are present in the gene trees. There are 21488 genes in total, 5285 of which contain all the species.

Chapter 2

A Framework for Quantitative Study of Phylogenetic Trees

Evolutionary tree models are used to represent genes' evolutions. They are reconstructed from genes' multiple alignment data using the Jukes-Cantor model and the neighbor-joining algorithm. This chapter focuses on setting up a framework for quantitative analysis on phylogenetic trees. I will discuss the space of phylogenetic trees that is used as the reference as well as a dissimilarity measure or a distance between phylogenetic trees in that space.

In Kim's space of evolutionary models [17], a gene's tree model defines a distribution of DNA characters observed at the tree's leaf nodes. These characters are assumed to be orthologous bases at an entire locus in the gene. Then for a pair of evolutionary models, there are two of such distributions implicitly defined by the models. I propose to use the Kullback-Leibler divergence (or KL divergence) between the two distributions as a dissimilarity measure between the two models.

However, this definition of the measure introduces a challenge to the computation feasibility. The time required to compute the exact marginal distribution of characters at the leaf nodes of a tree model alone would grow exponentially with the size of the tree, which roughly has 100 leaves. It is therefore impractical to use the exact divergences for further analysis. I instead aim to develop an efficient method for *estimating* the dissimilarity measures among evolutionary trees.

In the rest of this chapter, I will explain a motivation behind the dissimilarity measure and provide details on the methods and heuristics used to approximate the measure between a pair of gene tree models. Lastly, I will briefly explain the multidimensional scaling, which is used to find evolutionary models' representations in $\mathbb{R}^2$. This is the last important step of data preparation for performing cluster analysis in chapter 4.

2.1 Dissimilarity measures on evolutionary models

Phylogenetic trees are used to represent many types of biological relationships such as ancestral relationships between species. They can also represent evolutionary models that underlie the process of nucleotide substitution or mutations in orthologous genes. In order to conduct quantitative analysis on gene's evolutions, a framework, where a *distance* or a dissimilarity measure between phylogenetic trees (or evolutionary tree models) is well-defined, is required.

Robinson and Foulds [24] proposed a metric on phylogenetic trees that defined the distance as the smallest number of contractions and decontractions needed to transform one tree to the other. This method accounts for only the tree topologies and their labels but not the branch lengths, which might not be suitable for comparison of evolutionary models that are partly parameterized by the tree branch lengths.

Later, Billera *et al.* [5] defined the tree space by concatenating a number of Euclidean spaces whose dimensions are determined by the number of the internal edges in the trees. Each of these Euclidean spaces corresponds to a distinct topology of the trees (with a fixed number of leaves) and the coordinates specify the branch length parameters in the trees. Defining the space this way, there is a natural metric (called BHV) which measures the distance between any two trees by taking the length of the geodesic path between them. Although this metric might be useful in many applications, Pachter [19] pointed out some caveats when it was used with evolutionary tree models, especially when their branch lengths were extended to infinity.

Kim [17] defined a space of phylogenetic trees on a fixed number of leaves, or taxa. In this space, each phylogenetic tree is represented by a vector in $\mathbb{R}^{k^n}$ where k is the number of all characters (e.g. 4 for DNA bases or 20 for amino acids) and n is the number of leaves. This vector, which contains k^n real values between 0 and 1, corresponds to a probability distribution of all the k^n possible character patterns that can be observed at the leaves of the tree. The distribution obviously depends on some of the tree's parameters such as a rate matrix and its branch lengths. For each of the possible tree topologies, as these parameters vary over their respective valid ranges, they define a set of distributions or tree points in this space, which are collectively called a manifold. The manifold is embedded in the simplex in $\mathbb{R}^{k^n}$ as the sum of the probabilities need to be one. All of such manifolds (one for each valid topology on the fixed set of taxa) are glued together to make the space of the phylogenetic trees.

Kim's view on the evolutionary tree space aligns well with this research's scope, in which phylogenetic trees are used mainly to encode the distributions over character patterns at their leaf nodes. As Kim does not provide any specific metric in his paper, I propose to use the Kullback-Leibler divergence (KL divergence) or the Jensen-Shannon divergence (JS divergence), which is suitable for measuring the difference between probability distributions. But due to the limit on the algorithm time complexity, only the KL divergence will be used in this thesis.

Kullback-Leibler divergence

The KL divergence is commonly used to measure how much different two probability distributions are. Formally, let P and Q be two discrete probability distributions on the same sample space $\mathcal{S}$ such that P is dominated by Q, or equivalently, $Q(\mathbf{x}) = 0$ implies $P(\mathbf{x}) = 0$ for every $\mathbf{x} \in \mathcal{S}$. Then, the *Kullback-Leibler divergence* (KL divergence) between P and Q is defined as

$$D_{\mathrm{KL}}(P\|Q) = \sum_{\mathbf{x}\in\mathcal{S}} P(\mathbf{x}) \log \frac{P(\mathbf{x})}{Q(\mathbf{x})}.$$

It follows from the definition that the divergence is always non-negative and is 0 only when $P = Q$. Note also that the original definition of the KL divergence is not symmetric, while a reasonable dissimilarity measure between evolutionary models should be. So I will instead use a symmetric version of the divergence, which is

$$D(P\|Q) := \frac{D_{\mathrm{KL}}(P\|Q) + D_{\mathrm{KL}}(Q\|P)}{2}$$

For the rest of this dissertation, I will use the notation $D_{\mathrm{KL}}(P\|Q)$ to denote this symmetric version of the KL divergence.

Additionally, as stated above, the (symmetric) KL divergence $D(P\|Q)$ is only defined when the distribution P is absolutely continuous with respect to Q and vice versa. In this thesis, I compute KL divergences solely between marginal leaf-node distributions of evolutionary tree models. Under the Jukes-Cantor model, all the transition probability terms have exponential forms and are thus strictly positive, and so is the probability of each possible outcome in both distributions. This implies that either distribution is absolutely continuous with respect to the other.

2.2 Approximation of the marginal leaf-node distributions of tree models

In section 1.4, I describe the methods used to infer each gene's phylogenetic tree from the multiple alignment data. Under the Jukes-Cantor model, each phylogenetic tree (also known as a gene tree) implicitly defines an evolutionary model which in turn determines the (marginal) distribution of the character patterns on the tree's leaf nodes. The dissimilarity measure or the distance between any two gene trees is defined as the KL divergence between the marginal distributions of the two trees' leaf nodes. However, given the trees' sizes, the exact values of such marginal distributions cannot be computed, as they require to sum over all the possible states (or character patterns) of the trees' interior nodes and there are exponentially many of them. I instead look for a way to estimate them with a *tree distribution* – a distribution that can be defined by a set of random variables whose

dependence relations are representable by a tree, which I have an efficient algorithm for computing the KL divergence. This algorithm is discussed in chapter 3.

Given a discrete probability distribution, Chow and Liu [10] devise an algorithm that finds the tree distribution which has the least KL divergence to that probability distribution. This tree distribution (or the *Chow-Liu tree*) is thus considered as an optimal first-order dependence approximation to the original distribution. Next, I provide some details of this algorithm as well as state some relevant properties of Chow-Liu trees.

The Chow-Liu algorithm

Let P denote a discrete probability distribution on the random variables $(X_1, X_2, \ldots, X_n) \in \Sigma^n$ and let $\mathcal{T}$ be the set of all tree distributions on $(X_1, X_2, \ldots, X_n)$. The goal is to find the tree distribution

$$Q^* = \arg\max_{Q \in \mathcal{T}} D(P\|Q).$$

1. Compute the mutual information $I(X_i, X_j)$ between every pair of the random variables where
$$I(X_i, X_j) = \sum_{x_i, x_j \in \Sigma} P(x_i, x_j) \log\left(\frac{P(x_i, x_j)}{P(x_i)P(x_j)}\right).$$

2. Construct the complete weighted graph C_n where the n nodes are the n random variables and the edge (i, j) has the weight equal to $I(X_i, X_j)$.

3. Use the Prim's algorithm to find a maximum spanning tree (a subtree of the graph with maximal sum of the edge weights). Note that the original algorithm will find a minimum spanning tree, so in order to achieve the maximum version, I need to multiply all the weights by -1 before applying the algorithm.

4. The resulting maximum spanning tree is the optimal first-order dependence tree, which, together with the original marginal probabilities $P(X_i)$ and $P(X_i, X_j)$, defines a new distribution that approximates the original distribution.

The first step requires computing the sum over all possible values of $(x_i, x_j) \in \Sigma^2$ over $\binom{n}{2}$ pairs of nodes and its time complexity is thus $O(|\Sigma|^2 n^2)$. In step 3, I implement the Prim's algorithm using an adjacency matrix, and its running time is $O(n^2)$. Therefore, the total running time for this algorithm is $O(c^2 n^2)$ where n is the number of nodes and $c = |\Sigma|$.

Some properties of the Chow-Liu trees

In this project, a Chow-Liu tree is exclusively derived from an evolutionary tree model and, as a result, it shares some properties with the original tree model that may be exploited to simplify KL divergence calculations. Precisely, its marginal probabilities $P(x_i)$ are still the

same as the original values and so are the transition probabilities $P(x_i \mid x_j)$ between any two adjacent nodes in the new dependence tree. This implies that the transition probability matrix $[P(x_i \mid x_j)]_{i,j}$ will take the form $\exp(Qt_{ij})$ where Q is the rate matrix of the original evolutionary model and t_{ij} is a parameter proportional to the branch length between the node X_i and X_j. It then follows from the Chapman-Komolgorov equation that the conditional probability $[P(x_i \mid x_k)]_{i,k}$ between any two nodes will also have the form $\exp(Qt_{ik})$ where t_{ik} is again proportional to the distance between the two nodes in the Chow-Liu tree, which could be different from their distance in the original tree.

On a related note, the algorithm also simplifies under the Jukes-Cantor model. Precisely, the mutual information $I(X_i, X_j)$ is in fact decreasing in the length of the unique path between the two nodes. Let l be the path length between the nodes X_i and X_j. Then according to the Jukes-Cantor model, $P(x_i) = P(x_j) = 1/4$ and $P(x_i, x_j) = P(x_i \mid x_j)P(x_j) = f_{(x_i,x_j)}(l) \times (1/4)$ where

$$f_{(x_i,x_j)}(l) = \begin{cases} \frac{1}{4} + \frac{3}{4}e^{-4l/3} & \text{if } x_i = x_j \\ \frac{1}{4} - \frac{1}{4}e^{-4l/3} & \text{if } x_i \neq x_j \end{cases}.$$

Therefore, it follows that

$$I(x_i, x_j) = 4\left(\frac{1}{4}f_{(x_i=x_j)}(l)\log f_{(x_i=x_j)}(l)\right) + 12\left(\frac{1}{4}f_{(x_i\neq x_j)}(l)\log f_{(x_i\neq x_j)}(l)\right),$$

whose first-order derivative is

$$e^{-4l/3} \cdot \log \frac{f_{(x_i\neq x_j)}(l)}{f_{(x_i=x_j)}(l)} < 0.$$

This implies that $I(x_i, x_j)$ is a decreasing function of l. I can thus use the negative pairwise distances between leaf nodes as the weights (as defined in step 2) instead of using the mutual information when finding a maximum spanning tree.

2.3 Approximation of the distance between evolutionary models

Let P and Q be two evolutionary models of two different genes. As discussed in Section 2.1, I define the distance between P and Q as the symmetric KL divergence between their marginal leaf-node distributions, denoted by P_{leaf} and Q_{leaf}, respectively. Since P_{leaf} and Q_{leaf} cannot be computed exactly in practice, as the number of terms in their marginal sums grow exponentially with the tree size, I resort to the Chow-Liu algorithm to estimate them with other tree distributions P_{CL} and Q_{CL} (i.e. the Chow-Liu trees). In other word, I have

$$D(P_{leaf} \| Q_{leaf}) \approx D(P_{CL} \| Q_{CL}),$$

which can be computed in linear-time using the algorithm given in Chapter 3.

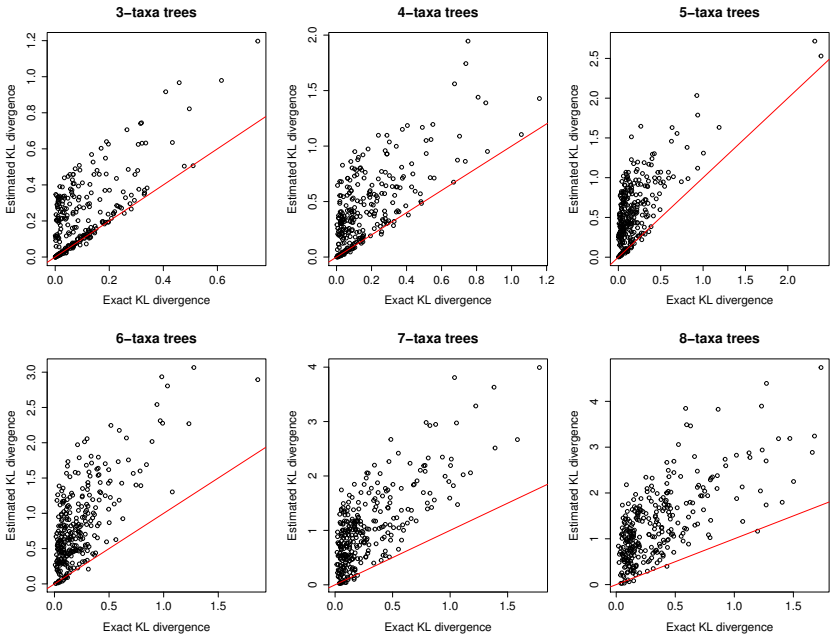

Figure 2.1: Plots of exact KL divergences vs. estimated KL divergences between pairs of randomly-selected gene trees trimmed down to n common taxa where $n = 3, 4, \ldots, 8$. The red straight line is $y = x$.

Analysis on the quality of approximation

To analyze the quality of the Chow-Liu tree approximation, I carry out the following method to see how the estimated values compare with their respective exact values. Note that since it requires an exponential time of the tree size to compute the exact divergences, I can only reasonably run the analysis on small trees with less than or equal 8 taxa.

1. Randomly sample 300 pairs of gene trees.
2. For each pair, randomly choose a common subset of the taxa of size $n = 3$. Then trim both trees down to the subtrees on the 3 taxa.
3. Compute the exact KL divergence between the leaf-node distributions of the two subtrees and its estimated value by the method.
4. Repeat but with the number of taxa replaced by $n = 4, 5, \ldots, 8$, so in total there are 1800 data points for analysis.

Based on the graphs in Figure 2.1, I can see that the estimation method generally overestimates the divergences. The source of such a discrepancy, of course, comes from the Chow-Liu tree estimation step, in which I try to approximate the marginal leaf-node distribution with a

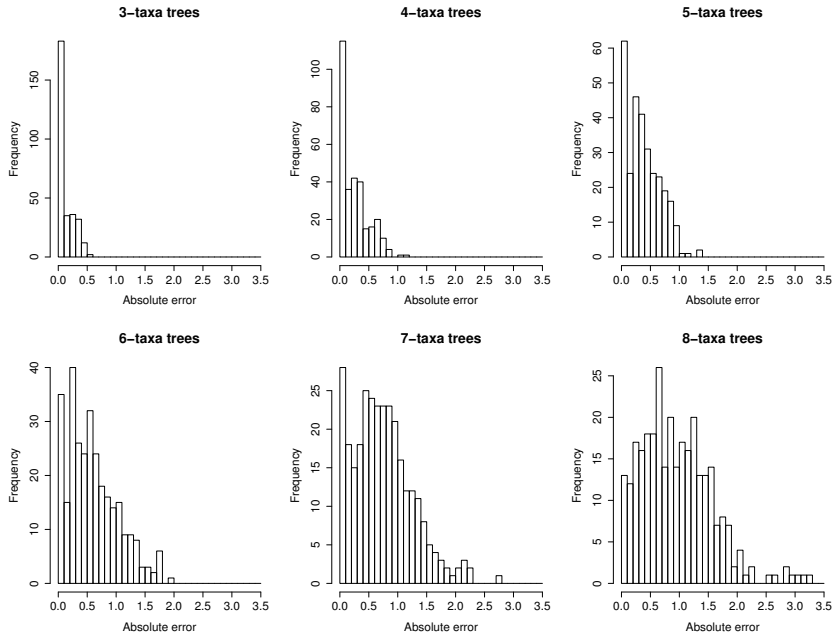

Figure 2.2: Histograms of the absolute errors from estimating (without correction) the KL divergences between pairs of gene trees with varying number of taxa.

first-order dependence distribution (i.e. a tree model). When taking a closer look especially at those points where the exact divergences are small but the estimated ones are large, I find that their original tree models have the same topology and slightly different branch lengths. The exact KL divergences on their leaf-node distributions are thus small. However, when I apply the Chow-Liu approximation, the output trees end up in different topologies and in turn their KL divergences (i.e. the *estimated* KL divergences) are amplified.

Also, from Figure 2.2, it can be observed that the estimation method seems to work better on smaller trees. For small trees with only few taxa, the dependence structure of the leaf nodes are better captured by the Chow-Liu trees than in larger trees. This explains why the algorithm does better on trees with 3 taxa and 4 taxa.

Heuristic correction to the measures

It is not certain of how this result extends to cases of even larger trees, especially those in the main experiment with one hundred taxa. So I decide to pose this as a prediction problem where I try to predict the exact KL divergence from the estimated value using the least-square linear model. Fitting a linear model on the 1800 data points above, I obtain the relation

$$\text{exact} = 0.290854 \times \text{estimated} + 0.006038.$$

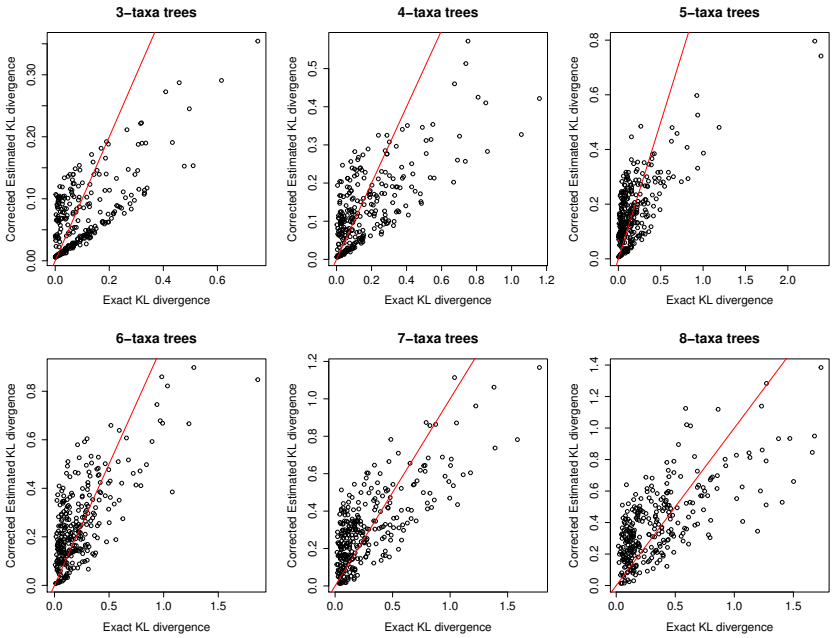

Figure 2.3: Plots of exact KL divergences vs. estimated KL divergences after heuristic corrections between pairs of gene trees with varying number of taxa. The red straight line is $y = x$.

Then I apply this heuristic correction to the divergences obtained from the algorithm to improve the estimates to the divergences. I can see some improvement from the histogram of the errors in Figure 2.4.

Another way that could potentially improve the estimates, though not implemented in this thesis, is to use the Chow-Liu tree approximations only when necessary. Recall the estimation formula that I use in order to avoid the exponential time complexity

$$D(P_{leaf}\|Q_{leaf}) \approx D(P_{CL}\|Q_{CL}).$$

In fact, I can efficiently compute $D(P_{CL}\|Q_{leaf})$ and $D(P_{leaf}\|Q_{CL})$, in linear time to be precise. So I could instead estimate the divergence by

$$D(P_{leaf}\|Q_{leaf}) \approx \frac{1}{2}D(P_{CL}\|Q_{leaf}) + \frac{1}{2}D(P_{leaf}\|Q_{CL}).$$

2.4 MDS of evolutionary tree models

To test for correlations between genes' evolutions and functions, I consider applying a cluster analysis framework to verify whether gene trees (which represent genes' evolutions) cluster naturally in their geometric space around their respective genes' functions. In this space, as

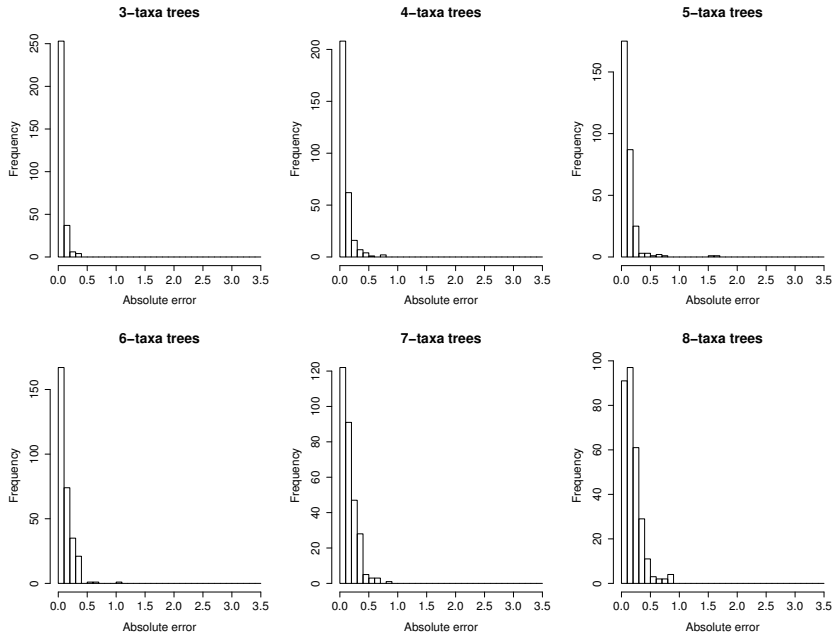

Figure 2.4: Histograms of the absolute errors from estimating the KL divergences (with heuristic correction) between pairs of gene trees with varying number of taxa.

Kim describes, gene trees are represented by real vectors of size 4^n where n is the number of leaf nodes in the trees, or 100 in this experiment. These real vectors in turn determine probability distributions of the character patterns on the trees' leaf nodes. Given all these representations, the concept of centroids is not well-defined in this space, while it is necessary for cluster analysis. I therefore need to find representations of these evolutionary models in the Euclidean $\mathbb{R}^2$ space using the multidimensional scaling (MDS). MDS tries to find points in $\mathbb{R}^2$ that represents the objects of interest while trying to preserve the pairwise distances between the objects. It requires a pairwise-distance matrix, which, in this case, are the KL divergence of the marginal leaf-node distributions between every pair of the gene tree models. Then it finds point coordinates in $\mathbb{R}^2$ that represent the gene tree models, which minimize the sum of the square of the difference between the KL divergence and the Euclidean distance of every pair of the gene tree models. In fact, MDS can find representations in a higher-dimensional space but I observe no significance difference from an experiment that uses $\mathbb{R}^3$ or $\mathbb{R}^4$ in place of $\mathbb{R}^2$. This will not only facilitate the cluster analysis on gene trees, but also improve the time complexity of the computations.

Chapter 3

An Algorithm for Computing KL Divergence of Tree Models

Given the proposed dissimilarity measure between gene evolutions, it is quite a challenge to efficiently carry out the required computations. Some approximation scheme is suggested to make the analysis feasible, given the size of the data set. Specifically, the marginal distribution of a tree model's leaf nodes is first estimated by a Chow-Liu tree, which is itself a tree model although it does not represent any evolutionary process. Then, the dissimilarity measure between two gene evolutions is estimated by the KL divergence between their corresponding Chow-Liu trees. In this chapter, I present an efficient algorithm that computes the KL divergence between Chow-Liu trees, or in fact, any general tree models under some mild conditions.

Details of the algorithm

Let P and Q be two directed tree models with n nodes $X_1, X_2, X_3, \ldots, X_n$ which take values on the character set Σ where $|\Sigma| = c$; and let $X = (X_1, X_2, \ldots, X_n) \in \Sigma^n$. Note that the two trees may have different topologies and I will interchangeably use the symbols P, Q for both the tree models themselves and the distributions of their node values. In the rest of this chapter, I will describe an efficient algorithm for computing the KL divergence

$$D(P\|Q) = \sum_{x \in \Sigma^n} P(X = x) \log \frac{P(X = x)}{Q(X = x)}.$$

Formula simplification

First, the tree distributions $P(X), Q(X)$ can be factorized as follow;

$$P(X) = \prod_{i=1}^{n} P(X_i \mid X_{p(i)}) \qquad Q(X) = \prod_{i=1}^{n} Q(X_i \mid X_{q(i)}),$$

where $p(i), q(i)$ are the parents of the node X_i in the tree P, Q, respectively and if the node X_i is the root, then $P(X_i|X_{p(i)}) := P(X_i)$. Note that $P(x_i \mid x_j)$ is used to denote $P(X_i = x_i \mid X_j = x_j)$ for convenience. Then the divergence formula simplifies to

$$D(P\|Q) = \sum_x P(x)\log P(x) - \sum_x P(x)\log Q(x) \tag{3.1}$$

$$= \sum_x P(x)\log\prod_{i=1}^{n} P(x_i \mid x_{p(i)}) - \sum_x P(x)\log\prod_{i=1}^{n} Q(x_i \mid x_{q(i)}) \tag{3.2}$$

$$= \sum_{i=1}^{n}\sum_x P(x)\log P(x_i \mid x_{p(i)}) - \sum_{i=1}^{n}\sum_x P(x)\log Q(x_i \mid x_{q(i)}) \tag{3.3}$$

$$= \underbrace{\sum_{i=1}^{n}\sum_{x_i,x_{p(i)}\in\Sigma} P(x_i, x_{p(i)})\log P(x_i \mid x_{p(i)})}_{A} - \underbrace{\sum_{i=1}^{n}\sum_{x_i,x_{q(i)}\in\Sigma} P(x_i, x_{q(i)})\log Q(x_i \mid x_{q(i)})}_{B} \tag{3.4}$$

Since the two tree models are given, the following probabilities can be accessed in constant time:

- **Distribution of the root nodes.** $P(x_{p_r})$ and $Q(x_{q_r})$, where p_r and q_r denote the roots of the tree P, Q, respectively.
- **Transition probabilities.** $P(x_i \mid x_{p(i)}), Q(x_i \mid x_{q(i)})$.

Term A

The term A can simply be computed in nested for-loops over i and over all possible c^2 values of $(x_i, x_{p(i)})$. Since each term $P(x_i, x_{p(i)})\log P(x_i \mid x_{p(i)})$ in the sum can be accessed in constant time, this computation will take $O(c^2 n)$ time.

Term B

Term B is more complicated as it involves the term $P(x_i, x_{q(i)})$ where the nodes X_i and $X_{q(i)}$ may not be adjacent in tree P, in which case the conditional probabilities $P(x_i \mid x_{q(i)})$ are not readily available. So the plan is to compute all such probabilities first and then compute the entire term B in nested for-loops over i and over all possible values of $(x_i, x_{q(i)})$. Next, I will describe an algorithm that computes all the marginal probabilities $P(x_i, x_j)$ for all edges (X_i, X_j) in tree Q. This algorithm will apply to a *general* tree model with some exceptions. But since the use cases in this thesis only involve KL divergence between Chow-Liu trees, which are derived from the leaf-node distribution of evolutionary models, the trees themselves have some special properties that allow for a simpler and faster algorithm, which will be discussed towards the end.

An algorithm for generalized tree models

Let X_r denote the root node in tree P.

1. Compute the marginal probabilities $P(x_i)$ for $i = 1, 2, \ldots, n$ and $x_i \in \Sigma$. These can be computed in $O(cn)-$time by the sum-product algorithm.

2. Compute the marginal probabilities $P(x_i, x_r)$ for $i = \{1, 2, \ldots, n\}/\{r\}$ and $x_i, x_r \in \Sigma$. To achieve this, I first use the sum-product algorithm to compute conditional probabilities $P(x_i \mid x_r)$ for all $i \neq r$. And since $P(x_r, x_i) = P(x_i \mid x_r) \cdot P(x_r)$, so I can easily compute $P(x_r, x_i)$ for all i in $O(c^2 n)-$time.

3. Construct a data structure that answers queries – in constant time – for the lowest common ancestor (LCA) of nodes X_i, X_j in tree P, for any i, j. This can be achieved in linear time by an algorithm devised by Bender et al. [2].

4. After all the pre-processing steps above, I now present a method to compute the marginal probabilities $P(x_i, x_j)$ for each edge $(i, j = q(i))$ in tree Q, which together takes constant time with respect to n. So it will take $\mathrm{O}(n)$ time to compute all such probabilities $P(x_i, x_j)$ for every edge in tree Q. To this end, I first query $LCA(X_i, X_j)$ from the structure constructed in step 3 and then proceed according to the position of the lowest common ancestor in tree P.

   - **Case 1.** $\mathrm{LCA}(X_i, X_j) = X_j$ It follows that $X_r \perp X_i \mid X_j$, in which case

$$\begin{aligned} P(x_r, x_i) &= \sum_{x_j \in \Sigma} P(x_r, x_i \mid x_j) P(x_j) \\ &= \sum_{x_j \in \Sigma} P(x_r \mid x_j) P(x_i \mid x_j) P(x_j), \qquad \because X_r \perp X_i \mid X_j \\ &= \sum_{x_j \in \Sigma} P(x_r \mid x_j) P(x_i, x_j). \end{aligned}$$

     Note that if $X_j = X_r$, then the desired probabilities are immediately available from computations in step 2. Otherwise, $P(x_i, x_j)$'s needs to be solved from a linear system of equations. Since $P(x_i)$'s, $P(x_r, x_i)$'s and $P(x_r, x_j)$'s are already computed in Step 1 and 2, the conditional probabilities $P(x_r \mid x_j) = P(x_r, x_j)/P(x_j)$ can be easily obtained. Therefore, there are c^2 of the unknown $P(x_i, x_j)$'s and c^2 equations as x_r, x_i, x_j vary over the c values in Σ. Precisely, the linear system $AX = B$ to be solved is the following:

$$\begin{pmatrix} P(x_r = 1 \mid x_j = 1) & P(x_r = 1 \mid x_j = 2) & \dots & P(x_r = 1 \mid x_j = c) \\ P(x_r = 2 \mid x_j = 1) & P(x_r = 2 \mid x_j = 2) & \dots & P(x_r = 2 \mid x_j = c) \\ \vdots & \vdots & \ddots & \vdots \\ P(x_r = c \mid x_j = 1) & P(x_r = c \mid x_j = 2) & \dots & P(X_r = c \mid X_j = c) \end{pmatrix}$$
$$\times \begin{pmatrix} P(x_i = 1, x_j = 1) & P(x_i = 2, x_j = 1) & \dots & P(x_i = c, x_j = 1) \\ P(x_i = 1, x_j = 2) & P(x_i = 2, x_j = 2) & \dots & P(x_i = c, x_j = 2) \\ \vdots & \vdots & \ddots & \vdots \\ P(x_i = 1, x_j = c) & P(x_i = 2, x_j = c) & \dots & P(x_i = c, x_j = c) \end{pmatrix}$$
$$= \begin{pmatrix} P(x_i = 1, x_r = 1) & P(x_i = 2, x_r = 1) & \dots & P(x_i = c, x_r = 1) \\ P(x_i = 1, x_r = 2) & P(x_i = 2, x_r = 2) & \dots & P(x_i = c, x_r = 2) \\ \vdots & \vdots & \ddots & \vdots \\ P(x_i = 1, x_r = c) & P(x_i = 2, x_r = c) & \dots & P(x_i = c, x_r = c) \end{pmatrix},$$

where Σ is assumed to be $\{1, 2, 3, \dots, c\}$ – the set of all character states. The time complexity for solving the system is the same as that for finding the inverse of A and multiplying two matrices which is $O(c^3)$ as all the matrices have dimension $c \times c$.

- **Case 2.** $\text{LCA}(X_i, X_j) = X_k$ where $k \neq i, j$. It then follows that $X_i \perp X_j \mid X_k$, in which case

$$\begin{aligned} P(x_i, x_j) &= \sum_{x_k \in \Sigma} P(x_i, x_j \mid x_k) \cdot P(x_k) \\ &= \sum_{x_k \in \Sigma} P(x_i \mid x_k) P(x_k) P(x_j \mid x_k) \cdot \frac{P(x_k)}{P(x_k)}, \qquad \because X_i \perp X_j \mid X_k \\ &= \sum_{x_k \in \Sigma} \frac{P(x_i, x_k) P(x_j, x_k)}{P(x_k)}. \end{aligned}$$

  Since X_k is the lowest common ancestor of both (X_i, X_k) and (X_j, X_k), I first use the method in case 1 to compute the probabilities $P(x_i, x_k)$ and $P(x_j, x_k)$ for all $x_i, x_j, x_k \in \Sigma$ in $O(c^3)$ time. With these probabilities available, I then compute each of $P(x_i, x_j)$'s by taking the sum of c terms on the right hand side. Since there are c^2 values of such $P(x_i, x_j)$, the latter process will take $O(c^3)$-time. The total running time for the computation in case 2 is thus $O(c^3)$.

Remark. Solving the linear system above relies on the matrix A being invertible. A solution to the system is guaranteed to exist because the corresponding probabilities derived from the tree models obviously satisfy the equations. But there are cases when there are multiple solutions to the linear system which happen precisely when

the matrix A is not invertible (or equivalently its determinant is 0), in which case the algorithm would fail. However, if the tree models in question are evolutionary trees (or their transition probabilities share the same property – satisfying the Chapman-Kolmogorov equation), I present next an alternative algorithm that achieves the same result without having to deal with a linear system.

A faster algorithm for a special class of tree models

In this project, I only need to compute the KL divergence between tree models obtained from applying the Chow-Liu algorithm to approximate the leaf-node distributions of evolutionary trees. As discussed in section 2.2, those output tree models inherit the transition probabilities from the input evolutionary models. Precisely, the transition probabilities between any pair of nodes in the Chow-Liu tree is the same those in the original tree and hence they can be computed using only the distance between the two nodes, where the distance is the total sum of the branch lengths of all the edges on the path between the two nodes. So this property will allow us to compute the marginal probabilities $P(x_i, x_j)$'s without having to solve the linear system. To do the computations in this case, I carry on the following steps.

1. Compute the distance between each node X_i and the root node in tree P. This can be done in $O(n)-$time using a tree traversal algorithm such as a breadth-first search or depth-first search.

2. Construct a data structure that answers queries – in constant time – for the lowest common ancestor (LCA) of the nodes X_i, X_j in tree P, for any i, j. This is achievable in linear time by an algorithm devised by [2]. Note that the results from step 1 and 2 allow us allow us to query the distance between *any* two nodes on the same tree in constant time because, given that $\text{LCA}(X_i, X_j) = X_k$,

$$\text{dist}(X_i, X_j) = \text{dist}(X_i, X_r) + \text{dist}(X_j, X_r) - 2 \cdot \text{dist}(X_k, X_r).$$

3. Compute the marginal probabilities $P(x_i)$ for $i = 1, 2, \ldots, n$ and $x_i \in \Sigma$. These can be computed through the formula

$$P(x_i) = \sum_{x_r \in \Sigma} P(x_i \mid x_r) P(x_r),$$

   where $P(x_r)$ are given and $P(x_i \mid x_r)$ can be easily computed from the distance between the nodes X_i and X_r from step 1.This step requires $O(cn)-$time.

4. For each edge $(x_i, x_{q(i)})$ in tree Q, compute $P(x_i, x_{q(i)})$ by first finding the distance between the node $X_i, X_{q(i)}$ in tree P, and then computing

$$P(x_i, x_{q(i)}) = P(x_{q(i)}) \cdot P(x_i \mid x_{q(i)})$$

using the transition probability formula (which is a function of the distance or branch-length) and the marginal probability from step 3. Then one can compute term B

$$\sum_{i=1}^{n} \sum_{x_i, x_{q(i)} \in \Sigma} P(x_i, x_{q(i)}) \log Q(x_i \mid x_{q(i)})$$

in $O(c^2 n)-$time as there are n edges in tree Q and c^2 possible states for $(x_i, x_{q(i)})$.

Remark. Some evolutionary models may have assumptions that further simplify the computations. For example, the Jukes-Cantor model assumes that the substitution probability from one state to another is the same, and as a result, the transition probabilities $P(x_i \mid x_j)$ between node X_i and X_j can have only 2 different values depending on whether the two states (x_i, x_j) are the same or not. Therefore, when computing the term such as

$$\sum_{x_i, x_{q(i)}} P(x_i, x_{q(i)}) \log Q(x_i \mid x_{q(i)}),$$

I do not need to sum over all possible c^2 values of the parameters but rather simplify it to

$$\begin{aligned} 4P(X_i = \sigma, X_{q(i)} = \sigma) \log Q(X_i = \sigma \mid X_{q(i)} = \sigma) \\ + 12P(X_i = \sigma, X_{q(i)} = \sigma') \log Q(X_i = \sigma \mid X_{q(i)} = \sigma'), \end{aligned}$$

where $\sigma \neq \sigma'$ and $\sigma, \sigma' \in \Sigma$.

Chapter 4

Significance Testing for Correlations

In the previous chapters, I describe the pipeline that is used to process the multiple alignment data into gene evolutionary models, and explain the methods used to estimate the distances between the evolutionary models. The pairwise distances between these models are used in the multidimensional scaling process to find gene evolutions' representations in the 2-dimensional Euclidean space. This process lays out a framework for studies related to gene evolution comparisons, which can be applied to many applications. One area of such applications that I am interested to explore in this dissertation is potential correlations between genes' evolution histories and their functions. Given the $\mathbb{R}^2$-point representations of gene evolutions, I can conveniently carry out cluster analysis necessary for this study. More specifically, I will examine if these gene tree points, or a subset of them, form clusters naturally according to their functions, which will require a statistical significance testing to see whether the clusters are formed simply by chance or by the common functions of the genes. In the rest of this chapter, I will first explain how the Gene Ontology database is used to determine gene functions and then discuss the techniques used to perform cluster analysis and relevant significance tests for correlations. The results are discussed in the last section.

4.1 Gene Ontology

The Gene Ontology database provides the annotations of genes and gene products with a list of controlled vocabularies (or GO terms) that are independent of their species [1]. The GO annotations are usually organized into three directed-acyclic graphs whose nodes represent the GO terms and edges describe one-way relations such as *is-a*, *part-of*. The roots of the three DAGs are GO:0003674 (molecular functions, MF), GO:0005575 (cellular components, CC) and GO:0008150 (biological processes, BP). As a matter of fact, the three DAGs have overlapping nodes, which means that a GO term may belong to more than one DAGs or even all three of them. As of August 24, 2014, the database contains 41596 GO terms in total where the molecular-function DAG (cellular-component, biological-process,

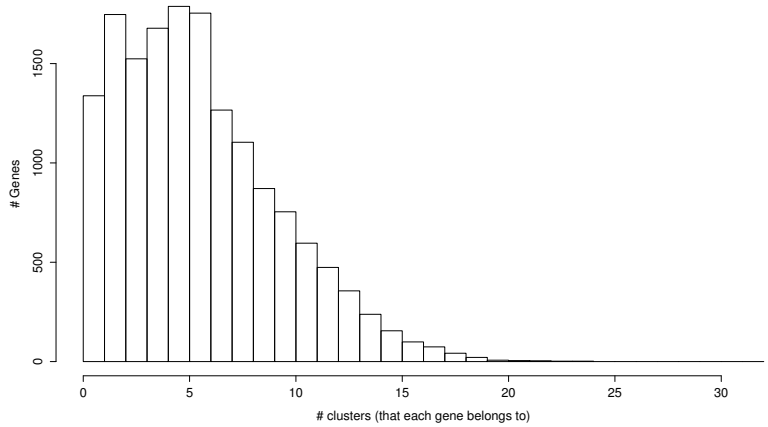

Figure 4.1: A histogram that shows how many genes there are that belong to different numbers of selected GO clusters. Among those 15899 genes, only 1338 of them belong to single clusters.

respectively) contain 10446 of them (3732, 36018, respectively).

To conduct a cluster analysis, I define clusters based on the GO terms, one cluster for each GO term, and assign a gene to a cluster if it is annotated with its respective GO term. When a gene is annotated with a GO term, it implies that the gene should also be annotated with all of the GO term's ancestors in the DAGs. This is because the relations *is-a* and *part-of* are transitive. Therefore, there is a strong dependency among these clusters, as every cluster is always contained in the clusters corresponding to its ancestral nodes in the DAGs. I anticipate that this type of dependency would introduce some undesirable complication to the significance testing analysis. It is therefore necessary that I choose only a subset of the clusters for the analysis and to avoid such dependency, I will choose them in such a way that no cluster is a descendant of another. In the experiment, I decide to use only the immediate children of the three root nodes and there are 56 of them. However, I still need to take out some of them due to the limitation of the permutation method to be discussed later.

In addition, due to the transitive annotations and overlapping DAGs, some genes are expected to belong to more than one clusters so it requires the fuzzy-clustering framework for cluster analysis. In this framework, each data element is allowed to belong to more than one clusters with associated *membership weights*, which determine how strong each data point is associated with each of the clusters. Since all the GO annotations are backed with some types of evidence, be it computational or experimental, I assume that all such cluster memberships are certain. Therefore, for every gene, I assign an equal membership weight to every cluster it belongs to.

4.2 Monte Carlo permutation tests

In this section, I explain the methods used to perform a significance test for clustering. I want to test the null hypothesis that the genes' evolutions have evolved independently from their functions versus the alternative hypothesis that there is a *natural* clustering structure based on their functions. Due to the complication of this problem, I cannot analytically compute the null distribution of the test statistics (to be stated later). I therefore resort to a Monte Carlo permutation test, which simulates data to approximate the null distribution instead. In the crisp or strong clustering framework, such a significance test would simply involve computing the F-ratio statistics and use its asymptotic distribution to estimate the p-value, which is known as *the maximum F test* [7]; or alternatively I could use an approximate permutation test that estimates the p-value by generating random cluster structures and compare their test statistics with the observed one. However, the matters get more complicated when, in fact, this experiment has to deal with the fuzzy clustering environment.

Let n be the number of genes and c be the number of clusters of interest. Fuzzy clusters of genes are represented by the *membership matrix*

$$M = [m_{ij}]_{n\times c}, \quad m_{ij} = \begin{cases} 1 & \text{if gene } i \text{ belongs to cluster } j \\ 0 & \text{otherwise.} \end{cases}$$

Define the row sums $(s_1, s_2, \ldots, s_n)$ and the column sums $(t_1, t_2, \ldots, t_c)$ by

$$s_i = \sum_{j=1}^{c} m_{ij}, \text{ for } i = 1, 2, \ldots, n \quad \text{and} \quad t_j = \sum_{i=1}^{n} m_{ij}, \text{ for } j = 1, 2, \ldots, c.$$

In particular, there are gene clusters based on the GO annotations and I would like to examine if this particular cluster structure is significantly "better" than other random structures. This requires a proper way of generating random membership matrices and a suitable test statistic that can compare the quality of any two cluster structures.

There are at least two ways to randomly "permute" such a membership matrix for significance tests. One is to permute only the rows, or equivalently to shuffle the gene labels, leaving the membership patterns for each row intact. This is to preserve any dependency among the clusters such as if a gene belongs to cluster A, it must also belong to cluster B and C. Another way is to also permute the entries in each row while keeping the marginal sums (row and column sums) fixed. In other words, I keep the size of each cluster and the number of clusters each gene belongs to the same. When carrying out hypothesis tests, this restriction could help remove the unwanted effect of nuisance parameters such as those related to the marginal sums, which definitely influence the null distribution of the cluster test statistics. Another example that imposes the same condition on marginal sums for tests can be found in [26]. Such condition on the marginal sums is also analogous to permutation tests for crisp clustering, where each element belongs to exactly one cluster and so the membership matrices always have row and column sums equal to one.

The first way of permuting the matrix simply requires a permutation algorithm while the second requires a much more sophisticated method such as sequential importance sampling and it is widely considered as an open problem in the community. Once a desired number of random permutations of the membership matrices are generated, their test statistics are computed and compared against the *observed* value obtained from the membership matrix derived from the GO clusters. This process yields a p-value, which is used to determine if the clusters induced by GO annotations are significant. If they are, then it implies that there is likely an association between genes' functions and their evolutions.

Cluster test statistics

To test the significance for clusters, a test statistic is needed for comparing the quality of different cluster structures. In general, a cluster structure is considered *good* when its clusters are well-separated from one another and each cluster is compact in the sense that its members are close to one another. Measures that quantifies the goodness of a cluster structure are referred to as *cluster validity indices* in the literature. Cluster validity indices are usually used to determine an optimal number of clusters as part of clustering algorithms and also to measure the validity or quality of the resulting clusters from a clustering algorithm. I can, therefore, use such indices to verify if the set of clusters induced by the GO terms is significantly "better" than other random clusters. To pick specific indices for this project, I consult with the article [30] by Wang and Zhang, which gives a good review of fuzzy validity indices available in the literature as well as comparisons of their performance on some widely used data sets. However, it is not straightforward to objectively choose ones that would work best with the data set. After looking at the comparison results and the number of citations to their respective articles, I decide to use two of the indices, i.e. Xie-Beni [31] and PBMF [21], whose formula are given below.

$$XB = \frac{\sum_{j=1}^{c}\sum_{i=1}^{n}\mu_{ij}^{2}\|V_j - X_i\|^2}{n\min_{j,k}\|V_j - V_k\|^2}, \tag{4.1}$$

$$PBMF = \left(\frac{1}{c}\times\frac{\sum_{i=1}^{n}\|X_i - V\|}{\sum_{j=1}^{c}\sum_{i=1}^{n}\mu_{ij}^{1.5}\|V_j - X_i\|}\times\max_{j,k}\|V_j - V_k\|^2\right)^2 \tag{4.2}$$

where V_i is the fuzzy centroid of the i^{th} cluster,

V is the centroid of the entire data set,

$\mu_{ij} \in [0, 1]$ is the membership weight,

n is the total number of points,

c is the number of clusters.

Note that when gene i belongs to c_i clusters, the membership weight μ_{ik} is $1/c_i$ if the gene belongs to cluster k and 0 otherwise.

Xie-Beni index

Xie and Beni proposed an index [31] that captured both the compactness and the separation of a cluster. From the formula (4.1), the numerator measures the compactness or how close data points from the same cluster are together while the denominator measures the separation or how well distinct clusters (or their centroids to be precise) are far from one another. A good cluster structure will produce a low value of the numerator and a high value of the denominator and in turn a low value of the Xie-Beni index.

To estimate the p-value, I therefore consider the random samples that produce lower Xie-Beni indices than the observed value. But note that the index is not robust to cases when there are two clusters that are very close together. From my observations, this index could be very bad when there are two almost identical clusters. This is likely to occur when the data contain clusters with large number of genes and hence many overlaps, and in turn the subsampling process could potentially pick up a lot of those overlapping genes, which result in similar clusters in the subsamples.

PBMF index

Pakhira et al. proposed a similar validity index [21] that also captured the two qualities of a cluster. But the compactness term uses different exponents and the separation uses the maximum instead of minimum. The remaining terms are constant for each data set, invariant between cluster-membership permutations. In contrast to the Xie-Beni index, a good cluster structure will attain a high value of PBMF index.

Generating random binary tables with fixed marginal sums

The problem of generating binary tables with fixed marginal sums from the uniform distribution arise in many fields, especially in statistical hypothesis test settings. Several methods have been proposed but all of them are observed to raise some issues when handling large- or moderate-size tables in the experiment described in [9]. Two of the most popular approaches are Monte Carlo Markov Chain (MCMC) and sequential importance sampling (SIS). The MCMC method [14] defines a Markov chain on the state space consisting of all valid binary tables and each move involves picking a pair of rows and a pair of columns at random and if the 2×2 sub-matrix, determined by the selected rows and columns, is one of the following

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix},$$

then switch to the other form, or otherwise do nothing. To generate one random sample from such a table, the methods begins with choosing a starting state, which can be any in the state space. It then makes a (large) number of moves from that state until it is believed to have reached the limit distribution. And the last state will be a random sample from the limiting distribution, which is hopefully close to the uniform distribution. This process

is repeated until enough samples are generated. A variation that involves an additional metropolis step is also discussed in [11] to make the limiting distribution closer to the uniform distribution. However, this approach still lacks analytical work that asserts a practical bound on the number of moves required to reach such limiting distribution. There is a report in [9] of an experiment where the SIS method is more efficient in estimating the p-value than the MCMC method given that they are allowed to run for the same amount of time. Therefore, I decide to use the SIS approach [9] in this thesis. It essentially samples a binary table from another distribution, known as the *importance distribution*, which is much easier to attain than sampling from the uniform distribution, and then uses the samples to estimate the p-value instead. The method sequentially samples one row (column) at a time from the conditional-Poisson distribution based on the current column sums (row sums, respectively). See next section for more details.

Sequential importance sampling method

I implement the SIS algorithm (the more delicate version) as proposed by Chen *et al.* in [9, sec. 4]. The algorithm simulates a binary table T of dimension $n \times c$ with fixed marginal sums in the row-by-row fashion. In order to gain better efficiency, I also adjust the algorithm to generate rows in order of their row sums, from largest to smallest, as suggested in [9]. To generate each row (a binary array of dimension $1 \times c$), it needs to randomly select which columns to be 1 (and the rest will be 0), where the probability of each column being 1 is proportional to its current column sum. This is precisely sampling from the conditional-Poisson distribution and can be done using the drafting sampling algorithm [8]. This step is rather expensive as it has the worst-case running time complexity of $O(c^4)$ where c is the number of columns. Therefore, there cannot be too many rows (if sampling column-by-column) nor too many columns (if sampling row-by-row) in order to make this step feasible. The most reasonable option is to restrict the number of columns (or clusters) and to simulate binary tables row-by-row.

Another crucial aspect of importance sampling is the *importance distribution* $q(T)$ (also known as the *trial distribution*), which defines the probabilities of each table T being simulated by the algorithm. These probabilities will be needed for estimating many quantities related to the hypothesis testing analysis later on. In this case, $q(T)$, the probability of producing a particular table T, is equal to the product of all the probabilities of each of its rows being sampled from the conditional Poisson distribution. To make it more precise, let T be a binary table T to be simulated. Suppose its rows are $\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_n \in \{0,1\}^c$ and its row sums and column sums are $\mathbf{s} = (s_1, s_2, \ldots, s_n), \mathbf{t} = (t_1, t_2, \ldots, t_c)$, respectively. The first row $\mathbf{r}_1$ is sampled according to the conditional Poisson distribution, conditioning on the given row and column sums, which means that its entries need to sum to s_1 and the probability of column i being 1 is proportional to t_i. Then, when the second row $\mathbf{r}_2$ is sampled, it needs to also condition on $\mathbf{r}_1$ as the column sums $\mathbf{t}$ need to be updated according to which columns of the first row are 1. Similarly, to sample the row i, it needs to condition on all the previously-sampled rows. Therefore, it follows that

$$q(T) = q(\mathbf{r}_1 \mid \mathbf{s}, \mathbf{t})q(\mathbf{r}_2 \mid \mathbf{r}_1, \mathbf{s}, \mathbf{t}) \dots q(\mathbf{r}_n \mid \mathbf{r}_1, \mathbf{r}_2, \dots, \mathbf{r}_{n-1}, \mathbf{s}, \mathbf{t}), \tag{4.3}$$

where $q(\cdot)$ on the right hand side represents the conditional Poisson distribution.

Let $p(T)$ be the target distribution, which in this case is the uniform distribution over all binary tables with given row and column sums. The exact value of $p(T)$ is presumably unknown as the problem of counting the exact number of binary tables with fixed marginal sums is still widely considered an open problem. But $p(T)$ is constant over all T. Now suppose I would like to estimate the quantity $\mu = \mathbb{E}_p(f(T))$, where f can be any function. For example, when $f(T) = \mathbb{1}\{I_{XB}(T) < I_{XB}(T_{\text{GO}})\}$, where $I_{XB}(T)$ computes the Xie-Beni index of the table T, then μ will be p-value in the hypothesis testing when using the Xie-Beni index. Similarly, if $f(T) = \mathbb{1}\{I_{PBMF}(T) > I_{PBMF}(T_{\text{GO}})\}$, then μ will be the p-value for the PBMF index.

To estimate μ, I first sample N tables $T_1, T_2, T_3, \dots, T_N$ from the trial distribution using the SIS algorithm. Then, I apply the following *self-normalized importance sampling* formula to estimate

$$\mu \approx \hat{\mu} = \frac{\sum_{i=1}^{N} f(T_i)w_i}{\sum_{i=1}^{N} w_i}, \text{ where } \quad w_i = p(T_i)/q(T_i).$$

The terms $w_i = p(T_i)/q(T_i)$ are known as the *importance weights*. Note that the less likely a table is to be produced by the SIS algorithm (i.e. low $q(T)$), the higher its weight is. This is to boost the importance of rare samples. However, since the exact values of $p(T_i)$ are not known, I use the following simplified formula

$$\frac{\sum_{i=1}^{N} f(T_i)w_i}{\sum_{i=1}^{N} w_i} = \frac{\sum_{i=1}^{N} f(T_i)p(T_i)/q(T_i)}{\sum_{i=1}^{N} p(T_i)/q(T_i)} = \frac{\sum_{i=1}^{N} f(T_i)/q(T_i)}{\sum_{i=1}^{N} 1/q(T_i)},$$

for computation in practice, exploiting the fact that $p(T_i)'s$ are all constant.

SIS diagnostics

The SIS algorithm produces samples with associated weights $p(T_i)/q(T_i)$. An ideal SIS algorithm would have the trail distribution identical to the target distribution, and so the weights would be all the 1.0 and their standard deviation would be 0. There are instances when the sampling algorithm is less than ideal and the weights are so disperse that only few of the samples dominate the estimated values of interest, which could make the results questionable. This happens when the samples contain some outliers with relatively extremely small probabilities $q(T_i)$, which is more likely to occur when simulating large tables.

To watch for such potential problems with the SIS samples, I use the following measures: standard deviation of the estimated p-value, the function-independent coefficient of variation of the weights, and its function-specific version, which are used to compute the effective sample size (ESS)

$$ESS = \frac{N}{1 + cv^2}.$$

1. *Standard deviation of the estimated p-value.* Based on the formula of $\hat{\mu}$ above, one can estimate its standard deviation using the delta method as derived in [18] as follows.

   Let $Y = f(T)p(T)/q(T)$ and $X = p(T)/q(T)$. The p-value can be estimated by

   $$\mu = \mathbb{E}_p(f(T)) = \sum_T f(T)p(T) = \frac{\sum_T f(T)\frac{p(T)}{q(T)} \cdot q(T)}{\sum_T \frac{p(T)}{q(T)} \cdot q(T)} = \frac{\mathbb{E}_q(Y)}{\mathbb{E}_q(X)}.$$

   Note that $\mathbb{E}_q(Y) = \mu$ and $\mathbb{E}_q(X) = 1$. Then the delta method gives us the approximation

   $$\begin{aligned}\mathrm{var}(\hat{\mu}) &\approx \frac{1}{N}\left[\mathrm{var}_q\left(f(T)\frac{p(T)}{q(T)}\right) + \mu^2\mathrm{var}_q\left(\frac{p(T)}{q(T)}\right) - 2\mu\,\mathrm{cov}_q\left(f(T)\frac{p(T)}{q(T)}, \frac{p(T)}{q(T)}\right)\right] \\ &\approx \frac{\sum_{i=1}^N \left(f(T_i) - \hat{\mu}\right)^2 \left(p(T_i)/q(T_i)\right)}{\sum_{i=1}^N \left(p(T_i)/q(T_i)\right)} = \frac{\sum_{i=1}^N \left(f(T_i) - \hat{\mu}\right)^2 /q(T_i)}{\sum_{i=1}^N 1/q(T_i)}\end{aligned}$$

   Note that in the above equation, the terms $p(T_i)$ disappear because they are constant with respect to all T_i so those in the numerators and denominators cancel out.

2. *Function-independent coefficient of variation or relative standard deviation.* One can estimate it using

   $$cv^2 = \frac{\mathrm{var}_q\left(\frac{p(T)}{q(T)}\right)}{E_q^2\left(\frac{p(T)}{q(T)}\right)} \approx \frac{\frac{1}{N-1}\sum_{i=1}^N \left(\frac{1}{q(T_i)} - \frac{1}{N}\left[\sum_{j=1}^N \frac{1}{q(T_i)}\right]\right)^2}{\left(\frac{1}{N}\left[\sum_{j=1}^N \frac{1}{q(T_i)}\right]\right)^2}.$$

3. *Function-dependent coefficient of variation.* The previous two measures do not depend on the function f while the efficiency of SIS should arguably depend on it. In this context, this function f is the indicator function that checks if a test statistic for a cluster is more extreme than the observed value or not. Owen [18] suggests another measure of efficiency, the coefficient of variation specific to f or $cv^2(\tilde{w}, f)$, which includes the function f as follows. Let

   $$\widetilde{w}_i(f) = \frac{|f(T_i)|p(T_i)/q(T_i)}{\sum_{j=1}^n |f(T_j)|p(T_j)/q(T_j)} = \frac{|f(T_i)|/q(T_i)}{\sum_{j=1}^n |f(T_j)|/q(T_j)}.$$

   Then, one can estimate

   $$cv^2(\tilde{w}, f) \approx n\sum_{i=1}^n \widetilde{w}_i^2(f) - 1,$$

   in which case the effective sample size is $1/\sum_{i=1}^n \widetilde{w}_i^2(f)$.

To interpret these diagnostic measures, it is generally desired for the standard deviation of the weights to be small and the effective sample sizes to be large. Based on the experiment, there are cases when the values $1/q(T_i)$ vary greatly among the samples, causing the three measures to have unfavorable values which indicate that the SIS is rather ineffective on those cases. Based on the examples given in Table 4.1, I observe that the test with the worst cv^2 produces 50000 tables with much wider range of $1/q(T_i)$ than the test with the best cv^2.

cv^2	0%	25%	50%	75%	100%
1466.28	2.25×10^{3943}	5.72×10^{3947}	3.77×10^{3948}	2.46×10^{3949}	1.40×10^{3954}
24.27	2.45×10^{4054}	2.40×10^{4057}	8.34×10^{4057}	2.87×10^{4058}	1.89×10^{4061}

Table 4.1: The quantiles of the values $1/q(T_i)$ from two tests in the experiment, where each test involves randomly generating 50,000 tables. These two tests produce the best (lowest) and the worst (highest) cv^2 among 200 of the tests.

When I look closely at them, I find that bad cases usually occur when the tables are far from being regular, where being regular means that all its row and column sums are approximately the same. From the comparison in Figure 4.2, I can clearly see the difference between the histograms of the row sums from both tests. The worst test contains a longer tail of low row sums (1's and 2's) and a higher number of high row sums (which are greatern than 10), while, in the best test, its row sums look more clustered between 3 and 10m which making its table look "closer" to being regular. This finding is also consistent to some extent with a finding in [4] that when using the SIS algorithm to count the number of distinct binary tables, whose row and column sums are of the form $(d, 1, 1, \ldots, 1)$ and fixed, it would be so ineffective that the result would be off by an exponential factor.

To investigate more on the issue, I look at some outlier samples with unusually large values of $1/q(T_i)$ (i.e. low $q(T_i)$). I find that such a low probability $q(T)$ is a result of the product in Equation 4.3 containing some unusually low terms, which correspond to the low probabilities of some rows being sampled from the conditional Poisson distribution. In these outlier tables, such rows are sampled with low probabilities because they contain 1's in columns with relatively low sums, and hence low weights; while in "normal" tables, the corresponding rows would contain 1's in columns with higher sums, which is much more likely to be chosen. The problem is more pronounced when the column sum configuration is so heavily skewed that it has more columns with relatively row sums and hence more likely getting chosen when sampling a row. Therefore, the cluster sizes should be made as balanced as possible. It is unfortunate that the distribution of the GO cluster sizes, as defined earlier, is heavily skewed, as some of them contain more than 50% of the genes while some contains only less than 1%. If I were to use all of them, the SIS algorithm would be prone to failures. I thus decide to take out these clusters, which have either more than 50% or less than 1% of the genes. This reduces the number of clusters from 56 to 35.

In addition, from Equation 4.3, the probability $q(T_i)$ is the product of n row-sampling probabilities where n is the number of rows. Thus, the higher number of rows, the more

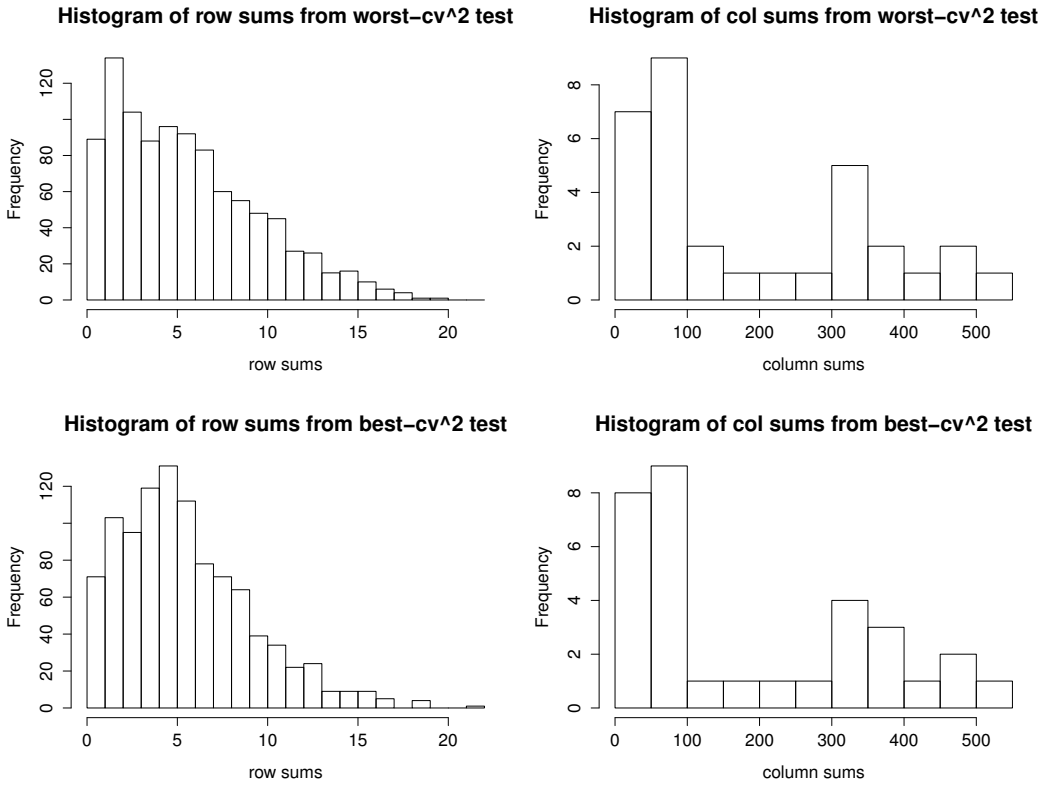

Figure 4.2: The histograms of the row and column sums from the test examples – those with the best and worst cv^2. The table in the best test contains 71 rows with row-sum 1 and 103 with 2 while the table in the worst test contains 89 rows with 1 and 134 with 2.

likely that the product will contain more unusually low terms, and the more likely that the SIS will be ineffective. This is to conclude that the table cannot have too many rows (or genes) at once; especially when the entire set of genes contains roughly 16,000 of them, it would be impossible to make SIS work with the table of that size.

To address this issue, I decide to instead perform a number of hypothesis tests on a smaller set of genes, randomly selected from the full set. Precisely, I will run 200 tests with 1000 genes in each test. Since I do not know how to combine those tests to make a single conclusion for the significance testing, I will instead treat them as independent tests.

4.3 Methodology

In summary, I take the following steps to carry out significance testing for clusters.

1. Assign each gene to appropriate Gene Ontology (GO) cluster(s). Due to limitations of the table sampling method, only 56 GO clusters are used in the analysis. They are

precisely the immediate children of the three root nodes. Let T_{GO} denote the membership matrix derived from the relationships between genes and GO terms. Discard all genes that are not assigned to any cluster.

2. For each gene, take the multiple sequence alignment data from the UCSC database and use the Jukes-Cantor model to compute the pairwise evolutionary distances between species. Then use the neighbor-joining algorithm to construct a gene tree, which also implicitly defines a corresponding evolutionary model.

3. Estimate the KL divergence between each pair of gene trees' leaf-node distributions with the following steps.

   a) For each gene, approximate the marginal distribution of all its leaf nodes with a tree model (first-order dependence model) using the Chow-Liu algorithm [10]. This tree model is referred to as the Chow-Liu tree.

   b) For each pair of gene trees, compute the KL divergence between their Chow-Liu trees, using the algorithm described in Chapter 3.

4. Once all the pairwise distances between gene trees are computed, apply multidimensional scaling (MDS) to find representations of the gene trees in the Euclidean space $\mathbb{R}^2$. These point representations are used for cluster analysis.

5. The membership matrix has dimension of 16929×56, which well exceeds the capacity of the SIS algorithm to efficiently handle it. So discard the GO clusters or columns by the following criteria.

   a) Discard the columns that either contain more than 50% or less than 1% of all the genes. This step removes 21 columns.

   b) Then for each pair of columns (considered as binary vectors), derive the contingency table and compute the χ^2 statistics. Discard one column (the one with the higher index) if the statistics is greater than a threshold of 10000, indicating that the two columns are highly correlated. This step removes 3 additional columns.

   The new membership matrix has dimension 15899×32, after removing the genes not belonging to any remaining clusters. The 32 clusters are classified as 10 cellular components (CCs), 13 biological processes (BPs) and 9 molecular functions (MFs) where the original 56 clusters are 20 CCs, 21 BPs and 15 MFs. However, the table still has too many rows so .. need to subsample the rows down to 1000, as discussed in the next step.

6. Perform 200 hypothesis tests independently where each test involves only a random subsample of 1000 genes or rows. To test for the association between gene functions and gene evolutions, carry out a permutation test by using the SIS algorithm to generate

50000 random binary matrices with the same row and column sums as T_{GO} and estimate the p-value. I use PBMF and Xie-Beni validity indices as the test statistics. Lastly, apply Benjamini and Hochberg's multiple testing correction to interpret the results.

4.4 Results and discussion

I carry out 3 separate hypothesis testing methods in total, one of which is the main work described in Section 4.3 above and the other two are experimental, to be described below. The original data consists of 16929 genes but after filtering out the GO clusters, I am left with 15899 of them, as the rest are not annotated with any of the retained features. Only 1338 of them belong to single classes, and each of the rest belongs to more than one groups. I decide to run a permutation test on these 1338 genes, which simply requires the well-known maximum F test as the clustering is crisp (one element belongs to one cluster). But when I want to include more data, I need to work with the fuzzy-clustering environment. And when I generate "permutations" of the data to estimate the null distribution, I experiment with two ways of permuting the clustering structure or the membership matrix, to be described in its respective section below.

Single classes - crisp clustering

As discussed before, I perform the maximum F test to see if these genes naturally cluster based on their GO classes or not. Again, I still use the permutation test to estimate the null distribution. It turns out that based on 1 million permutations, the estimated p-value is 0.0, which means all of one million randomly permuted tables have the F-ratio test statistics less than the observed value, or the p-value is 0.

From the Figure 4.4, one could probably tell that the points are not from a homogeneous population, although they are not clearly separated. This inhomogeneity is asserted by the F-ratio statistic and the significance test.

Row-shuffling permutations - fuzzy clustering

The first permutation method is to simply permute the gene labels or the rows in order to generate random clustering structures. In this case, I include all the 15899 genes in the permutation test and randomly simulate 200,000 of such structures to estimate the p-value and it turns out that the p-values are 0.0 when using either of the two cluster statistics.

SIS permutations - fuzzy clustering

The second method is to simulate random binary matrices with fixed row and column sums (which are the same as those of T_{GO}). For each test statistic, I carry out 200 independent tests/cases where each test involves only 1000 randomly-selected genes. Below are the results

Figure 4.3: Plot of MDS projections of gene trees that belong to only single classes. The original data contains 1338 points, but for clarity, only the elements that belong to the top 3 classes are plotted. There are 933 of them (or around 70% of 1338).

of the tests with 5 smallest p-values and 5 biggest p-values for each of the test statistics used. See the full results in Appendix B.

I apply Benjamini-Hochberg procedure to control the false discovery rate with level $\alpha = 0.05$. This procedure first sorts the p-values from smallest to largest, say $\{p_{(1)}, p_{(2)}, \ldots, p_{(200)}\}$ and compare the p-value $p_{(i)}$ with the *adjusted threshold* $i * \alpha/200$ in order to determine if the test i is significant (rejecting the null hypothesis H_0). The column $\hat{\sigma}$, N^* and N^f are the estimated standard deviation, the effective sample sizes, which are computed from the two types of the coefficient of variations described in Section 4.2. They are mainly for SIS diagnostics. The better these values are (i.e. lower for $\hat{\sigma}$, higher for N^*, N^f), the more reliable the test results would be. However, it is difficult to determine the absolute threshold for these measures as to when the test could be valid or not.

In summary, there are 69 Xie-Beni tests that have the p-values less than 0.05 and only 43 of them are declared significant after the correction procedure. For PBMF index, there are 18 tests with p-value less than 0.05 and none of them is significant after the procedure. Based on these mixed results, I could not make a decisive conclusion that such correlation between genes' functions and evolutions exists. But it is more likely the case that there is a group of genes whose functions and evolution histories are correlated and another group

Test #	**P-value**	$\hat{\sigma}$	N^*	N^f	**Adj. Threshold**	**Reject** H_0?	$T_<$
Xie-Beni index							
10	2.279e-06	2.282e-06	461.0	222.6	2.500e-04	True	1
69	6.859e-06	3.450e-06	686.7	442.0	5.000e-04	True	6
187	2.496e-05	1.680e-05	761.7	330.5	7.500e-04	True	8
1	3.303e-05	2.274e-05	626.9	323.2	1.000e-03	True	7
6	3.675e-05	2.018e-05	1896.6	404.4	1.250e-03	True	7
				⋮			
164	9.045e-01	7.289e-03	948.4	5710.4	4.900e-02	False	45100
62	9.278e-01	9.157e-03	190.0	2725.4	4.925e-02	False	46178
158	9.328e-01	8.493e-03	493.8	4331.1	4.950e-02	False	46527
172	9.370e-01	6.604e-03	685.3	5038.2	4.975e-02	False	47176
87	9.580e-01	3.054e-03	1035.0	6131.6	5.000e-02	False	47496
PBMF index							
41	1.801e-02	3.302e-03	769.2	1192.3	2.500e-04	False	1015
35	1.950e-02	3.231e-03	994.1	1312.8	5.000e-04	False	1285
95	2.181e-02	2.019e-03	1630.1	2339.6	7.500e-04	False	1491
56	2.618e-02	4.551e-03	949.8	1243.1	1.000e-03	False	1594
99	2.855e-02	3.315e-03	780.6	1892.2	1.250e-03	False	1827
				⋮			
146	4.051e-01	1.549e-02	1181.9	3647.4	4.900e-02	False	19901
182	4.237e-01	1.569e-02	932.7	4610.3	4.925e-02	False	23522
4	4.324e-01	1.358e-02	1249.7	5416.5	4.950e-02	False	22227
48	4.862e-01	1.642e-02	935.4	4034.4	4.975e-02	False	25166
157	4.951e-01	1.570e-02	1014.9	4498.7	5.000e-02	False	26675

Table 4.2: (Partial) Results which include only the cases with 5 least and 5 highest p-values when using each of the test statistics. $T_<$ denotes the number of samples/tables (out of 50000) that have respective test statistics more extreme than the observed value.

where genes, even with similar functions, do not have correlated evolutions. When including all these genes into the analysis, it is expected to see such mixed results.

A reasonable next step one could do is to find a set of genes' functions that would have some association with the genes' evolutions, although it is beyond the scope of this thesis. To this end, I could specifically look into the significant cases and find out if the sets of the GO terms labeled on those genes (every function in the database altogether, not just the 56 selected ones) share anything in common. And then I can re-select a new set of GO terms and run through the experiment again. Once I can confirm that there are some GO terms that are indeed related to how the genes evolve, there is another application that is worth investigating – to predict or discover a gene's function. This could be achieved by

running a clustering algorithm of the MDS of the gene trees, and, in each cluster, I look at the common known functions (which are also known to be correlated with genes' evolutions) that appear to be labeled on the majority of the genes. Then I could predict that those genes in the cluster that are currently not annotated with such functions should, in fact, have been annotated with them.

Pellegrini et al. [23] conduct a similar study that aims to discover functions of the proteins encoded in the *E. coli* genome. This study relies on the assumption that ''*functionally linked* proteins evolve in a correlated fashion, and, therefore, they have homologs in the same subset of organisms." So, as part of the process, they compute multiple sequence alignments of each of the proteins with the homologs found in 16 other fully-sequenced genomes (from bacteria, yeasts and archaea). Then they try to cluster the proteins based on the sets of genomes in which the proteins have homologs and allow these sets to differ by three elements. Using some statistical test, they find out that proteins in the same cluster are more likely functionally linked and, vice versa. The assumption of this study is essentially the hypothesis of this project. But I use a different measure for how much two genes or proteins evolve in a correlated fashion.

Although there is no conclusive result based on the current work, this thesis provide a framework for cluster analysis related to gene evolutions. Note that this framework can be applied to any work that aims to study potential association between genes' evolutions and genes' attributes, which are not necessarily derived from the GO databases, but rather any categorization scheme that divides genes into different classes.

Appendix A

Data and Software

A.1 Data

The data used in our experiment come from the following sources

Multiple Alignments of Assemblies

The Genome UCSC database which provides multiple alignments of genes' homologs within 100 species. The sequences are broken down into exons but I re-assemble the genes by simply concatenating all the exons. This is reasonable because the evolutionary model assumes the DNA sequences are generated independently from site to site. The data was retrieved on November 27, 2013 from
`http://hgdownload.cse.ucsc.edu/goldenPath/hg19/multiz100way/` .

Gene Ontology Annotations

The GO annotation data was downloaded on October 11, 2014 from the two following sources

- Gene-product (excluding Protein UnitProt) association with GO terms
  `http://archive.geneontology.org/latest-lite/`
- Unitprot GO annotation for Homo Sapiens
  `http://geneontology.org/page/download-annotations`

Gene Ontology Hierarchy

I use the Gene Ontology data to construct the directed-acyclic graph of the GO hierarchy. Note that I only consider the relation is_a and part_a when determining the GO hierarchy. The file was retrieved on August 24, 2014 from
`http://www.geneontology.org/ontology/obo_format_1_2/gene_ontology_ext.obo`

Gene ID conversion

The genes' IDs from the multi-alignment database are in UCSC format while those annotated with GO terms are in HGNC format.Therefore, I need to convert them in order to match a gene with its annotated GO terms. Below is the API call I made to retrieve the UCSC-HGNC conversion database on October 28, 2014.
`http://www.genenames.org/cgi-bin/download?col=gd_hgnc_id&col=gd_app_sym&col=gd_app_name&col=gd_aliases&col=md_ucsc_id&status=Approved&status_opt=2&where=&order_by=gd_hgnc_id&format=text&limit=&hgnc_dbtag=on&submit=submit`

A.2 Software

As part of this experiment, I write 2 software packages that compute the estimated distances between two gene trees (in Java) and implement the SIS algorithm [9] that randomly generates binary tables with fixed row and column sums (in Python). I also use the following open-source packages to aid our computation for the project:

1. Use R package `ape` to estimate the pairwise evolutionary distances between species from the multiple sequence alignments and construct phylogenetic trees using the Neighbor-Joining algorithm.

2. Use R function `cmdscale` use to apply multi-dimensional scaling (MDS) on the gene trees.

Appendix B

Complete Results

Xie-Beni index

Case	**P-value**	$\hat{\sigma}$	N^*	N^f	**Adj. Threshold**	**Reject** H_0?
10	2.279e-06	2.282e-06	461.0	222.6	2.500e-04	True
69	6.859e-06	3.450e-06	686.7	442.0	5.000e-04	True
187	2.496e-05	1.680e-05	761.7	330.5	7.500e-04	True
1	3.303e-05	2.274e-05	626.9	323.2	1.000e-03	True
6	3.675e-05	2.018e-05	1896.6	404.4	1.250e-03	True
72	1.034e-04	3.678e-05	891.3	623.3	1.500e-03	True
55	1.071e-04	6.939e-05	459.3	343.7	1.750e-03	True
168	1.187e-04	7.386e-05	1142.5	357.1	2.000e-03	True
73	1.322e-04	7.248e-05	1843.1	405.0	2.250e-03	True
63	1.674e-04	7.978e-05	853.8	465.8	2.500e-03	True
120	2.700e-04	1.299e-04	883.7	461.5	2.750e-03	True
86	3.904e-04	2.873e-04	252.5	303.0	3.000e-03	True
127	5.130e-04	1.169e-04	160.5	1023.9	3.250e-03	True
151	5.478e-04	1.427e-04	1353.4	847.8	3.500e-03	True
59	6.734e-04	1.873e-04	721.5	798.0	3.750e-03	True
81	8.217e-04	2.970e-04	1119.5	612.7	4.000e-03	True
20	9.071e-04	2.870e-04	638.9	701.9	4.250e-03	True
196	1.070e-03	6.751e-04	624.7	352.1	4.500e-03	True
14	1.165e-03	5.948e-04	1494.3	434.2	4.750e-03	True
52	1.201e-03	3.466e-04	37.6	920.4	5.000e-03	True
66	1.308e-03	6.065e-04	321.3	480.4	5.250e-03	True
39	1.391e-03	4.609e-04	612.7	670.2	5.500e-03	True
191	1.539e-03	1.193e-03	676.5	286.7	5.750e-03	True
159	1.914e-03	4.069e-04	562.2	1048.9	6.000e-03	True
163	2.223e-03	6.754e-04	689.6	729.4	6.250e-03	True
78	2.223e-03	8.772e-04	581.0	562.2	6.500e-03	True
139	2.294e-03	4.264e-04	634.3	1199.2	6.750e-03	True
96	3.224e-03	8.027e-04	1851.5	883.2	7.000e-03	True
71	3.415e-03	9.283e-04	734.8	814.1	7.250e-03	True
7	5.169e-03	2.610e-03	538.3	438.3	7.500e-03	True
36	5.533e-03	1.638e-03	889.8	744.7	7.750e-03	True

Case	**P-value**	$\hat{\sigma}$	N^*	N^f	**Adj. Threshold**	**Reject** H_0?
97	5.597e-03	1.232e-03	982.8	1000.6	8.000e-03	True
11	6.528e-03	1.980e-03	494.8	729.8	8.250e-03	True
48	6.804e-03	1.025e-03	935.4	1466.6	8.500e-03	True
25	7.021e-03	1.408e-03	622.0	1105.2	8.750e-03	True
82	7.041e-03	1.615e-03	654.2	963.4	9.000e-03	True
30	7.061e-03	9.834e-04	766.1	1598.7	9.250e-03	True
171	7.815e-03	1.689e-03	690.5	1021.5	9.500e-03	True
56	7.919e-03	1.040e-03	949.8	1685.3	9.750e-03	True
129	8.195e-03	2.885e-03	1114.8	624.4	1.000e-02	True
74	8.484e-03	1.303e-03	422.2	1478.1	1.025e-02	True
9	8.511e-03	1.486e-03	310.0	1308.5	1.050e-02	True
183	9.957e-03	1.716e-03	1155.3	1270.8	1.075e-02	True
34	1.285e-02	1.252e-03	1614.3	2241.0	1.100e-02	False
115	1.287e-02	4.771e-03	909.9	590.9	1.125e-02	False
84	1.367e-02	2.032e-03	1081.2	1472.0	1.150e-02	False
95	1.369e-02	1.580e-03	1630.1	1883.1	1.175e-02	False
189	1.370e-02	1.943e-03	1357.2	1537.1	1.200e-02	False
53	1.393e-02	1.814e-03	1071.1	1684.3	1.225e-02	False
119	1.412e-02	2.103e-03	898.7	1474.1	1.250e-02	False
2	1.506e-02	2.706e-03	828.6	1218.6	1.275e-02	False
142	1.784e-02	1.859e-03	1504.0	2086.3	1.300e-02	False
161	1.828e-02	4.022e-03	600.4	995.1	1.325e-02	False
83	2.133e-02	3.007e-03	1226.6	1537.1	1.350e-02	False
128	2.157e-02	3.362e-03	890.1	1397.6	1.375e-02	False
150	2.168e-02	3.871e-03	966.1	1215.1	1.400e-02	False
3	2.323e-02	4.142e-03	541.3	1231.1	1.425e-02	False
130	2.523e-02	3.753e-03	1153.2	1451.6	1.450e-02	False
85	2.545e-02	2.583e-03	635.2	2230.7	1.475e-02	False
195	3.165e-02	4.230e-03	794.2	1626.0	1.500e-02	False
192	3.326e-02	4.615e-03	551.2	1585.2	1.525e-02	False
16	3.420e-02	5.165e-03	1123.3	1417.4	1.550e-02	False
146	3.589e-02	3.325e-03	1181.9	2338.0	1.575e-02	False
169	3.614e-02	3.756e-03	1637.0	2047.8	1.600e-02	False
37	3.690e-02	2.223e-03	2059.8	3574.6	1.625e-02	False
28	3.775e-02	8.974e-03	802.5	898.3	1.650e-02	False
38	4.026e-02	4.076e-03	1552.6	2097.7	1.675e-02	False
108	4.600e-02	4.551e-03	1017.5	2174.1	1.700e-02	False
67	4.961e-02	5.419e-03	202.6	2417.6	1.725e-02	False
43	5.022e-02	5.881e-03	467.7	1897.7	1.750e-02	False
29	5.689e-02	7.945e-03	876.0	1507.5	1.775e-02	False
35	5.951e-02	6.846e-03	994.1	1829.8	1.800e-02	False
79	6.423e-02	1.079e-02	155.8	1374.7	1.825e-02	False
77	6.437e-02	4.647e-03	489.6	3461.0	1.850e-02	False
131	6.778e-02	5.957e-03	713.0	2488.0	1.875e-02	False
44	6.990e-02	9.628e-03	574.4	1532.3	1.900e-02	False
23	7.196e-02	6.139e-03	604.1	2618.7	1.925e-02	False
178	9.143e-02	1.158e-02	341.5	1706.2	1.950e-02	False
4	9.690e-02	7.884e-03	1249.7	2503.6	1.975e-02	False

Case	**P-value**	$\hat{\sigma}$	N^*	N^f	**Adj. Threshold**	**Reject** H_0?
193	1.003e-01	9.176e-03	245.0	2882.4	2.000e-02	False
113	1.022e-01	1.334e-02	1536.7	1512.6	2.025e-02	False
141	1.089e-01	6.321e-03	1622.8	3515.1	2.050e-02	False
26	1.096e-01	1.056e-02	438.2	2255.9	2.075e-02	False
157	1.149e-01	8.226e-03	1014.9	2878.2	2.100e-02	False
167	1.164e-01	1.100e-02	1357.1	2076.9	2.125e-02	False
17	1.190e-01	1.210e-02	1270.7	1922.0	2.150e-02	False
21	1.214e-01	6.609e-03	1188.1	3907.7	2.175e-02	False
148	1.215e-01	7.811e-03	1275.4	3158.0	2.200e-02	False
12	1.221e-01	6.303e-03	1780.1	3920.0	2.225e-02	False
54	1.221e-01	8.294e-03	860.8	3110.0	2.250e-02	False
61	1.239e-01	1.913e-02	494.7	1279.1	2.275e-02	False
0	1.382e-01	1.476e-02	476.8	1897.2	2.300e-02	False
173	1.392e-01	1.742e-02	531.4	1568.5	2.325e-02	False
170	1.397e-01	9.052e-03	996.9	3153.8	2.350e-02	False
140	1.408e-01	8.900e-03	924.1	3288.9	2.375e-02	False
188	1.443e-01	9.662e-03	984.5	3016.5	2.400e-02	False
51	1.445e-01	1.151e-02	831.1	2502.1	2.425e-02	False
98	1.473e-01	2.986e-02	34.1	1676.5	2.450e-02	False
147	1.526e-01	1.170e-02	381.2	3075.1	2.475e-02	False
88	1.533e-01	9.779e-03	1071.8	3125.1	2.500e-02	False
92	1.556e-01	9.713e-03	1363.9	3101.3	2.525e-02	False
175	1.577e-01	2.118e-02	927.8	1381.9	2.550e-02	False
76	1.595e-01	9.149e-03	986.1	3599.5	2.575e-02	False
122	1.794e-01	8.812e-03	1565.7	3931.8	2.600e-02	False
41	1.861e-01	1.143e-02	769.2	3337.2	2.625e-02	False
180	1.890e-01	1.122e-02	1885.7	3034.4	2.650e-02	False
116	1.905e-01	2.042e-02	324.6	1849.1	2.675e-02	False
138	1.910e-01	1.760e-02	280.7	2387.2	2.700e-02	False
93	1.927e-01	1.097e-02	1148.0	3367.6	2.725e-02	False
199	1.943e-01	2.653e-02	153.4	1539.4	2.750e-02	False
179	1.959e-01	1.652e-02	574.6	2273.7	2.775e-02	False
121	1.981e-01	1.555e-02	1061.8	2298.1	2.800e-02	False
33	2.071e-01	1.272e-02	1701.8	2864.1	2.825e-02	False
182	2.179e-01	1.134e-02	932.7	3846.1	2.850e-02	False
89	2.231e-01	1.841e-02	714.6	2167.1	2.875e-02	False
31	2.294e-01	2.417e-02	566.8	1647.5	2.900e-02	False
47	2.349e-01	1.391e-02	766.0	3251.4	2.925e-02	False
177	2.362e-01	1.260e-02	1114.3	3437.0	2.950e-02	False
194	2.491e-01	1.377e-02	1129.0	3189.5	2.975e-02	False
102	2.508e-01	1.342e-02	1362.3	3208.8	3.000e-02	False
197	2.586e-01	1.390e-02	1395.9	3131.6	3.025e-02	False
166	2.589e-01	2.133e-02	425.7	2228.7	3.050e-02	False
124	2.603e-01	1.671e-02	588.7	2966.9	3.075e-02	False
123	2.740e-01	1.204e-02	1105.4	4308.2	3.100e-02	False
107	2.785e-01	2.257e-02	176.4	4539.0	3.125e-02	False
19	2.793e-01	2.190e-02	282.0	2755.3	3.150e-02	False
8	3.054e-01	1.757e-02	502.1	3578.4	3.175e-02	False

Case	**P-value**	$\hat{\sigma}$	N^*	N^f	**Adj. Threshold**	**Reject** H_0?
46	3.102e-01	1.399e-02	1226.5	3669.6	3.200e-02	False
118	3.175e-01	2.605e-02	434.6	1951.7	3.225e-02	False
101	3.199e-01	1.441e-02	1020.3	3837.9	3.250e-02	False
144	3.200e-01	1.318e-02	1020.8	4575.8	3.275e-02	False
137	3.221e-01	2.454e-02	203.2	4131.5	3.300e-02	False
22	3.253e-01	1.826e-02	612.0	3185.3	3.325e-02	False
40	3.310e-01	1.662e-02	671.9	3754.1	3.350e-02	False
65	3.386e-01	2.452e-02	454.9	2202.7	3.375e-02	False
114	3.483e-01	1.527e-02	963.6	3837.9	3.400e-02	False
176	3.500e-01	1.387e-02	1149.3	4253.9	3.425e-02	False
18	3.743e-01	2.816e-02	395.3	1928.3	3.450e-02	False
104	3.822e-01	2.620e-02	448.0	2095.9	3.475e-02	False
15	3.825e-01	1.747e-02	756.1	3650.7	3.500e-02	False
133	3.929e-01	1.649e-02	987.4	3524.4	3.525e-02	False
75	4.076e-01	1.852e-02	671.8	3717.2	3.550e-02	False
125	4.151e-01	1.908e-02	668.6	3464.2	3.575e-02	False
165	4.460e-01	1.520e-02	1062.4	4525.3	3.600e-02	False
181	4.510e-01	1.143e-02	1883.7	5905.7	3.625e-02	False
153	4.602e-01	1.364e-02	1296.5	5482.1	3.650e-02	False
132	4.648e-01	2.302e-02	481.8	2879.5	3.675e-02	False
49	4.725e-01	5.014e-02	91.4	2763.9	3.700e-02	False
70	4.756e-01	1.349e-02	1399.3	4730.9	3.725e-02	False
105	4.792e-01	1.558e-02	1055.3	4022.2	3.750e-02	False
64	4.858e-01	3.186e-02	235.8	4431.8	3.775e-02	False
184	5.052e-01	1.272e-02	1549.5	6062.1	3.800e-02	False
32	5.219e-01	1.946e-02	636.7	3258.8	3.825e-02	False
42	5.284e-01	1.692e-02	896.1	5065.3	3.850e-02	False
126	5.293e-01	1.344e-02	1347.2	4955.3	3.875e-02	False
27	5.462e-01	2.222e-02	519.6	3834.5	3.900e-02	False
45	5.519e-01	2.050e-02	588.4	3738.1	3.925e-02	False
174	5.543e-01	2.113e-02	521.0	3200.8	3.950e-02	False
152	5.579e-01	1.305e-02	1466.9	5850.4	3.975e-02	False
58	5.632e-01	1.802e-02	831.1	5336.0	4.000e-02	False
154	5.752e-01	1.268e-02	1622.2	6664.3	4.025e-02	False
94	5.755e-01	3.394e-02	165.7	1662.2	4.050e-02	False
111	5.776e-01	1.282e-02	1467.6	5725.1	4.075e-02	False
110	5.823e-01	1.338e-02	1428.6	6126.2	4.100e-02	False
5	5.856e-01	3.699e-02	235.3	5183.7	4.125e-02	False
198	5.953e-01	1.682e-02	802.9	4243.0	4.150e-02	False
109	5.974e-01	1.286e-02	1213.9	4732.9	4.175e-02	False
190	5.989e-01	2.607e-02	323.8	2725.3	4.200e-02	False
24	6.012e-01	2.088e-02	600.7	4312.8	4.225e-02	False
112	6.143e-01	3.462e-02	205.0	2465.9	4.250e-02	False
80	6.190e-01	1.260e-02	1424.6	5743.2	4.275e-02	False
143	6.315e-01	1.474e-02	1184.3	5898.9	4.300e-02	False
103	6.729e-01	2.845e-02	415.3	4796.2	4.325e-02	False
156	6.765e-01	2.070e-02	643.5	4854.2	4.350e-02	False
99	6.786e-01	1.543e-02	780.6	4427.1	4.375e-02	False

Case	**P-value**	$\hat{\sigma}$	N^*	N^f	**Adj. Threshold**	**Reject** H_0?
145	6.794e-01	1.764e-02	907.5	5776.9	4.400e-02	False
149	6.800e-01	2.186e-02	255.7	2385.0	4.425e-02	False
186	6.858e-01	6.029e-02	124.6	6917.7	4.450e-02	False
90	7.066e-01	1.805e-02	391.2	3097.2	4.475e-02	False
117	7.087e-01	1.177e-02	1137.8	5296.2	4.500e-02	False
100	7.391e-01	1.112e-02	1092.4	5303.0	4.525e-02	False
136	7.478e-01	1.476e-02	543.0	3828.4	4.550e-02	False
50	7.635e-01	1.429e-02	832.4	5035.1	4.575e-02	False
60	7.657e-01	1.484e-02	436.6	3475.8	4.600e-02	False
91	7.797e-01	1.813e-02	328.7	3141.1	4.625e-02	False
57	7.819e-01	1.405e-02	936.2	5514.7	4.650e-02	False
155	7.835e-01	1.662e-02	271.4	2799.9	4.675e-02	False
134	7.837e-01	9.483e-03	1386.3	6213.1	4.700e-02	False
160	7.977e-01	1.375e-02	797.9	5047.6	4.725e-02	False
68	8.038e-01	1.182e-02	507.1	3852.8	4.750e-02	False
162	8.241e-01	1.158e-02	711.0	4714.7	4.775e-02	False
185	8.396e-01	8.741e-03	949.5	5387.9	4.800e-02	False
13	8.517e-01	1.852e-02	900.5	6529.9	4.825e-02	False
106	8.593e-01	1.142e-02	838.2	5347.6	4.850e-02	False
135	9.006e-01	1.377e-02	332.4	3540.9	4.875e-02	False
164	9.045e-01	7.289e-03	948.4	5710.4	4.900e-02	False
62	9.278e-01	9.157e-03	190.0	2725.4	4.925e-02	False
158	9.328e-01	8.493e-03	493.8	4331.1	4.950e-02	False
172	9.370e-01	6.604e-03	685.3	5038.2	4.975e-02	False
87	9.580e-01	3.054e-03	1035.0	6131.6	5.000e-02	False

Table B.1: Results based on the use of Xie-Beni index. N^* denotes the effective sample size based on the function-independent cv^2 and N^f denotes the effective sample size based on the function-dependent cv^2.

PBMF index

Case	**P-value**	$\hat{\sigma}$	N^*	N^f	**Adj. Threshold**	**Reject** H_0?
41	1.801e-02	3.302e-03	769.2	1192.3	2.500e-04	False
35	1.950e-02	3.231e-03	994.1	1312.8	5.000e-04	False
95	2.181e-02	2.019e-03	1630.1	2339.6	7.500e-04	False
56	2.618e-02	4.551e-03	949.8	1243.1	1.000e-03	False
99	2.855e-02	3.315e-03	780.6	1892.2	1.250e-03	False
156	3.156e-02	5.532e-03	643.5	1236.3	1.500e-03	False
78	3.175e-02	3.398e-03	581.0	2102.5	1.750e-03	False
170	3.292e-02	3.313e-03	996.9	2166.8	2.000e-03	False
37	3.436e-02	2.775e-03	2059.8	2634.7	2.250e-03	False
54	3.576e-02	3.481e-03	860.8	2258.7	2.500e-03	False
120	4.223e-02	3.821e-03	883.7	2426.6	2.750e-03	False
167	4.464e-02	3.896e-03	1357.1	2449.9	3.000e-03	False
7	4.721e-02	9.871e-03	538.3	1019.1	3.250e-03	False

Case	**P-value**	$\hat{\sigma}$	N^*	N^f	**Adj. Threshold**	**Reject** H_0?
103	4.912e-02	6.131e-03	415.3	1785.3	3.500e-03	False
77	4.928e-02	3.845e-03	489.6	3135.3	3.750e-03	False
135	4.944e-02	5.242e-03	332.4	2236.8	4.000e-03	False
128	4.981e-02	4.824e-03	890.1	2233.4	4.250e-03	False
187	4.984e-02	4.158e-03	761.7	2676.7	4.500e-03	False
34	5.067e-02	3.849e-03	1614.3	2793.9	4.750e-03	False
49	5.095e-02	7.682e-03	91.4	1879.2	5.000e-03	False
27	5.099e-02	1.340e-02	519.6	805.0	5.250e-03	False
158	5.731e-02	5.867e-03	493.8	2190.3	5.500e-03	False
123	5.844e-02	5.776e-03	1105.4	2138.3	5.750e-03	False
9	5.906e-02	8.697e-03	310.0	1499.8	6.000e-03	False
192	5.978e-02	5.716e-03	551.2	2338.9	6.250e-03	False
23	6.004e-02	1.218e-02	604.1	1033.5	6.500e-03	False
40	6.125e-02	5.481e-03	671.9	2469.2	6.750e-03	False
38	6.223e-02	3.680e-03	1552.6	3642.9	7.000e-03	False
82	6.258e-02	7.050e-03	654.2	1905.8	7.250e-03	False
195	6.436e-02	5.985e-03	794.2	2318.1	7.500e-03	False
43	6.662e-02	9.524e-03	467.7	1493.6	7.750e-03	False
51	6.691e-02	6.666e-03	831.1	2135.1	8.000e-03	False
50	6.917e-02	5.454e-03	832.4	2772.9	8.250e-03	False
191	6.938e-02	5.736e-03	676.5	2687.3	8.500e-03	False
97	6.957e-02	5.298e-03	982.8	2835.5	8.750e-03	False
172	6.980e-02	5.146e-03	685.3	3092.2	9.000e-03	False
90	7.029e-02	9.179e-03	391.2	1665.3	9.250e-03	False
10	7.042e-02	6.390e-03	461.0	2530.1	9.500e-03	False
89	7.042e-02	1.168e-02	714.6	1250.7	9.750e-03	False
190	7.089e-02	8.839e-03	323.8	1790.8	1.000e-02	False
150	7.216e-02	1.323e-02	966.1	1120.6	1.025e-02	False
98	7.235e-02	1.584e-02	34.1	1474.0	1.050e-02	False
107	7.444e-02	7.449e-03	176.4	2951.3	1.075e-02	False
118	7.696e-02	8.435e-03	434.6	2005.3	1.100e-02	False
152	7.708e-02	5.321e-03	1466.9	3029.6	1.125e-02	False
148	7.873e-02	5.381e-03	1275.4	3095.1	1.150e-02	False
161	7.937e-02	7.254e-03	600.4	2391.1	1.175e-02	False
55	7.981e-02	5.738e-03	459.3	3495.9	1.200e-02	False
79	8.022e-02	8.391e-03	155.8	2877.8	1.225e-02	False
76	8.095e-02	7.356e-03	986.1	2297.3	1.250e-02	False
11	8.425e-02	9.404e-03	494.8	1920.8	1.275e-02	False
169	8.460e-02	6.000e-03	1637.0	2894.2	1.300e-02	False
60	8.474e-02	8.489e-03	436.6	2215.1	1.325e-02	False
115	8.558e-02	7.034e-03	909.9	2571.4	1.350e-02	False
127	8.779e-02	8.600e-03	160.5	3278.5	1.375e-02	False
68	8.862e-02	7.546e-03	507.1	2648.2	1.400e-02	False
91	9.304e-02	8.595e-03	328.7	2585.3	1.425e-02	False
109	9.358e-02	6.003e-03	1213.9	3290.8	1.450e-02	False
119	9.370e-02	7.729e-03	898.7	2540.0	1.475e-02	False
132	9.391e-02	7.577e-03	481.8	2858.2	1.500e-02	False
46	9.394e-02	6.554e-03	1226.5	2982.4	1.525e-02	False

Case	**P-value**	$\hat{\sigma}$	N^*	N^f	**Adj. Threshold**	**Reject** H_0?
112	9.430e-02	9.397e-03	205.0	2685.9	1.550e-02	False
75	9.446e-02	8.813e-03	671.8	2265.8	1.575e-02	False
25	9.473e-02	1.872e-02	622.0	1019.4	1.600e-02	False
130	9.605e-02	6.664e-03	1153.2	3012.8	1.625e-02	False
188	9.774e-02	1.452e-02	984.5	1345.7	1.650e-02	False
100	9.975e-02	7.732e-03	1092.4	2658.4	1.675e-02	False
94	1.024e-01	1.038e-02	165.7	2893.8	1.700e-02	False
124	1.025e-01	8.674e-03	588.7	2564.2	1.725e-02	False
153	1.038e-01	1.104e-02	1296.5	1868.7	1.750e-02	False
6	1.053e-01	6.306e-03	1896.6	3358.8	1.775e-02	False
145	1.055e-01	1.091e-02	907.5	1950.5	1.800e-02	False
185	1.077e-01	8.397e-03	949.5	2649.9	1.825e-02	False
137	1.081e-01	1.257e-02	203.2	2047.2	1.850e-02	False
143	1.102e-01	8.482e-03	1184.3	2629.6	1.875e-02	False
18	1.103e-01	8.752e-03	395.3	3027.8	1.900e-02	False
65	1.112e-01	1.491e-02	454.9	1522.4	1.925e-02	False
168	1.114e-01	6.523e-03	1142.5	3628.1	1.950e-02	False
61	1.117e-01	1.456e-02	494.7	1561.7	1.975e-02	False
114	1.122e-01	8.845e-03	963.6	2598.1	2.000e-02	False
64	1.125e-01	1.230e-02	235.8	2147.2	2.025e-02	False
164	1.129e-01	1.005e-02	948.4	2268.3	2.050e-02	False
52	1.139e-01	2.166e-02	37.6	1932.7	2.075e-02	False
36	1.153e-01	1.068e-02	889.8	2174.2	2.100e-02	False
155	1.160e-01	1.680e-02	271.4	1448.4	2.125e-02	False
189	1.178e-01	9.733e-03	1357.2	2388.1	2.150e-02	False
62	1.188e-01	1.984e-02	190.0	1265.3	2.175e-02	False
47	1.193e-01	9.462e-03	766.0	2622.4	2.200e-02	False
19	1.216e-01	9.912e-03	282.0	3275.4	2.225e-02	False
39	1.232e-01	1.141e-02	612.7	2228.8	2.250e-02	False
151	1.236e-01	1.001e-02	1353.4	2422.5	2.275e-02	False
92	1.249e-01	1.291e-02	1363.9	1870.2	2.300e-02	False
129	1.259e-01	1.065e-02	1114.8	2332.9	2.325e-02	False
13	1.264e-01	1.402e-02	900.5	1763.4	2.350e-02	False
42	1.264e-01	1.201e-02	896.1	2084.8	2.375e-02	False
113	1.269e-01	1.279e-02	1536.7	1906.2	2.400e-02	False
140	1.323e-01	1.558e-02	924.1	1641.5	2.425e-02	False
44	1.329e-01	1.376e-02	574.4	1944.6	2.450e-02	False
174	1.363e-01	9.078e-03	521.0	3540.9	2.475e-02	False
131	1.368e-01	1.710e-02	713.0	1549.2	2.500e-02	False
67	1.379e-01	1.889e-02	202.6	1568.9	2.525e-02	False
66	1.388e-01	1.250e-02	321.3	2553.3	2.550e-02	False
173	1.391e-01	1.190e-02	531.4	2452.7	2.575e-02	False
162	1.393e-01	1.633e-02	711.0	1655.0	2.600e-02	False
8	1.406e-01	1.859e-02	502.1	1479.3	2.625e-02	False
178	1.408e-01	1.102e-02	341.5	3147.6	2.650e-02	False
193	1.418e-01	1.255e-02	245.0	2915.0	2.675e-02	False
33	1.418e-01	7.534e-03	1701.8	3717.8	2.700e-02	False
88	1.422e-01	1.432e-02	1071.8	1897.5	2.725e-02	False

Case	**P-value**	$\hat{\sigma}$	N^*	N^f	**Adj. Threshold**	**Reject** H_0?
59	1.427e-01	9.331e-03	721.5	3292.8	2.750e-02	False
70	1.433e-01	8.126e-03	1399.3	3520.2	2.775e-02	False
138	1.436e-01	1.132e-02	280.7	3421.3	2.800e-02	False
186	1.447e-01	1.563e-02	124.6	2948.5	2.825e-02	False
57	1.467e-01	9.855e-03	936.2	3016.7	2.850e-02	False
1	1.477e-01	1.072e-02	626.9	2922.7	2.875e-02	False
197	1.492e-01	7.192e-03	1395.9	4289.0	2.900e-02	False
0	1.506e-01	1.414e-02	476.8	2183.9	2.925e-02	False
31	1.531e-01	1.281e-02	566.8	2450.9	2.950e-02	False
58	1.539e-01	1.062e-02	831.1	2939.1	2.975e-02	False
117	1.541e-01	1.097e-02	1137.8	2721.0	3.000e-02	False
81	1.549e-01	1.045e-02	1119.5	2899.1	3.025e-02	False
14	1.557e-01	8.409e-03	1494.3	3640.2	3.050e-02	False
121	1.588e-01	9.737e-03	1061.8	3260.9	3.075e-02	False
20	1.597e-01	1.185e-02	638.9	2780.3	3.100e-02	False
2	1.598e-01	9.758e-03	828.6	3430.9	3.125e-02	False
175	1.618e-01	1.022e-02	927.8	3196.7	3.150e-02	False
171	1.620e-01	1.367e-02	690.5	2331.1	3.175e-02	False
80	1.622e-01	9.048e-03	1424.6	3493.0	3.200e-02	False
159	1.631e-01	1.456e-02	562.2	2231.6	3.225e-02	False
134	1.631e-01	8.782e-03	1386.3	3657.2	3.250e-02	False
139	1.642e-01	9.476e-03	634.3	4037.2	3.275e-02	False
199	1.644e-01	1.739e-02	153.4	2549.0	3.300e-02	False
104	1.646e-01	3.143e-02	448.0	970.5	3.325e-02	False
84	1.648e-01	1.001e-02	1081.2	3264.1	3.350e-02	False
29	1.684e-01	1.620e-02	876.0	1944.7	3.375e-02	False
5	1.709e-01	1.499e-02	235.3	2915.1	3.400e-02	False
166	1.715e-01	2.711e-02	425.7	1176.5	3.425e-02	False
45	1.720e-01	1.444e-02	588.4	2362.8	3.450e-02	False
106	1.740e-01	1.089e-02	838.2	3243.0	3.475e-02	False
154	1.745e-01	1.290e-02	1622.2	2466.4	3.500e-02	False
86	1.756e-01	1.642e-02	252.5	2480.1	3.525e-02	False
63	1.756e-01	8.768e-03	853.8	4529.6	3.550e-02	False
30	1.757e-01	8.762e-03	766.1	4763.6	3.575e-02	False
136	1.806e-01	2.661e-02	543.0	1236.6	3.600e-02	False
142	1.836e-01	1.119e-02	1504.0	3030.4	3.625e-02	False
17	1.837e-01	8.756e-03	1270.7	4242.9	3.650e-02	False
12	1.839e-01	7.961e-03	1780.1	4489.3	3.675e-02	False
93	1.851e-01	8.767e-03	1148.0	4390.7	3.700e-02	False
181	1.883e-01	8.508e-03	1883.7	4180.5	3.725e-02	False
141	1.896e-01	8.849e-03	1622.8	4108.8	3.750e-02	False
15	1.938e-01	1.212e-02	756.1	3224.4	3.775e-02	False
108	1.938e-01	9.132e-03	1017.5	4544.8	3.800e-02	False
83	1.945e-01	1.051e-02	1226.6	3547.7	3.825e-02	False
28	1.949e-01	1.212e-02	802.5	3198.5	3.850e-02	False
198	1.983e-01	1.160e-02	802.9	3475.3	3.875e-02	False
73	1.989e-01	8.218e-03	1843.1	4637.1	3.900e-02	False
110	1.992e-01	1.059e-02	1428.6	3508.7	3.925e-02	False

Case	**P-value**	$\hat{\sigma}$	N^*	N^f	**Adj. Threshold**	**Reject** H_0?
116	2.011e-01	1.940e-02	324.6	2101.8	3.950e-02	False
165	2.038e-01	1.227e-02	1062.4	3121.8	3.975e-02	False
74	2.075e-01	1.670e-02	422.2	2536.3	4.000e-02	False
196	2.100e-01	1.320e-02	624.7	3290.5	4.025e-02	False
24	2.109e-01	1.485e-02	600.7	2802.0	4.050e-02	False
163	2.112e-01	1.565e-02	689.6	2542.3	4.075e-02	False
53	2.138e-01	1.664e-02	1071.1	2259.1	4.100e-02	False
69	2.212e-01	1.306e-02	686.7	3458.0	4.125e-02	False
147	2.257e-01	2.601e-02	381.2	1555.2	4.150e-02	False
26	2.258e-01	1.305e-02	438.2	4639.1	4.175e-02	False
176	2.321e-01	1.208e-02	1149.3	3555.1	4.200e-02	False
71	2.328e-01	1.462e-02	734.8	3027.0	4.225e-02	False
160	2.361e-01	1.394e-02	797.9	3225.6	4.250e-02	False
180	2.414e-01	9.155e-03	1885.7	4848.0	4.275e-02	False
126	2.424e-01	1.007e-02	1347.2	4664.8	4.300e-02	False
149	2.430e-01	1.763e-02	255.7	4019.4	4.325e-02	False
125	2.463e-01	1.404e-02	668.6	3543.5	4.350e-02	False
21	2.466e-01	1.620e-02	1188.1	2566.9	4.375e-02	False
102	2.476e-01	1.302e-02	1362.3	3299.2	4.400e-02	False
3	2.494e-01	1.495e-02	541.3	3530.5	4.425e-02	False
111	2.525e-01	1.061e-02	1467.6	4380.6	4.450e-02	False
105	2.560e-01	1.106e-02	1055.3	4692.0	4.475e-02	False
32	2.593e-01	1.462e-02	636.7	3603.4	4.500e-02	False
194	2.606e-01	1.087e-02	1129.0	4816.6	4.525e-02	False
22	2.645e-01	1.857e-02	612.0	2541.9	4.550e-02	False
177	2.711e-01	1.288e-02	1114.3	3803.1	4.575e-02	False
101	2.807e-01	1.603e-02	1020.3	2924.4	4.600e-02	False
183	2.909e-01	1.401e-02	1155.3	3534.0	4.625e-02	False
16	2.955e-01	1.386e-02	1123.3	3672.3	4.650e-02	False
184	2.981e-01	1.174e-02	1549.5	4332.2	4.675e-02	False
122	3.155e-01	1.207e-02	1565.7	4340.1	4.700e-02	False
72	3.368e-01	1.732e-02	891.3	3080.3	4.725e-02	False
179	3.460e-01	2.081e-02	574.6	2713.5	4.750e-02	False
144	3.542e-01	1.415e-02	1020.8	4453.0	4.775e-02	False
87	3.641e-01	1.419e-02	1035.0	4532.2	4.800e-02	False
96	3.729e-01	1.096e-02	1851.5	5603.2	4.825e-02	False
85	3.836e-01	2.021e-02	635.2	2936.2	4.850e-02	False
133	3.878e-01	1.549e-02	987.4	4048.9	4.875e-02	False
146	4.051e-01	1.549e-02	1181.9	3647.4	4.900e-02	False
182	4.237e-01	1.569e-02	932.7	4610.3	4.925e-02	False
4	4.324e-01	1.358e-02	1249.7	5416.5	4.950e-02	False
48	4.862e-01	1.642e-02	935.4	4034.4	4.975e-02	False
157	4.951e-01	1.570e-02	1014.9	4498.7	5.000e-02	False

Table B.2: Results based on the use of PBMF index. N^* denotes the effective sample size based on the function-independent cv^2 and N^f denotes the effective sample size based on the function-dependent cv^2.

Bibliography

[1] Michael Ashburner et al. "Gene Ontology: tool for the unification of biology". In: *Nature genetics* 25.1 (2000), pp. 25–29.

[2] Michael A Bender et al. "Lowest common ancestors in trees and directed acyclic graphs". In: *Journal of Algorithms* 57.2 (2005), pp. 75–94.

[3] Jeremy M Berg, John L Tymoczko, and Lubert Stryer. "Biochemistry". In: *475–477](WH Freeman and Co., New York, 2011)* (2002).

[4] Ivona Bezáková et al. "Negative examples for sequential importance sampling of binary contingency tables". In: *Algorithms–ESA 2006*. Springer, 2006, pp. 136–147.

[5] Louis J Billera, Susan P Holmes, and Karen Vogtmann. "Geometry of the space of phylogenetic trees". In: *Advances in Applied Mathematics* 27.4 (2001), pp. 733–767.

[6] Mathieu Blanchette et al. "Aligning multiple genomic sequences with the threaded blockset aligner". In: *Genome research* 14.4 (2004), pp. 708–715.

[7] Hans-Hermann Bock. "On some significance tests in cluster analysis". In: *Journal of classification* 2.1 (1985), pp. 77–108.

[8] Sean X Chen and Jun S Liu. "Statistical applications of the Poisson-binomial and conditional Bernoulli distributions". In: *Statistica Sinica* 7.4 (1997), pp. 875–892.

[9] Yuguo Chen et al. "Sequential Monte Carlo methods for statistical analysis of tables". In: *Journal of the American Statistical Association* 100.469 (2005), pp. 109–120.

[10] C Chow and C Liu. "Approximating discrete probability distributions with dependence trees". In: *Information Theory, IEEE Transactions on* 14.3 (1968), pp. 462–467.

[11] George W Cobb and Yung-Pin Chen. "An application of Markov chain Monte Carlo to community ecology". In: *American Mathematical Monthly* (2003), pp. 265–288.

[12] Charles Darwin. "On the origin of species". In: *Murray, London* (1859), p. 360.

[13] Richard Desper and Olivier Gascuel. "The minimum evolution distance-based approach to phylogenetic inference". In: *Mathematics of evolution and phylogeny* (2005), pp. 1–32.

[14] Persi Diaconis and Anil Gangolli. *Rectangular arrays with fixed margins*. Springer, 1995.

[15] Joseph Felsenstein. "Cases in which parsimony or compatibility methods will be positively misleading". In: *Systematic Biology* 27.4 (1978), pp. 401–410.

[16] Thomas H Jukes and Charles R Cantor. "Evolution of protein molecules". In: *Mammalian protein metabolism* 3 (1969), pp. 21–132.

[17] Junhyong Kim. "Slicing hyperdimensional oranges: the geometry of phylogenetic estimation". In: *Molecular phylogenetics and evolution* 17.1 (2000), pp. 58–75.

[18] Art B. Owen. *Monte Carlo theory, methods and examples.* 2013.

[19] Lior Pachter. *The phylogeny of oranges is a point in a phylogenetic orange.* [Online; posted 26-August-2013]. Aug. 2013. URL: `https://liorpachter.wordpress.com/2013/08/26/the-phylogeny-of-orange-is-a-point-in-a-phylogenetic-orange/`.

[20] Lior Pachter and Bernd Sturmfels. *Algebraic statistics for computational biology.* Vol. 13. Cambridge University Press, 2005.

[21] Malay K Pakhira, Sanghamitra Bandyopadhyay, and Ujjwal Maulik. "Validity index for crisp and fuzzy clusters". In: *Pattern recognition* 37.3 (2004), pp. 487–501.

[22] E. Paradis, J. Claude, and K. Strimmer. "APE: analyses of phylogenetics and evolution in Rlanguage". In: *Bioinformatics* 20 (2004), pp. 289–290.

[23] Matteo Pellegrini et al. "Assigning protein functions by comparative genome analysis: protein phylogenetic profiles". In: *Proceedings of the National Academy of Sciences* 96.8 (1999), pp. 4285–4288.

[24] DavidF Robinson and Leslie R Foulds. "Comparison of phylogenetic trees". In: *Mathematical Biosciences* 53.1 (1981), pp. 131–147.

[25] Naruya Saitou and Masatoshi Nei. "The neighbor-joining method: a new method for reconstructing phylogenetic trees." In: *Molecular biology and evolution* 4.4 (1987), pp. 406–425.

[26] Tom AB Snijders. "Enumeration and simulation methods for 0–1 matrices with given marginals". In: *Psychometrika* 56.3 (1991), pp. 397–417.

[27] Gergely J Szöllősi and Vincent Daubin. "Modeling gene family evolution and reconciling phylogenetic discord". In: *Evolutionary Genomics.* Springer, 2012, pp. 29–51.

[28] Simon Tavaré. "Some probabilistic and statistical problems in the analysis of DNA sequences". In: *Lectures on mathematics in the life sciences* 17 (1986), pp. 57–86.

[29] *Vertebrate Multiz Alignment & Conservation (100 Species).* `http://genome.ucsc.edu/cgi-bin/hgTrackUi?db=hg19&g=cons100way`. Accessed: 2013-11-27.

[30] Weina Wang and Yunjie Zhang. "On fuzzy cluster validity indices". In: *Fuzzy sets and systems* 158.19 (2007), pp. 2095–2117.

[31] Xuanli Lisa Xie and Gerardo Beni. "A validity measure for fuzzy clustering". In: *IEEE Transactions on pattern analysis and machine intelligence* 13.8 (1991), pp. 841–847.