BACHELOR'S THESIS

Degree in Mathematics

Title: Linear invariants for the equal-input evolutionary model

Author: Alba Puy Tapia

Advisor: Marta Casanellas Rius

Department: Matemàtiques (MAT)

Academic year: : 2016 - 2017

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat de Matemàtiques i Estadística

Universitat Politècnica de Catalunya
Facultat de Matemàtiques i Estadística

Degree in Mathematics

Bachelor's Degree Thesis

Linear invariants for the equal-input evolutionary model

Alba Puy Tapia

Supervised by Marta Casanellas Rius

June, 2017

Thanks to Marta Casanellas for being my supervisor. Thanks to my family, my boyfriend and friends for giving me hope even when I couldn't see the light during these years. Thanks to Jaume Franch, Pere Pascual, Narciso Román and other professors for teaching me not only the mathematic issues a mathematician should know, but also for making me become one.

Abstract

Mathematical models for describing biological systems are becoming more and more important nowadays. In this thesis we will study Markov processes on phylogenetic trees and deeply study the equal-input model, which is the first non-trivial model that presents the interesting properties of simple models. We will study widely this model, talking about its invariants, computing the linear ones for the case of $n = 5$ and studying the linear equations that satisfy all the distributions arising from it. We will also talk about some notions of phylogenetic trees and the distinction between topology and model invariants, understanding the generalization of these processes to phylogenetic mixtures.

Keywords

Algebraic invariant, algebraic phylogenetic tree, evolutionary model

Contents

1. Introduction

"Every new body of discovery is mathematical in form, because there is no other guidance we can have." Charles Darwin.

Through all times, biologists have tried to find out the origin of all the species trying to piece together parts of this 'tree of life' based on the observations they have. However, new improvements in mathematics made us, the mathematicians, play an important role in transforming genomic data into phylogenetic trees, useful to find more hints about where we came from.

The modeling of the evolution of the species in a phylogenetic tree is done according to Darwin's theory of natural selection. In a phylogenetic tree the leaves represent the current species, the root the common ancestor and the edges the speciation process.

Nowadays, studying the evolutionary relationship of the species implies the study of the relation between the molecules of DNA associated to them. These molecules usually correspond to genes and, thanks the double helix symmetry, could be thought as a sequence of nucleotides: adenine (A), cytosine (C), guanine (G) or thymine (T).

The main goal of phylogenetics is to get, from the DNA sequences of a group of current species, the reconstruction of the ancestral relations between them. To this aim evolutionary nucleotide substitution models have been introduced since 1969 and these have been since then, the basis of phylogenetic reconstruction.

In this line, the main goals of this project are:

- Studying Markov processes on phylogenetic trees and deeply study the properties of the equal-input model (explained in section 2.3). Understanding the generalization of these processes to phylogenetic mixtures (explained in section 4).
- Understanding the notions of invariants on phylogenetic trees and the distinction between topology and model invariants (explained in section 3.1).
- Study the linear equations that satisfy all the distributions arising from an equal-input Markov process on a phylogenetic tree based on the previous study of Marta Casanellas and Mike Steel (explained in section 3.2).
- Compute explicitly generators of the space of linear invariants for phylogenetic trees on five leaves evolving under the equal-input model (explained in section 5).

2. Background

2.1 Phylogenetic trees

We introduce here some notions that we are going to use in the following pages:

Definition 2.1. A *graph* $G = (N, E)$ consists of a set N of nodes and a set $E \subseteq \binom{N}{2}$ of edges connecting pairs of nodes.

Definition 2.2. The *degree* of a node in a graph is the number of edges adjacent to it.

Definition 2.3. A *subgraph* of a graph is another graph, formed from a subset of the nodes of the graph and all of the edges connecting pairs of nodes in that subset.

Definition 2.4. A *tree* is a graph that is connected and has no cycles. There are many alternative characterizations of trees; for example, the following are equivalent:

- $T = (N, E)$ is a tree.
- $T = (N, E)$ is connected and it has $|N| - 1$ edges.
- For any two nodes u and v of T, there is a unique path from u to v.

Definition 2.5. The *leaves* of a tree are the nodes whose degree is one. We call also the edges of a tree *branches*.

Definition 2.6. A *phylogenetic tree* on a set of species S is a tree T together with a bijection ψ from the set of leaves of T to S is a branching diagram showing the inferred evolutionary relationships among various biological species based on similarities and differences in their genetic characteristics. Each leaf is related with a existing species and the internal nodes are the common ancestors. The branches between two nodes represent the evolutionary processes between them.

The bijection ψ labels the leaves of the tree with species in S. So, two trees are topologically equivalent if there exists a graph isomorphism between both that preserves the labeling of the leaves.

Example 2.7.

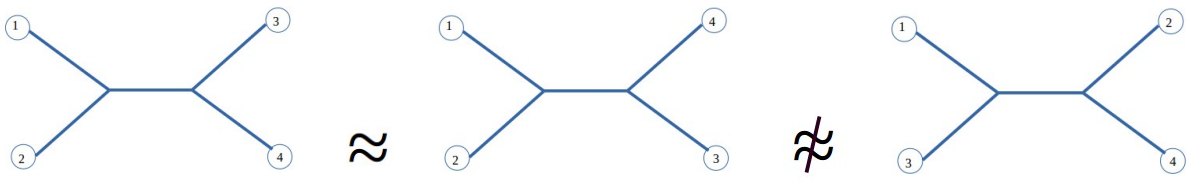
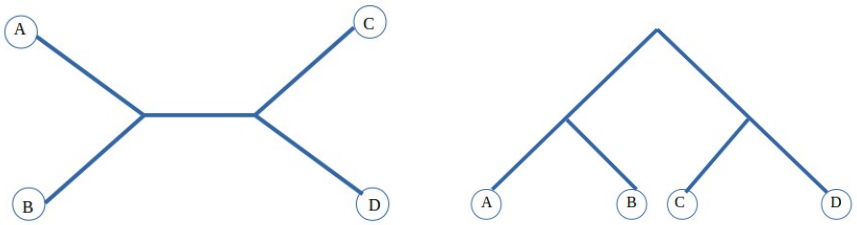


Figure 1: Phylogenetic trees with different topology

Phylogenetic trees can have a distinguished interior node, called the root, and then edges which are directed out of the root. A root represents a common ancestor for all the current species of the tree. Rooted trees are called binary if every branch is divided in two other branches until we arrive to the leaves. It is impossible with only the DNA of the current species and the evolutionary models to reconstruct the exact position of this root, so that, the most phylogenetic methods consider trees without root.

Example 2.8.

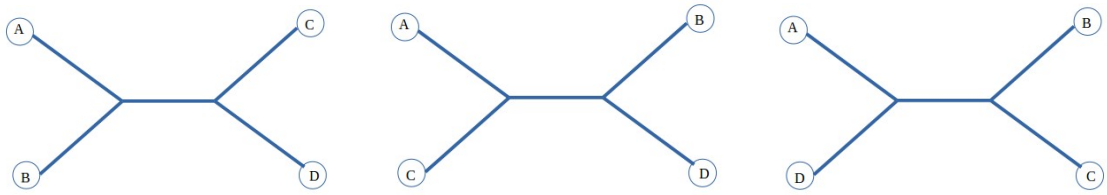


(a) Tree rootless (b) Tree with root

Figure 2: Trees without and with root

The next notation is going to be used to express the different topologies of a tree: we will write: $AB|CD$ if we consider a four leaves tree to express that leaves A and B are together and C and D. See example 2.9.

Example 2.9. This example shows the three possible different topologies for a four leaved tree:



(a) $AB|CD$ (b) $AC|BD$ (c) $AD|BC$

Figure 3: The three different topologies for unrooted trees with 4 leaves.

Phylogenetic trees can come also with lengths assigned to the branches. In phylogenetics the length of a branch represents the evolutionary distance between the nodes in the branch (and it is usually measured as the elapsed amount of mutations between both DNA sequences).

2.2 DNA alignments

Nowadays it is usual to look at the DNA molecules in the genome of the species to reconstruct phylogenetic trees. The DNA is a nucleic acid, formed by smaller molecules called nucleotides. A nucleotide is composed of a sugar, a phosphate group and a nitrogen-containing base, which determines the type of the nucleotide.

Due to various processes of mutation, suppression or insertion of nucleotides, the DNA of the same gen in different species is not equal, in fact, it usually present some similar regions and other that are difficult to compared. Also, the genes might appear in different regions due that the genome of different species has different number of nucleotides, chromosomes and genes. That is why, before starting to study the ancestral relations between the species is really important to know which parts of the genome of each species give the same information. This information is kept in an alignment of DNA sequences.

Definition 2.10. An *alignment* is a table in which rows are the DNA sequences of each species of study are and each column corresponds to the nucleotides which have evolved from the same nucleotide in the common ancestor.

In order to simplify, in this thesis only are going to be taken the mutation events into account, forgetting about the suppression and insertion events, as is usually done in phylogenetics.

Example 2.11.

Gorilla Gorilla	AACTTCGAGGCTTACCGCTG
Homo Sapiens	AACGTCTATGCTCACCGATG
Pan Troglodytes	AAGGTCGATGCTCACCGATG

Table 1: Alignment of DNA chains of the species *Homo Sapiens* (humans), *Pan Troglodytes* (chimpanzee) and *Gorilla Gorilla* (gorilla).

Once we have an alignment and an evolutionary model, we can compute the relative frequencies of each combination of nucleotides. Since each phylogenetic tree has its own evolutionary characteristics, the probability of obtaining a certain alignment is different for each tree.

There are different methods that use these probabilities to determine the tree that has originated the alignment.

2.3 Markov models: Equal-input model

In order to model the evolutionary process of the species considered to study, the following hypothesis are assumed:

- We consider only binary trees (the internal nodes have degree tree or two if it is the root).
- The evolutionary processes on the adjacent branches only depends on the common node.
- The mutations of the DNA chain occur randomly.
- The different positions in the DNA chain evolve independently one from each other and with the same probability of mutation.

Due to the last hypothesis it is enough to model one position: to each node i it is assigned a discrete random variable X_i with values in $\{A, C, G, T\}$. The variables at the leaves represent observations from the nucleotides of the actual species. Each column of the alignment corresponds to one observation of the random vector $X = (X_1, \dots, X_n)$, being n the number of species considered.

The random variable at the interior nodes are hidden variables (we don't have observations about them). Let T be a phylogenetic tree on a set of n species, either rooted or unrooted. If the tree is rooted then the root naturally defines an orientation on the edges of the tree. If it is not rooted, then we choose an internal node and direct the edges out of it.

Following a Markov process, a substitution matrix S_e is assigned to the edge e. The entries of S_e are the probabilities $P(x|y,e)$ of the nucleotide y at the predecessor node being substituted by the nucleotide x at the child node thorough the evolutionary process represented by e.

$$S_e = \begin{array}{c} \begin{matrix} A & C & G & T \end{matrix} \\ \begin{pmatrix} P(A|A,e) & P(C|A,e) & P(G|A,e) & P(T|A,e) \\ P(A|C,e) & P(C|C,e) & P(G|C,e) & P(T|C,e) \\ P(A|G,e) & P(C|G,e) & P(G|G,e) & P(T|G,e) \\ P(A|T,e) & P(C|T,e) & P(G|T,e) & P(T|T,e) \end{pmatrix} \end{array} \begin{matrix} A \\ C \\ G \\ T \end{matrix}$$

Note that, as they represent probabilities, the sum of the rows must be equal to one. The entries of S_e are unknown and together with the nucleotides distribution at the root $(\pi_A, \pi_C, \pi_G, \pi_T)$ are the parameters of the model where $\pi_A + \pi_C + \pi_G + \pi_T = 1$.

Depending on the structure of the matrix different models can be obtained. For example, if no restriction is imposed at the entries of S_e the most general model is obtained, named *general model of Markov GMM*:

$$S_e = \begin{pmatrix} a_e & b_e & c_e & d_e \\ e_e & f_e & g_e & h_e \\ j_e & k_e & l_e & m_e \\ n_e & o_e & p_e & q_e \end{pmatrix}$$

But if, for example, $\pi_A = \pi_T$ and $\pi_C = \pi_G$ are imposed, the *Strand symmetric model* is obtained, reflecting the symmetry of the DNA molecule. If $\pi_A = \pi_C = \pi_G = \pi_T$, then $b_e = c_e = d_e$ is imposed and the *Jukes-Cantor Model JC* is obtained:

$$S_e = \begin{pmatrix} a_e & b_e & b_e & b_e \\ b_e & a_e & b_e & b_e \\ b_e & b_e & a_e & b_e \\ b_e & b_e & b_e & a_e \end{pmatrix}$$

The parameter a_e in the last model is the probability for a nucleotide of not being substituted.

Remark 2.12. As the addition of all the elements of a row of the matrix should be one, we have just one free parameter per edge for the JC model: $a_e + 3b_e = 1$.

For the general Markov Model there are three free parameters at the root and twelve at each substitution matrix.

Example 2.13. Lets focus on the following phylogenetic tree:

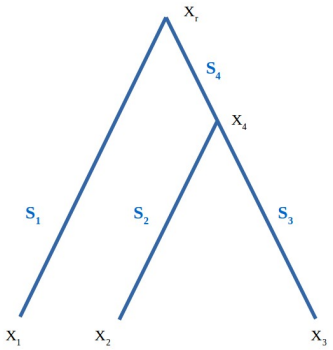


Figure 4: Statistical model of a tree of three leaves.

Knowing the substitution matrices and the distribution of the nucleotides at the root $\pi = (\pi_A, \pi_C, \pi_G, \pi_T)$, the probability of obtaining $X_1 = x$, $X_2 = y$ and $X_3 = z$ can be obtained under Markov process on this tree is:

$$p^T_{xyz} = P(X_1 = x, X_2 = y, X_3 = z|\,T) = \sum_{x_r, x_4 \in \{A,C,G,T\}} \pi_{x_r} S_1(x_r, x) S_4(x_r, x_4) S_2(x_4, y) S_3(x_4, z), \qquad (1)$$

If the Jukes-Cantor model is considered, we have:

$$p^T_{AAA} = p^T_{CCC} = p^T_{TTT} = p^T_{GGG}$$

where $$p^T_{AAA} = \frac{1}{4}(a_1a_4a_2a_3 + 3b_1b_4a_2a_3 + 3a_1b_4b_2b_3 + 3b_1a_4b_2b_3 + 6b_1b_4b_2b_3)$$

$$p^T_{ACC} = p^T_{ATT} = p^T_{AGG} = p^T_{CAA} = p^T_{CTT} = p^T_{CGG} = p^T_{GAA} = p^T_{GCC} = p^T_{GTT} = p^T_{TAA} = p^T_{TCC} = p^T_{TGG}$$

where $$p^T_{ACC} = \frac{1}{4}(a_1a_4b_2b_3 + 2a_1b_4b_2b_3 + a_1b_4a_2a_3 + 6b_1b_4b_2b_3 + b_1a_4a_2a_3 + 2b_1a_4b_2b_3 + 2b_1b_4a_2a_3)$$

$$p^T_{ACA} = p^T_{ATA} = p^T_{AGA} = p^T_{CAC} = p^T_{CTC} = p^T_{CGC} = p^T_{GAG} = p^T_{GCG} = p^T_{GTG} = p^T_{TAT} = p^T_{TCT} = p^T_{TGT}$$

where $$p^T_{ACA} = \frac{1}{4}(a_1b_4a_2b_3 + 2a_1b_4b_2b_3 + a_1a_4b_2a_3 + b_1a_4a_2b_3 + 3b_1b_4b_2a_3 + 3b_1b_4b_2b_3 + 2b_1b_4a_2b_3 + 2b_1a_4b_2b_3)$$

$$p^T_{AAC} = p^T_{AAT} = p^T_{AAG} = p^T_{CCA} = p^T_{CCT} = p^T_{CCG} = p^T_{GGA} = p^T_{GGC} = p^T_{GGT} = p^T_{TTA} = p^T_{TTC} = p^T_{TTG}$$

where $$p^T_{AAC} = \frac{1}{4}(a_1a_4a_2b_3 + a_1b_4b_2a_3 + 2a_1b_4b_2b_3 + b_1a_4b_2a_3 + 3b_1b_4a_2b_3 + 4b_1b_4b_2b_3 + 2b_1a_1b_2b_3 + 2b_1b_4b_2a_3)$$

$$p^T_{ACG} = p^T_{ACT} = p^T_{AGC} = p^T_{AGT} = p^T_{ATC} = p^T_{ATG} = p^T_{CAG} = p^T_{CAT} = p^T_{CGA} = p^T_{CGT} = p^T_{CTA} = p^T_{CTG} = p^T_{GAC} = p^T_{GAT} = p^T_{GCA} = p^T_{GCT} = p^T_{GTA} = p^T_{GTC} = p^T_{TAC} = p^T_{TAG} = p^T_{TCA} = p^T_{TCG} = p^T_{TGA} = p^T_{TGC}$$

where $$p^T_{ACG} = \frac{1}{4}(a_1a_4b_2b_3 + a_1b_4a_2b_3 + a_1b_4b_2a_3 + a_1b_4b_2b_3 + b_1a_4a_2b_3 + 4b_1b_4b_2b_3 + 2b_1b_4b_2a_3 + b_1a_4b_2a_3 + 2b_1b_4a_2b_3 + b_1a_4b_2b_3)$$

This can be done for any tree, but the expression of the probabilities in terms of the parameters gets more complicated. We will call $p^T_{x_1,\dots,x_n}$ the probability of observing pattern $x_1, \dots, x_n$ at the leaves $1, \dots, n$ of T, and p^T will be the corresponding probability vector $p^T = (p^T_{A\dots A}, \dots, p^T_{T\dots T})$.

Remark 2.14. The root of a phylogenetic tree is not identifiable. That is, different root placements can lead to the same vector of probabilities at the leaves, if we allow different transition matrices.

The *Equal Input model (EI)* for a set S of k states is a particular type of Markov process on a tree, defined as follows.Let π be a distribution on the set S. This distribution π will play the role of the stationary distribution for this model and, as the model is time-reversible, this will also be the distribution at the root of the tree. Throughout this thesis we shall assume that the distribution π is known (as it is the stationary distribution it can be inferred from the data) and is strictly positive.

Given a root vertex v_o let π be a distribution of states at v_o and for each (directed edge $e = (u, v)$ (directed away from v_0). In the *EI* model, each transition matrix S_e has the property that for some value $\theta_e \in [0, 1]$ and all states $\alpha, \beta \in S$ with $\alpha \neq \beta$ we have:

$$S_e^{\alpha\beta} = \pi_\beta \theta_e$$

This model generalizes the familiar *fully symmetric model* of k states (such as the 'Jukes-Cantor model' when $k = 4$) to allow each state to have its own stationary probability. The fully symmetric model on k states assumes π to be the uniform distribution, $\pi = (\frac{1}{k}, \dots, \frac{1}{k})$ and for k=4 this is the Jukes-Cantor model we presented above. In the case $k = 4$ with S equal to the four nucleotide bases, the equal input model is known as the *Felsenstein 1981 model*. The defining property of the model is that the probability of transition from α to β (two distinct states) is the same, regardless of the initial state $\alpha(\neq \beta)$.

Example 2.15. The *Felsenstein 1981 model*, like the Jukes-Cantor model, assumes that all substitutions are equally likely, but can model an arbitrary stationary distribution, $\pi = (\pi_A, \pi_C, \pi_G, \pi_T)$ where $\pi_A \neq \pi_C \neq \pi_G \neq \pi_T$. The Felsenstein model transition matrix is:

$$S_e = \begin{pmatrix} 1-(\pi_C+\pi_G+\pi_T)\theta_e & \pi_C\theta_e & \pi_G\theta_e & \pi_T\theta_e \\ \pi_A\theta_e & 1-(\pi_A+\pi_G+\pi_T)\theta_e & \pi_G\theta_e & \pi_T\theta_e \\ \pi_A\theta_e & \pi_C\theta_e & 1-(\pi_A+\pi_C+\pi_T)\theta_e & \pi_T\theta_e \\ \pi_A\theta_e & \pi_C\theta_e & \pi_G\theta_e & 1-(\pi_A+\pi_C+\pi_G)\theta_e \end{pmatrix}$$

3. Phylogenetic invariants for the equal-input model

3.1 Phylogenetic invariants

From now T is going to be the *topology* of the phylogenetic tree, and n is going to be the number of species considered (n leaves). Let's call M the evolutionary model with d free parameters. The next polynomial map is going to be associated to the leaves:

$$\varphi_T^M : \mathbb{R}^d \longrightarrow \mathbb{R}^{k^n}$$
$$\theta = (\theta_1, \dots, \theta_d) \longrightarrow p^T$$

For instance, the Jukes-Cantor model on the tree of *Figure 4* has associated the following polynomial map:

$$\varphi_T^M : \mathbb{R}^d \longrightarrow \mathbb{R}^{4^3} = \mathbb{R}^{64}$$
$$(b_1, b_2, b_3, b_4) \longrightarrow p^T = (p^T_{AAA}, p^T_{AAC}, \dots, p^T_{TTT})$$

and $p^T_{AAA}, p^T_{AAC}, \dots, p^T_{TTT}$ are written in therms of the parameters as (1), where we substitute a_e by $1-3b_e$.

Although the parameters of the model are probabilities and are at the interval $[0, 1]$ the map is going to be considered at $\mathbb{R}^d$ in order to be able to apply the results and techniques of algebraic geometry.

$Im\varphi_T^M$ contains all the distributions generated by the parameters of T. Also, considering $S \subseteq \mathbb{R}^{4^n}$, $I(S)$ is defined as the set of polynomials that are zero on S, is an ideal and has a finite set of generators due to the Hilbert basis theorem.

Definition 3.1. Given a tree topology T of n leaves and a evolutionary model M, the polynomials of $I_M(T) = I(Im\varphi_T^M)$ are called *invariants of* T.

Definition 3.2. The polynomials of $I_M(T)$ that are not in $I_M(T')$ for some other tree T' are called *topology invariants of* T and the ones that lie in $I_M(T)$ for any tree topology on n leaves are called *model invariants*.

3.2 Linear invariants

Linear invariants are invariants of degree one. Some models do not allow linear invariants but some models do. The models that have linear invariants have very nice properties, some of which we are going to study in the next pages.

From now, some of the following theoretical results are going to be illustrated as set of applied examples, the general proof can be found at the article of Marta Casanellas and Mike Steel: *Phylogenetic mixtures and linear invariants for equal-input models* [3].

Lemma 3.3. *For variables $\theta_1, \dots, \theta_d$, consider polynomials $f_1(\boldsymbol{\theta}), \dots, f_M(\boldsymbol{\theta}) \in \mathbb{R}[\theta_1, \dots, \theta_d]$ of the following form:*

$$f_i(\boldsymbol{\theta}) = \sum_{A \subseteq [d]} c_A^{(i)} \prod_{j \in A} \theta_j, \qquad c_A^{(i)} \in \mathbb{R}.$$

(i) Then $f_0 \equiv 0$ (i.e. $c_A^0 = 0$ for all $A \subseteq [d]$) if and only if for any $t \neq 0$, $f_0(\boldsymbol{\theta}) = 0$ for all $\boldsymbol{\theta} \in \{0, t\}^d$.

(ii) Let $f = (f_0, \dots, f_M) : \mathbb{R}^d \to \mathbb{R}^M$ and let $L : \mathbb{R}^M \to \mathbb{R}$ be a linear map. Define an equivalence relation among the elements of $\{0, t\}^d$ by $\boldsymbol{\theta} \sim \boldsymbol{\theta}'$ if $f(\boldsymbol{\theta}) = f(\boldsymbol{\theta}')$, and let $\theta_1, \dots, \theta_s$ be the representatives of these equivalence classes. We call $q_i = f(\theta_i)$, $i = 1, \dots, s$. Then $L(f(\boldsymbol{\theta})) = 0$ for all $\boldsymbol{\theta} \in \mathbb{R}^d$ if and only if $L(q_j) = 0$ for $j = 1, \dots, s$.

The proof of this lemma relies on multilinear algebra. Although the statement requires a lot of notation, it only says that to check whether a linear function f_0 is an invariant on T, one just needs to check whether f_0 vanishes when the parameters are 0 or 1 (or any t instead of 1).

We are going to illustrate it with the example on the Jukes Cantor model on three leaves used at the previous section.

Let $f_1 = P^T_{AAA}$ and $f_2 = P^T_{CCC}$ and consider $f_0 = f_1 - f_2 = 0$. Then, we could check that $P^T_{CCC} = P^T_{AAA}$ without having to compute P^T_{CCC} or P^T_{AAA}: one just needs to compute P^T_{AAA} and P^T_{CCC} for parameters $b_i \in \{0, 1\}$ and check if they are equal in all the cases.

The previous lemma is useful for a evolutionary models. However, from now on we need to restrict to the equal input model. At the section 5 we will compute the linear invariants for an equal-input model of a tree with 5 leaves.

In the Equal Input model, once we fix π, the probability $\mathbb{P}_T(\chi, \theta)$ of observing a character at the leaves of T is:

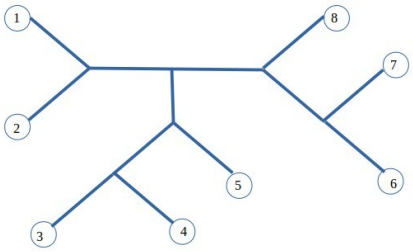
$$\mathbb{P}_T(\chi|\theta) = \sum_{(s_v)_v \in S^{Int(T)}} \pi_{s_{v_0}} \prod_{(u,w) \in E(T)} S_e^{s_u, s_w}$$

A *subforest* F of T is a subgraph comprised of a collection of disjoint trees $\{T_1, \dots, T_r\}$ such that the only nodes of degree ≤ 1 in T_i are leaves of T.

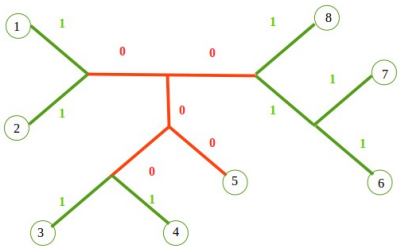
From now, we will say that a subforest $F = \{T_1, \dots, T_r\}$ is a *full subforest* of T if it the union of the sets of leaves of $T_1, \dots, T_r$ includes all the leaves of T.

We define Θ_F as the collection of edge parameters θ_e such that $\theta_e = 0$ if $e \in E(T_i)$ for some $T_i \in E$ and $\theta_e = 1$ for all other edges e. We will define the partition that F generates over the leaves as $\sigma(F)$.

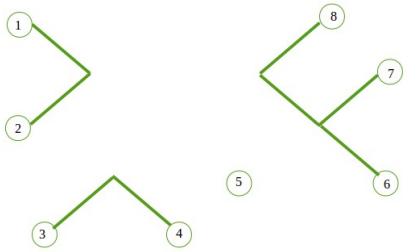
Example 3.4. Considering the tree in the *Figure 5*(a) we give values to the edges to get the full subforest F. Which defines the following partition $\sigma(F) = \{1, 2|3, 4|5|6, 7, 8\}$.



(a) Tree of 8 leaves



(b) Giving values to the edges



(c) Subforest F

Figure 5: Illustrative example

Lemma 3.5. *(a) Let Θ be a collection of parameters $(\theta_e)_{e \in E(T)}$ such that θ_e is either 0 or 1 for all $e \in E(T)$. Then there exists a unique full subforest $F \in \mathscr{F}_T$ such that $\mathbb{P}_{T,\Theta} = \mathbb{P}_{T,\Theta_F}$.*

(b) A degree 1 polynomial $\sum_\chi \lambda_\chi x_\chi$ is a linear phylogenetic invariant for a tree T if and only if

$$\sum_{\chi} \lambda_\chi \mathbb{P}_T(\chi|\Theta_F) = 0$$

for any full subforest $F \in \mathscr{F}_T$.

We illustrate the proof with an example:

(a) Let us consider the following tree:

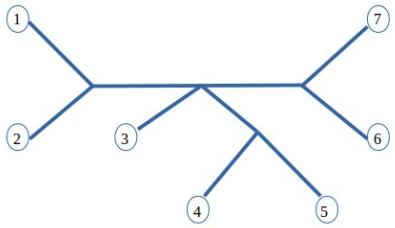
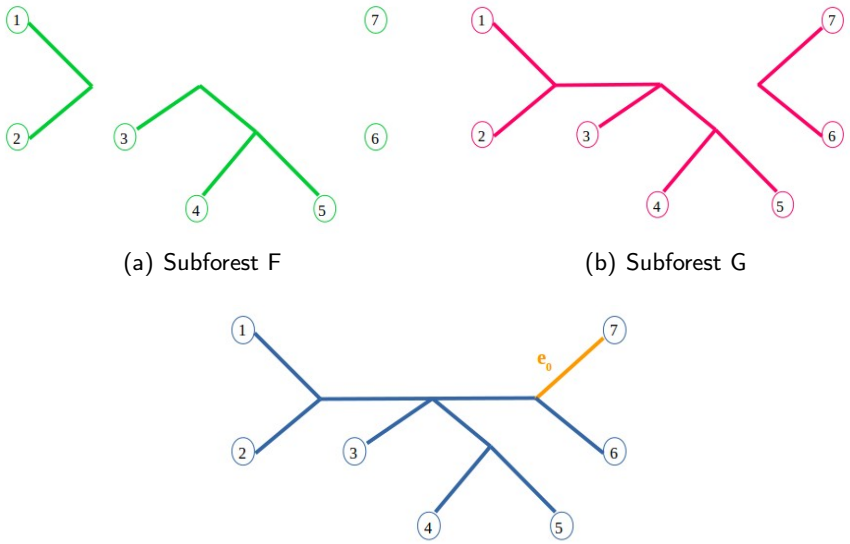


Figure 6: Graph with seven nodes.

First, we consider different full subforests F and G are different subforests of the tree and prove that there exists χ that makes $\mathbb{P}(\chi|\Theta_G) \neq \mathbb{P}(\chi|\Theta_F)$ (this illustrates the "uniqueness" part of the proof).

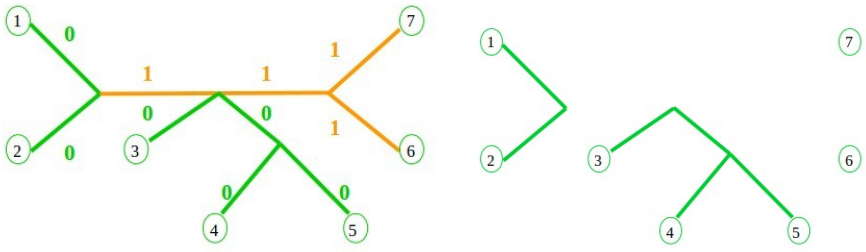


(a) Subforest F (b) Subforest G

(c) Edge e_0

Figure 7: Subforests

Due to F and G are subforests completed, they generate different partitions: $\sigma(F) = \{1, 2|3, 4, 5|6|7\}$ and $\sigma(G) = \{1, 2, 3, 4, 5|6, 7\}$.
There exist an edge e_0 such that is compatible with $\sigma(F)$ (if each "block" of $\sigma(F)$ are inside the "blocks" produced by $\sigma(e_0)$). The partition of e_0 is $\sigma(e_0) = \{1, 2, 3, 4, 5, 6|7\}$ and is not compatible with $\sigma(G)$.
So, if χ is the character that assign x to one connected part from $T - e_0$ and $y \neq x$ to the other component we have $\mathbb{P}(\chi|\Theta_G) = 0$ while $\mathbb{P}(\chi|\Theta_F) \neq 0$.
Now let Θ satisfy the statement (a) of the lemma. Lets define $A \subseteq E(T)$ edges from T that have $\theta_e = 1$.



(a) Tree with values at the edges (b) Subforest F

Figure 8: Deleting the nodes equal to 1

Considering $\sigma(T \backslash A)$ the partition obtained deleting all the edges of A. If F is the subforest $F_T(\sigma(T \backslash A))$, then we have $\mathbb{P}_{T,\Theta} = \mathbb{P}_{T,\Theta_F}$.

(b) Is a direct consequence from lemma 3.3(ii) and lemma 3.5(a).

□

Let Θ be a collection of edge parameters on a tree T. For a site character χ, lets define:

$$\tilde{p}_{\chi}^{T}(\Theta) = \frac{\mathbb{P}_{T}(\chi|\Theta)}{\pi_{\chi_1}\pi_{\chi_2}\ldots\pi_{\chi_n}}$$

Calling:

$$\tilde{x}_{\chi} = \frac{x_{\chi}}{\pi_{\chi_1}\pi_{\chi_2}\ldots\pi_{\chi_n}}$$

Example 3.6. In order to use the previous lemma in an applied case we explain the "Lake invariants" in the tree of four leaves.

The Lake invariants are:

$$P_{ACAC} + P_{ACGT} = P_{ACGC} + P_{ACAT}, \tag{2}$$
$$P_{ACCA} + P_{ACGT} = P_{ACCG} + P_{AACT} \tag{3}$$

The equation (2) is invariant for the trees $T_1 = 12|34$ and $T_2 = 14|23$ while it is not invariant for $T_3 = 13|24$.

We compute the probabilities for each tree:

Table 2: Table of calculations equation (2):

	$\mathbf{T_1 = 12\|34}$	$\mathbf{T_2 = 14\|23}$	$\mathbf{T_3 = 13\|24}$
$\mathbf{p_{ACAC}}$	0	0	$\frac{1}{\pi_A\pi_C}$
$\mathbf{p_{ACGT}}$	0	0	0
$\mathbf{p_{ACGC}}$	0	0	0
$\mathbf{p_{ACAT}}$	0	0	0

Getting:

$$0 = 0 \quad at \quad T_1$$
$$0 = 0 \quad at \quad T_2$$
$$\frac{1}{\pi_A\pi_C} = 0 \quad at \quad T_3$$

Being clear that T_3 does not have the linear invariant while T_1 and T_2 do.

The equation (3) is invariant for the trees $T_1 = 12|34$ and $T_3 = 13|24$ while it is not invariant for $T_2 = 14|23$.

We compute the probabilities for each tree:

Table 3: Table of calculations equation (3):

	$\mathbf{T_1 = 12\|34}$	$\mathbf{T_2 = 14\|23}$	$\mathbf{T_3 = 13\|24}$
$\mathbf{p_{ACAC}}$	0	$\frac{1}{\pi_A\pi_C}$	0
$\mathbf{p_{ACGT}}$	0	0	0
$\mathbf{p_{ACGC}}$	0	0	0
$\mathbf{p_{ACAT}}$	0	0	0

Getting:

$$\begin{aligned} 0 = 0 \quad & at \quad T_1 \\ \frac{1}{\pi_A\pi_C} = 0 \quad & at \quad T_2 \\ 0 = 0 \quad & at \quad T_3 \end{aligned}$$

Being clear that T_2 does not have the linear invariant while T_1 and T_3 do.

Lemma 3.7. *We say that two characters χ and χ' are equivalent, $\chi \equiv \chi'$, if $\sigma(\chi) = \sigma(\chi')$ and $\chi_i = \chi'_i$ for any leaf i that belongs to a block of the partition of cardinality greater than or equal to two. Let $\chi \equiv \chi'$ be two characters on the set $X = [m]$.*

(a) If $\chi \equiv \chi'$ then $\tilde{x}_\chi - \tilde{x}_{\chi'}$ is a linear invariant.

(b) If π is not invariant by any permutation of the set of states, then for any tree T the following equality $\tilde{p}^T_\chi(\Theta) = \tilde{p}^T_{\chi'}(\Theta)$ for every Θ implies that $\chi \equiv \chi'$ (i.e. in this case every linear phylogenetic invariant of type $\tilde{x}_\chi - \tilde{x}_{\chi'}$ satisfies $\chi \equiv \chi'$).

We illustrate the proof with an example:

(a) Having $\chi = AACGG$ and $\chi' = AATGG$, we want to prove that $\chi \equiv \chi'$. It is easy to see that

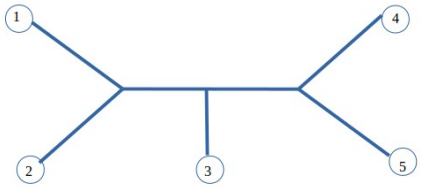


Figure 9: Graph with five nodes.

$\sigma(\chi) = \{12|3|45\} = \sigma(\chi') := \sigma$.
By lemma 2.6.(b) we have to prove $\tilde{p}_\chi(\Theta_F) = \tilde{p}_{\chi'}(\Theta_F)$ for any full subforest $F \in \mathscr{F}_T$.

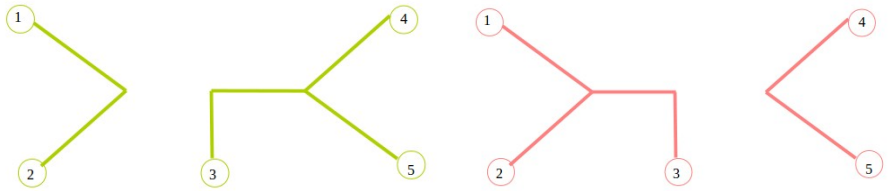
- If $\sigma(F)$ does not refine σ:
  For example $\sigma(F) = \{12345\}$:
  The probabilities are: $\tilde{p}_\chi(\Theta_F) = \tilde{p}_{\chi'}(\Theta_F) = 0$.

- If $\sigma(F)$ refines:
  For example $\sigma(F) = \{1|2|3|45\}$:
  The probabilities are: $\tilde{p}_\chi(\Theta_F) = \pi_A\pi_C\pi_G$ and $\tilde{p}_{\chi'}(\Theta_F) = \pi_A\pi_T\pi_G$.
  Computing the new probability:

$$\tilde{p}^T_\chi(\Theta_F) = \frac{\pi_A\pi_C\pi_G}{\pi_A\pi_A\pi_C\pi_G\pi_G} = \frac{1}{\pi_A\pi_G}$$
$$\tilde{p}_{\chi'}(\Theta_F) = \frac{\pi_A\pi_T\pi_G}{\pi_A\pi_A\pi_T\pi_G\pi_G} = \frac{1}{\pi_A\pi_G}$$

  So $\tilde{p}^T_\chi(\Theta_F) = \tilde{p}^T_{\chi'}(\Theta_F)$.

(b) It is going to be assumed that π is not invariant by any permutation of the set of states, for example $\pi = \{\pi_A, \pi_C, \pi_G, \pi_T\} = \{0.1, 0.2, 0.3, 0.4\}$, so that $\pi_s = \pi_t$ is and only if $s = t$. Also, that for a tree T (the one at the Figure 5) we have $\tilde{p}_\chi(\Theta_T) = \tilde{p}_{\chi'}(\Theta_T)$ for any collection of edge parameters Θ_T. Then, for each block B_i of $\sigma(\chi) = \{12|3|45\}$ of size b_i greater or equal that 2 ($B_1 = \{1, 2\}$ and $B_2 = \{4, 5\}$ with $b_1 = b_2 = 2 \geq 2$) consider the forest $F_i = \{T_{B_i}, \cup_{l \notin B_i}\{l\}\}$, where T_{B_i} is the smallest subtree of T joining the leaves of B_i.



(a) $F_1 = \{1, 2|3, 4, 5\}$ (b) $F_2 = \{1, 2, 3|4, 5\}$

Figure 10: Subforests

Then $\tilde{p}^T_\chi(\Theta_{F_i}) = \frac{1}{\pi^{b_i-1}_{s_i}}$ if s_i is the state of χ at the leaves of B_i. By hypothesis this is equal to $\tilde{p}^T_{\chi'}(\Theta_{F_i})$. But $\tilde{p}^T_{\chi'}(\Theta_{F_i})$ is zero if $\sigma(\chi')$ does not contain the block B_i. Performing the same argument for any block we obtain $\sigma(\chi) = \sigma(\chi')$. Now for each B_i we have $\tilde{p}^T_\chi(\Theta_{F_i}) = \tilde{p}^T_{\chi'}(\Theta_{F_i})$ and hence $\frac{1}{\pi^{b_i-1}_{s_i}} = \frac{1}{\pi^{b_i-1}_{s'_i}}$ if s'_i is the state of χ' at the leaves of B_i. As $b_i \geq 2$, the assumption on π implies $s_i = s'_i$ and thus, $\chi \equiv \chi'$.

Remark 3.8. If π is the uniform distribution, then we have $\mathbb{P}_T(\chi|\Theta) = \mathbb{P}_T(\chi'|\Theta)$ if and only if $\sigma(\chi) = \sigma(\chi')$. Indeed, in this case if we consider any permutation g of the set of states S, the polynomials $x_\chi - x_{g\Delta\chi}$ are linear phylogenetic invariants for any tree (proved by Marta Casanellas), where $g\Delta\chi$ stands for the corresponding permutation of states at the leaves. But these polynomials can also be rewritten as $x_\chi - x_{\chi'}$ for $\sigma(\chi) = \sigma(\chi')$.

□

Example 3.9. For $n = 4$. The previous lemma gives the following. If we consider different states x, y, z, t and another set of four different states x', y', z', t' the linear phylogenetic invariants are:

$$\tilde{x}_{xyzt} - \tilde{x}_{x'y'z't'}, \quad \tilde{x}_{xxyz} - \tilde{x}_{xxy'z'}, \quad \tilde{x}_{xxxy} - \tilde{x}_{xxxy'}$$

and the analogous invariants obtained for other partitions of [4] involving singletons.

4. Phylogenetic mixtures

Definition 4.1. Fix a distribution π on the set of states. Given a particular tree T, we denote by $\mathbb{P}_{T,\Theta}$, the distribution of an Equal-Input model with parameters π, Θ on T. We define the *space of mixtures on* T as:

$$\mathscr{D}_T^\pi = \{p = \sum_i \lambda_i \mathbb{P}_{T,\Theta_i} | \sum_i \lambda_i = 1\}$$

In other words, a mixture p is a distribution resulting from a combination of distributions from the Equal Input model in the tree T.

If $\mathscr{T}$ is the set of phylogenetic trees on $[n]$, we define *the space of phylogenetic mixtures* on $[n]$ as:

$$\mathscr{D}^\pi = \{p = \sum_i \lambda_i \mathbb{P}_{T_i,\Theta_i} | \sum_i \lambda_i = 1,\ T_i \in \mathscr{T}\}$$

That is, in $\mathscr{D}^\pi$ we consider mixtures of distribution from de *EI* model on different trees.

From now on, we fix $n \geq 4$ throughout the rest of the section. We call Σ_k the set of partitions of $[n]$ of size at most k (note that if $k \geq n$, this is the whole set of partitions of $[n]$). If σ is a partition of $[n]$ compatible with trees T and T', and we consider $F = F_T(\sigma)$ and $F' = F_{T'}(\sigma)$, then one has $\mathbb{P}_{T,\Theta_F} = \mathbb{P}_{T',\Theta_{F'}}$. This point will be briefly denoted as q_σ (because it does not depend on the chosen tree compatible with σ). For example, if $\sigma = 1|234$ we can take T as *any of the trees of four leaves*.

Example 4.2. Using the alignment of the example 2.11, it could be that different parts of the alignment came from different substitution matrices on the same tree.

Gorilla Gorilla	AACTTCG	AGGCTTACCGCTG
Homo Sapiens	AACGTCT	ATGCTCACCGATG
Pan Troglodytes	AAGGTCG	ATGCTCACCGATG
	$<-\lambda_1->$	$<---\lambda_2---->$

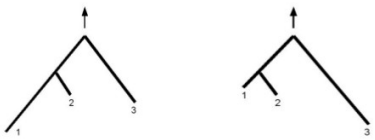


Figure 11: Example of mixtures

Being the same model, same tree topology T but different substitution parameters we obtain that the mixture is $p = \lambda_1 \mathbb{P}_T(\theta_1) + \lambda_2 \mathbb{P}_T(\theta_2)$, with $\lambda_1 = \frac{7}{20}$ and $\lambda_2 = \frac{13}{20}$.

The space of phylogenetic mixtures is clearly related to the space of linear invariants. Indeed if we let E_T^π be the set of vectors in $\mathbb{R}^{k^n}$ where all the linear invariants for T vanish (that is, E_T^π is the dual to the space of linear invariants for T), then E_T^π is the director space of the affine linear variety $\mathscr{D}_T^\pi$. Similarly if E^π is the set of vectors in $\mathbb{R}^{k^n}$ where all the linear model invariants vanish, then E^π is the director space

of the affine linear variety $\mathscr{D}^\pi$. As a consequence, studying $\mathscr{D}_T^\pi$ and $\mathscr{D}^\pi$ is equivalent to studying the set of linear invariants.

The following theorem computes the dimension of the space of phylogenetic mixtures on trees (and as a consequence, the dimension of the space of linear model invariants).

Theorem 4.3. *(Casanellas-Steel) If π is a distribution on k states with positive entries, then $\{q_\sigma | \sigma \in \Sigma_k\}$ are affine linearly independent points. Moreover, if π is the uniform distribution or generic distribution, or if $k \geq n$, then $\mathscr{D}^\pi$ coincides with $\langle q_\sigma | \sigma \in \Sigma_n \rangle_a$ and has dimension $|\Sigma_k| - 1$ (which equals $B_n - 1$ if $k \geq n$).*

In the next chapter we are going to verify htis theorem by computing this set of points for $n = 5$ and $k = 4$.

5. Computational part

In this section we are going to compute the linear independent points for $\mathscr{D}_T^\pi$ for $n = 5$ and therefore will verify the theorem 4.3. First of all, in order to make it easier to understand the code, we are going to explain what every function does:

- **initializer**(**w**) : this function turns into -1 all the components of the vector w.
- **are_equal**(**v1**, **v2**) : return true if v1 and v2 are equal.
- **canonizer**(**original**, **v**) : return the representative of the equivalence relation, which is calculated by giving for each character an alphabetical order. If we have *yyzzz* (for the function it is 11222) it will return 11222 (*xxyyy*).
- **select_row**(**M**, **k**) : it returns the row k of the matrix M.
- **add_row**(**M**, **u**) : this function add the vector u as the last row of the matrix M.
- **check_is_new**(**final_container**, **cont**, **u**, **v**) : this function checks if the vector u given is already in the '*final_container*' (it or its canonical representative, which are considered to be the same). If it is already in, it does nothing. Else, it puts it in the '*final_container*'.
- **fill**(**final_container**, **cont**, **m**, **v**) : this function generates all the possible combinations we could have in a vector of length $n = 5$ with $m = 4$ possible values each component could have, checking if we have kept its canonical representative and adding it if not (using the function previously presented *check_is_new*).
- **printer_numbers**(**M**) : print the matrix M, which is a matrix of numbers.
- **printer**(**M**) : print the matrix M, changing each number by the letter we have assigned (0 for x, 1 for y, 2 for z and 3 for t).
- **point_machine**(**point_container**, **n**) : this function fills the matrix which in each row has a possible q_σ. We will use the following notation: $q_{12} = 11000$ (which force the leaf 1 and 2 to have the same state) and $q_{12|34} = 11220$ (which force the leaf 1 and 2 and 3 and 4 to have the same state). This points are going to be kept in the matrix *point_container*.
- **valor**(**n**) : return the letter assigned to each number n.
- **calculator**(**fc**, **pc**) : for each $\tilde{x}$ which we name *fc* and q_σ named *pc* computes the probability of this combination.
- **probabilitator**(**prob_container**, **point_container**, **final_container**, **cont**) : this function computes the linearly independent points for $\mathbb{D}_T$ from the matrices that keep the $\tilde{x}$ and q_σ and keep them in *prob_container*.
- **latex**(**M**) : returns the code we need to have the function in latex language (is implemented in sage).

For computing, we used sage:

```
# n is the quantity of leaves.
n = 5

# m is the quantity of values the leaf can take.
m = 4

# v is an auxiliar vector.
v = vector(QQ,n)

def initializer(w):
    """
    This function changes all the components of the vector w for -1.
    """
    for i in range(n):
        w[i] = -1
    return w

def are_equal(v1, v2):
    """
    This function says if v1 and v2 are equal or not.
    """
    for i in range(n):
        if v1[i] != v2[i]:
            return false
    return true

def canonizer(original,v):
    cnt = 0
    """
    This function returns the canonical representative of a vector.
    We use it to keep at the final_container only one
    representative of each class.
    """
    initializer(v)
    for i in range(n):
        if v[i] == -1:
            for j in range(n):
                if original[i] == original[j]:
                    v[j] = cnt
            cnt = cnt + 1
    return v
```

```
def select_row(M,k):
    """
    This function returns the row k of the Matrix M.
    """
    f = M.nrows()
    c = M.ncols()
    v = vector(QQ,n)
    for i in range(c):
        v[i] = M[k,i]
    return v

def add_row(M,u):
    """
    This function returns M adding u as the final row.
    """
    f = M.nrows()
    c = M.ncols()
    A = matrix(f+1,c)
    for i in range(f + 1):
        for j in range(c):
            if i == f:
                A[i,j] = u[j]
            else:
                A[i,j] = M[i,j]
    return A

#final_container is where the representatives are saved, it is \
   initialized
#by taking the trivial representative (all the leaves with the same \
   value)

final_container = matrix(QQ,1,n)

# cont is the number of representatives

cont = final_container.nrows()

def check_is_new(final_container,cont,u,v):
    """
    Checks if the vector u given is already in the 'final_container'
    (itself or its canonical representative, which are considered to
    be the same).
    If it is already in, it does nothing.
    Else, it puts it in the 'final_container'.
    """
    canonized = canonizer(u,v)
```

```
    for i in range(cont):
        existing_word = select_row(final_container,i)
        if are_equal(canonized,existing_word) == true:
            return (final_container,cont)
    cont = cont + 1
    A = add_row(final_container,canonized)
    return (A,cont)

def fill(final_container,cont,m,v):
    """
    This function returns the final_container.
    """
    for i in range(m):
        for j in range(m):
            for k in range(m):
                for l in range(m):
                    for h in range(m):
                        u = [i,j,k,l,h]
                        (final_container,cont) = check_is_new(\
   final_container,cont,u,v)
    return (final_container,cont)

(final_container,cont) = fill(final_container,cont,m,v)

def printer_numbers(M):
    """
    This function prints M.
    """
    f = M.nrows()
    c = M.ncols()
    for i in range(f):
        for j in range(c):
            print M[i,j],
        print " "
    return None

def printer(M):
    """
    This function prints M changing each number for the letter \
   assigned.
    """
    f = M.nrows()
    c = M.ncols()
    for i in range(f):
        for j in range(c):
            if M[i,j] == 0: print "x",
```

```
            elif M[i,j] == 1: print "y",
            elif M[i,j] == 2: print "z",
            else: print "t",
        print " "
    return None

s = "There are"
saux = "representatives."
print s, cont, saux

s = "The matrix of representatives is:"
print s
printer(final_container)

#Imposing leaves to have the same value
#point_container is where the points are going to be saved

point_container = matrix(QQ,cont,n)

def point_machine(point_container,n):
    """
    This function returns all the points
    """
    aux = 0
    #couples
    for i in range (n):
        for j in range (i + 1,n):
            point_container[aux,i] = 1
            point_container[aux,j] = 1
            aux = aux + 1
    #double couples
    aux2 = aux - 1
    for a in range(aux2):
        p1 = -1
        p2 = -1
        have_one = false
        for i in range(n):
            if (point_container[a,i] == 1):
                if (have_one == false):
                    have_one = true
                    p1 = i
                else:
                    p2 = i
        for i in range(a,aux2 + 1):
            p3 = -1
            p4 = -1
```

```
            have_one_two = false
            for j in range(n):
                if (point_container[i,j]==1):
                    if (have_one_two == false):
                        have_one_two = true
                        p3 = j
                    else:
                        p4 = j
            if(p1 != p3 and p1 != p4 and p2 != p3 and p2 != p4):
                point_container[aux,p1] = 1
                point_container[aux,p2] = 1
                point_container[aux,p3] = 2
                point_container[aux,p4] = 2
                aux = aux + 1
    aux3 = aux - 1
    # triplets
    for i in range (n):
        for j in range (i + 1,n):
            for k in range (j + 1,n):
                point_container[aux,i] = 1
                point_container[aux,j] = 1
                point_container[aux,k] = 1
                aux = aux + 1
    aux4 = aux - 1
    #couple+triplet
    for a in range(aux2 + 1):
        for i in range(n):
            if(point_container[a,i] == 1):
                point_container[aux,i] = 1
            else:
                point_container[aux,i] = 2
        aux = aux + 1
    #fourtets
    for i in range (n):
        for j in range (i + 1,n):
            for k in range (j + 1,n):
                for l in range (k + 1,n):
                    point_container[aux,i] = 1
                    point_container[aux,j] = 1
                    point_container[aux,k] = 1
                    point_container[aux,l] = 1
                    aux = aux + 1
    #all the same value
    for i in range (n):
        point_container[aux,i] = 1
    return point_container
```

```
s = "The matrix of (point_container) is:"
print s

point_container = point_machine(point_container,n)

printer_numbers(point_container)

#Imposing the values
#prob_container is where the probabilities are going to be saved

prob_container = matrix(SR,cont,cont)

def valor (n):
    if(n == 0): return var('x')
    elif(n == 1): return var('y')
    elif(n == 2): return var('z')
    else: return var('t')
    return None

def calculator (fc,pc):
    error = False #if it's not compatible the combination it will \
   finish being true
    hihauns = False #if pc has ones
    hihados = False #if pc has twos
    value = 1 #we are going to change it by multipling and dividing to\
    get the value
    for m in range(5):
        value = value / valor(fc[m]) #we apply the values we got from \
   fc
        if(pc[m] == 0): value = value*valor(fc[m]) #if we dont have \
   restriction we multiply
        elif(pc[m] == 1): hihauns = True
        else: hihados = True
    if (hihauns == False and hihados == False): return 1
    if (hihauns == True):
        found = False #we want to know the value of the ones
        i = 0
        while (not found):
            if(pc[i] == 1): #when we find it
                found = True #we mark that we have it
                value = value*valor(fc[i]) #multiply by the value
                for l in range(i,5):
                    if(pc[l] == 1 and fc[i] != fc[l]): error = True
                    #if there is one in pc but doesnt have the same \
   value there is an error
```

```
            i = i + 1
    if (hihados == True):
        found = False
        i = 0
        while (not found):
            if(pc[i] == 2):
                found = True
                value = value*valor(fc[i])
                for l in range(i,5):
                    if(pc[l] == 2 and fc[i] != fc[l]): error = True
            i = i + 1
    if (error == True): value = value*0
    return value

def probabilitator(prob_container,point_container,final_container,cont\
   ):
    """
    This function returns the matrix of probabilities
    """
    for i in range(cont):
        for j in range(cont):
            pc = select_row(point_container,i)
            fc = select_row(final_container,j)
            prob_container[i,j] = calculator(fc,pc)
    return prob_container

s = "The matrix of (prob_container) is:"
print s

prob_container = probabilitator(prob_container,point_container,\
   final_container,cont)

latex(prob_container) #to get the latex code
```

We wanted also to prove that the rows of *prob_container* generate a complete subspace of dimension 51. In order to prove it, we compute its determinant using a python code (which we don't consider enough important to add here) because in sage it was extremely slow, uncomputable.

$$det(prob\_container) = -\frac{1}{tx^{60}y^{37}z^{11}} \neq 0 \quad \forall x, y, z, t \in (0, 1)$$

Since is not zero for all values of x, y, z, t *prob_container* always generate a complete space.

6. Results

There are 51 representatives.

The matrix of representatives is:

```
x x x x x
x x x x y
x x x y x
x x x y y
x x x y z
x x y x x
x x y x y
x x y x z
x x y y x
x x y y y
x x y y z
x x y z x
x x y z y
x x y z z
x x y z t
x y x x x
x y x x y
x y x x z
x y x y x
x y x y y
x y x y z
x y x z x
x y x z y
x y x z z
x y x z t
x y y x x
x y y x y
x y y x z
x y y y x
x y y y y
x y y y z
x y y z x
x y y z y
x y y z z
x y y z t
x y z x x
x y z x y
x y z x z
x y z x t
x y z y x
```

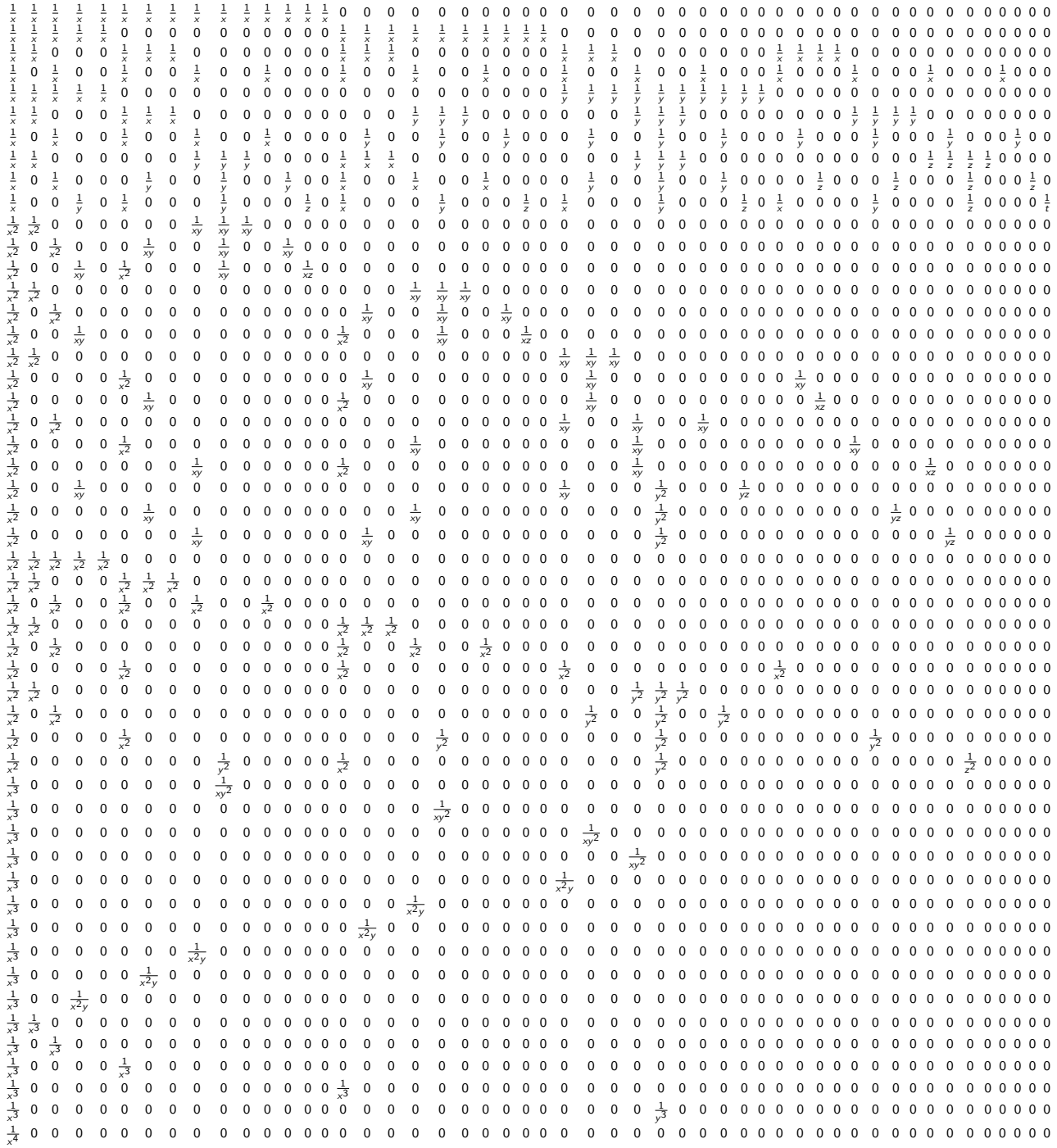
```
x y z y y
x y z y z
x y z y t
x y z z x
x y z z y
x y z z z
x y z z t
x y z t x
x y z t y
x y z t z
x y z t t

The matrix of (point_container) is:

1 1 0 0 0
1 0 1 0 0
1 0 0 1 0
1 0 0 0 1
0 1 1 0 0
0 1 0 1 0
0 1 0 0 1
0 0 1 1 0
0 0 1 0 1
0 0 0 1 1
1 1 2 2 0
1 1 2 0 2
1 1 0 2 2
1 2 1 2 0
1 2 1 0 2
1 0 1 2 2
1 2 2 1 0
1 2 0 1 2
1 0 2 1 2
1 2 2 0 1
1 2 0 2 1
1 0 2 2 1
0 1 1 2 2
0 1 2 1 2
0 1 2 2 1
1 1 1 0 0
1 1 0 1 0
1 1 0 0 1
1 0 1 1 0
1 0 1 0 1
1 0 0 1 1
0 1 1 1 0
0 1 1 0 1
```

0 1 0 1 1
0 0 1 1 1
1 1 2 2 2
1 2 1 2 2
1 2 2 1 2
1 2 2 2 1
2 1 1 2 2
2 1 2 1 2
2 1 2 2 1
2 2 1 1 2
2 2 1 2 1
2 2 2 1 1
1 1 1 1 0
1 1 1 0 1
1 1 0 1 1
1 0 1 1 1
0 1 1 1 1
1 1 1 1 1

The matrix of (prob_container) is:



7. Conclusion

Throughout this project, we have achieved all the objectives we had proposed at the beginning. We studied the Markov processes on phylogenetic trees and deeply studied the properties of the Equal-Input model, studying also the linear equations that all the distributions arising from it. Not only that, we computed explicitly, using sage, the generators of the space of linear invariants for phylogenetic trees on five leaves evolving under the Equal-Input model.

References

[1] E S Allman and J A Rhodes. Phylogenetic invariants. In O Gascuel and M A Steel, editors, *Reconstructing Evolution*. Oxford University Press, 2007.

[2] M Casanellas. Algebraic tools for evolutionary biology. *La Gaceta de la RSME*, 15:521–536, 2012.

[3] M Casanellas and M A Steel. Phylogenetic mixtures and linear invariants for equal input models. *Journal of Mathematical Biology*, 2016.