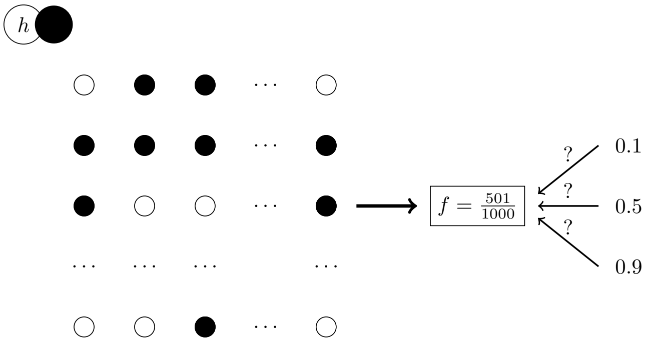
Chapter 1

Phylogenetic Reconstruction from a Probabilistic Perspective

We have already discussed distance-based phylogenetic trees and parsimony trees. By using these different algorithms, each based on distinct principles, we aim to find the most probable tree (rather than a 'true' tree) from the given dataset. This fact prompts us to consider phylogenetic reconstruction from a probabilistic perspective.

In probabilistic terms, we are inferring a phylogenetic tree that best explains the observed sequence data. Firstly, let us consider the example of a coin toss: if we roll a coin 1000 times and count the number of times it shows 'heads', the observed count of 'heads' (data, D) helps us infer the probability of 'heads' (parameter, p), which is intrinsic to the coin. If we observe 501 'heads,' we might reasonably estimate that p is closer to 0.5 than to 0.1 or 0.9. This intuition comes from noting that, if $p = 0.5$, it is more probable to obtain a result close to 501 'heads' than if $p = 0.1$ or $p = 0.9$. In other words, we are inferring the parameter by examining the conditional probability of the observed data given different parameter values.



Returning to phylogenetic reconstruction, our observed data consists of the sequence data

(D), from which we aim to infer the tree structure (T). Calculating the exact probability of a particular tree given the data is challenging, or even impossible. However, similar to inferring the probability of heads in a coin toss, we can calculate the conditional probability of observing the sequence data given a specific tree:

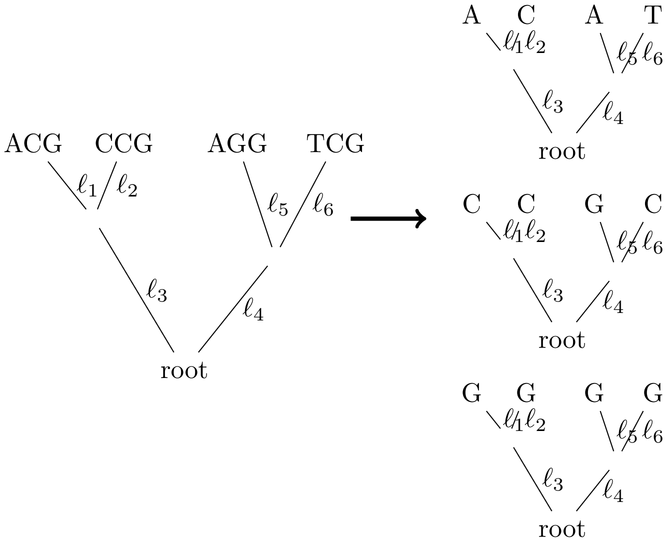
$$Pr(D|T) = ?$$

1.1 Conditional Probability of Sequence Data Given a Tree

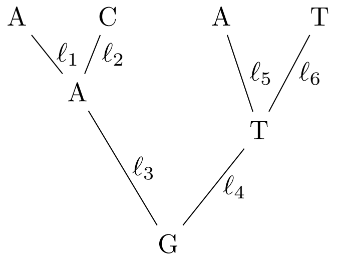
A basic assumption is that all alignment columns evolve independently of each other. Please note that this assumption does not always hold true. For instance, in the so-called 'CpG' sites — DNA regions where a cytosine is followed by a guanine — the cytosine is more likely to be methylated and subsequently mutate into thymine. In this case, the mutation G → T depends on the state of the adjacent site (whether it is a C or not). Despite this, we focus here on the general case where all alignment columns are assumed to be independent. Under this independence assumption, the probability of the data D given the tree T is expressed as:

$$Pr(D|T) = \prod_i Pr(d_i|T) \tag{1.1}$$

where d_i represents the ith alignment column.



For a tree where the state of each inner node is specified, it is straightforward to calculate the probability of that particular tree.



We assume that the probability of a nucleotide x being replaced by y depends on the branch length ℓ and is represented by $P_{x\to y}(\ell)$. Additionally, the probability that the root has state x is denoted as p_x. For example, for tree T_1 with specified states at the inner nodes, its probability can be calculated as:

$$Pr(D|T) = p_G \cdot P_{G\to A}(\ell_3) \cdot P_{G\to T}(\ell_4) \qquad \cdot P_{A\to A}(\ell_1) \cdot P_{A\to C}(\ell_2) \cdot P_{T\to A}(\ell_5) \cdot P_{T\to T}(\ell_6)$$

However, the states of the inner nodes are usually unknown. As a result, the probability of the tree cannot be directly calculated by simply multiplying all the replacement probabilities along each branch, since the replacements themselves are not determined.

Notice that there are many duplicated calculations when different trees share the same subtree. For Example, consider ((A,T):A,T) and ((A,T):A,G).

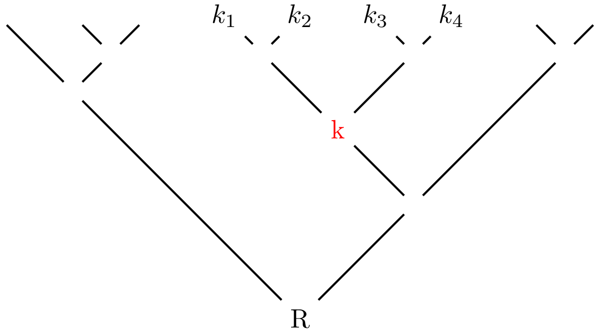
So, what can we do? The simplest approach is to calculate probabilities for every possible combination of inner node states. This involves assigning each inner node one of the four nucleotides (A, C, G, T) and iterating through all possible configurations.

However, a more effcient approach is to use **Felsenstein's pruning algorithm**.

Definition 1.1.1. ***Felsenstein's Pruning Algorithm***
For each node k let D_k be the part of the data d_i that are labeled to tips that stem from k and define

$$w_k(x) = Pr(D_k | k \text{ has an } x \text{ at this site})$$

for every nucleotide x.

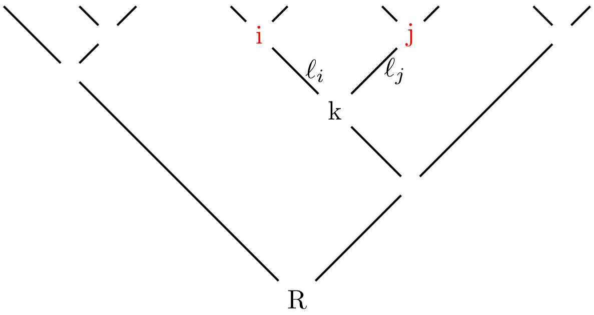


For any leave b with nucleotide y we have

$$w_b(x) = \begin{cases} 0, & \textit{if } x \neq y \\ 1, & \textit{if } x = y \end{cases}$$

If k is a node with child nodes i and j and corresponding branch lengths ℓ_i and ℓ_j, then

$$w_k(x) = \left(\sum_{y \in \{A,C,G,T\}} P_{x \to y}(\ell_i) \cdot w_i(y) \right) \cdot \left(\sum_{z \in \{A,C,G,T\}} P_{x \to z}(\ell_j) \cdot w_j(z) \right) \tag{1.2}$$



Compute all $w_k(x)$ from the tips to the root by dynamic programming for all k and all x. Then for the root r we can compute:

$$Pr(T|D) = \sum_{x \in \{A,C,G,T\}} p_x \cdot w_r(x) \tag{1.3}$$

where p_x is the probability that the root node have the state x.

A more concise way to express (1.2) is in matrix form. Let W_k represent the matrix of $w_{k,s}(x)$ where k denotes the node, s indicates the sequence position (n positions in total), and x represent the nucleotide:

$$W_k = \begin{pmatrix} w_{k,1}(A) & w_{k,2}(A) & w_{k,3}(A) & \dots & w_{k,n}(A) \\ w_{k,1}(C) & w_{k,2}(C) & w_{k,3}(C) & \dots & w_{k,n}(C) \\ w_{k,1}(G) & w_{k,2}(G) & w_{k,3}(G) & \dots & w_{k,n}(G) \\ w_{k,1}(T) & w_{k,2}(T) & w_{k,3}(T) & \dots & w_{k,n}(T) \end{pmatrix}$$

Matrix multiplication $A \cdot B = C$, also called Matmul Product, in which $c_{ij} = \sum_{k=1}^{m} a_{ik} \cdot b_{kj}$.

Entrywise product $A \circ B = D$, also called Hadamard Product, in which $d_{ij} = a_{ij} \cdot b_{ij}$

And we define a transition matrix $P(\ell_i) = (P_{x \to y}(\ell_i))_{x,y \in \{A,C,G,T\}}$ which encapsulates all transition probabilities with the branch length of ℓ_i:

$$P(\ell_i) = \begin{pmatrix} P_{A\to A}(\ell_i) & P_{A\to C}(\ell_i) & P_{A\to G}(\ell_i) & P_{A\to T}(\ell_i) \\ P_{C\to A}(\ell_i) & P_{C\to C}(\ell_i) & P_{C\to G}(\ell_i) & P_{C\to T}(\ell_i) \\ P_{G\to A}(\ell_i) & P_{G\to C}(\ell_i) & P_{G\to G}(\ell_i) & P_{G\to T}(\ell_i) \\ P_{T\to A}(\ell_i) & P_{T\to C}(\ell_i) & P_{T\to G}(\ell_i) & P_{T\to T}(\ell_i) \end{pmatrix}$$

Thus, the equation (1.2) can be written as:

$$W_k = (P(\ell_i) \cdot W_i) \circ (P(\ell_j) \cdot W_j) \tag{1.4}$$

1.2 Jukes-Cantor Model: Sequence Evolve as a Stochastic Process

To calculate the probability of a certain tree using Felsenstein's pruning algorithm, we require both the nucleotide transition probabilities ($P_{x\to y}$) and the initial distribution of nucleotides (p_x). Firstly, consider a discrete time condition: in each generation, the probability of a mutation occurring is p. Let X represent a random variable indicating the generation in which the first mutation is observed.

$$Pr(X = k) = (1-p)^{k-1} \cdot p$$

which is a geometrical distribution with $\mathbb{E}X = \frac{1}{p}$.

$$\begin{aligned} \mathbb{E}X &= \sum_{k=1}^{\infty} k \cdot (1-p)^{k-1} \cdot p \\ &= \sum_{k=0}^{\infty} (k+1) \cdot (1-p)^{k} \cdot p \\ &= \sum_{k=1}^{\infty} k \cdot (1-p)^{k} \cdot p + \sum_{k=0}^{\infty} (1-p)^{k-1} \cdot p \\ &= (1-p) \cdot \mathbb{E}X + p \cdot \frac{1}{p} \end{aligned}$$

Thus, the probability that a mutation has not occurred before the k-th generation is:

$$Pr(X > k) = (1-p)^k$$

The geometric distribution is characterized by the no-memory condition:

$$Pr(X = k + n | X > k) = Pr(X = n)$$

To obtain a continuous version, we divide the time period t into very short intervals ϵ, in which transition can only occur once with probability λ. As a result, the mutation probability is $p = \epsilon \cdot \lambda$. Therefore, the probability that a transition has not occured (i.e. the random variable $X > \frac{t}{\epsilon}$) over a time period t is: Notice that this represents a **Poisson process**, as we are only concerned with the occurrence of mutations, not the specific outcomes of those mutations.

$$Pr(X > \frac{t}{\epsilon}) = (1 - \lambda\epsilon)^{t/\epsilon} \overset{\lim\ \epsilon \to 0}{\longrightarrow} e^{-\lambda t}$$

which is an exponential distribution with $\mathbb{E}X = \frac{1}{\lambda}$.

Then, if the nucleotide changes randomly to one of {A, C, G, T}, we have:

$$\begin{aligned} P_{x \to y}(t) &= Pr(X < \frac{t}{\epsilon}) \cdot Pr(\text{last mutation leads to } y) \\ &= \frac{1}{4} \cdot (1 - Pr(X > \frac{t}{\epsilon})) \\ &= \frac{1}{4}\left(1 - e^{-\lambda t}\right) \end{aligned}$$

and:

$$\begin{aligned} P_{x \to x}(t) &= Pr(X > \frac{t}{\epsilon}) + Pr(X < \frac{t}{\epsilon}) \cdot Pr(\text{last mutation leads to } x) \\ &= Pr(X > \frac{t}{\epsilon}) + \frac{1}{4} \cdot (1 - Pr(X > \frac{t}{\epsilon})) \\ &= \frac{1}{4}\left(1 + 3e^{-\lambda t}\right) \end{aligned}$$

As a result, we obtain the simplest nucleotide trasition model:

Definition 1.2.1. ***Jukes-Cantor Model***

With the following assumption:

- *all sites independent of each other (given the tree)*
- *all p_x equal*
- *"mutations" appear at rate λ*
- *a "mutation" lets the site forget its state and sample the new one uniformly from {A,C,G,T}.(i.e. A can be replaced by another A)*

(in original paper for protein sequences)

Thus, we have:

$$P_{x \to y} = \begin{cases} \frac{1}{4}\left(1 - e^{-\lambda t}\right) \ldots\ldots (x \neq y) \\ \frac{1}{4}\left(1 + 3e^{-\lambda t}\right) \ldots\ldots (x = y) \end{cases}$$

and

$$p_x = \frac{1}{4} \ldots\ldots (x \in \{A, C, G, T\})$$

More generally, we could express Jukes-Cantor model in matrix form. First, we consider the discrete-time condition with intercal ϵ. In each step, the probability of changing to another nucleotide is $\frac{1}{4}\lambda\epsilon$. Then the transition probability could be written in a matrix $S_{epsilon}$:

$$S_\epsilon = \begin{pmatrix} 1-\frac{3}{4}\lambda\epsilon & \frac{1}{4}\lambda\epsilon & \frac{1}{4}\lambda\epsilon & \frac{1}{4}\lambda\epsilon \\ \frac{1}{4}\lambda\epsilon & 1-\frac{3}{4}\lambda\epsilon & \frac{1}{4}\lambda\epsilon & \frac{1}{4}\lambda\epsilon \\ \frac{1}{4}\lambda\epsilon & \frac{1}{4}\lambda\epsilon & 1-\frac{3}{4}\lambda\epsilon & \frac{1}{4}\lambda\epsilon \\ \frac{1}{4}\lambda\epsilon & \frac{1}{4}\lambda\epsilon & \frac{1}{4}\lambda\epsilon & 1-\frac{3}{4}\lambda\epsilon \end{pmatrix}$$

Let nucleotide distribution after n steps be denoted as X_n. Given the initial distribution X_0, the distribution after one step, X_1, is given by:

$$X_1 = X_0 \cdot S_\epsilon$$

and generally, X_n:

$$X_n = X_0 \cdot S_\epsilon^n$$

Here, the transitions of nucleotides are modeled by a Markov chain. To simplify the structure of $S = S_{epsilon}$, we assume, without loss of generality, that $\alpha = \lambda\epsilon$

Markov Chain

The equilibrium distribution in markov process is defined as $\boldsymbol{\pi}$, which fit:

$$\boldsymbol{\pi} \cdot S_\epsilon = \boldsymbol{\pi}$$

Note that $\boldsymbol{\pi}$ is an eigenvector of the matrix S with the eigenvalue 1.

$$S \cdot \boldsymbol{\pi} = \lambda \cdot \boldsymbol{\pi}$$

$$(S - I)\boldsymbol{\pi} = 0 \ldots\ldots (\lambda = 1)$$

To solve this matrix function, notice here $S = (1-\alpha)I + \frac{1}{4}\alpha J$, where J is an all ones matrix

The solution is $\boldsymbol{\pi} = \{\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}\}$ (to make sure $\sum \pi = 1$)

Next, we attempt to extend this to a continuous condition. Let $t = n\epsilon$,

$$S(t) = S_\epsilon(n)$$

To obtain the continuous trasition matrix $S(t)$, we need to calculate S_ϵ^n. A common approach is to diagonalize the matrix S_ϵ. If we can find an invertible matrix U, such that $S_\epsilon = U \cdot D \cdot U^{-1}$, where D is a diagonal matrix, then the power of matrix S_ϵ could be easily calculated as:

$$S_\epsilon^n = (U \cdot D \cdot U^{-1})^n = U \cdot D^n \cdot U^{-1}$$

For a invertible matrix U, $U \cdot U^{-1} = U^{-1} \cdot U = I$

Here, matrix S_ϵ can be easily diagonalize due to its excellent symmetry. First, its eigenvalue, obtained by solving $det|S - \lambda I| = 0$, are $\lambda = \{1, 1-\alpha, 1-\alpha, 1-\alpha\}$. And the corresponding eigenvectors, found by solving $(S - \lambda_i I)v_i = 0$, are $v_1 = [1,1,1,1]$, $v_2 = [1,-1,0,0]$, $v_3 = [0,1,-1,0]$, $v_4 = [0,0,1,-1]$. Thus,

$$U = (v_1, v_2, v_3, v_4) = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 1 & 0 \\ 1 & 0 & -1 & 1 \\ 1 & 0 & 0 & -1 \end{pmatrix} \quad U^{-1} = \frac{1}{4} \begin{pmatrix} -3 & 1 & 1 & 1 \\ 1 & -3 & 1 & 1 \\ 1 & 1 & -3 & 1 \\ 1 & 1 & 1 & -3 \end{pmatrix}$$

$$D = diag(\lambda) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1-\alpha & 0 & 0 \\ 0 & 0 & 1-\alpha & 0 \\ 0 & 0 & 0 & 1-\alpha \end{pmatrix}$$

here we obtain:

$$S_\epsilon^n = U \cdot D^n \cdot U^{-1} = \begin{pmatrix} \frac{3+(1-\alpha)^n}{4} & \frac{1-(1-\alpha)^n}{4} & \frac{1-(1-\alpha)^n}{4} & \frac{1-(1-\alpha)^n}{4} \\ \frac{1-(1-\alpha)^n}{4} & \frac{3+(1-\alpha)^n}{4} & \frac{1-(1-\alpha)^n}{4} & \frac{1-(1-\alpha)^n}{4} \\ \frac{1-(1-\alpha)^n}{4} & \frac{1-(1-\alpha)^n}{4} & \frac{3+(1-\alpha)^n}{4} & \frac{1-(1-\alpha)^n}{4} \\ \frac{1-(1-\alpha)^n}{4} & \frac{1-(1-\alpha)^n}{4} & \frac{1-(1-\alpha)^n}{4} & \frac{3+(1-\alpha)^n}{4} \end{pmatrix}$$

Substituting $\alpha = \lambda\epsilon$ and $t = n\epsilon$ into the matrix above, and let $lim\epsilon \to 0$,

$$(1-\alpha)^n = (1-\lambda\epsilon)^{t/\epsilon} \overset{\lim\ \epsilon \to 0}{\longrightarrow} e^{-\lambda t}$$

Then,

$$S(t) = \lim_{\epsilon \to 0} S_\epsilon^n = \frac{1}{4} \begin{pmatrix} 1+3e^{-\lambda t} & 1-e^{-\lambda t} & 1-e^{-\lambda t} & 1-e^{-\lambda t} \\ 1-e^{-\lambda t} & 1+3e^{-\lambda t} & 1-e^{-\lambda t} & 1-e^{-\lambda t} \\ 1-e^{-\lambda t} & 1-e^{-\lambda t} & 1+3e^{-\lambda t} & 1-e^{-\lambda t} \\ 1-e^{-\lambda t} & 1-e^{-\lambda t} & 1-e^{-\lambda t} & 1+3e^{-\lambda t} \end{pmatrix}$$

Here the solution for $S(t)$ can also be written as $S(t) = e^{Rt}$, where $R = S_\epsilon - I$. We will discuss this in more details later.

The result derived using Markov chain in matrix form is equivalent to the previous one obtained through the Poisson process.

1.3 Transition Probabilities and Equilibrium Distribution in General

In a more general case, we consider the nucleotide transition process as a continuous Markov process, without focusing on a specific example.

As we discussed earlier, what matters are the transition matrix and the equilibrium distribution. Again, we begin with a discrete model. The multistep transition matrix is given by:

$$S_\epsilon(n) = S_\epsilon^n \tag{1.5}$$

Let $t = n\epsilon$,

$$S(t) = S_\epsilon(n)$$

Then, the difference of trasition matrix at time t and $t + \epsilon$ is:

$$\frac{S(t+\epsilon) - S(t)}{\epsilon} = \frac{S_\epsilon^{n+1} - S_\epsilon^n}{\epsilon} = S(t) \cdot \frac{(S_\epsilon - I)}{\epsilon}$$

Note here $\frac{(S_\epsilon - I)}{\epsilon}$ is a matrix without time interval ϵ, we define it as the **Transition Rate Matrix** R:

$$R = \frac{(S_\epsilon - I)}{\epsilon}$$
$$= \begin{pmatrix} R_{AA} & R_{AC} & R_{AG} & R_{AT} \\ R_{CA} & R_{CC} & R_{CG} & R_{CT} \\ R_{GA} & R_{GC} & R_{GG} & R_{GT} \\ R_{TA} & R_{TC} & R_{TG} & R_{TT} \end{pmatrix}$$

In Jukes-Cantor model,

$$R = \begin{pmatrix} -3\alpha & \alpha & \alpha & \alpha \\ \alpha & -3\alpha & \alpha & \alpha \\ \alpha & \alpha & -3\alpha & \alpha \\ \alpha & \alpha & \alpha & -3\alpha \end{pmatrix}$$

shows the 'inflow' and 'outflow' of a state Then the derivative of transition matrix is given by:

$$\frac{\mathrm{d}S(t)}{\mathrm{d}t} = \lim_{\epsilon \to 0} \frac{S(t+\epsilon) - S(t)}{\epsilon} = S(t) \cdot R$$

The solution for this matrix differential equation is:

$$S(t) = e^{Rt}$$

This exponential form of matrix represents:

$$e^{Rt} = UT^tU^{-1} \tag{1.6}$$

Notice that matrices

$R = S_\epsilon - I$ and S_ϵ share the same eigenvectors. This is because for a vector x that satisfies $S \cdot x = \lambda \cdot x$,it must also hold that $(S - I) \cdot x = (\lambda - 1) \cdot x$. where U is the eigenvalue matrix of transition matrix R such that $R = UDU^{-1}$. And T is diagonal matrix whose entries are the exponentials to eigenvalues of R, $T = diag[e^\lambda]$

Matrix exponential

The Matrix exponential e^{Rt} is actually first defined as the solution of fuction $\frac{\mathrm{d}S(t)}{\mathrm{d}t} = S(t) \cdot R$. Another definition of matrix exponential e^{tR} is more effcient for calculating:

$$e^{tR} = \sum_{n=0}^{\infty} \frac{(tR)^n}{n!}$$

which can be made more stable by chosing $\beta > max\{\lambda_1, ..., \lambda_m\}$ and then using the variant:

$$e^{tR} = e^{-\beta t} \cdot \sum_{n=0}^{\infty} \frac{(\beta t)^n \cdot (I + R/\beta)^n}{n!}$$

Another approach is to use the limit, from which we calculated the discrete-time multistep trasition matrix, and its variant:

$$e^{tR} = \lim_{\epsilon \to 0} S_{epsilon}^{t/\epsilon} = \lim_{n \to \infty} \left(I + \frac{t}{n} R \right)^n = \lim_{n \to \infty} \left(\left(I - \frac{t}{n} R \right)^{-1} \right)^n$$

With a large n, the approximation:

$$e^{tR} \approx \left(I + \frac{t}{n} R \right)^n \approx \left(\left(I - \frac{t}{n} R \right)^{-1} \right)^n$$

This method works because β controlls the overflow caused by excessive exponential growth due to a large matrix norm of R

In the above we discussed trasition probabilities, which could be written as the matrix exponential of the so called transition rate matrix R: $S(t) = e^{tR}$. Now what matters is the equilibrium distribution π.

Firstly, note the markov property of the transition process:

$$S(t+s) = S(t) \cdot S(s)$$

The equilibrium distribution π should fit:

$$\forall t \geq 0, \pi \cdot S(t) = \pi \tag{1.7}$$

A easy way to obtain a solution is to choose $t = \epsilon$, as we did at the begining. This gives us:

$$\pi \cdot S_\epsilon = \pi$$

As we discussed before, π is the eigenvector of S_ϵ with eigenvalue $\lambda = 1$.

More general, take the derivative of the stationary distribution condition $\pi\pi \cdot S(t) = \pi$:

$$\begin{aligned} \pi \cdot S(t) &= \pi \\ \frac{\mathrm{d}\pi \cdot S(t)}{\mathrm{d}t} &= 0 \end{aligned}$$

Also, we know the derivative of of left part is:

$$\frac{\mathrm{d}\pi \cdot S(t)}{\mathrm{d}t} = \pi \cdot R \cdot S_(t)$$

which holds for all t, then we obtain:

$$\frac{\mathrm{d}\pi \cdot S(t)}{\mathrm{d}t}|_{t=0} = \pi \cdot R$$

$$\pi \cdot R = 0 \tag{1.8}$$

which means that the equilibrium distribution (π) remains unchanged, as the rate matrix R represents the inflow and outflow of the states. By equation (1.8) the equilibrium distribution can be calculated from the rate matrix R, which is necessary when we measure the probability of a certain tree using Felsenstein pruning algorithm.