Diego Giovane Pasqualin

SFREEMAP - MAPEAMENTO ESTOCÁSTICO DE ÁRVORES FILOGENÉTICAS LIVRE DE SIMULAÇÕES

Brasil

2016

Diego Giovane Pasqualin

SFREEMAP - MAPEAMENTO ESTOCÁSTICO DE ÁRVORES FILOGENÉTICAS LIVRE DE SIMULAÇÕES

Universidade Federal do Paraná

Departamento de Informática

Programa de Pós-Graduação em Informática

Orientador: Dr. Fabiano Silva

Coorientador: Dr. Marcos Soares Barbeitos

Brasil

2016

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DO PARANÁ
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
Setor CIÊNCIAS EXATAS
Programa de Pós Graduação em INFORMÁTICA
Código CAPES: 40001016034P5

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em INFORMÁTICA da Universidade Federal do Paraná foram convocados para realizar a arguição da Dissertação de Mestrado de **DIEGO GIOVANE PASQUALIN**, intitulada: "**SFREEMAP - MAPEAMENTO ESTOCÁSTICO DE ÁRVORES FILOGENÉTICA LIVRE DE SIMULAÇÕES** ", após terem inquirido o aluno e realizado a avaliação do trabalho, são de parecer pela sua APROVAÇÃO.

Curitiba, 23 de Fevereiro de 2016.

Prof FABIANO SILVA (UFPR)
(Presidente da Banca Examinadora)

Prof DANIEL WEINGAERTNER (UFPR)

Prof MÁRCIO ROBERTO PIE (UFPR)

Prof MARCOS SOARES BARBEITOS (UFPR)
Coorientador

MINISTÉRIO DA EDUCAÇÃO · UNIVERSIDADE FEDERAL DO PARANÁ · Setor de Ciências Exatas · Coordenação de Pós-Graduação em Informática

Agradecimentos

Tenho plena consciência da importância dos bons amigos que me acompanham, do carinho da minha namorada e da grande dedicação dos meus orientadores, que tanto apoiaram essa jornada. Sem dúvida as valiosas ideias passadas por todos serão base para o desenrolar da minha vida e carreira científica.

Em especial, agradeço também aos meus pais, os grandes responsáveis por tudo de bom que posso ter feito e vivido até hoje.

Se vi mais longe, foi por estar de pé sobre ombros de gigantes.
(Isaac Newton)

Resumo

Desde que Charles Darwin revolucionou a biologia em 1859 com “A Origem das Espécies”, a árvore da vida vem crescendo e seus ramos são gradativamente preenchidos e catalogados. Inicialmente um esforço exclusivo da Biologia, tal atividade recebeu reforço da estatística para validação de seus modelos evolutivos e também da computação para acelerar a análise.

A reconstrução de ancestrais hipotéticos dos organismos de interesse é de suma importância para a compreensão das forças que guiam a evolução das espécies e essa atividade ganhou especial interesse com o aprimoramento da tecnologia de sequenciamento de DNA. Atualmente existem vários métodos focados nessa tarefa, que recebem características dos organismos de interesse, morfológicas ou genéticas, e produzem como resultado uma árvore filogenética representando a história evolutiva dos organismos estudados.

Este trabalho descreve a implementação computacional de um desses métodos, desenvolvido por Vladimir Minin e Marc Suchard, que fornece não só a história evolutiva, mas um detalhamento das transições entre os diferentes organismos ao longo dos ramos na árvore. Esse método possui como diferencial o fato de ser analítico, eliminando assim a necessidade de simulações, que apresentam problemas com erros de aproximação e falta de critérios claros para uma convergência apropriada.

A principal contribuição do trabalho é uma implementação de alto desempenho do método em um pacote de software para o sistema R, contendo não somente uma implementação livre e eficiente, mas também ferramentas gráficas para análise de seus resultados.

Abstract

Since Charles Darwin permanently revolutionized biology in 1859 with the “The Origin of Species” the tree of life has been growing and its branches are gradually fulfilled and cataloged. Initially, and effort exclusive of biology, such activity has recently being reinforced by statistics in order to validade its evolutionary models and also by computer science, to improve the calculation process.

The reconstruction of ancestral states of organisms is of utmost importance to comprehend the guiding forces of evolution and this task has gained special interest with the improvement of the technics to decode DNA and genome sequencing. Currently, there are several methods focused on this task that receive traits of organisms of interest, which can be based on its behaviour or genetics, and produce as a result a phylogenetic tree representing the evolutionary history of the studied organisms.

This work describes the computer implementation of one of this methods, developed by Vladimir Minin and Marc Suchard, which provides not only the evolutionary history but also a breakdown of the transitions among the different organisms within the tree. What sets it apart from all different available methods is its characteristic of being analitical, not relying on simulations, which are subject to aproximation errors.

Lista de ilustrações

Lista de tabelas

Sumário

Introdução

A Filogenética é uma área de estudo da Biologia Evolutiva cujo objetivo é estabelecer as relações evolutivas entre grupos de organismos. Ela é em geral empregada através da análise morfológica ou, mais recentemente, do sequenciamento molecular de espécies de interesse e posterior aplicação de um modelo evolutivo sobre esse sequenciamento. O resultado do processo é uma hipótese sobre a história evolutiva do conjunto, usualmente representada através de uma árvore filogenética, ou simplesmente filogenia.

Uma filogenia exibe portanto as relações evolutivas como árvore de topologia[1] definida, indicando a sequência de ancestrais hipotéticos que deram origem aos organismos objetos de interesse. Os organismos, seus caracteres (qualquer atributo intrínseco e herdável do organismo, ex: antena) e estados dos caracteres (ex: antena birreme) considerados relevantes para a aplicação dos processos da filogenética são definidos pelo biólogo interessado na pesquisa.

As árvores filogenéticas podem ser representadas computacionalmente através de um grafo acíclico e conexo. Os nós terminais, ou folhas da árvore, representam os organismos de interesse. Os nós internos representam os ancestrais hipotéticos dos organismos, sendo portanto a raiz da árvore o ancestral comum à todos os seres representados. Dado um conjunto qualquer com n organismos distintos, o número de árvores que podem representar sua história evolutiva é dada pela equação 1 (4).

$$N_a = \frac{(2n-3)!}{2^{n-2}(n-2)!}, n >= 2 \tag{1}$$

O valor de N_a cresce de tal forma que para $n = 70$, $N_a = 5.86^{96}$, quantidade superior ao número estimado de prótons existentes no universo. É evidente que nem todas as árvores possíveis representam a realidade com igual exatidão, com isso em mente diversos métodos heurísticos foram desenvolvidos para reduzir o espaço de busca e encontrar a árvore, ou as árvores, que melhor representam o conjunto de organismos observados. Como medida de qualidade ou critério de otimização para as heurísticas podemos citar a Máxima Verossimilhança, Máxima Parcimônia e Inferência Bayesiana, que serão descritas ao longo do trabalho.

Uma árvore filogenética pode representar além da topologia e da história evolutiva, também a quantidade de mutações que ocorreram entre dois estados quaisquer, ou o tempo de persistência de um estado no tempo evolutivo, proporcional ao tamanho da aresta que liga dois nós consecutivos. Uma das características mais utilizadas na reconstrução de

[1] Forma da árvore, representada pelos seus ramos e nós, além das etiquetas nos nós folha, que representam os organismos.

caracteres discretos é o mapeamento estocástico (5), que permite o estudo de inúmeras características relevantes, tais como a correlação evolutiva entre espécies e identificação de mutações sinônimas e não-sinônimas (3).

Existem programas computacionais para auxiliar no processo de mapeamento estocástico. O BEAST (6) e PhyloBayes (7), por exemplo, possuem funções capazes de fornecer estimativas das taxas de transição entre nucleotídeos e aminoácidos (no caso da caracteres moleculares), ou entre estados de um caráter (para filogenias baseadas em dados morfológicos). Outros programas disponibilizam ferramental para o mapeamento estocástico propriamente dito, como o Bio++ (8) e o SIMMAP (9). O que esses possuem em comum é o uso de Cadeias de Markov de Tempo Contínuo (10) (CMTC) para simulação das trajetórias na árvore, método estatístico que possui parâmetros difíceis de ajustar e que podem prejudicar a precisão dos resultados (3).

Na tentativa de solucionar o problema com os parâmetros e erros de aproximação inerentes às CMTC, Minin e Suchard propõe em 2008 (3) um método analítico de mapeamento estocástico, que utiliza a decomposição espectral do gerador infinitesimal Q da cadeia de Markov para estimar características de interesse na filogenia. O método foi proposto mas nenhuma implementação livre foi disponibilizada até a presente data.

O objetivo desse trabalho é portanto implementar de forma eficiente o método analítico de Minin e Suchard para mapeamento estocástico. Em uma primeira etapa o algoritmo será implementado totalmente em R (11), que é a plataforma comumente utilizada pelos pesquisadores da área. Em seguida, serão identificados os pontos de computação mais intensiva, que serão então otimizados, paralelizados e reescritos em uma linguagem de alto desempenho. Como resultado deste trabalho, será disponibilizado como software livre um pacote para R contendo, além do método de mapeamento em si, uma série de ferramentas gráficas e textuais para análise dos mapeamentos.

O presente documento está organizado como segue. O capítulo 1 dispõe a fundamentação teórica, detalhando termos e conceitos da biologia e estatística que serão relevantes para o desenvolvimento do trabalho. Em seguida, o capítulo 2 apresenta uma revisão bibliográfica, enumerando a base e métodos alternativos de mapeamento estocástico. O capítulo 3 detalha o artigo de Minin e Suchard para mapeamento estocástico, objeto principal deste trabalho. O processo de implementação e experimentação são apresentados nos capítulos 4 e 5, respectivamente. Após o capítulo de conclusão e referências bibliográficas, dois apêndices são disponibilizados. O apêndice A corresponde a documentação oficial do pacote, enquanto o apêndice B contém exemplos de uso detalhados.

1 Fundamentação Teórica

Este capítulo visa estabelecer a base teórica para acompanhamento deste trabalho, detalhando os principais termos e conceitos que serão utilizados posteriormente na descrição do método de mapeamento estocástico e implementação do algoritmo. A seção 1.1 apresenta alguns conceitos importantes relacionados a estatística. Em seguida, a seção 1.2 apresenta um breve histórico e descrição do processo de análise filogenética. Por fim, a seção 1.3 detalha alguns métodos para definição da topologia da árvore.

1.1 Estatística

1.1.1 Cadeias de Markov

Uma **cadeia de Markov** é um **processo estocástico**[1] que gera uma sequência (ou cadeia) de estados discretos, dados um conjunto finito de estados possíveis e uma **matriz de taxa de transição instantânea** Q, que descreve a taxa com que as mudanças entre os estados ocorrem na cadeia em uma unidade de tempo. A matriz Q, também conhecida como matriz de Markov ou matriz estocástica, pode ser definida como $P_{ij} = Pr(X_{n+1} = j|X_n = i)$, significando que, P_{ij} é igual a probabilidade do próximo estado na cadeia (X_{n+1}) ser igual a j, dado que o estado atual (X_n) é i. Como consequência dessa definição, fica evidente que a probabilidade de transição para o próximo estado depende somente do estado atual, não importando o caminho percorrido para chegar até ele.

Cadeias de Markov podem ser utilizadas em qualquer processo que possa ser descrito como uma sequência de estados, bastando enumerar os estados, as possíveis transições entre eles e a probabilidade de cada transição ocorrer. Para exemplificar, utilizaremos uma agência de seguros de automóveis (13), que define o preço do serviço enquadrando os clientes nas categorias de baixo e alto risco, indicando respectivamente situação onde o condutor não cometeu nenhuma infração no ano anterior, ou situação onde ele cometeu alguma infração.

Dados históricos da agência de seguros indicam que a probabilidade de um cliente, atualmente considerado de alto risco, continuar nessa categoria na próxima renovação anual é de 60%, de forma que a chance de ele passar para a categoria de baixo risco é de 40%. Por outro lado, a chance de um motorista de baixo risco continuar na mesma categoria após renovação é de 85% e 15% é a chance de ele ser movido para alto risco.

[1] Um processo estocástico é qualquer processo cuja evolução possa ser descrita e analisada em termos probabilísticos (12).

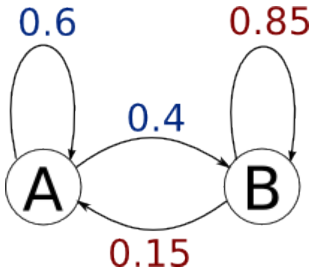


Figura 1 – Ilustração de uma cadeia de Markov de tempo discreto com número finito de estados.

Com esses dados em mãos, a empresa pode então questionar, por exemplo, qual será a distribuição de clientes entre as categorias de alto e baixo risco nos próximos anos.

A Figura 1 exibe uma representação tradicional do problema modelado como uma cadeia de Markov, com "A" representando o estado de alto risco e "B" o estado de baixo risco. Os arcos indicam possíveis transições entre os estados em cada passo da cadeia, que no problema definimos como o intervalo de um ano. Os valores acima dos arcos indicam as probabilidades de transição nesse tempo.

Também é possível modelar as transições através da matriz Q mencionada anteriormente, como abaixo:

$$\mathrm{P} = \begin{matrix} \\ \textit{Alto} \\ \textit{Baixo} \end{matrix} \begin{matrix} \begin{matrix} \textit{Alto} & \textit{Baixo} \end{matrix} \\ \begin{bmatrix} 0.6 & 0.4 \\ 0.15 & 0.85 \end{bmatrix} \end{matrix}$$

.

Na matriz, as linhas podem ser lidas como "origem" e as colunas como "destino", ou seja, o valor 0.4 correspondente a linha "Alto", coluna "Baixo", indica a probabilidade de um cliente na categoria alto risco ser promovido para a categoria baixo risco.

Em seguida precisamos definir o estado inicial. Em nosso exemplo, vamos considerar que a probabilidade de um cidadão escolhido aleatoriamente ter cometido uma infração no ano anterior é de 10% e, consequentemente, de não ter cometido qualquer infração é de 90%. Multiplicando o estado inicial pela matriz Q temos então a probabilidade do cidadão se encontrar nos estados de alto e baixo risco no ano seguinte que, como demonstrado abaixo, seriam de 19.5% e 80.5% respectivamente.

$$P^1 = \begin{matrix} \begin{matrix} \textit{Alto} & \textit{Baixo} \end{matrix} \\ \begin{bmatrix} 0.1 & 0.9 \end{bmatrix} \end{matrix} \begin{matrix} \\ \textit{Alto} \\ \textit{Baixo} \end{matrix} \begin{matrix} \begin{matrix} \textit{Alto} & \textit{Baixo} \end{matrix} \\ \begin{bmatrix} 0.6 & 0.4 \\ 0.15 & 0.85 \end{bmatrix} \end{matrix} = \begin{matrix} \begin{matrix} \textit{Alto} & \textit{Baixo} \end{matrix} \\ \begin{bmatrix} 0.195 & 0.805 \end{bmatrix} \end{matrix}$$

Utilizando o resultado acima como estado inicial para uma segunda iteração, teremos então as probabilidades para o segundo ano. Repetindo o processo N vezes, a

companhia pode saber qual será a proporção de clientes nos estados de baixo e alto risco em N anos. Por exemplo, para $N = 2$:

$$P^2 = P^1 . \begin{matrix} \\ Alto \\ Baixo \end{matrix} \begin{matrix} \begin{matrix} Alto & Baixo \end{matrix} \\ \begin{bmatrix} 0.6 & 0.4 \\ 0.15 & 0.85 \end{bmatrix} \end{matrix} = \begin{matrix} \begin{matrix} Alto & Baixo \end{matrix} \\ \begin{bmatrix} 0.237 & 0.762 \end{bmatrix} \end{matrix}$$

Uma forma mais geral, no entanto, consiste em multiplicar o vetor de estados iniciais v pela exponenciação da matriz de transições P, com N representando o número de anos que se deseja avançar, como segue:

$$P_N = vP^N \tag{1.1}$$

Uma propriedade interessante das cadeias de Markov é a estabilidade alcançada quando N tende a um valor alto. Em nosso exemplo, com $N > 25$ já não existe variabilidade e as probabilidades para alto e baixo risco se estabilizam, respectivamente, em 0.272 e 0.727. Usando essas probabilidades como vetor inicial v temos então que $vP = v$, para valores elevados de N. Com essa propriedade torna-se possível calcular o **vetor de estado estacionário** sem o conhecimento prévio do estado inicial, substituindo os valores do vetor inicial por variáveis:

$$P = \begin{matrix} \begin{matrix} Alto & Baixo \end{matrix} \\ \begin{bmatrix} x & y \end{bmatrix} \end{matrix} \begin{matrix} \\ Alto \\ Baixo \end{matrix} \begin{matrix} \begin{matrix} Alto & Baixo \end{matrix} \\ \begin{bmatrix} 0.6 & 0.4 \\ 0.15 & 0.85 \end{bmatrix} \end{matrix} = \begin{matrix} \begin{matrix} Alto & Baixo \end{matrix} \\ \begin{bmatrix} x & y \end{bmatrix} \end{matrix}$$

Executando as operações algébricas necessários concluí-se que $x = 0.272$ e $y = 0.727$, exatamente o mesmo resultado obtido anteriormente, eliminando a necessidade de exponenciar a matriz P repetidas vezes e com uma conclusão interessante, a de que o estado inicial na verdade não tem qualquer influência quando a cadeia é longa o suficiente.

1.1.2 Expectância

A **expectância** (também conhecida como **valor esperado**, ou **esperança matemática**) de uma variável representa a média obtida pela repetição do experimento um número de vezes que tende ao infinito. Mais precisamente, para obter a expectância soma-se cada possível valor da variável aleatória ponderado pela sua probabilidade de ocorrer. Para uma variável discreta X com k valores possíveis x_i e suas respectivas probabilidades p_i, temos:

$$E[X] = \sum_{i=1}^{k} x_i p_i \tag{1.2}$$

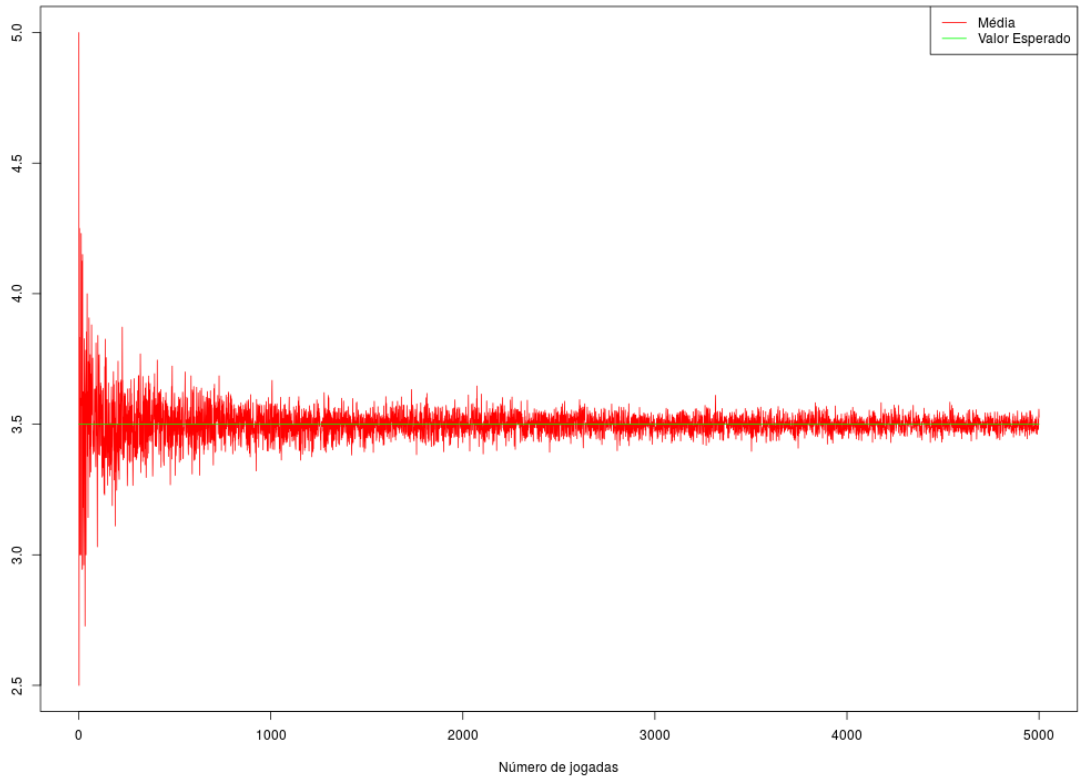


Figura 2 – Ilustração da convergência da média para a expectância.

Utilizando o lançamento de dados como exemplo temos que:

$$\begin{aligned} E[X] &= \sum_{i=1}^{6} x_i p_i \\ &= 1 \times \frac{1}{6} + 2 \times \frac{1}{6} + ... + 6 \times \frac{1}{6} \\ &= 3.5 \end{aligned} \tag{1.3}$$

A figura 2 ilustra a convergência da média para a expectância, conforme o número de lançamentos de dados aumenta.

Para variáveis contínuas que possuem uma **função densidade de probabilidade**[2] $f(x)$ o conceito é o mesmo, porém utiliza-se a integral para cálculo da expectância, como segue:

$$E[X] = \int_{-\infty}^{+\infty} x f(x) dx \tag{1.4}$$

[2] Uma função densidade de probabilidade representa as ocorrências de uma variável contínua em sua distribuição de probabilidade.

1.1.3 Verossimilhança

Verossimilhança (no inglês, *likelihood*) é um termo frequentemente confundido com probabilidade, porém ambos servem a propósitos diferentes. A probabilidade é utilizada para indicar a chance de um evento futuro ocorrer quando todos os resultados possíveis são conhecidos. Ela pode indicar, por exemplo, a probabilidade em se obter cara cinco vezes consecutivas no lançamento de uma moeda não viciada. A verossimilhança, em exemplo similar, indica a probabilidade de uma moeda ser viciada, dado que obtivemos cara cinco vezes consecutivas. Na filogenética a verossimilhança é proporcional à probabilidade de uma árvore ser verdadeira, de acordo com a distribuição dos estados nas folhas da árvore, que correspondem a observações coletadas de organismos fósseis e/ou viventes.

Mais formalmente, a verossimilhança de um conjunto de parâmetros m, dados os resultados observados D, é proporcional à probabilidade dos resultados observados dados os parâmetros, ou seja:

$$L(m|D) \propto P(D|m) \tag{1.5}$$

1.1.4 Intervalo de Credibilidade

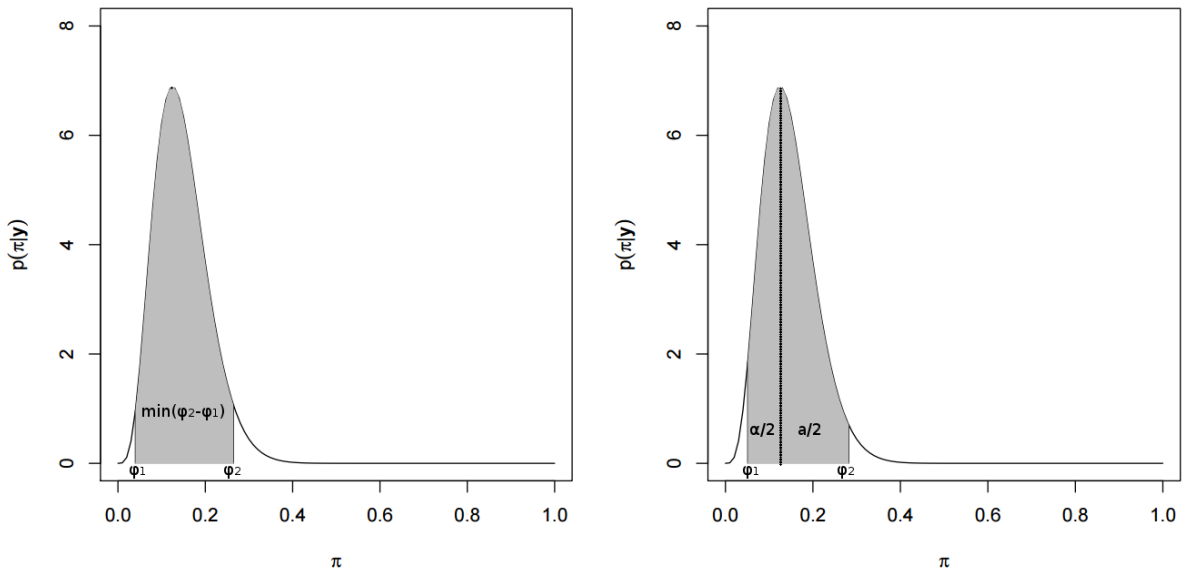
O **intervalo de credibilidade** é o equivalente bayesiano do intervalo de confiança da estatística frequentista, porém ambos costumam ser usados de forma indistinta. O conceito envolve definir um valor α tal que $1-\alpha$ da **massa de probabilidade** (na prática, a área sob a curva do gráfico) esteja contida no intervalo $[\phi_1, \phi_2]$ de uma distribuição de probabilidade que delimita uma certa credibilidade para um parâmetro ϕ, estimado pelo teorema de bayers (equação 1.6).

$$P(m|D) = \frac{P(D|m)P(D)}{P(m)}, \tag{1.6}$$

Um valor comum para α é 0.05, significando que a probabilidade do parâmetro ϕ residir no intervalo $[\phi_1, \phi_2]$ é de $1-\alpha$, ou 95%. Em outras palavras, 95 entre 100 vezes o valor real estará dentro do intervalo delimitado por ϕ_1 e ϕ_2.

Existem valores diferentes que podem ser atribuídos a $[\phi_1, \phi_2]$ que resultam na mesma credibilidade desejada. As duas principais convenções para escolha dos valores são (14):

- Escolher o menor intervalo $[\phi_1, \phi_2]$ que compreenda a credibilidade desejada. Isso é chamado de **maior densidade a posteriori**, ou **HPD** (do inglês, *highest posterior density*).



(a) (b)

Figura 3 – (a) ilustra a maior densidade a posteriori (HPD) e (b) o intervalo de credibilidade simétrico.

- Escolher o intervalo tal que exista um valor igual da função massa de probabilidade em ambos os lados do intervalo. Ou seja, deve-se escolher $[\phi_1, \phi_2]$ tal que $P(\phi > \phi_1|D) = P(\phi < \phi_2|D) = \frac{\alpha}{2}$. Isso é chamado de **intervalo de credibilidade simétrico**.

A Figura 3 ilustra as duas convenções.

1.1.5 Bootstrapping

Bootstrapping é um método estatístico proposto por Bradley Efron em 1979 (15). Carateriza-se por estimar distribuições através da criação de conjuntos de dados artificiais, utilizando para tal reamostragens sucessivas do conjunto de dados original (16). É especialmente útil quando tais distribuições revelam-se difíceis de serem calculadas de forma analítica.

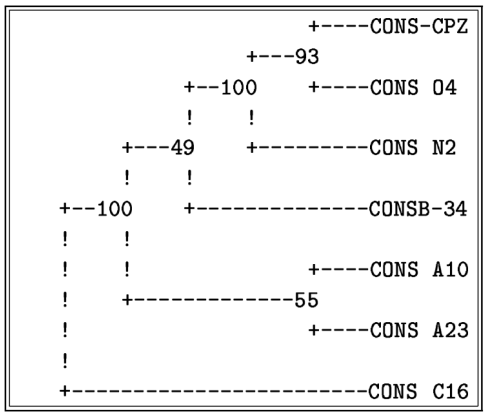
Aplicado a filogenia primeiramente por Felseinstein (17), o método ajuda a estimar a confiança dos **clados**[3] de uma árvore obtida via Máxima Verossimilhança ou Máxima Parcimônia (seção 2.1), de acordo com a frequência com que aparecem no processo de reamostragem. A Figura 4a apresenta uma matriz D de sequenciamento de DNA de espécies do vírus HIV previamente **alinhada**[4], onde as linhas representam as $n = 7$ espécies estudadas e as colunas os $m = 30$ caracteres.

[3] Ancestral hipotético e seus organismos descendentes, representado como uma subárvore gerada a partir de um nó.

[4] O alinhamento objetiva a homologia posicional, ou seja, a ideia de que uma coluna na matriz de sequenciamento representa uma mesma característica nos diferentes organismos.

```
1 CONS_A-10    KKEEEEALLT GADTVVVLEE INLGGKKKPK
2 CONS_A23     RREEEEALLT GADTVVVLEE INLGGKKKPK
3 CONSB-34     KKEEEEALLT GADTVVVLEE MNLGGRKKPK
4 CONS_C16     KKEEEEALLT GADTVVVLEE INLGGKKKPK
5 CONSEN_N2    RREEEEALLT GADTVVVIEE ?QLGGKKKPK
6 CONSEN_04    CCEEEEVLLT GADTVVVLNN IQLGGKTTPK
7 CONSENCPZ    ??EEEEALLT GADTVVVIDD IQLGG?RRPK
```

(a) Matriz de sequenciamento de DNA com as espécies representadas nas linhas e os caracteres nas colunas. Os pontos de interrogação em certos caracteres indicam incerteza no processo de sequenciamento e seus estados podem ser estimados pelos métodos da seção 2.1 ou simplesmente ignorados pelo *bootstrapping*.



(b) Árvore consenso, resultado do método de *bootstrapping* aplicado a matriz em (a). Nota-se que o clado constituído pelas espécies CONS-CPZ e CONS 04 e CONS N2 aparecem em 100% das amostragens e possuem portanto suporte ideal.

Figura 4 – Matriz de sequenciamento de DNA (a) e árvore consenso (b). Ambas as figuras foram retiradas do artigo (1), com dados provenientes do banco de dados de HIV LANL (2).

O método descrito por Felsenstein (17) procede através da geração de uma matriz D^* com as mesmas dimensões de D, composta pela seleção aleatória, com reposição, de m colunas da matriz D. Assim, D^* pode conter, por exemplo, em sua 1^a coluna a 7^a coluna de D, em sua 21^a coluna a 3^a coluna de D, enquanto que a 5^a coluna de D pode aparecer duas vezes em D^* e a 28^a coluna de D sequer ser amostrada em D^*.

Todo o processo é repetido B vezes e em seguida é feita uma contagem da frequência de amostragem de cada clado, gerando a **matriz consenso** (17), a qual chamaremos de D_c, que contém os clados não conflitantes com maior frequência de aparição nas amostras feitas.

A topologia exibida na Figura 4b pode ser construída de diversas formas, por exemplo através do cálculo da distância Euclidiana entre as espécies (18). Para o cálculo da distância interpreta-se (A, C, T, G) numericamente como $(1, 2, 5, 6)$, por exemplo, gerando uma matriz 7×7 contendo a distância entre as espécies. Em seguida conecta-se as espécies com menor distância entre si em um clado, gerando uma nova matriz de distâncias com dimensões 6×6, composta pelas cinco espécies não envolvidas na junção mais a codificação que representa o clado recém formado pelas duas espécies. Esse processo é repetido n vezes, quando todos os clados estarão conectados e a topologia completamente definida. Outras formas mais sofisticadas de construir a topologia a partir dos dados de sequenciamento são descritas no artigo (18).

Segundo Berry (19), *"valores altos para o bootstrapping*[5] *(pertos de 100%) indicam*

[5] O autor utiliza a expressão "alto valor de *bootstrapping*" como sinônimo de um clado com alta

suporte uniforme, isto é, se o valor de um certo clado é próximo de 100%, aproximadamente todos os caracteres relevantes para este grupo concordam que ele é um grupo". Normalmente, porém, considera-se um clado com alto suporte aquele que possui frequência maior ou igual a 70% (20).

O método clássico de *bootstrapping* introduzido por Felsenstein (17) e amplamente utilizado pressupõe que os caracteres evoluem independentemente, o que nem sempre é verdade (1). Existem **modelos evolutivos** que descrevem matematicamente a dinâmica da evolução do caráter e que podem ser utilizados em conjunto com o método para contornar esse problema, mas dificuldades de ajuste e tempo computacional excessivo (21) dificultam seu uso prático.

1.2 Análise Filogenética

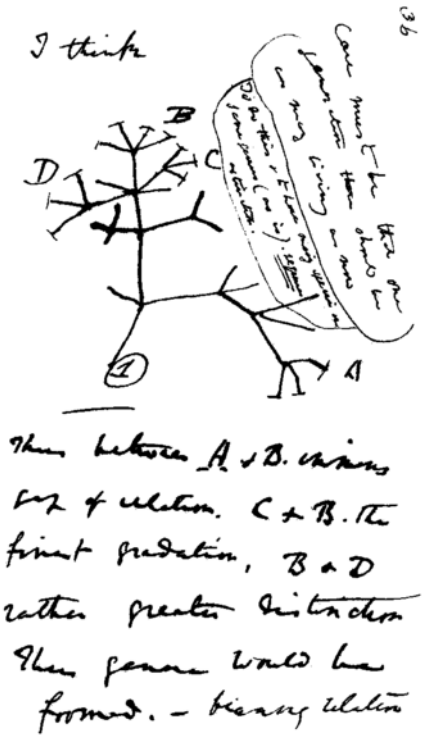


Figura 5 – Árvore filogenética feita por Charles Darwin em 1837.

Ao biólogo alemão Ernst Haeckel é comumente atribuído o crédito pela criação da palavra Filogenia, cunhada no livro "Generelle Morphologie der Organismen" (22) de 1866. O crédito sobre a primeira representação moderna de uma árvore filogenética é dado a Charles Dawin, devido ao rascunho encontrado em um dos seus livros de anotações escrito por volta de julho de 1837 (Figura 5). Entretanto, a formalização da teoria só ocorreu na

frequência de aparição no processo de amostragem.

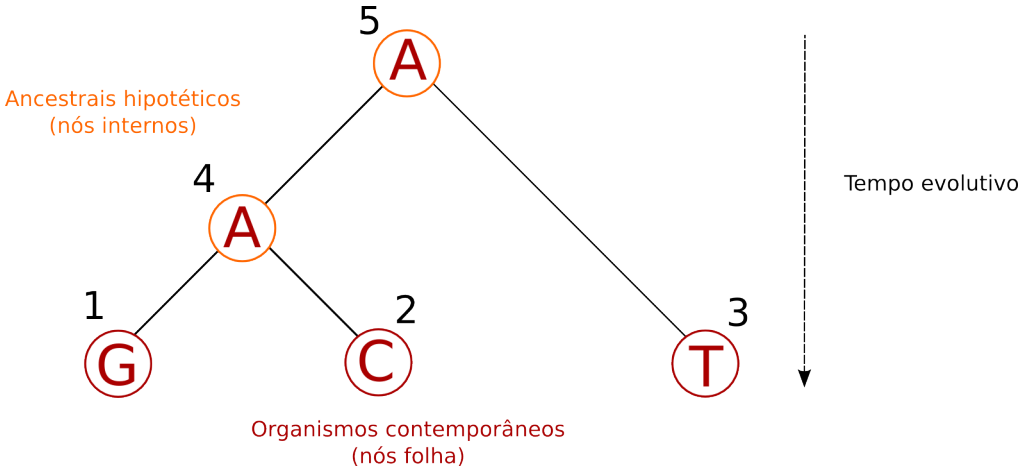


Figura 6 – Árvore filogenética moderna. Em destaque os nós folhas (organismos de interesse) e nós internos (ancestrais hipotéticos) enumerados, dentro dos nós os estados do caráter genético em estudo (A, C, T e G) e ao lado a indicação do tempo evolutivo da árvore.

década de 1950, com o trabalho de Willi Hennig em seu livro que, traduzido para o inglês, veio a se chamar *Phylogenetic Systematics* (23).

A Figura 6 destaca uma representação moderna de árvore filogenética. Os nós folha representam os **organismos de interesse**, ou **unidades taxonômicas**[6] **operacionais** (UTO). Os nós internos, ou **unidades taxonômicas hipotéticas** (UTH), são ancestrais **hipotéticos**, estimados pelos métodos de **reconstrução de caracteres**[7] apresentados na seção 2.1. A descendência entre os organismos é indicada pela conexão dos ramos (arestas no grafo) aos nós, de forma que os nós 1 e 2 possuem 4 como ancestral comum, que por sua vez compartilha com 3 o ancestral 5, raiz da árvore.

Os organismos de interesse são definidos por **caracteres** e seus **estados**. Um caráter, em princípio, pode ser qualquer atributo genotípico, fenotípico, morfológico, fisiológico ou mesmo comportamental do organismo, desde que intrínseco e herdável. Os estados são versões particulares dos caracteres. O caráter "cabelo", por exemplo, pode possuir os estados "liso" e "ondulado", ou também "cabelo liso" pode ser considerado um caráter por si só, com os estados "presente" e "ausente" para identificar se o organismo apresenta o caráter em questão ou não. Voltando a Figura 6, é possível observar a evolução de um caráter genético que pode assumir os estados A, C, T ou G.

A composição de caracteres e estados é a base fundamental da análise filogenética e é definida pelo pesquisador como uma reflexão da relevância, comparabilidade e homologia[8] das características dos organismos.

---

[6] Unidade em um sistema de classificação dos seres vivos. Um táxon pode ser uma espécie, reino, ou mesmo um único indivíduo em particular.

[7] Processo responsável por estimar o estado do caráter nas unidades taxonômicas hipotéticas.

[8] "Mesmo órgão em diferentes organismos com respeito a toda variação de forma e função" (24).

Como mencionado na introdução deste trabalho, o método de **mapeamento estocástico** introduzido por Nielsen (5) permitiu não somente a reconstrução dos caracteres ancestrais mas também um mapeamento das transições ocorridas entre os nós, facilitando o estudo de importantes propriedades da trajetória evolutiva, entre elas a classificação das mutações genéticas em sinônimas[9] e não-sinônimas (3), a detecção de co-evolução[10] (26) e a seleção positiva[11] (9).

Em resumo, o processo de reconstrução de caráteres ancestrais, com os organismos de interesse nos nós folha e seus ancestrais hipotéticos nos nós internos, segue o seguinte procedimento:

1. Amostragem de caracteres dos organismos de interesse que irão compor os nós folha;
2. Obtenção dos caracteres e estados que melhor os descrevem. Tais caracteres podem ser genotípicos, fenotípicos, morfológicos ou fisiológicos, obtidos através de observação visual e estudo dos hábitos dos invidívuos ou sequenciamento de DNA;
3. Geração da topologia, ou conjunto de topologias, de acordo com os métodos propostos na seção 1.3;
4. Reconstrução de caracteres ancestrais, cujos métodos estão descritos na seção 2.1;
5. Mapeamento estocástico para detalhamento dos ramos das árvores geradas, seção 2.2.

Ao longo do texto o termo nó folha será utilizado como sinônimo para organismo de interesse e nó interno como sinônimo para ancestral hipotético.

1.3 Métodos para Escolha da Topologia da Árvore

Os métodos de construção de árvores filogenéticas (seção 2.1) recebem como parâmetro a **topologia** da árvore, que nada mais é do que a forma da árvore, representada pelos seus ramos e nós, além das etiquetas nos nós folha, que representam os estados do caráter em estudo nos organismos de interesse. Um exemplo de topologia é exibido na Figura 7.

A necessidade de escolher certas topologias para estudo vem da impossibilidade em se analisar todas as topologias possíveis, quantidade esta que cresce de forma fatorial de

[9] Mutações sinônimas são aquelas que implicam na geração de um mesmo aminoácido e consequentemente de uma mesma proteína, não havendo portanto alteração no gene e não representando qualquer papel na evolução da espécie. Mutações não-sinônimas, em contrapartida, são as que produzem de fato uma modificação no gene.

[10] Co-evolução, nesse contexto, indica correlação entre estados de caracteres (25). Por exemplo, se a presença de exibição de cio está correlacionada a sistema de acasalamento promíscuo e se sua ausência está correlacionada à reprodução monogâmica.

[11] Modificação genética que aumenta a aptidão à sobrevivência do organismo.

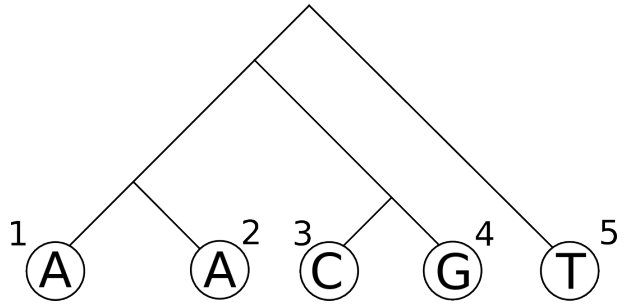


Figura 7 – Topologia de uma árvore filogenética. Apenas organismos de interesse presentes, comprimento dos ramos podem ou não fazer parte da topologia.

acordo com o número de organismos de interesse estudados. Considerando esse problema, alguns métodos foram propostos para reduzir o espaço de busca e limitar o número de árvores analisadas, procurando sempre garantir que a árvore, ou o conjunto de árvores selecionadas, represente satisfatoriamente a árvore real.

O método de análise Bayesiana foi desenvolvido por Rannala e Yang (27) e tem como objetivo gerar uma distribuição a *posteriori* de topologias, com o sequenciamento de DNA e modelo evolutivo como parâmetros. Neste método a matriz inicial é constante e a frequência de um clado na distribuição é interpretada como probabilidade, de forma que uma frequência de 100% no *bootstrapping*, em linhas gerais, equivale a probabilidade igual a 1 na análise Bayesiana.

A implementação do método sugerida por (28) utiliza simulação de Monte Carlo via Cadeia de Markov através do algoritmo de subida de encosta *Metropolis-Hastings* (29), com a topologia da árvore, comprimento dos ramos e modelo evolutivo como parâmetros de otimização.

No algoritmo de *Metropolis-Hastings* (29), somente a topologia atual e a nova topologia gerada são armazenadas (assim como nas Cadeias de Markov, os estados passados são desconsiderados para a geração do novo estado). Para iniciar o processo, uma topologia τ_t com parâmetros aleatórios é gerada e sua probabilidade a *posteriori* é calculada (detalhes sobre o cálculo na seção 2.1.3). Em seguida, uma modificação aleatória é feita em um dos parâmetros, gerando uma nova topologia τ'. Se a nova topologia possui maior probabilidade a *posteriori* ela é aceita e $\tau_{t+1} = \tau'$, caso contrário ela é aceita com uma determinada probabilidade r, definida por $P(\tau_t)/P(\tau')$.

É importante notar que o método de análise Bayesiana possui as mesmas falhas inerentes aos algoritmos e métodos que utiliza, como as questões sobre convergência da Cadeia de Markov e máximos locais na subida de encosta.

2 Revisão Bibliográfica

O capítulo anterior apresentou termos e conceitos importantes e descreveu alguns passos para obtenção da árvore filogenética, em especial a enumeração dos organismos do estudo, definição dos caracteres e estados e geração da topologia. Este capítulo descreve e compara métodos para reconstrução dos estados ancestrais na seção 2.1 e, posteriormente, detalha métodos de mapeamento estocástico (seção 2.2), comparando o método implementado nesse trabalho com outros já estabelecidos e amplamente utilizados.

2.1 Métodos para Reconstrução dos Estados Ancestrais

Após escolha da topologia da árvore, o próximo passo é descobrir a história evolutiva dos organismos de interesse. Os três métodos apresentados nas subseções seguintes estão entre os mais utilizados na área (30)(31) e possuem características distintas.

A Máxima Parcimônia assume que árvores com o menor número de transições de estados estão mais próximas do que seria a árvore correta, enquanto que o método de Máxima Verossimilhança é fundamentado no conceito estatístico que carrega seu nome e permite a inserção de um modelo evolutivo ao estimar a validade da árvore. O último método analisado é o de Inferência Bayesiana, que retorna não uma árvore específica, mas uma função de densidade probabilística, um conjunto de árvores com uma distribuição de probabilidade.

2.1.1 Máxima Parcimônia

A máxima parcimônia tem como princípio filosófico a "Navalha de Ockham", argumento dialético sugerido por Guilherme Ockham que estabelece como preferencial a escolha pela hipótese mais simples frente a duas hipóteses que explicam um fenômeno com igual precisão, onde por hipótese mais simples entende-se a que possui o menor número de suposições e pressupostos. Na filogenia, a hipótese mais simples é a que implica na menor quantidade de transições entre estados na árvore. O número de transições é portanto o critério de otimização e a árvore com menor transições é dita a mais parcimoniosa. É importante ressaltar que podem existir várias árvores a cumprir o critério de otimização e se apresentarem como igualmente parcimoniosas, situação onde as implementações costumam retornar todas as árvores.

O algoritmo de Sankoff e Rousseau (32), proposto em 1975, resolve o problema utilizando programação dinâmica[1] e encontra todas as árvores mais parcimoniosas para

[1] Técnica de programação que pode ser aplicada no caso onde a combinação das soluções ótimas de partes

uma dada entrada. O algoritmo recebe como entrada: a topologia da árvore, com os organismos de interesse e seus estados pré-determinados nos nós folha e uma matriz de transição instantânea de estados Q, que indica o custo de transição entre os estados e que pode ser definida diretamente pelo pesquisador ou estimada através de modelos evolutivos. O resultado do algoritmo são as árvores, ou a árvore, com os estados estimados nos nós internos.

São dois os passos necessários para resolução e ambos envolvem o percurso completo da árvore. No primeiro, chamado de percurso pré-ordem, o algoritmo percorre a árvore das folhas para a raiz, registrando os custos de todos os estados possíveis em cada vértice e atribuindo os possíveis valores finais para a raiz. O segundo passo é chamado de percurso pós-ordem e percorre a árvore da raiz para as folhas, atribuindo então os estados nos nós internos.

Seja V o conjunto de nós da árvore e $v \in V$, denotaremos $d(v)$ como o nó filho direito de v e $e(v)$ como o nó filho esquerdo de v. A operação $Q(s \to s')$ retorna o custo de transição do estado s (mais basal[2]) para o estado s', de acordo com a matriz de transição Q. Definimos por fim R_v como o custo acumulado em cada vértice interno, dado pela equação 2.2, que descreve o percurso pré-ordem. Como passo inicial do algoritmo, o custo do conjunto D dos organismos de interesse é definido de acordo com a equação 2.1, que indica intuitivamente a impossibilidade da existência no organismos de interesse de um estado diferente do observado. Os passos de percurso pré-ordem e pós-ordem são ilustrados através das Figuras 8 e 9, respectivamente.

$$Q_{v \in D}(s \to s') = \begin{cases} 0, & \text{se } s = s' \\ \infty, & \text{caso contrário} \end{cases} \tag{2.1}$$

$$R_v(s) = min[Q(s \to s') + R_{d(v)}(s')] + min[Q(s \to s') + R_{e(v)}(s')] \tag{2.2}$$

O método da parcimônia é muito simples e intuitivo, fato que mantém sua relevância para a área tanto no uso prático como para o ensino da Filogenia (33). Apesar disso, porém, outros métodos tem se mostrado mais eficazes. A discussão sobre a acurácia da parcimônia começa em seu princípio filosófico, com o pressuposto de que a simplicidade da árvore é um indicativo de sua corretude, afirmação nem sempre verdadeira (34).

Alguns autores apontam que o método é muito sensível a pequenas falhas no sequenciamento do DNA (33) e apresenta inconsistência estatística[3] em árvores com uma

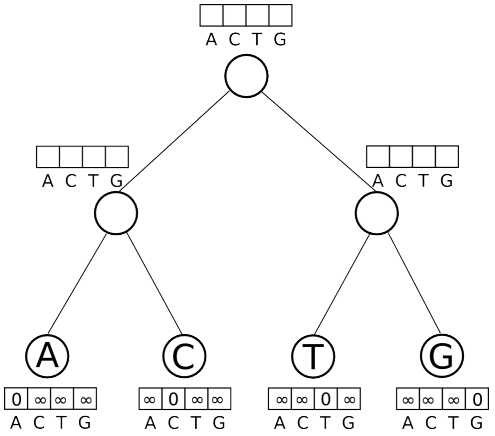
de um problema é também a solução ótima do problema como um todo. Na prática a programação dinâmica armazena resultados parciais para uso futuro, reduzindo recálculo.

[2] Mais antigo, ou mais próximo da raíz.

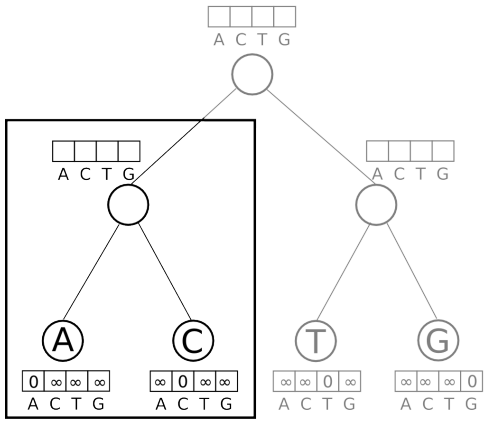
[3] Um método de reconstrução de caracteres ancestrais é dito estatisticamente consistente se a probabilidade de reconstrução da árvore correta converge para a certeza conforme a quantidade de dados coletados aumenta.

	A	C	T	G
A	0	9	3	4
C	9	0	4	4
T	3	4	0	2
G	4	4	2	0

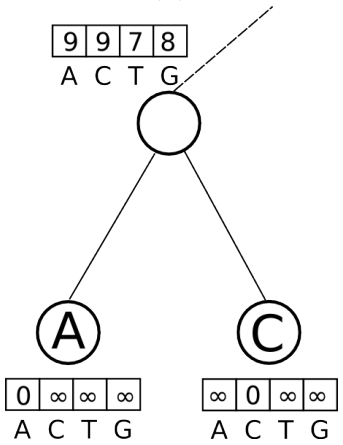
(a)



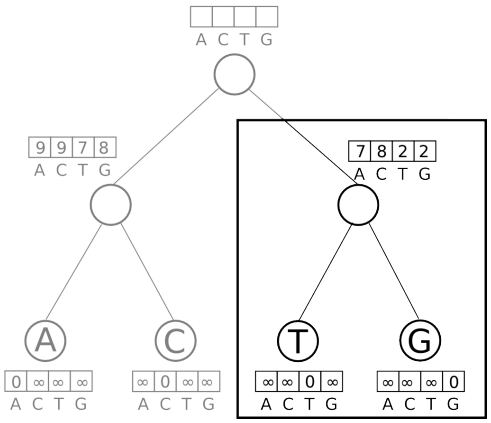
(b)



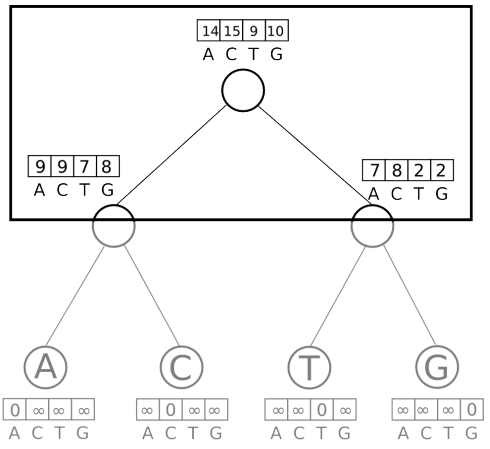
(c)



(d)



(e)



(f)

Figura 8 – Ilustração gráfica do percurso pré-ordem do algoritmo de Sankoff e Rousseau. (a) é a matriz de custos de transição de estados Q. (b) mostra os estados dos caracteres nos nós folha, parâmetro do algoritmo. Qualquer estado diferente do observado possui custo infinito de transição. (c) ilustra que a primeira parte do cálculo será feita no ramo esquerdo da árvore e (d) destaca este ramo, onde para cada estado possível foi calculada a soma do custo de transição para os estados observados nos nós filhos ($Q(A \to A) + Q(A \to C)$, $Q(C \to A) + Q(C \to C)$, e assim sucessivamente). (e) ilustra o cálculo no ramo direito da árvore e por fim (f) finaliza o passo pré-ordem com os custos de transição acumulados calculados para a raiz.

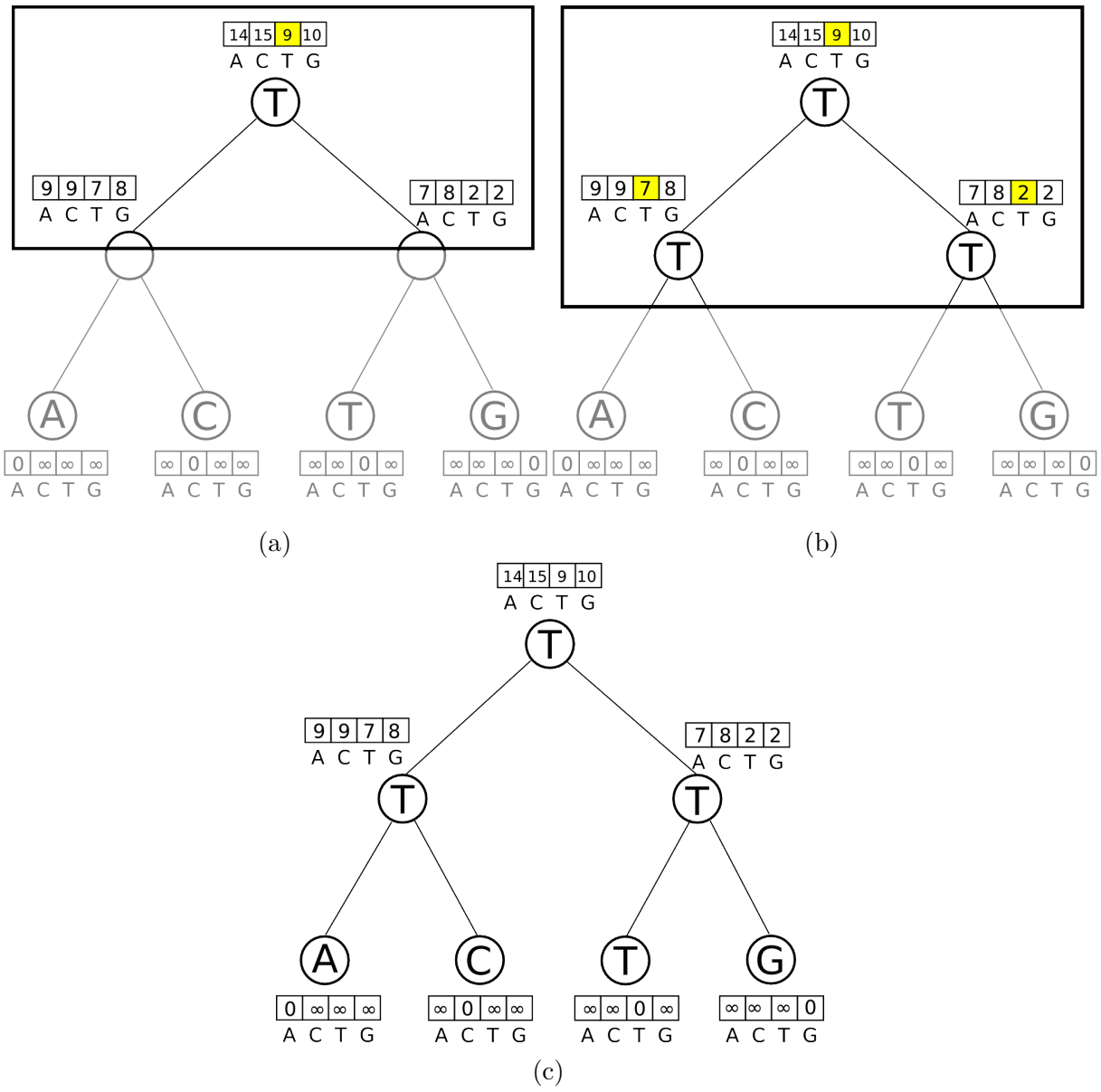


Figura 9 – Ilustração gráfica do percurso pós-ordem do algoritmo de Sankoff e Rousseau. Em (a) a raiz recebe seu estado hipotético de acordo com o menor custo acumulado, calculado ao fim do percurso pré-ordem (Figura 8). Partindo dessa atribuição, os demais ramos recebem também seus valores (b). A reconstrução de caracteres final com todos os estados atribuídos é exibida em (c).

certa combinação de tamanho dos ramos (35)(36)(37), problema este conhecido como "atração de ramos longos" (ou *long branch attraction*, do inglês). Outra crítica ao método é sobre o fato de ser não-paramétrico, ou seja, ele desconsidera distribuições estatísticas e modelos evolutivos, utilizando apenas o critério geral da minimização do número de transições entre estados. A afirmação sobre a incapacidade de considerar um modelo evolutivo é porém controversa, visto que ao estabelecer os custos de transição entre os estados na matriz Q, um conhecimento sobre as características evolutivas dos estados estudados é evidenciada. Outros autores afirmam que a não-parametricidade pode ser positiva ao se considerar evidências favoráveis a evolução não heterogênea, que sugere que a evolução não seguiria portando uma distribuição estatística (38).

2.1.2 Máxima Verossimilhança

Uma vez que um modelo estatístico seja definido e dados sobre os objetos de estudo sejam coletados, deseja-se saber o quão bem os dados representam o modelo. Esta medida é chamada de qualidade do ajuste (do inglês, *goodness of fit*). O método de máxima verossimilhança (MV) estima esta medida, encontrando os parâmetros e valores do modelo evolutivo que melhor descrevem os dados observados (39). Em filogenia, dado um modelo evolutivo m, a máxima verossimilhança se encarrega de reconstruir os estados ancestrais de forma a tornar mais provável o conjunto de estados D, observado nas espécies de interesse. Em outras palavras, a máxima verossimilhança não calcula a probabilidade de uma árvore qualquer ser verdadeira, mas a probabilidade dos dados representarem a realidade sob um determinado modelo, dada uma árvore τ específica, ou $P(D; m|\tau)$.

O exemplo a seguir foi baseado no artigo de Mark Pagel (40) e ilustra a execução do método de máxima verossimilhança para três árvores de topologia definida, apresentando nas folhas duas espécies A e B, que compartilham de um caráter binário com estados $D = \{0, 1\}$. O tamanho dos ramos é o único parâmetro variável entre as árvores e o nó ancestral $n = \{a\}$, de estado indefinido, será estimado pelo método. Estas informações estão dispostas graficamente na Figura 10.

Podemos afirmar que a máxima verossimilhança dos dados observados é proporcional

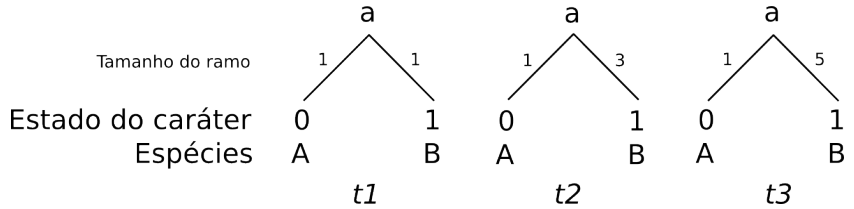


Figura 10 – Três árvores filogenéticas ($\tau 1$, $\tau 2$ e $\tau 3$) com duas espécies e um caráter binário. O valor ao lado dos ramos indica seu comprimento e a é o ancestral hipotético a ser reconstruído.

a probabilidade de sua ocorrência dado o modelo evolutivo (41), ou seja:

$$L(D;m) \propto P(D|m) \tag{2.3}$$

Utilizaremos um modelo m de evolução de caracteres descrito por Pagel (42) que supõe que as transições entre estados podem acontecer em qualquer direção ($1 \rightarrow 0$ e $0 \rightarrow 1$) e representa a probabilidade de transição como função dos parâmetros α e β, onde α é a taxa de transição entre o estado i e j e β representa a taxa de transição na direção oposta, entre j e i. Neste contexto, a probabilidade de transição de estado em um ramo de tamanho t é dado por $P_{ij}(t) = m(\alpha, \beta, t)$.

A equação 2.4 estabelece que a probabilidade de encontrarmos os estados apresentados nas folhas é o produto de duas probabilidades, uma considerando $a = 0$ e outra considerando $a = 1$. As duas probabilidades são ponderadas pela função $w(a)$ que indica a expectativa a *priori* de encontrar os respectivos estados em a, o que pela falta de mais informações é considerada igual para ambos os estados, ou seja, $w(a) = 0.5$.

$$\begin{aligned} L(D;m) \propto P(D|m) &= \sum_{a=0}^{1} w(a)P(D|m;a) \\ &= \sum_{a=0}^{1} w(a)(P_{a0}(t).P_{a1}(t)) \\ &= w(0)(P_{00}(t).P_{01}(t) + w(1)(P_{10}(t).P_{11}(t)) \end{aligned} \tag{2.4}$$

A solução do problema da máxima verossimilhança consiste portanto em encontrar os valores de α e β que maximizem a equação 2.4. Executando o método de Pagel(42) temos que para a árvore com ramos de tamanho igual a 1, $\alpha = \beta = 8.0$ e a equação 2 fornece o valor 0.250, que representa a verossimilhança dos dados, independente do estado hipotético em a. Já para encontrar o estado mais provável em a, faz-se necessário estimar a máxima verossimilhança separadamente para todos os estados possíveis, como demonstrado na equação 2.5, onde $i = 0, 1$ corresponde aos possíveis estados para o nó a.

$$L(m; a = i) \propto w(a = i)P(D|m, a = i) \tag{2.5}$$

Como na máxima verossimilhança os dados são considerados fixos e o que varia é o parâmetro m podemos simplificar $L(D;m)$ para $L(m)$.

A Figura 11 apresenta o resultado da execução do algoritmo de máxima verossimilhança para a desconhecido e, posteriormente, considerando a reconstrução de caráter ancestral baseando-se em seus dois possíveis estados.

Podemos notar que $L(m; a = 0)$ é maior quanto maior for o comprimento do ramo $a \rightarrow B$. Isto ocorre pelo fato de um ramo com maior comprimento possuir mais tempo

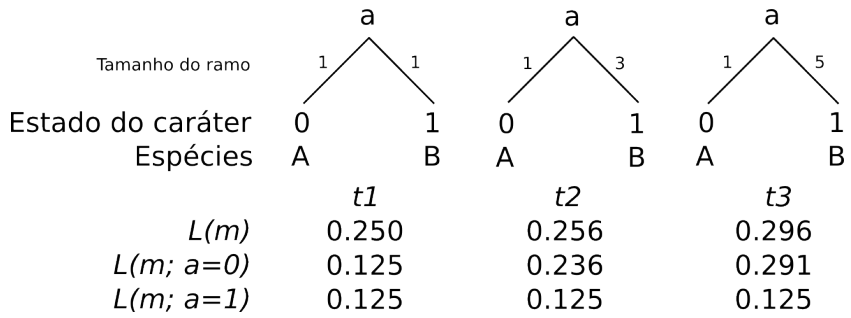


Figura 11 – Árvore filogenética da Figura 10 já com o resultado da máxima verossimilhança considerando os diferentes estados possíveis para a.

para sofrer mutações e, portanto, uma maior probabilidade de ocorrência da transição do estado 0 para 1.

A árvore escolhida pelo método da máxima verossimilhança pode ou não ser a mais parcimoniosa (40). Considerando o modelo m do exemplo apresentado, uma árvore cujos ramos possuem o mesmo tamanho ($\tau 1$, por exemplo) apresentaria o mesmo resultado para ambos os métodos (42).

Por fim, vale constatar que o método da máxima verossimilhança é estatisticamente consistente (43) para modelos evolutivos simples ou quando utilizado na análise o mesmo modelo que deu origem aos dados (44). Esta característica, junto com a possibilidade de anexar informação do tamanho dos ramos na árvore, torna o método preferível para muitos biólogos em comparação ao método da máxima parcimônia.

2.1.3 Inferência Bayesiana

O método de inferência Bayesiana foi o último a ser desenvolvido entre os três aqui apresentados (45)(46). Ele se baseia no Teorema de Bayes, o qual podemos utilizar para traçar um paralelo entre os métodos de máxima verossimilhança e inferência Bayesiana, utilizando a equação a seguir:

$$P(m|D) = \frac{P(D|m)P(D)}{P(m)} \tag{2.6}$$

No método da máxima verossimilhança procuramos maximar $P(D|m)$, considerando $\frac{P(D)}{P(m)}$ uma constante e encontrando como resultado um valor único. Já o método de inferência Bayesiana permite inserir não somente uma crença a *priori* sobre os dados, mas também acomodar na equação novas evidências apresentadas (denominador da função). O resultado da aplicação do método não é um ponto, mas uma função de densidade probabilística, que representa a incerteza sobre os dados.

O método pode ser utilizado para estimar funções de densidade probabilísticas para topologias de árvores (seção 1.3), matriz de transição de estados Q e também

para reconstrução de caracteres ancestrais, o qual recebe como parâmetro idealmente um conjunto de topologias junto com um conjunto de matrizes Q, que representam satisfatoriamente a incerteza inerente ao modelo evolutivo. Vale observar aqui que os métodos anteriores tratavam apenas uma única topologia e matriz de transição.

Diversos programas (47)(6) implementam essas funcionalidades, mas utilizaremos aqui o algoritmo implementado no programa BayesTraits e descrito no artigo (48) para exemplificar o método de inferência Bayesiana.

Como visto no método de máxima parcimônia, a matriz de taxa transição instantânea Q é suficiente para calcular os estados nos nós ancestrais e ela pode ser definida arbitrariamente pelo pesquisador ou estimada através de algum método, como o de máxima verossimilhança. É possível, porém, gerar não somente uma matriz Q, mas um conjunto $Q = \{Q_i\}$ de matrizes, representando sua distribuição de probabilidade e integrando assim a incerteza sobre seu valor real.

Suponha uma árvore $\tau \in T$, então

$$P(Q_i|D,\tau) = \frac{P(D|Q_i)P(Q_i)}{\int_Q P(D|Q)P(Q)dQ}, \tag{2.7}$$

onde $P(D|Q_i)$ é a probabilidade dos dados considerando as matrizes Q_i, $P(Q_i)$ é a probabilidade a *priori* de Q_i e o denominador integra sobre todos os elementos de Q. A equação sugere que a probabilidade de Q_i é uma proporção da probabilidade total, fruto da integração de todos os valores possíveis de seus coeficientes. O número de valores possíveis é intratável, mas podem ser aproximados através de simulações em uma Cadeia de Markov que implemente o modelo evolutivo do caráter observado no conjunto D.

A equação anterior integra possíveis valores de Q sobre uma árvore $\tau \in T$ específica. Porém, é necessário também considerar a incerteza entre as possíveis árvores integrando Q sobre todo o conjunto T, como segue:

$$P(Q_i|D) = \frac{\int_\tau P(D|Q_i,\tau)P(Q_i)P(\tau)d\tau}{\int_\tau \int_Q P(D|Q)P(Q)P(\tau)dQd\tau}. \tag{2.8}$$

Finalmente, em posse do conjunto de árvores T com seus respectivos estados atribuídos nas folhas (conjunto D), mais o conjunto de matrizes de transição Q estimadas de forma a representar a incerteza em T, podemos então gerar a função de densidade de probabilidade relacionada aos ancestrais hipotéticos. Fazemos isso integrando sobre as topologias a probabilidade de observar o conjunto D dado que o nó i adota o estado j, dividindo pela probabilidade de se observar qualquer outro estado diferente de j no nó i,

$$P(S_{ij}|D)_{i\in T} = \frac{\int_T \int_Q P(D|S_{ij},Q,T)P(S_{ij})P(Q)P(T)dQdT}{\int_T \int_Q P(D|Q,T)P(Q)P(T)dQdT}, \tag{2.9}$$

onde S_{ij} indica que j é o estado do nó i e portanto $P(S_{ij}|D)$ é a probabilidade a *posteriori* do estado j ser observado em i.

Alguns autores, como (49), limitam o conjunto T somente às árvores que possuem o nó i, mas como demonstrado em BayesTraits isso leva o método a desconsiderar a incerteza das árvores e superestimar a probabilidade $P(S_{ij}|D)$.

O método de inferência Bayesiana e máxima verossimilhança possuem em comum sua base estatística, que é a função de verossimilhança. Sendo assim, compartilham de características interessantes, como a consistência estatística e a possibilidade de acomodar complexos modelos evolutivos. O método também permite adicionar informação de incerteza na árvore filogenética e matriz Q, proporcionando assim uma forma intuitiva de medir o suporte às evidências.

Apesar da aparente complexidade matemática, o método é computacionalmente eficiente quando implementado com simulações utilizando cadeias de Markov de tempo contínuo (CMTC). Entretanto, CMTCs também são fontes de grande discussão: considerando que a CMTC gera um resultado aproximado, quantas iterações devem ser executadas até que o erro seja mínimo? Uma possibilidade é executar a CMTC até que a variância entre sucessivas árvores geradas seja pequena o suficiente, onde teríamos uma situação de estabilidade. Essa convergência por estabilidade é suscetível ao problema dos máximos locais e não é portando considerada confiável (50). A alternativa é a execução paralela de múltiplas CMTC com inicialização randômica, onde probabilidades semelhante em árvores geradas de forma independente seriam um bom sinal de convergência (51).

Por fim, a definição das probabilidades a *priori* é também um ponto de importante discussão, principalmente pela dificuldade em identificar sua influência nos resultados e possibilidade de torná-los tendenciosos frente a crenças mal formuladas sobre os dados. Alguns autores apontam que a execução do procedimento sobre diferentes probabilidades a *priori* elucidaria essa questão (50) e algumas pesquisas de fato mostram que, em longas execuções da CMTC, presunções errôneas sobre os dados tendem a ser esquecidas (52) frente as evidências apresentadas.

2.2 Métodos de Mapeamento Estocástico

Os métodos de reconstrução de caráter ancestral apresentados na seção 2.1 consideram que as transições de estados ocorrem apenas nos nós internos, onde acontece a cladogênese (separação de linhagens). Tal premissa permite responder diversas questões sobre a história evolutiva de organismos, mas o conhecimento dos padrões de mutações ao longo dos ramos possibilita o levantamento de novas hipóteses sobre as forças envolvidas no processo de seleção natural. O mapeamento estocástico (5) cumpre então esta tarefa, ao considerar que novidades evolutivas surgem por uma transformação gradual da espé-

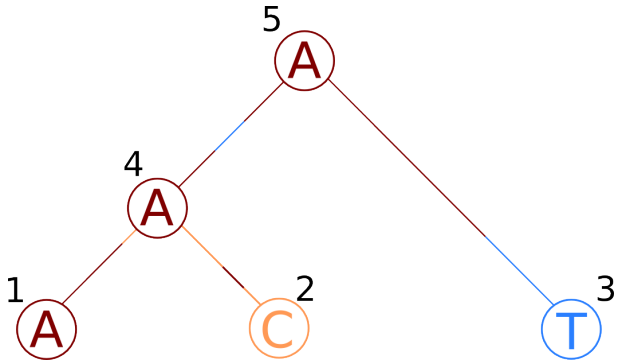


Figura 12 – Árvore filogenética com o mapeamento de estados nos nós e também ao longo dos ramos. Cada cor representa um nucleotídeo, com A em vermelho, C em laranja e T em azul.

cie, permitindo que substituições de estado ocorram não somente nos nós internos, mas também ao longo dos ramos. Esta visão concede um novo nível de inferência, ampliando a compreensão sobre alguns processos evolutivos (53)(26) não tratáveis com o método de reconstrução de caracteres ancestrais tradicional. A Figura 12 exibe um exemplo de filogenia com o mapeamento dos estados ao longo dos ramos.

Dois métodos de mapeamento estocástico serão apresentados. O primeiro, chamado SIMMAP (9), utiliza o método Bayesiano e Cadeias de Markov para estimar o mapeamento dos caracteres. O segundo método, cuja implementação é a proposta deste trabalho, dispensa o uso das Cadeias de Markov e será chamado de SFREEMAP (de ***S****imulation* ***Free Map****ping*).

Quanto ao SIMMAP, seu algoritmo é descrito pelo autor através dos seguintes passos:

1. Calcular a verossimilhança condicional para cada estado em todos os nós da árvore;

2. Simular os estados ancestrais para cada nó interno a partir de amostragem da distribuição a *posteriori* de estados;

3. Simular a história de mutações, amostrando da distribuição a *posteriori* condicionada nas reconstruções do passo 2 e observadas nas folhas da topologia. Os tempos de permanência entre substituições são amostradas de uma distribuição exponencial, com a taxa sendo a diagonal dos elementos do modelo da matriz de transição de estados Q, condicionados ao estado corrente.

O primeiro passo implica no cálculo da verossimilhança condicional, L, para cada estado em cada nó interno da árvore. Para isso precisamos da matriz de probabilidade de transição entre estados $P = P_{ij}(t)$, obtida através da exponenciação da matriz de

transição instantânea Q, e multiplicada pelo comprimento do ramo t, que indica um tempo decorrido, como segue:

$$P = P_{ij}(t) = e^{Qt} \tag{2.10}$$

Com P calculado é possível seguir o exemplo da seção 2.1.2 ou utilizar o algoritmo de poda de Felsenstein (54) para obter a verossimilhança condicional. Em seguida são amostrados os estados ancestrais nos nós internos. Iniciando na raiz da árvore, denotada por σ, o estado d pode ser simulado através da equação 2.11, onde $\Omega = \{A, C, G, T\}$, para o caso de DNA, θ é um vetor com os parâmetros do modelo evolutivo, τ é a topologia e $\pi = \{\pi_1, ..., \pi_m\}$ é a distribuição raiz, ou distribuição estacionária da cadeia de Markov.

$$Pr(d_\sigma = i|D, \tau, \theta) = \frac{L_{\sigma,i}\pi_i}{\sum_{j\in\Omega}, L_{\sigma,j}\pi_j} \tag{2.11}$$

Uma vez amostrados os estados ancestrais na raiz, podemos gerar uma amostragem dos estados em cada nó interno a partir da seguinte equação:

$$Pr(d_{\sigma-1} = j|d_{\sigma-1} = i, D, \tau, \theta) = \frac{l_{\sigma-1,i}P_{ij}(t_{\sigma-1})}{\sum_{k\in\Omega}, l_{\sigma-1,k}P_{ik}(t_{\sigma-1})} \tag{2.12}$$

Neste ponto os nós ancestrais já possuem estados atribuídos. É importante notar que o passo dois não difere substancialmente do algoritmo de reconstrução de caracteres por inferência Bayesiana apresentado na seção 2.1.3, de fato os primeiros passos do mapeamento estocástico usualmente envolvem a reconstrução dos caracteres nos nós internos, através dos métodos já apresentados de Máxima Parcimônia, Inferência Bayesiana ou Máxima Verossimilhança.

O último passo do algoritmo simula a história evolutiva ao longo de cada ramo da árvore, condicionado aos estados observados nas folhas da topologia. Cada ponto gerado na história é simulado através de uma cadeia de Markov de tempo contínuo (CMTC), com as transições modeladas como um processo de Poisson, de foma que os tempos de permanência de um estado em um ramo até que ele sofra mutação é calculado como

$$\lambda e^{\lambda t}, \tag{2.13}$$

com t igual ao comprimento do ramo e $\lambda = -q_{ii}$, valor oriundo da diagonal principal da matriz de transição de estados Q e que pode ser interpretado como a taxa média de saída do estado i, ou probabilidade de ocorrência de uma transição de i para qualquer estado $j \neq i$.

Seja t_1 o tempo de permanência calculado do estado i, caso $t_1 >= t$ e os estados em ambos os nós conectados ao ramo sejam iguais, assumimos que nenhuma mutação

ocorreu e a simulação para o ramo está completa. Caso contrário, recalculamos 2.13 com $\lambda = p_{ij} = \frac{q_{ij}}{-q_{ii}}$ e o novo comprimento de ramo $t = t - t_1$. Verificamos novamente se o novo tempo de permanência excede o comprimento do ramo e o estado final da simulação coincide com o estado do nó descendente. Se afirmativo, a simulação está terminada para o ramo.

Em ambos os casos, quando o tempo de permanência ultrapassa o comprimento do ramo e o estado final é diferente do nó descendente, uma inconsistência foi gerada. Nesta situação, o mapeamento é rejeitado e o processo deve ser completamente refeito a partir do nó ancestral, garantindo assim que os tempos de permanência sejam exponencialmente distribuídos.

O autor aponta que seria possível efetuar o mapeamento sem antes reconstruir os estados nos nós internos (passo 2). Porém, a quantidade de mapeamentos rejeitados seria muito alta e como o processo precisaria sempre ser refeito para toda a árvore, ao invés de apenas para o ramo problemático, esta característica aumentaria substancialmente o tempo computacional envolvido na operação.

Por ser um método de mapeamento baseado em simulações, o SIMMAP exibe as mesmas questões descritas na seção 2.1.3. Em resumo, não há consenso quanto aos parâmetros da simulação que levam a execução a uma boa convergência. Em 2008 Minin e Suchard (3) propõe um método analítico capaz de resolver esses problemas. A definição desse método, cuja implementação nesse trabalho recebeu o nome de SFREEMAP, é detalhada no próximo capítulo.

3 Sfreemap - Mapeamento Estocástico Livre de Simulação

Vladimir Minin e Marc Suchard apresentaram em 2008, no artigo intitulado *"Fast, accurate and simulation-free sthochastic mapping"* (3), um algoritmo que não utiliza cadeias de Markov de tempo contínuo (CMTC) para o mapeamento de trajetórias de estados em filogenias. A relevância para a área consiste em eliminar problemas relacionados a simulações, como erros de aproximação e parâmetros controversos como o número de iterações necessárias para se obter um resultado satisfatório. A solução analítica também promete redução e previsibilidade no tempo de computação. Segundo os autores, a eficácia do algoritmo foi verificada com experimentação mas nenhuma implementação livre foi disponibilizada para a comunidade científica.

O artigo de Minin e Suchard possui como principal referência o artigo de Nielsen (2002), intitulado *"Mapping Mutations on Phylogenies"* (5), onde são introduzidos os conceitos de mapeamento estocástico em filogenias. Nielsen propõe testar hipóteses evolucionárias através da expectância a *priori* $E[H(M_\theta)]$ e a expectância condicional a *posteriori* $E[H(M_\theta)|D]$, onde $M_\theta = \{X_{1t}, ..., X_{B_nt}\}$ é uma coleção de trajetórias sob um caráter X que possui m estados, ao longo dos ramos da árvore τ, de topologia e comprimento dos ramos definidos, $H(M_\theta)$ é uma função que sumariza a coleção M_θ sob algum aspecto e, por fim, D é o conjunto de estados observados nas folhas de τ.

Nielsen recorre a simulação na implementação do método por considerar que o número de mapeamentos possíveis em $E[H(M_\theta)]$ e $E[H(M_\theta)|D]$ é infinito e portanto intratável, mas Minin e Suchard sugerem que estabelecendo um H específico esse problema é eliminado, como demonstrado na equação 3.1, que implica em $E[H(M_\theta)] = Pr(D)$, função de verossimilhança que pode ser calculada de forma analítica (55).

$$H(M_\theta) = \begin{cases} 1, & \text{se } M_\theta \text{ é consistente com } D, \\ 0, & \text{caso contrário.} \end{cases} \tag{3.1}$$

O artigo de Minin e Suchard apresenta então duas funções H passíveis de resolução analítica. H_L representa o número de transições entre estados nos ramos da árvore τ, enquanto H_W representa a "recompensa evolucionária" da árvore τ, que pode ser compreendida como o tempo que uma linhagem apresentou um determinado estado, ou permanência (duração) deste estado no tempo evolutivo. As funções H_L e H_W serão detalhadas a seguir e em seguida o algoritmo 1 exibe todos os passos necessários para o mapeamento completo.

Expectância de transições de estados (H_L).

A função H_L representa a expectância de transições de estados nos ramos da árvore τ no intervalo de tempo $[0, t)$. Segue da teoria estatística da renovação e do processo de contagem por cadeias de Markov(56):

$$E(H_L, t) = \int_0^t e^{Qz} Q_{\mathcal{L}} e^{Q(t-z)} dz, \tag{3.2}$$

onde $\mathcal{L} \subset \{1, .., m^2\}$ é um conjunto de pares de índices que enumeram transições entre estados de interesse, $Q_{\mathcal{L}} = \{q_{ij} \times 1_{\{(i,j)\in\mathcal{L}\}}\}$ é a matriz de transição de estados Q com as transições não presentes no conjunto $\mathcal{L}$ removidas e 1 é o símbolo que define uma função indicadora, que retorna o valor numérico 1 quando a condição indicada é satisfeita e 0 caso contrário.

Para remover a integral e eliminar a necessidade de simulação utilizaremos a decomposição espectral[1] da matriz de transição de estados Q, de forma que $Q = U \times diag(d_1, ..., d_m) \times U^{-1}$, onde $diag(d_1, ..., d_m)$ é uma matriz diagonal com os autovalores de Q, U é uma matriz $m \times m$ cuja coluna i é formada pelo autovetor i e por fim U^{-1} é a inversa de U. Sabemos também pelo processo de Markov que a matriz de probabilidade de transição pelo tempo é dada por (57):

$$P(t) = e^{Qt} = U \times diag(e^{d_1 t}, ..., e^{d_m t}) \times U^{-1} = \sum_{i=1}^{m} S_i e^{d_i t}, \tag{3.3}$$

onde $S_i = U E_i U^{-1}$, E_i é uma matriz $m \times m$ com o valor 1 na posição ii e zero nas demais posições. Utilizando a equação 3.3 podemos então reescrever a equação 3.2 eliminando a integral, obtendo o seguinte (57):

$$E(H_L, t) = \sum_{i=1}^{m} \sum_{j=1}^{m} S_i Q_{\mathcal{L}} S_j I_{ij}(t), \tag{3.4}$$

onde $I_{ij}(t)$ é definido como segue:

$$I_{ij}(t) = \begin{cases} t e^{d_i t} & \text{se } d_i = d_j, \\ \frac{e^{d_i t} - e^{d_j t}}{d_i - d_j} & \text{caso contrário.} \end{cases} \tag{3.5}$$

Uma consideração importante é que o cálculo da equação 3.4 (além das funções que a utilizam, descritas adiante) deve ser executado uma vez para cada transição de interesse descrita no conjunto $\mathcal{L}$.

[1] Transforma a matriz em sua forma canônica, representando-a através de seus autovetores e autovalores.

Recompensa evolucionária - (H_W).

A função H_W indica o tempo de permanência dos estados nos ramos da árvore e é calculada como um processo de renovação com recompensa. Considere $w = (w_1, ..., w_m)$ o conjunto de recompensas[2] associadas com cada estado i que ocorre na árvore τ, então o caráter X é recompensado com a soma dos valores $t_i \times w_i$, onde t_i é o tempo acumulado de permanência do estado i nos ramos da árvore τ. É possível desconsiderar o conceito de recompensa atribuindo o valor 1 para w_i, obtendo dessa forma simplesmente o tempo de permanência dos estados nos ramos.

Similarmente a H_L, é possível remover a integral da resolução da CMTC (equação 3.6), transformando-a em um somatório (equação 3.7).

$$E(H_W, t) = \int_0^t e^{Q_z} diag(w_1, ..., w_m) e^{Q(t-z)} dz \tag{3.6}$$

$$E(H_W, t) = \sum_{i=1}^{m} \sum_{j=1}^{m} S_i diag(w_1, ..., w_m) S_j I_{ij}(t) \tag{3.7}$$

Assim como para o cálculo do número de transições, a equação 3.7 deve ser calculada uma vez para cada valor diferente de zero do conjunto w.

[2] Valores do universo dos reais, atribuídos pelo pesquisador com o intuito de representar a relevância da permanência do caráter no estado i.

3.1 Algoritmo de Mapeamento Estocástico Livre de Simulações.

Os principais passos da implementação são descritos no algoritmo 1 e detalhados em seguida.

Entrada: Topologia de uma árvore filogenética com organismos de interesse e comprimento dos ramos definidos.

1 Mapeamento da função H nos ramos da árvore.
2 Obter a decomposição espectral da matriz de transição de estados Q;
3 Usar a decomposição para calcular a matriz de transição $P_{ij}(t)$ para cada ramo b da árvore τ;
4 Empregando a mesma decomposição espectral, calcular a expectância local de H para cada ramo;
5 Percorrer τ uma primeira vez para calcular a verossimilhança parcial (F_u) e direcional (S_b) para cada nó u e ramo b;
6 Calcular a máxima verossimilhança pós-ordem da árvore como o produto escalar de F_{1i} e a distribuição estacionária π;
7 Percorrer τ uma segunda vez e calcular a verossimilhança pré-ordem G_u para todos os nós u;
8 Aplicar a equação 3.12 para obter a expectância a *priori* para H nos ramos de interesse b^*;
9 Retornar a expectância a *posteriori* de acordo com a equação 3.13;

Algoritmo 1: Mapeamento estocástico de Minin e Suchard

Passo 1 - Decomposição Espectral.

A matriz Q descreve o custo de transição instantânea entre os estados da árvore e pode ser fornecida como parâmetro ao programa ou estimada através de vários métodos.

A função de decomposição espectral retorna os autovetores e autovalores da matriz Q e possui uma implementação nativa[3] na linguagem R. A complexidade da função varia entre $O(n^2 log_n + (nlog_n^2))$ e $O(n^3)$ dependendo da implementação (58). Apesar do alto custo computacional, a função é executada apenas uma vez em uma matriz quadrada de dimensões pequenas, usualmente 2×2 para um caráter morfológico binário, ou 4×4 para nucleotídeos.

Passo 2 - Matriz de probabilidade de transição $P(t_b)$.

A matriz de transição de estados Q indica em termos matemáticos a probabilidade

[3] http://stat.ethz.ch/R-manual/R-patched/library/base/html/espectral.html

instantânea de transição de um estado i para um estado j. Devemos calcular então a probabilidade de transição entre os estados após um tempo t, indicado pelo comprimento dos ramos na árvore. Na equação 3.8 os autovetores e autovalores da decomposição espectral da matriz Q são representados pelo símbolo U e o conjunto $\{d_1, ..., d_m\}$, respectivamente, e utilizados para gerar uma matriz de probabilidade de transição para cada ramo b de comprimento t.

$$P_{ij}(t) = U \times diag(e^{d_1 t}, ..., e^{d_m t}) \times U^{-1} \tag{3.8}$$

Assim, $P_{ij}(t)$ é a probabilidade de transição do estado i para o estado j em um ramo de comprimento t.

Passo 3 - Calcular a expectância utilizando H_L ou H_W.

Seja t_b o comprimento do ramo b, calcular $E(H, t_b)$, onde H é a equação 3.4 (H_L) ou 3.7 (H_W), ambas já descritas anteriormente.

Nesse passo temos a expectância para o número de transições de estados e o tempo de permanência em cada ramos da árvore, mas calculados de forma independente. A evolução em um ramo depende dos ramos ao seu redor e também do estado nos terminais folha. Os próximos passos adicionam essas variáveis no cálculo.

Passo 4 - Verossimilhança parcial e direcional (F_u e S_b).

Seja $F_u = (F_{u1}, ..., F_{um})$ o vetor de verossimilhança parcial do nó u, o elemento F_{ui} representa a probabilidade dos dados observados nas folhas que descendem do nó u se o estado de u for i. Quando u é um nó folha com estados conhecidos a probabilidade F_{ui} é 1 para o estado observado e 0 para os demais estados, ou formalmente, $F_{ui} = 1_{i=D_u}$ (função indicadora).

Para os nós internos a recursão abaixo é executada:

$$F_{ui} = \underbrace{\left[\sum_{j=1}^{m} F_{c(b_1)j} P_{ij}(t_{b_1})\right]}_{S_{b_1 i}} \times \underbrace{\left[\sum_{j=1}^{m} F_{c(b_2)j} P_{ij}(t_{b_2})\right]}_{S_{b_2 i}} \tag{3.9}$$

Na recursão, b_1 e b_2 são os índices dos ramos que descendem do nó u e $c(b_1)$ e $c(b_2)$ são os índices dos nós filhos dos ramos $b1$ e $b2$, respectivamente. A verossimilhança direcional $S_b = (S_{b1}, ..., S_{bm})$ é dada por $S_{b_1 i}$ e $S_{b_2 i}$ e elas representam as probabilidades considerando lado esquerdo e direito da subárvore que tem o nó u como raiz. A Figura 13 exibe uma árvore que ilustra a abrangência da verossimilhança F_{ui} para $u = 3$ junto com os valores de S_{b_1} e S_{b_2} correspondentes.

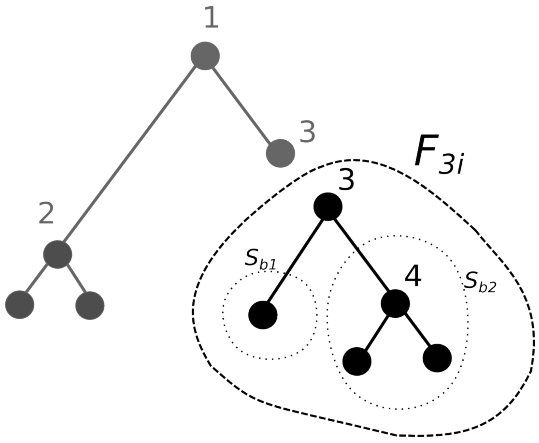


Figura 13 – Ilustração de F_{ui} com $u = 3$.

Passo 5 - Máxima Verossimilhança (Pr(D))

A máxima verossimilhança é proporcional à probabilidade de se observar a distribuição dos estados nos terminais de uma árvore e é calculada como o produto escalar de F_{1i} e π, onde F_{1i} é F_{ui} para o nó raiz (equação 3.9) e $\pi = \{\pi_i, ..., \pi_m\}$ é a distribuição estacionária da cadeia de Markov, a qual identifica a probabilidade a longo prazo (tendendo ao infinito) da ocorrência dos estados no nó raiz, o que a priori se assume como igualmente provável para todos os estados. O pesquisador pode utilizar valores diferentes para a distribuição estacionária caso julgue necessário através de argumentos passados ao programa.

$$Pr(D) = F_{1i} \cdot \pi_i \tag{3.10}$$

Passo 6 - Verossimilhança pré-ordem (G_u)

A verossimilhança pré-ordem $G = (G_{u1}, ..., G_{um})$ é a probabilidade de se observar o estado i no nó u, junto com os outros estados nas folhas da subárvore de τ obtida pela remoção da linhagem abaixo de u. Seu valor pode ser obtido utilizando o vetor S_b (passo 4) e percorrendo uma segunda vez a árvore τ, seguindo a recorrência

$$G_{ui} = S_{c'(p(u))i} G_{p(u)i}, \tag{3.11}$$

onde $p(u)$ é o indice do pai do nó u e $c'(p(u))$ é o segundo nó descendente de $p(u)$, o qual chamaremos de nó "irmão". Supondo que o nó raiz tenha índice 1, iniciamos o cálculo de G com $G_{1i} = \pi_i$, onde π_i é o vetor contendo a distribuição estacionária discutida no passo anterior. Essa inicialização representa uma árvore de um único nó onde a verossimilhança pré-ordem é equivalente às probabilidades atribuídas a priori para os possíveis estados que esse único nó pode assumir (a distribuição estacionária). Uma representação gráfica da abrangência de G_{ui} é exibida na Figura 14, considerando $u = 3$.

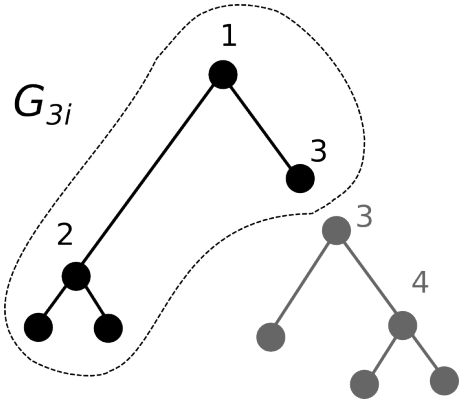


Figura 14 – Ilustração de G_{ui} com $u = 3$.

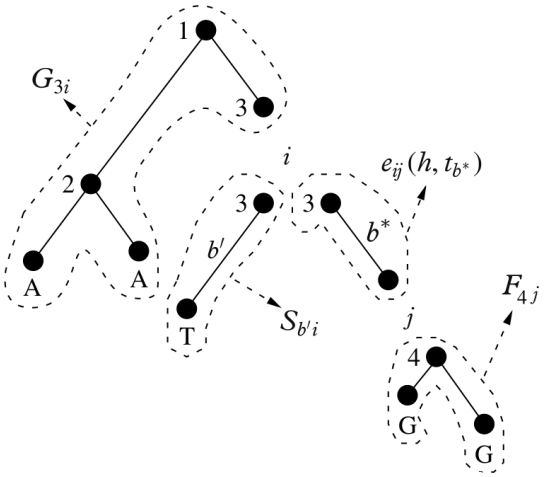


Figura 15 – Representação gráfica da equação 3.12. Imagem retirada do artigo de Minin e Suchard (3).

Passo 7 - Obter a expectância da função H (H_W ou H_L)

Seja b^* um ramo qualquer e b' seu ramo irmão, então

$$E[h(\{X_{b^*t}\})1_D] = \sum_{i=1}^{m}\sum_{j=i}^{m} G_{p(b^*)i} S_{b'i} e_{ij}(H, t_{b^*}) F_{c(b^*)j}, \tag{3.12}$$

onde $e_{ij}(H, t_{b^*})$ é o resultado do passo 3 para H_L ou H_W, considerando a transição de i para j em um ramo de comprimento t. A figura 15 apresenta graficamente todas as variáveis envolvidas na equação acima considerando o ramo $b*$, que conecta os nós 3 e 4.

Nesse momento temos, para cada ramo, a expectância para o número de transições e tempo de permanências em todos os estados do caráter, considerando probabilidades pré-ordem e pós-ordem e os valores de H_L e H_W calculadas anteriormente. O último passo, abaixo, envolve esse cálculo com a verossimilhança da árvore.

Passo 8 - Expectância a *posteriori* de H

Como último passo o algoritmo retorna o valor da equação

$$E[H(M_\theta)|D] = 1/Pr(D) \sum_{b^* \in \tau} E[h(\{X_b^* t\})1_D], \tag{3.13}$$

onde $Pr(D)$ foi definida na equação 3.10 e $E[h(\{X_b^* t\})1_D]$ na equação 3.12. Temos assim o cálculo completo da função H para todos os ramos e estados da árvore τ, consideradas as probabilidades pré e pós-ordem da árvore junto com sua verossimilhança calculada.

3.2 Considerações

Minin e Suchard fazem uma comparação com o método tradicional de mapeamento estocástico apresentando o erro acumulado da Cadeia de Markov de Tempo Contínuo (CMTC) para duas cepas do vírus *HIV*, uma com rápida evolução e outra com evolução lenta[4]. A tabela 1 possui os resultados do experimento, demonstrando que apesar do erro diminuir conforme o número de iterações da CMTC aumenta, a redução não ocorre de forma linear. Também é apontada a falta de estudos relacionados a decisão sobre o número ideal de iterações da simulação, concluindo assim que o método livre de simulação é preferível.

Tabela 1 – Comparativo da redução do erro absoluto em uma CMTC para evolução rápida e lenta.

Iterações CMTC	Erro - Evolução Rápida	Erro - Evolução Lenta
100	0.0598	0.4624
500	0.0255	0.3319
1.000	0.0259	0.2905
10.000	0.0205	0.2809

No artigo, não são apresentados experimentos relacionados ao tempo de processamento ou comparativos com outros algoritmos de mapeamento estocástico. Apenas uma nota é adicionada pelos autores comparando o tempo de execução com o método de máxima verossimilhança. O algoritmo proposto seria 1,5 vezes mais lento, o que foi interpretado como positivo devido ao grau de detalhes superior do mapeamento estocástico.

3.3 Análise de Complexidade

Os dois fatores relevantes no cálculo da complexidade do algoritmo são o número de arestas na árvore (a) e o número de estados do caráter em estudo (n). Considerando que as árvores são sempre enraizadas e dicotômicas (cada nó possui sempre dois descendentes),

[4] Termos relacionados com a velocidade com que se observam mutações nos genes.

o valor de a depende somente da quantidade de nós internos presentes na árvore, que por sua vez é igual a $c - 1$, onde c é número de organismos de interesse (nós folha). Dessa forma, é direto que $a = (c - 1) \times 2$.

Assim sendo, seguindo implementação das equações apresentadas nos passos 2 até passo 8, a complexidade do algoritmo proposto pode ser descrita assintoticamente como $O(a.n^3)$.

4 Implementação

Este capítulo descreve a implementação do método de Minin e Suchard de mapeamento estocástico livre de simulações, o qual chamaremos de SFREEMAP. A seção 4.1 irá descrever o R, linguagem de programação escolhida para a implementação. Em seguida são apresentadas as bibliotecas *Rcpp* e *RcppArmadillo*, utilizadas para otimização do tempo de execução. A seção 4.3 apresenta otimização desenvolvida para geração da matriz Q, processo apenas indiretamente relacionado ao mapeamento, mas que se mostrou extremamente importante. As estruturas de dados utilizadas na implementação são detalhadas na seção 4.4, seguidas da possibilidade de ganho de performance alavancada por técnicas de programação dinâmica, na seção 4.5. Discutimos por fim a importância do uso do Software Livre no desenvolvimento da aplicação e apresentamos, na seção 4.7, informações sobre instalação e uso do SFREEMAP.

4.1 R

R é uma linguagem de alto nível e um ambiente para análise de dados e geração de gráficos (59), características que a tornaram muito popular entre estatísticos e biólogos. A familiaridade dos pesquisadores com a linguagem, unida a vasta disponibilidade de pacotes relacionados a filogenética (60) fizeram do R a escolha para implementação do método descrito neste trabalho. A linguagem R foi desenvolvida por Ross Ihaka e Robert Gentleman na Universidade de Auckland (Nova Zelândia) e atualmente faz parte do projeto GNU[1], onde é mantida por desenvolvedores do mundo todo e licenciada sob a *General Public License*, GPL3 (61).

Uma característica importante do ambiente R é a possibilidade de expandi-lo com a instalação de pacotes através do sistema de criação e gerenciamento de pacotes integrado, em conjunto com CRAN (*The **C**omprehensive **R** **A**rchive **N**etwork*)[2], rede de servidores que armazenam e replicam os pacotes desenvolvidos pela comunidade. A implementação do SFREEMAP fez uso dessa modularidade, importando pacotes básicos de estatística e filogenia, notoriamente o *phytools* (62) e o *ape* (63), sendo também ele próprio disponibilizado como um pacote, que pode então ser instalado e incorporado em outras aplicações.

R é uma linguagem interpretada e multi-paradigma que oferece algumas estruturas de dados como primitivas da linguagem. Entre elas podemos citar os vetores, matrizes, *data frames* e listas. Matrizes são generalizações bi-dimensionais de vetores e podem abrigar

[1] http://www.gnu.org/
[2] http://cran.r-project.org/

somente elementos de um mesmo tipo (como inteiros ou caracteres). *Data frames* são como tabelas em banco de dados relacionais, permitem agregação de objetos de diferentes tipos e possuem funções específicas de manipulação de dados. Listas, por sua vez, permitem reunir elementos não relacionados em um mesmo objeto, cada elemento da lista pode conter qualquer estrutura de dados.

Em R, vetorização é o nome dado a possibilidade de utilizar um mesmo operador ou função sobre objetos de diferentes tipos e estruturas. Um exemplo é a função *sum*, que soma todos os elementos de um vetor ou matriz de forma equivalente, ou o operador $+$, capaz de somar vetores com vetores, vetores com matrizes ou matrizes entre si. Funções vetorizadas em R não apenas simplificam a escrita de código, mas costumam ser computacionalmente mais eficientes que seus equivalentes mais tradicionais com laços do tipo *for* ou *while*. O exemplo abaixo demonstra a diferença de tempo de execução de um trecho de código que multiplica um vetor por uma constante, utilizando laço e também vetorização.

```
> A <- rep(0, 10^6)
> B <- 1:10^6
> x <- 3.14
>
> system.time(for (i in 1:length(A)) A[i] <- B[i] * x)
   usuario    sistema  decorrido
     1.176      0.000      1.175
>
> system.time(A <- B * x)
   usuario    sistema  decorrido
     0.004      0.000      0.003
```

A primeira linha do código acima cria e inicializa com zeros o vetor A de tamanho 10^6, enquanto a segunda linha cria um vetor B e o preenche com uma sequência de inteiros, do número 1 ao número 10^6. É notável que o tempo de execução do método com laço é ordens de grandeza maior quando comparado ao uso do operador de multiplicação nativo do R.

Um outro exemplo é o cálculo de ocorrências de um determinado valor em um vetor. Suponha uma amostragem de 10^6 lançamentos de um dado e posterior contagem do número de ocorrências de cada número, como no código abaixo.

```
> rolls <- sample(1:6, 10^6, replace=T)
>
> count <- rep(0,6)
> system.time(for (i in rolls) count[i] <- count[i] + 1)
   usuario    sistema  decorrido
     1.174      0.000      1.173
```

```
>
> system.time(table(rolls))
   usuario    sistema  decorrido
      0.28       0.00       0.28
```

A grande diferença no tempo de execução é devida principalmente a função *table* ser implementada em C, enquanto o laço é interpretado em tempo de execução pelo R. Essa integração com outras linguagens é um ponto positivo do R, pois permite aproveitar a potencial de linguagens de baixo nível na execução de funções onde o desempenho é um fator crítico, tornando assim viável a utilização do R para aplicações de computação de intensiva.

4.2 Rcpp e RcppArmadillo

A linguagem R possui integração nativa com as linguagens C e C++, mas essa integração demanda uma série de cuidados relacionados a conversão de tipos e gerenciamento de memória que tornam o trabalho excessivamente complexo. Felizmente, existem pacotes que abstraem alguns desses aspectos e integram muito facilmente as duas linguagens. Utilizamos como facilitadora a biblioteca *Rcpp* (64), que além de mapear objetos do R em estruturas de dados adequadas ao C, também permite o uso de funcionalidades mais avançadas do C++, como classes, *templates* e a biblioteca *Standard Template Library* (STL), segundo os autores praticamente sem *overhead* se comparado ao método nativo de integração do R.

Para operações de álgebra linear utilizamos o *RcppArmadillo* (65), pacote que integra ao *Rcpp* a biblioteca C++ *Armadillo* (66), de álgebra linear. A escolha se deu pela facilidade de integração e pelo desempenho, comparável às principais bibliotecas de álgebra linear disponibilizadas livremente (66). Comparativos e considerações sobre ganho de desempenho serão apresentadas no capítulo 5.

4.3 Otimização da Geração da Matriz Q

O método descrito por Minin & Suchard assume que a matriz Q é dada como argumento ao programa, por isso sua geração não foi descrita no artigo. Porém, normalmente o pesquisador não tem conhecimento a priori da matriz Q e deseja estimá-la com base nos dados das árvores sendo analisadas. Para execução desse procedimento, inicialmente foi utilizada a função presente no pacote *phytools*, mesma função utilizada pelo programa SIMMAP, onde a matriz Q é estimada através de processo de otimização, que têm como parâmetros os estados nos nós folha e o comprimento dos ramos.

Usando funções de *profiler* da linguagem R foi possível identificar que esse processo de geração da matriz era responsável por uma porção significativa do tempo total de execução do programa, eclipsando o tempo de mapeamento propriamente dito. Grande parte do tempo de execução era concentrado na função *matexpo*, do pacote *ape* (importado pelo *phytools*), responsável por exponenciar a matriz Q repetidas vezes, para todos os ramos presentes na árvore. O próprio mapeamento estocástico também efetua exponenciação da matriz Q, como descrito nos passos um e dois do método, no capítulo 3. Tentamos então substituir a função *matexpo* pela alternativa com decomposição espectral e o ganho de desempenho foi expressivo. Em seguida, o código para exponenciação também foi reescrito em C com a ajuda do *Rcpp*, melhorando ainda mais o tempo de processamento. Essa modificação pode ser incorporada no SIMMAP e possivelmente em programas similares. Detalhes sobre o o ganho de desempenho atingido podem ser vistos na seção 5.2.

4.4 Estruturas de Dados

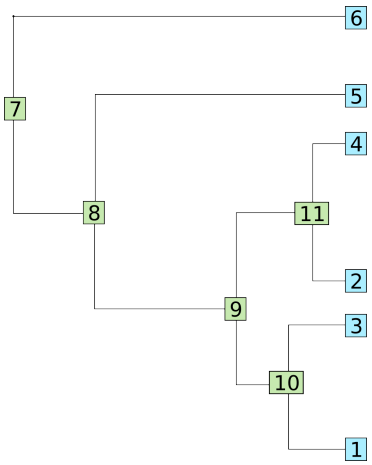
Uma árvore filogenética é representada no programa como um objeto da classe *phylo*, implementada no pacote *ape* e largamente utilizada em outros programas de filogenia. Considerando a o número de arestas na árvore, d o número de nodos, n o número de estados e c o número de organismos de interesse, podemos dizer que essa classe possui os seguintes atributos:

- *edge*: uma matriz $a \times 2$ contendo o nó de origem e destino de cada uma das arestas da árvore. O pacote *ape* permite ordenar essas arestas de forma a facilitar o percurso pré-ordem da árvore. A Figura 16 ilustra essa ordenação;
- *edge.length*: um vetor contendo a elementos que representam o comprimento de todas as arestas da árvore, ordenada de forma equivalente ao atributo *edge*, com o elemento na posição i de *edge.length* correspondendo a aresta representada pela linha i na matriz do atributo *edge*;
- *tip.label*: um vetor contendo as etiquetas dos nós folha da árvore (organismos de interesse), indexada de forma que a etiqueta i esteja associada ao organismo representado pelo nó folha i;
- *Nnode*: um inteiro armazenando a quantidade de nós internos da árvore.

Cada um desses objetos é utilizado em algum passo do algoritmo de mapeamento descrito na seção 3.1. O "Passo 2 - Matriz de probabilidade de transição" utiliza o vetor *edge.length* junto com a matriz resultante da decomposição espectral de Q para calcular a probabilidade de transição entre todos os estados para cada comprimento de aresta. O resultado dessa função é armazenado em uma matriz de dimensões $n \times n \times a$, com os

	Nó	
Aresta	**Origem**	**Destino**
1	10	1
2	10	3
3	11	2
4	11	4
5	9	10
6	9	11
7	8	9
8	8	5
9	7	8
10	7	6

(a)



(b)

Figura 16 – Representação matricial (a) e gráfica (b) da ordenação pré-ordem dos nós de uma árvore filogenética.

eixos x e y representando a transição entre estados e o eixo z a aresta ao qual os valores se referem. O "Passo 3 - Calcular a expectância utilizando HL ou HW" recebe e retorna estrutura de mesmas dimensões, com a expectância calculada para cada comprimento de ramo em *edge.length*.

O "Passo 4 - Verossimilhança parcial e direcional (F_u e S_b)" utiliza a ordenação conveniente do atributo *edge* para executar o percurso pré-ordem. Para cada duas linhas na matriz da Figura 16a temos um nó pai, representado pela primeira coluna de ambas as linhas, e seus dois descendentes diretos, representados na segunda coluna. Percorrendo a matriz dessa forma é garantido que o cálculo dos nós internos só será efetuado após seus descendentes terem sido calculados. Assim sendo, o cálculos para F_u e S_b são executados com uma única leitura da matriz *edge*, garantindo rápido acesso aos dados armazenados de forma contígua na memória. As matrizes resultado, F_u e S_b, possuem dimensões $d \times e$.

No "Passo 5 - Máxima Verossimilhança (Pr(D))", o resultado é um número real obtido pela multiplicação vetorial de F_u, onde u é a raiz da árvore (7, na Figura 16), pelo vetor de probabilidade estacionária, cujo valor é determinado por parâmetros do programa. O passo "Passo 6 - Verossimilhança pré-ordem (G_u)" produz resultado com as mesmas dimensões do passo 4 e necessita de um segundo percurso na matriz *edge*, dessa vez na ordem inversa (pós-ordem), além dos valores de S_b já calculados.

Em seguida, o "Passo 7 - Obter a expectância da função H (H_W ou H_L)" efetua uma série de múltiplicações de matrizes utilizando resultados anteriores e retorna também uma matriz contendo a expectância das funções H em cada aresta da árvore para cada estado, quando calculando H_W, e cada transição entre estado, para H_L. O resultado é então dividido no passo 8 pela verossimilhança, gerando a expectância a posteriori de H, que por fim é adicionado como atributo ao objeto da classe *phylo* que representa a

árvore original, de forma a compor o resultado final do programa. Em outras palavras, o retorno da execução do SFREEMAP é o mesmo conjunto de árvores de entrada, com dois atributos adicionais:

- *mapped.edge*, uma matriz $n \times a$ contendo o tempo de permanência de cada estado em cada aresta da árvore;
- *mapped.edge.lmt*, matriz com um número de colunas igual a combinação de pares de estados possíveis, desconsiderando transições de um estado para ele mesmo, indicando a expectância do número de transições entre cada par, para cada aresta da árvore.

4.5 Programação Dinâmica

Programação Dinâmica (67) corresponde a um paradigma de programação que busca otimizar o tempo de processamento de um programa. Como requisito de sua aplicação está a possibilidade do problema ser dividido em problemas menores que, quando agrupados, correspondem a solução do problema global. Na prática, soluções parciais são armazenadas e reutilizadas quando necessário ao invés de recalculadas. Usualmente, o pesquisador busca efetuar o mapeamento em um conjunto contendo centenas de árvores com os mesmos organismos de interesse. Levantou-se então a possibilidade de reaproveitamento dos cálculos efetuados em uma árvore desse conjunto em árvores semelhantes analisadas posteriormente.

Porém, para que tal aproveitamento seja possível, não basta que dois clados (subárvores) apresentem exatamente os mesmos organismos de interesse, eles precisam possuir também a mesma topologia e comprimentos idênticos para todos os ramos do clado. Em testes empíricos foi possível identificar que tais requisitos raramente são obedecidos, o que invariavelmente torna o processo de identificar clados idênticos em diferentes árvores mais custoso do que refazer os cálculos necessários. Além disso, buscar um reaproveitamento de cálculo iria implicar na necessidade de sincronização e troca de mensagens entre processos, o que poderia prejudicar o desempenho paralelo do programa.

Pelos motivos de baixo ganho potencial e aumento na complexidade das soluções paralelas, a utilização de programação dinâmica para reaproveitamento de cálculo em diferentes árvores foi descartada.

4.6 Software Livre

De acordo com a *Free Software Foundation* (FSF)[3], um programa é Software Livre se garantidas as seguintes liberdades essenciais aos seus usuários[4]:

[3] https://www.fsf.org/pt-br
[4] Tradução livre da versão original em inglês (68)

0. A liberdade de executar o programa como desejar, para qualquer propósito;

1. A liberdade de estudar o funcionamento do programa e adaptá-lo as suas necessidades; Acesso ao código fonte é uma pré-condição para isso;

2. A liberdade de redistribuir cópias de forma que você possa ajuda ao próximo;

3. A liberdade de redistribuir cópias da sua versão adaptada. Fazendo isso toda a comunidade pode ser beneficiar das suas melhorias. Acesso ao código fonte é uma pré-condição para isso.

Apesar dos benefícios relacionados ao compartilhamento de conhecimento proporcionado pela filosofia, tanto para aprendizado quanto para aprimoramento da computação, o Software Livre permaneceu por muito tempo como uma cultura marginal, praticada somente por entusiastas e considerada inviável do ponto de vista comercial. Foi só recentemente, através da notoriedade alcançada pelo sistema operacional Gnu/Linux[5], além de ferramentas importantes como o navegador Firefox[6] e o servidor *web* Apache[7] que o Software Livre saiu do meio acadêmico e ganhou destaque também entre empresas e usuários.

A divulgação da filosofia por instituições como a FSF e a criação de modelos de desenvolvimento eficazes como os propostos no livro *A Catedral e o Bazar* (69), permitiram que hoje exista uma comunidade de desenvolvedores dispostos a dedicar tempo em projetos de interesse pessoal e comercial, encontrando no Software Livre uma alternativa viável as limitações relacionadas a sua contra parte, o chamado Software Proprietário.

Como resultado desse esforço coletivo existem hoje diversos repositórios de projetos e código fonte de pacotes de software, disponíveis para que entusiastas possam aprender sobre seu funcionamento, aperfeiçoá-lo e beneficiar a comunidade com suas modificações. Entre esses repositórios é possível citar o *SourceForge*[8] e o *Github*[9]. Ao publicar um pacote de software em um repositório como o *Github* o desenvolvedor pode, além de divulgar seu trabalho, abrir um canal de comunicação com seus usuários, receber críticas e sugestões, além de eventual contribuição direta com melhorias e correções.

É com esse objetivo, de contribuir ao máximo para com a comunidade e retribuir os esforços de outros ao proporcionar pacotes utilizados no desenvolvimento do SFREEMAP, que este também se encontra disponível publicamente para apreciação de biólogos e cientistas da computação interessados em seu uso e aperfeiçoamento contínuo. Detalhes sobre instalação e acesso a documentação estão dispostos na próxima seção.

[5] http://www.linux.org/
[6] http://www.mozilla.org/
[7] http://apache.org/
[8] https://sourceforge.net/
[9] https://github.com/

4.7 Instalação e Documentação

Todo o código desenvolvido foi disponibilizado como Software Livre sob licença *GPL*. O código fonte é acessível no repositório de código *Github* no endereço <https://github.com/dpasqualin/sfreemap> e pode ser instalado através do pacote *devtools* com os comandos abaixo, que se encarregam de satisfazer, baixar e instalar todas as dependências do SFREEMAP.

```
> install.packages('devtools')
> require(devtools)
> install_github('dpasqualin/sfreemap')
```

A documentação detalhada de todas as funções de mapeamento e análise do SFREEMAP encontram-se no Anexo 1. O Anexo 2 possui exemplos de uso com os respectivos resultados.

4.7.1 Pacote de testes e experimentação

O pacote *sfreemap.tests* foi desenvolvido para auxiliar na reprodução dos experimentos apresentados no capítulo 5 . Ele possui funções de validação e comparação com o algoritmo SIMMAP, além de métodos específicos para verificação de desempenho em diversos cenários. O pacote foi utilizado para geração dos resultados e gráficos apresentados no capítulo 5 e foi disponibilizado livremente no repositório de código *Github*, sob licença *GPL*, no endereço `https://github.com/dpasqualin/sfreemap.tests`.

5 Experimentos

Neste capítulo, a seção 5.1 apresenta experimentos relacionados a eficácia do algoritmo, que tem como objetivo identificar a validade e precisão dos resultados do SFREEMAP. Em seguida, a seção 5.2 detalha experimentos de desempenho de tempo de processamento e comparativos com o SIMMAP.

O código fonte da implementação do SIMMAP sugerida no artigo original (9) não é multiplataforma e não é disponibilizado livremente. Por esse motivo, os testes de desempenho e comparativos de eficácia foram realizados utilizando uma versão livre implementada no pacote *phytools* (62), chamada de *make.simmap*. Por efeitos de simplicidade, no entanto, iremos nos referir a ela como SIMMAP.

5.1 Validação

A validação do método proposto foi realizada com histórias simuladas (70), cujo processo consiste em gerar uma árvore e simular a história de um caráter hipotético, seguindo um modelo de evolução conhecido. O SFREEMAP, agnóstico ao modelo utilizado na simulação, deve então estimar a história evolutiva nessa mesma árvore. Um resultado satisfatório é aquele onde a diferença entre a estimativa e a simulação seja pequena.

Para efeitos comparativos, foi executado o mesmo procedimento de validação no programa SIMMAP. Pela natureza analítica do SFREEMAP a hipótese atribuída ao experimento foi de que o erro na estimativa estaria localizado próximo a média associada a múltiplas execuções do SIMMAP.

5.1.1 Método para Simulação

Para validação do método executamos testes com árvores ultramétricas[1], geradas pelo método *pure-birth*[2] e simuladas pela função *pbtree* do pacote *phytools*. O número de taxa foi fixado em 128 e o tempo evolutivo entre especiações é contínuo e obtido de uma distribuição exponencial. A função *pbtree* retorna uma árvore com a topologia e comprimento dos ramos definidos. Essa árvore é então utilizada na função *sim.history*, também do pacote *phytools*, para simular uma história evolutiva de um caráter binário, indicando o tempo de permanência dos estados em cada ramo da árvore e o estado final nos nós folha. Os parâmetros da função são a árvore gerada pelo *pbtree* e uma matriz Q de

[1] Árvore ultramétrica é aquela cuja distância de qualquer folha até a raiz é sempre a mesma.

[2] Caso especial do modelo *birth-death*, cadeia de Markov com somente dois tipos de transições de estados, *birth* (nascimento) e *death* (morte). No modelo *pure-birth* a taxa de morte é igual a zero.

transição instantânea entre estados, no experimento definida como uma matriz simétrica com taxas de transição iguais e estados nomeados como a e b, como segue:

$$\mathrm{Q} = \begin{array}{c} \\ a \\ b \end{array} \begin{array}{c} \begin{array}{cc} a & b \end{array} \\ \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix} \end{array}$$

.

5.1.2 Método para Estimativa

A verdadeira matriz Q não é conhecida a priori e deve ser definida arbitrariamente pelo pesquisador ou estimada pelo programa, com base na topologia da árvore e comprimento dos ramos. É comum a utilização de várias matrizes Q para se incorporar a incerteza relacionada a estimativa. Portanto, em nossos experimentos foram estimadas cinquenta matrizes através da distribuição de probabilidade a posteriori implementada via *MCMC* (*Monte Carlo Markov Chain*) no programa *phytools*. Em seguida, a árvore de estudo (mesma utilizada na simulação) teve sua história estimada de forma independente para cada matriz Q.

O SIMMAP, como algoritmo baseado em simulação, aceita como parâmetro a frequência de amostragem, que representa a quantidade de iterações (ou gerações) da simulação para cada amostra gerada. Quanto maior essa frequência, mais tempo de execução é necessário e, em teoria, mais precisa é a resposta. Utilizamos o valor 2000 para frequência de amostragem, retirando 25 amostras por execução do programa. Dessa forma, a primeira amostra é retirada após 2000 iterações, a segunda após 4000, e assim sucessivamente até que a última amostra tenha sido extraída após 50000 iterações da simulação.

Os testes foram executados vinte vezes e os resultados são exibidos nos diagramas[3] abaixo, com as caixas ilustrando os resultados do SIMMAP e a linha vermelha representando a expectância, obtida pelo SFREEMAP. As figuras 17 e 18 exibem os valores médios estimados para tempo de permanência nos estados a e b, respectivamente, e a figura 19 exibe o número médio estimado de transições entre estados na árvore analisada.

Observando agora a diferença entre os valores estimados e simulados, a figura 20 exibe o erro acumulado no tempo de permanência nos estados (erro no estado "a" mais erro no estado "b"), enquanto que a figura 21 apresenta o erro acumulado no número de transições entre os estados. Como esperado, o valor obtido pelo SFREEMAP encontra-se

[3] No diagrama de caixas as linhas verticais superior e inferior em cada intervalo representam respectivamente o maior e menor valor obtido para aquele intervalo. A linha destacada no centro do retângulo indica o valor médio e os limites superior e inferior do retângulo representam o desvio padrão. Os círculos pretos acima e abaixo das linhas verticais são considerados *outliers*, valores atípicos com um grande afastamento do restante da série.

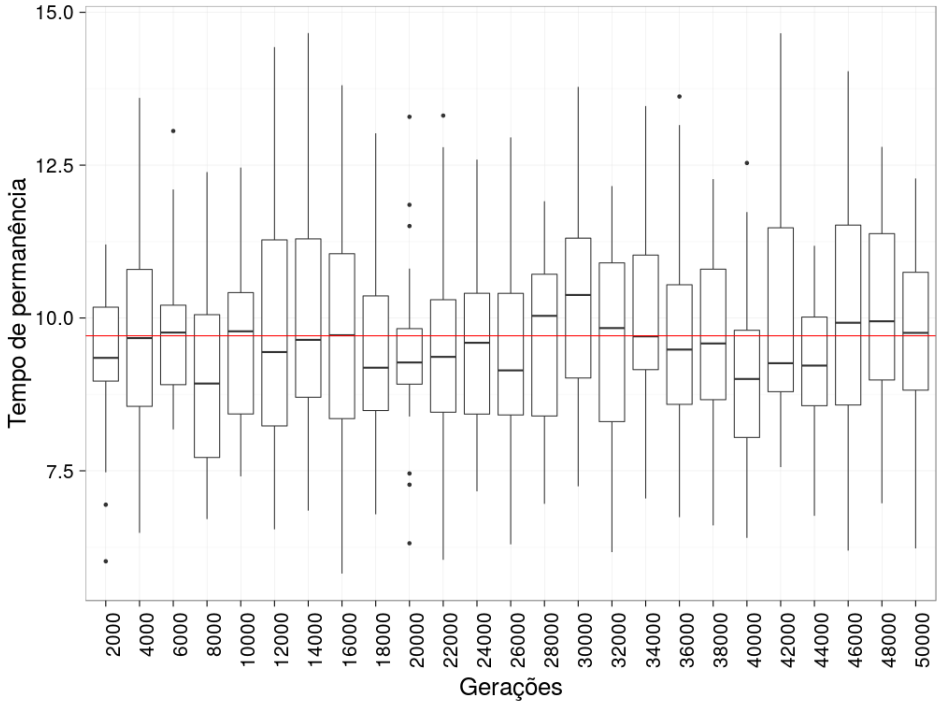


Figura 17 – Diagrama de caixas representando o tempo de permanência no estado "a" para diferentes valores de gerações do SIMMAP. A linha vermelha indica a expectância calculada pelo SFREEMAP.

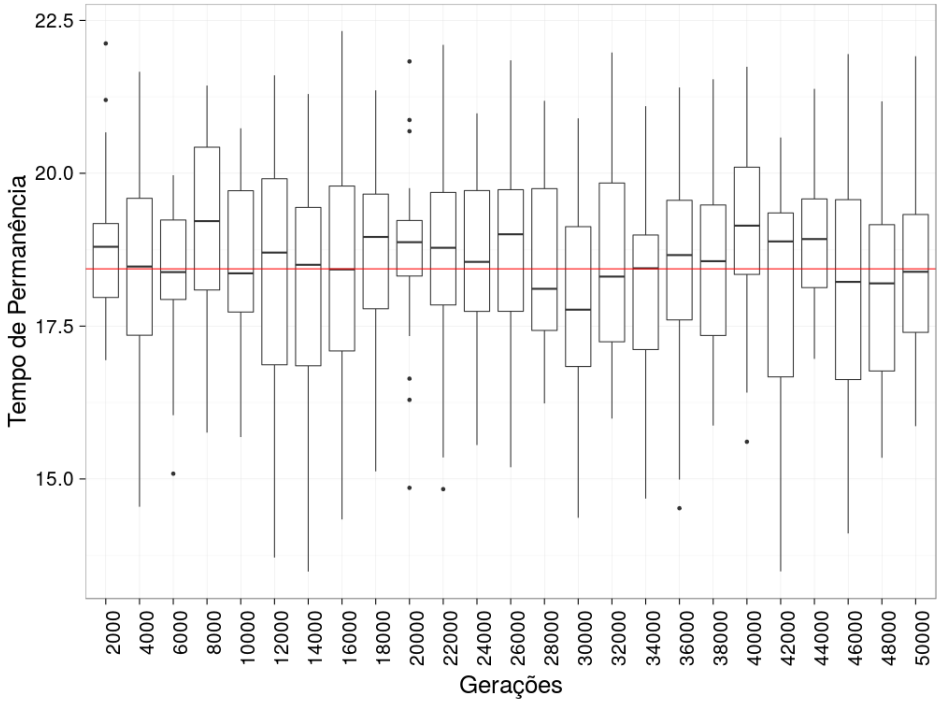


Figura 18 – Diagrama de caixas representando o tempo de permanência no estado "b" para diferentes valores de gerações do SIMMAP. A linha vermelha indica a expectância calculada pelo SFREEMAP.

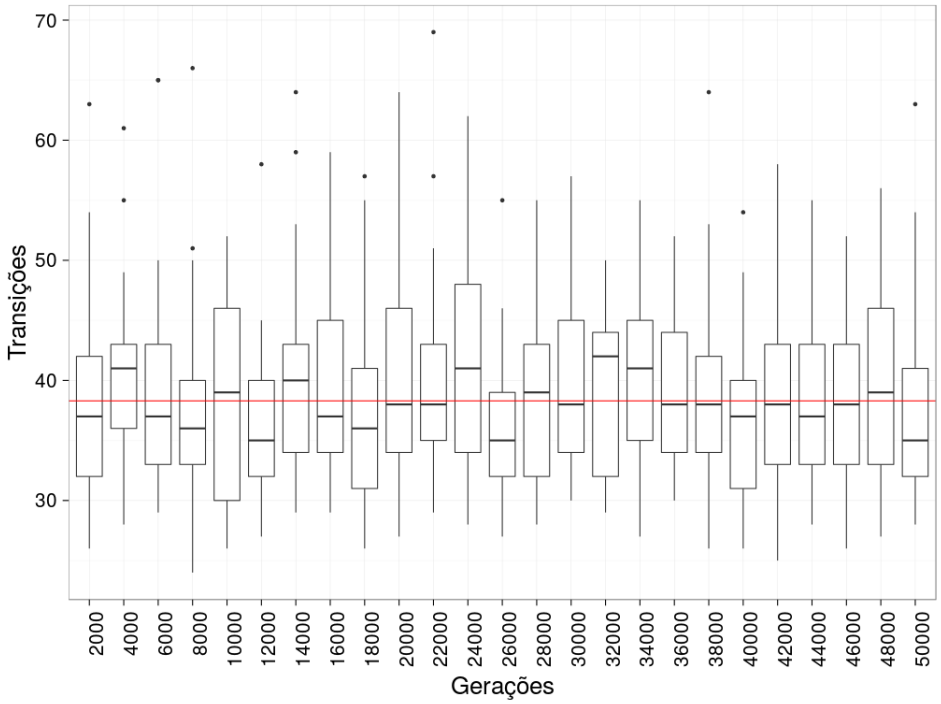


Figura 19 – Número de transições de estados. As caixas indicam valor máximo e mínimo, média e desvio padrão para dez execuções do SIMMAP para cada quantidade de gerações por amostragem (eixo x). A linha vermelha indica o resultado retornado pelo SFREEMAP.

próximo a média das execuções do método baseado em simulações, sugerindo que o método analítico de fato é comparável no que diz respeito a precisão de sua resposta. A seção seguinte analisa e compara novamente os algoritmos, mas em relação ao tempo de execução.

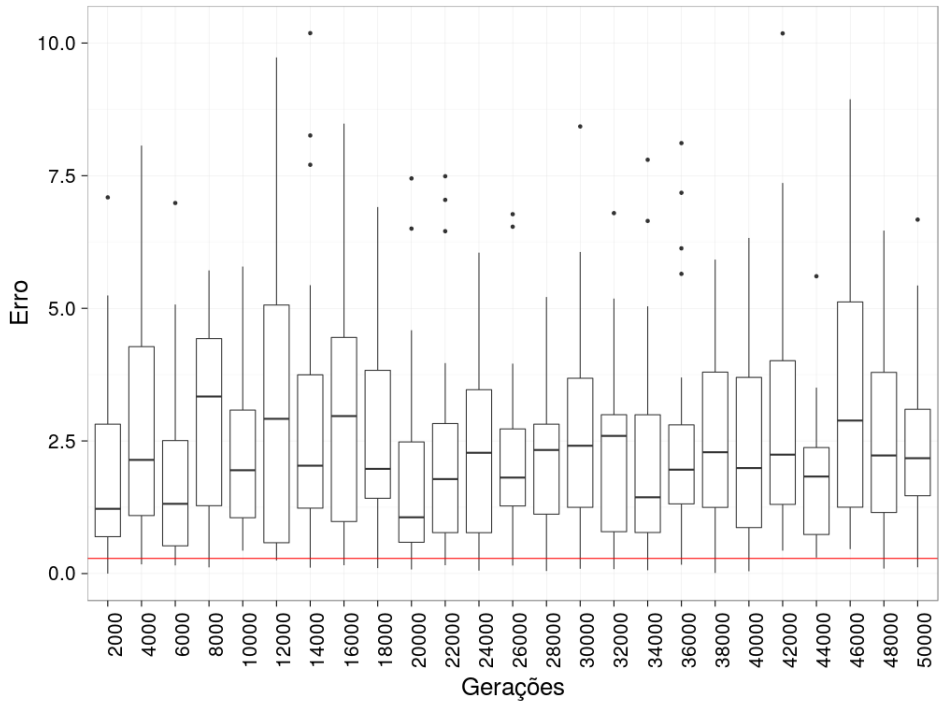


Figura 20 – Erro acumulado do tempo de permanência nos estados do valor estimado em relação ao valor simulado, comparando resultados do SIMMAP (caixas) com SFREEMAP (linha vermelha).

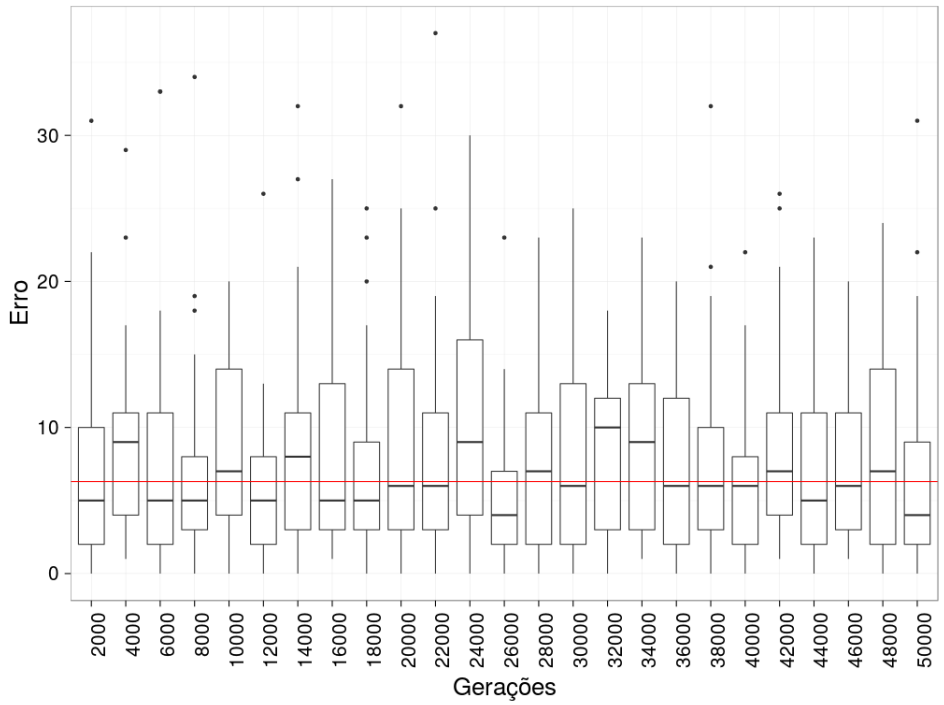


Figura 21 – Erro no número de transições entre estados do valor estimado em relação ao valor simulado, comparando resultados do SIMMAP (caixas) com SFREEMAP (linha vermelha).

5.2 Desempenho

Os testes de desempenho de tempo de processamento foram executados em um sistema operacional Linux, distribuição Debian Jessie com *kernel* 4.0.1, rodando em uma máquina com 128GB de memória RAM e processador de 32 núcleos AMD Opteron 6136, com frequência de 2.4GHz.

A análise considerou variação nos três parâmetros de execução capazes de impactar no tempo de execução do algoritmo: número de taxa presente na árvore, quantidade de árvores e número de estados do caráter avaliado.

As seções seguintes detalham a análise de desempenho considerando os parâmetros acima definidos, na versão em R do programa, que por simplificação será chamada aqui de SFREEMAP-R e na versão com parte do código reescrita em C, a qual será referenciada como SFREEMAP-C. Por fim, ambas serão confrontadas com a implementação livre do SIMMAP, *make.simmap*.

5.2.1 Impacto da Geração da Matriz Q

Na seção 4.3 foi mencionado que o processo para estimar a matriz Q não está diretamente relacionado ao método de mapeamento estocástico aqui apresentado. Porém, ele é necessário para tal e deve ser parte integrante da ferramenta. Esta seção detalha o impacto desse procedimento no tempo de execução do programa e demonstra as melhorias aplicadas que podem ser incorporadas em programas similares. A geração da matriz Q é um processo de otimização e como tal possui variáveis aleatórias envolvidas que causam certas anomalias no tempo de execução. Por esse motivo, todos os gráficos de experimentos são exibidos com curvas de aproximação, que tem como objetivo facilitar a visualização do padrão de crescimento dos tempos de execução. Todos os resultados representam a média de dez execuções do programa.

A Figura 22 exibe o resultado do experimento de análise da geração da matriz Q nos programas SIMMAP, SFREEMAP-R e SFREEMAP-C, em um comparativo de tempo de execução conforme variação no número de estados do caráter em estudo. O SFREEMAP-C foi em média 2.89 vezes mais rápido que o SFREEMAP-R e 8.04 vezes mais rápido que o SIMMAP. Teste semelhante foi feito variando o número de taxa, enquanto número de estados se manteve constante em 4. A Figura 23 exibe os resultados, onde o SFREEMAP-C foi novamente superior, em média 2.1 vezes mais rápido que o SFREEMAP-R e 4.9 vezes mais rápido que o SIMMAP.

A Tabela 2 apresenta um compilado dos tempos, em segundos, para gerar a matriz Q e executar o mapeamento propriamente dito, considerando casos extremos dos experimentos anteriores, com um caráter de 20 estados e 1024 taxa. A Tabela 3, por sua vez, exibe a média da proporção do tempo de processamento ocupado pela geração da matriz Q, em

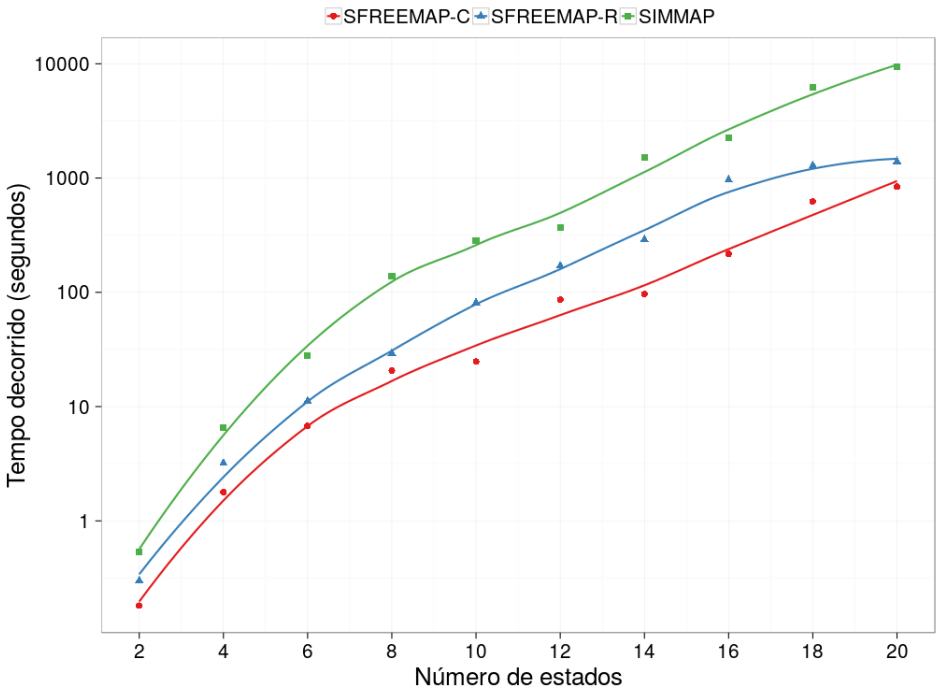


Figura 22 – Comparativo de tempo de execução da função de geração da matriz Q para SFREEMAP-C, SFREEMAP-R e SIMMAP, variando número de estados.

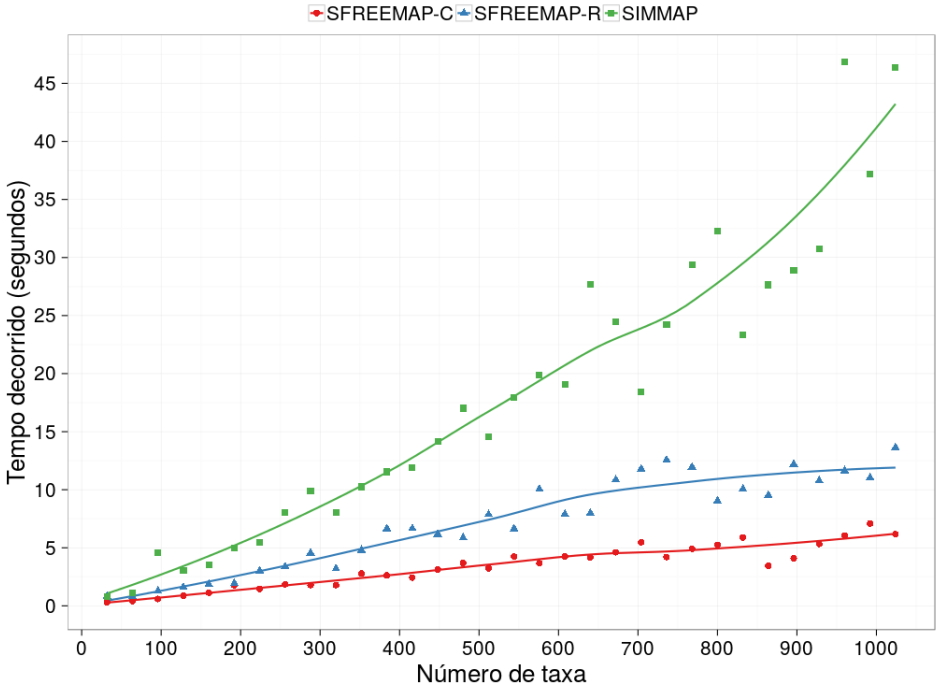


Figura 23 – Comparativo de tempo de execução da função de geração da matriz Q para SFREEMAP-C, SFREEMAP-R e SIMMAP, variando taxa.

Tabela 2 – Comparativo entre SFREEMAP-C, SFREEMAP-R e SIMMAP relacionado ao tempo de geração da matriz Q e mapeamento, em segundos, para dois casos extremos dos experimentos executados.

		SFREEMAP-C	SFREEMAP-R	SIMMAP
1 árvore,	Estimar Q	841	1392	9358
20 estados e	Mapear	471	1931	3
256 taxa	Total	1312	3323	9361
1 árvore,	Estimar Q	7	13	47
4 estados e	Mapear	0.2	9	3
1024 taxa	Total	7.2	22	50

Tabela 3 – Proporção média do tempo de execução utilizado pelo programa para a geração da matriz Q.

	SIMMAP	**SFREEMAP-R**	**SFREEMAP-C**
Estados	93.35	51.14	79.76
Taxa	93.7	59.8	95.77

relação tempo total de execução do programa (geração da matriz mais mapeamento).

As três implementações testadas exibiram uma proporção superior no tempo de geração da matriz em comparação ao mapeamento em si, indicando a grande importância da otimização desse procedimento. Foi possível observar também que as melhorias aplicadas surtiram efeito positivo, com a versão mais otimizada implementada no SFREEMAP-C alcançando desempenho cerca de oito vezes superior a sua equivalente no SIMMAP.

5.2.2 SFREEMAP em R e Otimização com Rcpp

A implementação original do SFREEMAP foi feita integralmente em R, mas diversas limitações da linguagem, apontadas no capítulo 4, sugeriam a necessidade de reescrever partes da aplicação em linguagem de mais baixo nível. Fez-se então uma análise dos pontos de desempenho crítico, que foram então reescritos em C com a ajuda das bibliotecas *Rcpp* e *RcppArmadillo*. A modificação é transparente para o usuário, que continua a executar e interagir com o programa através do interpretador R. É importante destacar que os tempos aqui apresentados representam a execução real do programa, ou seja, consideram o mapeamento propriamente dito mais a geração da matriz Q.

A Figura 24 exibe uma comparação do tempo entre as versões variando o número de árvores analisadas enquanto o número de estados do caráter e taxa se manteve constante em 4 e 256, respectivamente. No caso mais extremo, com 128 árvores analisadas, o tempo total de execução foi de 734 segundos para SFREEMAP-R e 223 segundos para SFREEMAP-C, em média 3.2 vezes mais rápido. É importante observar que o crescimento no tempo de execução é linear em relação ao número de árvores analisadas.

Em experimento seguinte observamos o impacto da variação na quantidade de taxa

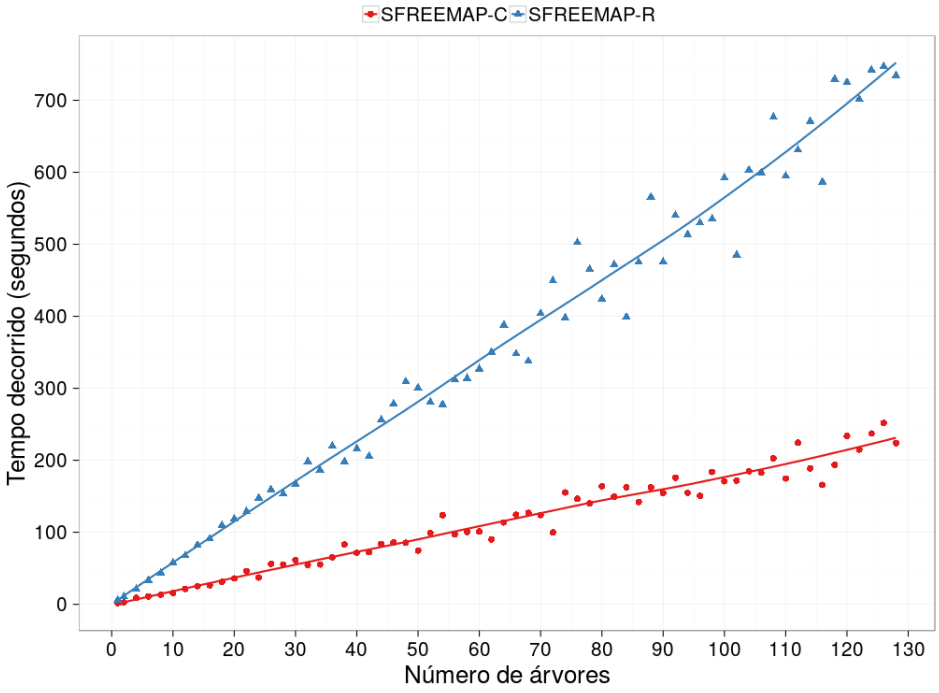


Figura 24 – Comparação entre SFREEMAP-R e SFREEMAP-C, variando o número de árvores e mantendo os demais parâmetros constantes (4 estados e 256 taxa). Os pontos representam os tempos observados e as linhas a respectiva curva de suavização.

no tempo de execução, com uma única árvore e caráter de 4 estados. A figura 25 exibe os resultados desse experimento, que também apresentou crescimento linear no tempo de execução com ganho médio de performance de 3.33 vezes para o SFREEMAP-C.

O algoritmo é composto essencialmente de multiplicações de matrizes, cujas dimensões são iguais ao número de estados do caráter em estudo. Por esse motivo, a quantidade de estados é o fator determinante no tempo de execução do problema. A figura 26 exibe uma curva cúbica de crescimento, padrão característico desse tipo de algoritmo. O SFREEMAP-C nesse caso foi em média 3.42 vezes mais rápido que o SFREEMAP-R. É importante destacar que a análise mais comum se dá com estados binários ou nucleotídeos, que possuem quatro estados apenas. O tempo de execução aumenta consideravelmente para aminoácidos, que possuem vinte estados.

Considerando o desempenho superior apresentado pelo SFREEMAP-C em todos os testes, passaremos a considerá-lo como a versão definitiva do SFREEMAP. As demais seções irão se referir a SFREEMAP como sinônimo de SFREEMAP-C.

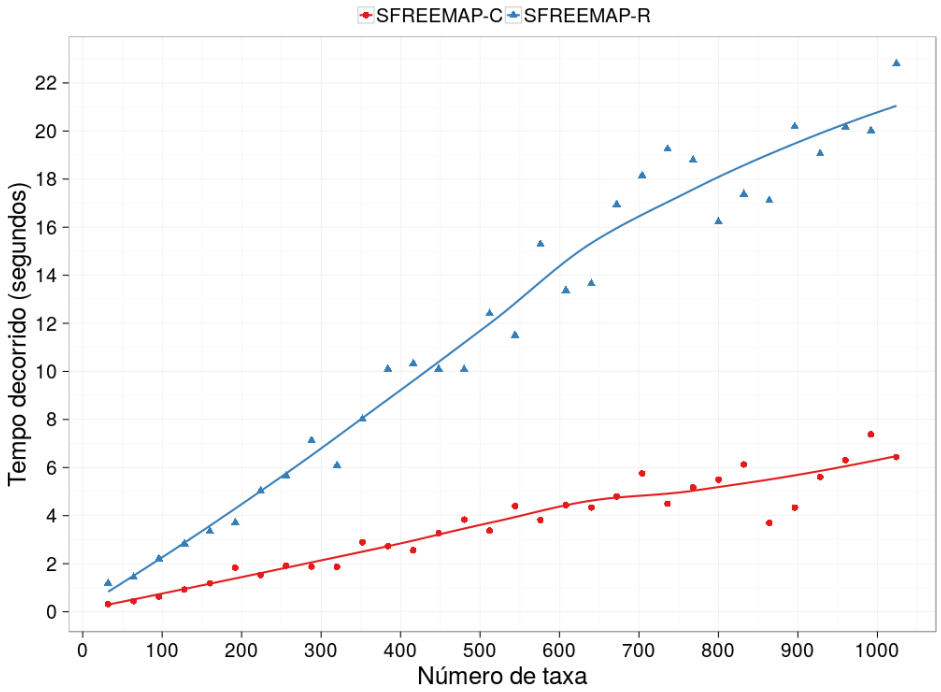


Figura 25 – Experimento com taxa variável, caráter com quatro estados e uma única árvore. O crescimento observado é linear e o ganho é de 3.33 vezes em média para o SFREEMAP-C

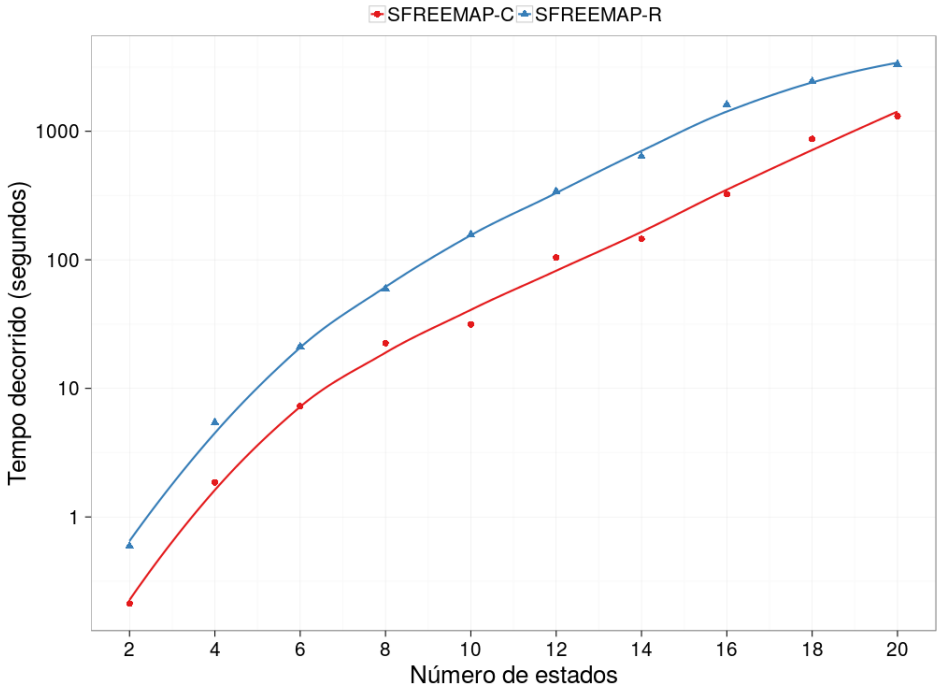


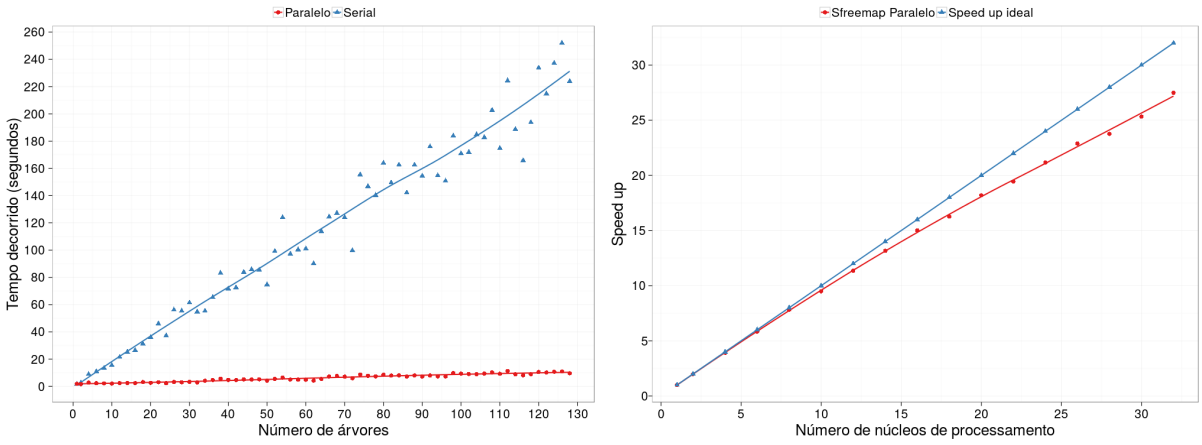
Figura 26 – Experimento com uma árvore, 256 taxa e número de estados variável. O crescimento apresentado é cúbico e o ganho médio de desempenho para o SFREEMAP-C foi de 2.99 vezes.

5.2.3 Paralelismo

O SFREEMAP pode ser executado de forma sequencial, ou com duas formas distintas de paralelismo, uma com otimização baseada na distribuição das árvores entre os núcleos de processamento e outra procurando otimizar o processamento individual de cada árvore.

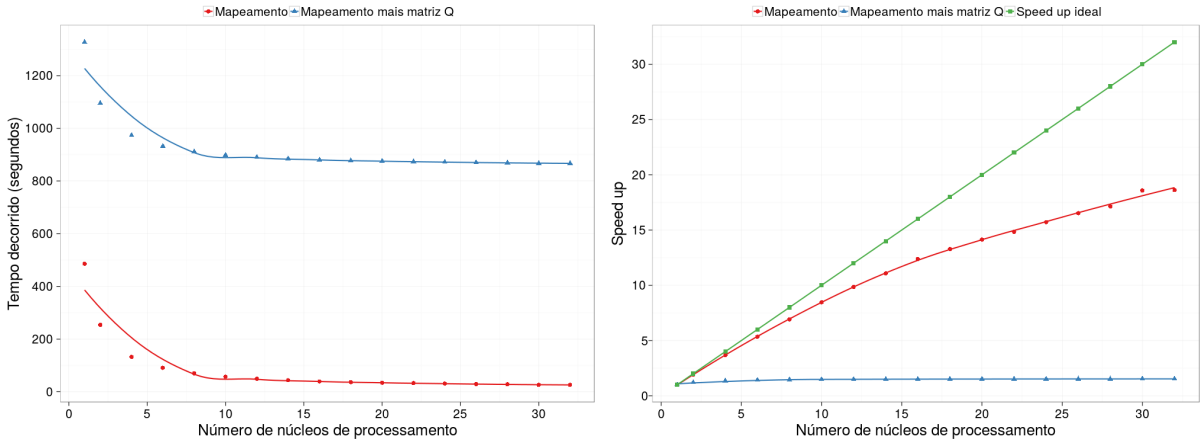
No primeiro modelo de paralelismo, o conjunto total de árvores é dividido igualmente entre os núcleos da máquina. O processamento é independente e cada núcleo executa sua tarefa sem necessidade de troca de mensagens. Além disso, usualmente as árvores analisadas possuem tamanho e forma muito semelhantes, tornando trivial o processo de balanceamento de carga. Esse modelo demonstrou-se especialmente eficiente na análise de uma grande quantidade de árvores, como pode ser visto na Figura 27a, que ilustra experimento com um número crescente de árvores analisadas, assim como o número de núcleos de processamento alocados. Uma vez que todos os 32 núcleos da máquina tenham sido alocados, se forma uma fila de espera para processamento. A Figura 27b exibe uma representação alternativa do ganho de desempenho seguindo a Lei de Amdahl (71), onde o *speed up* ideal define a linha esperada para ganho de desempenho, que ocorre quando o tempo de execução cai pela metade a medida que se dobra o número de núcleos de processamento.

O segundo modelo de paralelismo foi implementado com a biblioteca *OpenMP* (72) e visa acelerar o processamento individual de cada árvore, otimizando mapeamentos com número elevado de espécies e caráteres com muitos estados. Foram adicionadas instruções do *OpenMP* na multiplicação de matrizes do passo 2 do algoritmo e na paralelização do cálculo pelos ramos da árvore nos passos 3 e 7 (seção 3.1). A Figura 28a apresenta os tempos de execução do programa variando o número de núcleos de processamento,



(a) Paralelo Vs Serial (b) Paralelo Vs Speedup ideal

Figura 27 – Comparativo de desempenho entre execução sequencial e paralela, com um número de árvores crescente e número de taxa e estados constante.



(a) (b)

Figura 28 – Teste de desempenho com utilização do *OpenMP*, variando número de núcleos de processamento e considerando tempo para mapeamento e tempo total de execução do programa (mapeamento mais geração da matriz Q).

considerando somente mapeamento e execução completa do programa, enquanto que a Figura 28b exibe a representação alternativa do experimento destacando *speed up* da solução. O experimento utilizou apenas uma árvore com 256 taxa e caráter com 20 estados.

Os resultados apontam ganho considerável no uso do *OpenMP* para o mapeamento da instância testada. Porém, esse ganho torna-se desprezível quando o tempo para geração da matriz Q é considerado, problema esse já evidenciado no estudo desenvolvido na seção 5.2.1. Como foi demonstrado, a função para geração da matriz Q, que não foi paralelizada, é responsável por mais de 80% do tempo de execução total do programa, restando assim menos de 20% do tempo para ser otimizado por paralelismo ou qualquer outra técnica.

Nessa seção foram apresentados dois métodos para paralelismo, um que distribui as árvores entre os núcleos da máquina e outro que paraleliza o processamento individual de cada árvore. Supondo então um cenário futuro, onde o tempo para geração da matriz Q seja menos expressivo, fica a questão sobre qual seria a combinação ideal dos métodos. Com o objetivo de reduzir o tempo total de processamento, dado um conjunto de tamanho arbitrário contendo árvores com qualquer número de taxa e estados, quantos núcleos deveriam ser alocados para o processamento individual via *OpenMP* e em quantos subgrupos deveriam ser divididas as árvores? Atualmente o programa permite a escolha manual desses parâmetros, mas em um cenário ideal eles poderiam ser definidos de forma automatizada.

5.2.4 Comparativo entre SFREEMAP e SIMMAP

O SIMMAP teve o seu desempenho avaliado seguindo a mesma sequência de experimentos, mas por não dispor de paralelismo utilizaremos como base comparativa a versão sequencial do SFREEMAP. Uma diferença fundamental entre as duas aplicações é

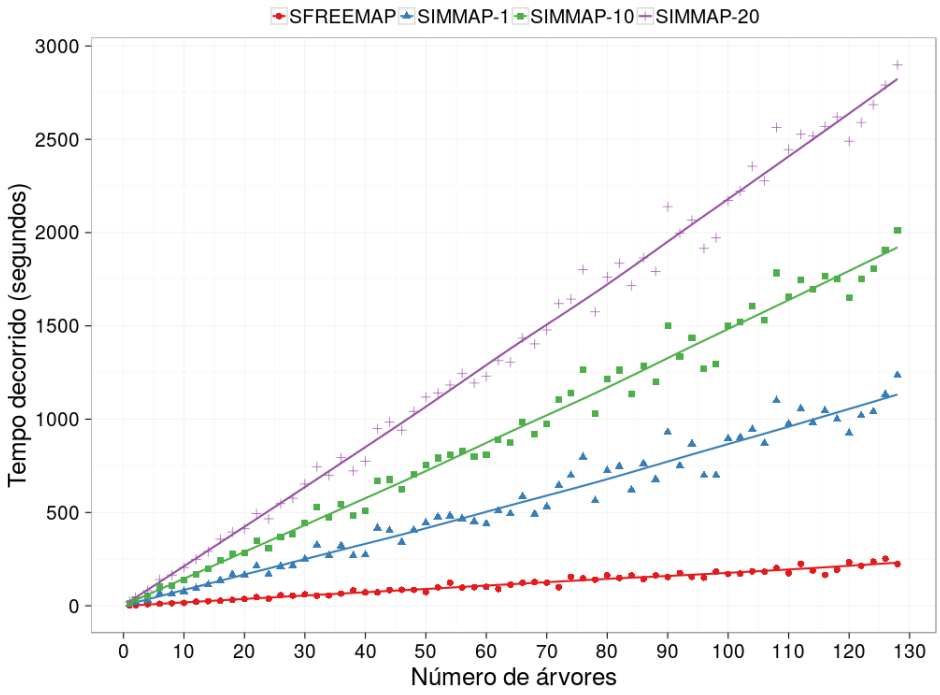


Figura 29 – Comparativo entre SFREEMAP e SIMMAP, com um número de árvores crescente e número de taxa e estados fixados em 256 e 4, respectivamente.

que, enquanto o SFREEMAP retorna a expectância para o tempo de permanência e o número de transições, o SIMMAP retorna por padrão uma única simulação nos ramos. O usuário porém possui a opção de utilizar um parâmetro de execução do SIMMAP que altera seu comportamento para que múltiplas simulações sejam executadas por ramo, permitindo assim o cálculo da média e comparação mais direta com os resultados do SFREEMAP. Sendo assim, foram consideradas no experimento a seguir execuções do SIMMAP para uma única simulação, dez e vinte simulações. Os tempos de cada caso foram contabilizados em segundos e, ao final da seção, a tabela 4 sumariza os resultados.

As figuras 29, 30 e 31 exibem respectivamente o comparativo dos programas quanto a variação no número de árvores, taxa e estados. Este último, fator mais impactante no tempo de processamento do SFREEMAP, também se demonstrou problemático para o SIMMAP. A tabela 4, por sua vez, sumariza o ganho de desempenho do SFREEMAP indicando quantas vezes ele é mais rápido em média quando comparado ao SIMMAP, para os três aspectos apresentados, considerando uma, dez e vinte simulações.

Tabela 4 – Ganho de desempenho do SFREEMAP quando comparado ao SIMMAP executado com diferentes quantidades de simulações.

	SIMMAP-1	**SIMMAP-10**	**SIMMAP-20**
Taxa	4.8	8.2	12.1
Árvores	5.1	8.6	12.3
Estados	6.3	10.8	13.8

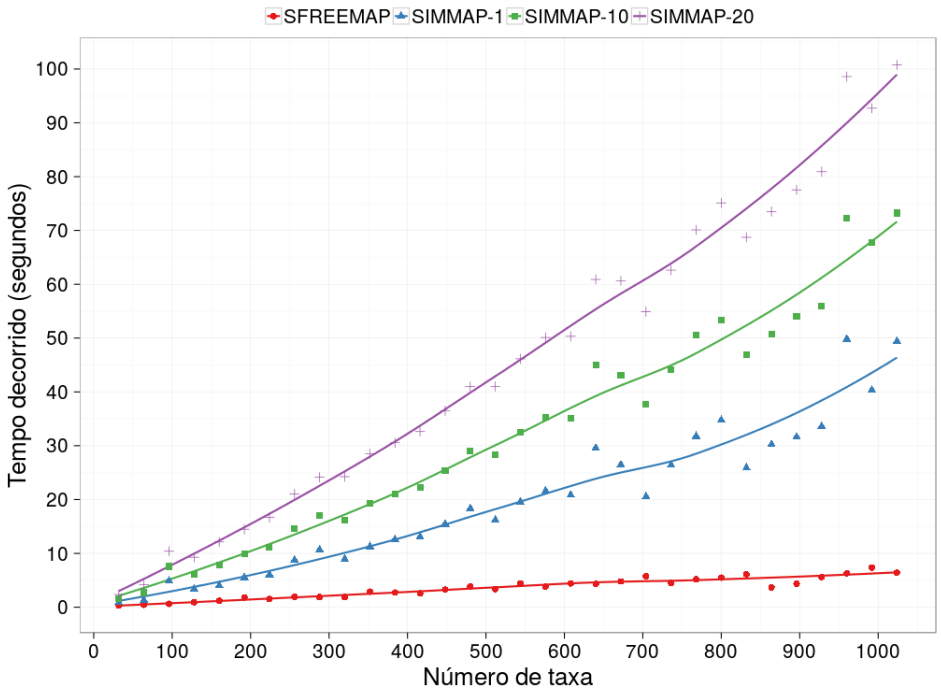


Figura 30 – Comparativo entre SFREEMAP e SIMMAP com uma única árvore, caráter com 4 estados e taxa variável.

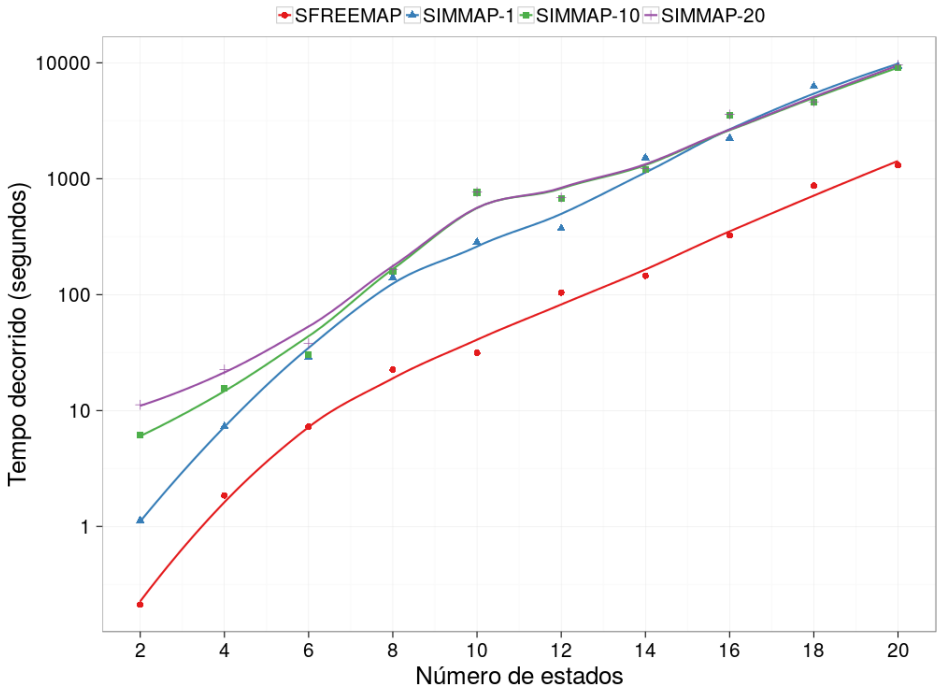


Figura 31 – Comparativo entre SFREEMAP e SIMMAP, número de estados crescente e número de árvores e taxa fixo.

6 Conclusão

A análise filogenética é de importância fundamental na compreensão dos mecanismos da evolução e história das espécies. Este trabalho detalhou processos relacionados a construção das árvores filogenéticas, estimativa das histórias evolutivas e, por fim, demonstrou o potencial de uma ferramenta de mapeamento estocástico livre de simulações, denominada aqui como SFREEMAP. Uma primeira implementação aberta de um método proposto em 2008, que pode agora ser estudada, aprimorada e utilizada por todos que queiram contribuir com sua continuidade.

O SFREEMAP demonstrou eficácia e desempenho quando comparado com ferramenta de objetivo similar, o SIMMAP, com resultados positivos nos dois critérios. O mapeamento do SFREEMAP, que visa retornar o valor esperado para número de transições e tempo de permanência nos estados nos ramos das árvores, realmente correspondente a média obtida através de sucessivas simulações no SIMMAP, confirmando assim a validade do método e precisão nos resultados. Quanto ao desempenho, o SFREEMAP apresentou tempo de execução consideravelmente inferior, por vezes frações do tempo necessário para o SIMMAP executar análise semelhante. O pacote R desenvolvido que contém, além do programa de mapeamento, diversas ferramentas para análise dos resultados, foi disponibilizado sob licença GPL no repositório de código público *Github* e possui vasta documentação, exemplos de uso e testes automatizados, facilitando a divulgação e compartilhamento da ferramenta. Todo o conjunto de testes de eficácia e desempenho utilizado neste documento também foi disponibilizada livremente, permitindo a reprodução dos experimentos de forma simplificada.

Trabalho futuros

Apesar do desempenho satisfatório, ainda há pontos que podem ser melhorados consideravelmente, principalmente quanto a otimização da análise individual de árvores através do *OpenMP* e o procedimento para geração da matriz Q, não diretamente relacionado ao método mas de grande importância para uso prático da ferramenta. Ainda relacionado a desempenho, seria possível automatizar a decisão quanto a quantidade de núcleos atribuídos aos diferentes métodos de paralelismo e também pesquisar sobre formas mais eficientes de executar operações de álgebra linear na implementação do algoritmo.

Os testes de eficácia indicam similaridade no mapeamento executado pelo SFREEMAP em relação ao SIMMAP, algoritmo baseado em simulação já amplamente utilizado pela comunidade. Porém, experimentos mais exaustivos podem ser conduzidos, procurando identificar se essa similaridade se mantém com configurações diversas de parâmetros, como o número de estados do caráter, quantidade de organismos de interesse e modelos evolutivos.

Referências

1 HOLMES, S. Bootstrapping phylogenetic trees: Theory and methods. *Statistical Science*, Institute of Mathematical Statistics, v. 18, n. 2, p. pp. 241–255, 2003. ISSN 08834237. Disponível em: <http://www.jstor.org/stable/3182854>. Citado 3 vezes nas páginas 9, 23 e 24.

2 LANL. *LANL - HIV database*. 2002. Disponível em: <http://hiv-web.lanl.gov/content/hiv-db/>. Citado 2 vezes nas páginas 9 e 23.

3 MININ, V. N.; SUCHARD, M. A. Fast, accurate and simulation-free stochastic mapping. *Philosophical Transactions of the Royal Society B: Biological Sciences*, v. 363, n. 1512, p. 3985–3995, 2008. Disponível em: <http://rstb.royalsocietypublishing.org/content/363/1512/3985.abstract>. Citado 6 vezes nas páginas 10, 16, 26, 40, 41 e 47.

4 FELSENSTEIN, J. The number of evolutionary trees. *Systematic Biology*, Oxford University Press, v. 27, n. 1, p. 27–33, 1978. Citado na página 15.

5 NIELSEN, R. Mapping mutations on phylogenies. *Systematic Biology*, v. 51, n. 5, p. 729–739, 2002. Disponível em: <http://sysbio.oxfordjournals.org/content/51/5/729.abstract>. Citado 4 vezes nas páginas 16, 26, 37 e 41.

6 DRUMMOND, A.; RAMBAUT, A. Beast: Bayesian evolutionary analysis by sampling trees. *BMC Evolutionary Biology*, v. 7, n. 1, p. 214, 2007. ISSN 1471-2148. Disponível em: <http://www.biomedcentral.com/1471-2148/7/214>. Citado 2 vezes nas páginas 16 e 36.

7 LARTILLOT, N.; LEPAGE, T.; BLANQUART, S. Phylobayes 3: a bayesian software package for phylogenetic reconstruction and molecular dating. *Bioinformatics*, v. 25, n. 17, p. 2286–2288, 2009. Disponível em: <http://bioinformatics.oxfordjournals.org/content/25/17/2286.abstract>. Citado na página 16.

8 DUTHEIL, J. et al. Bio++: a set of c++ libraries for sequence analysis, phylogenetics, molecular evolution and population genetics. *BMC Bioinformatics*, v. 7, n. 1, p. 188, 2006. ISSN 1471-2105. Disponível em: <http://www.biomedcentral.com/1471-2105/7/188>. Citado na página 16.

9 BOLLBACK, J. Simmap: Stochastic character mapping of discrete traits on phylogenies. *BMC Bioinformatics*, v. 7, n. 1, p. 88, 2006. ISSN 1471-2105. Disponível em: <http://www.biomedcentral.com/1471-2105/7/88>. Citado 4 vezes nas páginas 16, 26, 38 e 59.

10 METROPOLIS, N. et al. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, v. 21, n. 6, 1953. Citado na página 16.

11 R Development Core Team. *R: A Language and Environment for Statistical Computing*. Vienna, Austria, 2008. ISBN 3-900051-07-0. Disponível em: <http://www.R-project.org>. Citado na página 16.

12 MONTROLL, E. *Fluctuation phenomena*. [S.l.]: Elsevier, 2012. v. 7. 13 p. Citado na página 17.

13 FOLTZ, B. *Finite Math: Long-run Markov Chain Probabilities*. 2012. <https://www.youtube.com/watch?v=IYaOMor9qvE>. Citado na página 17.

14 LEVY, R. Probabilistic models in the study of language. *Ms, University of California, San Diego*, 2010. Citado na página 21.

15 FRONT Matter. In: THE Jackknife, the Bootstrap and Other Resampling Plans. [S.l.: s.n.], 1979. cap. 0, p. i–xi. Citado na página 22.

16 WHELAN, S.; LIÒ, P.; GOLDMAN, N. Molecular phylogenetics: state-of-the-art methods for looking into the past. *TRENDS in Genetics*, Elsevier, v. 17, n. 5, p. 262–272, 2001. Citado na página 22.

17 FELSENSTEIN, J. Confidence limits on phylogenies: an approach using the bootstrap. *Evolution*, JSTOR, p. 783–791, 1985. Citado 3 vezes nas páginas 22, 23 e 24.

18 EFRON, B.; HALLORAN, E.; HOLMES, S. Bootstrap confidence levels for phylogenetic trees. *Proceedings of the National Academy of Sciences*, National Acad Sciences, v. 93, n. 23, p. 13429–13429, 1996. Citado na página 23.

19 BERRY, V.; GASCUEL, O. On the interpretation of bootstrap trees: appropriate threshold of clade selection and induced gain. *Molecular Biology and Evolution*, Chicago, Ill.: University of Chicago Press, c1983-, v. 13, n. 7, p. 999–1011, 1996. Citado na página 23.

20 SOLTIS, P. S.; SOLTIS, D. E. et al. Applying the bootstrap in phylogeny reconstruction. *Statistical Science*, Institute of Mathematical Statistics, v. 18, n. 2, p. 256–267, 2003. Citado na página 24.

21 FELSENSTEIN, J. Statistical inference of phylogenies. *Journal of the Royal Statistical Society. Series A (General)*, Wiley for the Royal Statistical Society, v. 146, n. 3, p. pp. 246–272, 1983. ISSN 00359238. Disponível em: <http://www.jstor.org/stable/2981654>. Citado na página 24.

22 HAECKEL, E. *Generelle Morphologie der Organismen — Allgemeine Grundzüge der organischen Formen-Wissenschaft, mechanisch begründet durch die von Charles Darwin reformierte Descendenz-Theorie*. [S.l.: s.n.], 1866. Citado na página 24.

23 HENNIG, W.; DAVIS, D. D.; ZANGERL, R. *Phylogenetic systematics*. [S.l.]: University of Illinois Press, 1999. Citado na página 25.

24 OWEN, R.; COOPER, W. W. *Lectures on the comparative anatomy and physiology of the invertebrate animals : delivered at the Royal College of Surgeons, in 1843 / By Richard Owen ; From notes taken by William White Cooper and revised by Professor Owen.* London :Longman, Brown, Green, and Longmans„ 1843. 406 p. Http://www.biodiversitylibrary.org/bibliography/6788. Disponível em: <http://www.biodiversitylibrary.org/item/29826>. Citado na página 25.

25 YIP, K. Y. et al. An integrated system for studying residue coevolution in proteins. *Bioinformatics*, v. 24, n. 2, p. 290–292, 2008. Disponível em: <http://bioinformatics.oxfordjournals.org/content/24/2/290.abstract>. Citado na página 26.

26 HUELSENBECK, J. P.; NIELSEN, R.; BOLLBACK, J. P. Stochastic mapping of morphological characters. *Systematic Biology*, v. 52, n. 2, p. 131–158, 2003. Disponível em: <http://sysbio.oxfordjournals.org/content/52/2/131.abstract>. Citado 2 vezes nas páginas 26 e 38.

27 RANNALA, B.; YANG, Z. Probability distribution of molecular evolutionary trees: A new method of phylogenetic inference. *Journal of Molecular Evolution*, Springer-Verlag, v. 43, n. 3, p. 304–311, 1996. ISSN 0022-2844. Disponível em: <http://dx.doi.org/10.1007/BF02338839>. Citado na página 27.

28 CUMMINGS, M. P. et al. Comparing bootstrap and posterior probability values in the four-taxon case. *Systematic Biology*, Oxford University Press, v. 52, n. 4, p. 477–487, 2003. Citado na página 27.

29 HASTINGS, W. K. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, v. 57, n. 1, p. 97–109, 1970. Disponível em: <http://biomet.oxfordjournals.org/content/57/1/97.abstract>. Citado na página 27.

30 NEINHUIS, C. et al. Phylogeny of aristolochiaceae based on parsimony, likelihood, and bayesian analyses of trnl-trnf sequences. *Plant Systematics and Evolution*, Springer-Verlag, v. 250, n. 1-2, p. 7–26, 2005. ISSN 0378-2697. Disponível em: <http://dx.doi.org/10.1007/s00606-004-0217-0>. Citado na página 29.

31 LEACHé, A. D.; REEDER, T. W. Molecular systematics of the eastern fence lizard (sceloporus undulatus): A comparison of parsimony, likelihood, and bayesian approaches. *Systematic Biology*, v. 51, n. 1, p. 44–68, 2002. Disponível em: <http://sysbio.oxfordjournals.org/content/51/1/44.abstract>. Citado na página 29.

32 SANKOFF, D.; ROUSSEAU, P. Locating the vertices of a steiner tree in an arbitrary metric space. *Mathematical Programming*, Springer-Verlag, v. 9, n. 1, p. 240–246, 1975. ISSN 0025-5610. Disponível em: <http://dx.doi.org/10.1007/BF01681346>. Citado na página 29.

33 EYRE-WALKER, A. Problems with parsimony in sequences of biased base composition. *Journal of Molecular Evolution*, Springer-Verlag, v. 47, n. 6, p. 686–690, 1998. ISSN 0022-2844. Disponível em: <http://dx.doi.org/10.1007/PL00006427>. Citado na página 30.

34 GRANT, T.; KLUGE, A. G. Perspective: Parsimony, explanatory power, and dynamic homology testing. *Systematics and Biodiversity*, v. 7, n. 4, p. 357–363, 2009. Disponível em: <http://dx.doi.org/10.1017/S147720000999017X>. Citado na página 30.

35 FELSENSTEIN, J. Cases in which parsimony or compatibility methods will be positively misleading. *Systematic Biology*, v. 27, n. 4, p. 401–410, 1978. Disponível em: <http://sysbio.oxfordjournals.org/content/27/4/401.abstract>. Citado na página 33.

36 FELSENSTEIN, J. Maximum likelihood and minimum-steps methods for estimating evolutionary trees from data on discrete characters. *Systematic Biology*, v. 22, n. 3, p. 240–249, 1973. Disponível em: <http://sysbio.oxfordjournals.org/content/22/3/240.abstract>. Citado na página 33.

37 KIM, J. General inconsistency conditions for maximum parsimony: Effects of branch lengths and increasing numbers of taxa. *Systematic Biology*, v. 45, n. 3, p. 363–374, 1996. Disponível em: <http://sysbio.oxfordjournals.org/content/45/3/363.abstract>. Citado na página 33.

38 KOLACZKOWSKI, B.; THORNTON, J. W. Performance of maximum parsimony and likelihood phylogenetics when evolution is heterogeneous. *Nature*, v. 431, n. 7011, p. 980–984, Oct 2004. ISSN 0028-0836. Disponível em: <http://dx.doi.org/10.1038/nature02917>. Citado na página 33.

39 MYUNG, I. J. Tutorial on maximum likelihood estimation. *Journal of Mathematical Psychology*, v. 47, n. 1, p. 90 – 100, 2003. ISSN 0022-2496. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0022249602000287>. Citado na página 33.

40 PAGEL, M. The maximum likelihood approach to reconstructing ancestral character states of discrete characters on phylogenies. *Systematic Biology*, v. 48, n. 3, p. 612–622, 1999. Disponível em: <http://sysbio.oxfordjournals.org/content/48/3/612.short>. Citado 2 vezes nas páginas 33 e 35.

41 EDWARDS, A. *Likelihood, 1972*. [S.l.]: Cambridge University Press, Cambridge. Citado na página 34.

42 PAGEL, M. Detecting correlated evolution on phylogenies: A general method for the comparative analysis of discrete characters. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, v. 255, n. 1342, p. 37–45, 1994. Disponível em: <http://rspb.royalsocietypublishing.org/content/255/1342/37.abstract>. Citado 2 vezes nas páginas 34 e 35.

43 CHANG, J. T. Full reconstruction of markov models on evolutionary trees: Identifiability and consistency. *Mathematical Biosciences*, v. 137, n. 1, p. 51 – 73, 1996. ISSN 0025-5564. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0025556496000752>. Citado na página 35.

44 STEEL, M.; PENNY, D. Parsimony, likelihood, and the role of models in molecular phylogenetics. *Molecular Biology and Evolution*, v. 17, n. 6, p. 839–850, 2000. Disponível em: <http://mbe.oxfordjournals.org/content/17/6/839.abstract>. Citado na página 35.

45 MAU, B.; NEWTON, M. A.; LARGET, B. Bayesian phylogenetic inference via markov chain monte carlo methods. *Biometrics*, Blackwell Publishing Ltd, v. 55, n. 1, p. 1–12, 1999. ISSN 1541-0420. Disponível em: <http://dx.doi.org/10.1111/j.0006-341X.1999.00001.x>. Citado na página 35.

46 LI, S.; PEARL, D. K.; DOSS, H. Phylogenetic tree construction using markov chain monte carlo. *Journal of the American Statistical Association*, v. 95, n. 450, p. 493–508, 2000. Disponível em: <http://amstat.tandfonline.com/doi/abs/10.1080/01621459.2000.10474227>. Citado na página 35.

47 RONQUIST, F.; HUELSENBECK, J. P. Mrbayes 3: Bayesian phylogenetic inference under mixed models. *Bioinformatics*, v. 19, n. 12, p. 1572–1574, 2003. Disponível em: <http://bioinformatics.oxfordjournals.org/content/19/12/1572.abstract>. Citado na página 36.

48 PAGEL, M.; MEADE, A.; BARKER, D. Bayesian estimation of ancestral character states on phylogenies. *Systematic Biology*, v. 53, n. 5, p. 673–684, 2004. Disponível em: <http://sysbio.oxfordjournals.org/content/53/5/673.abstract>. Citado na página 36.

49 HUELSENBECK, J. P.; BOLLBACK, J. P. Empirical and hierarchical bayesian estimation of ancestral states. *Systematic Biology*, v. 50, n. 3, p. 351–366, 2001. Disponível em: <http://sysbio.oxfordjournals.org/content/50/3/351.abstract>. Citado na página 37.

50 GILKS, W. R.; RICHARDSON, S.; SPIEGELHALTER, D. J. Introducing markov chain monte carlo. *Markov chain Monte Carlo in practice*, London: Chapman and Hall, v. 1, p. 19, 1996. Citado na página 37.

51 GEYER, C. J. Markov chain monte carlo maximum likelihood. Citado na página 37.

52 HUELSENBECK, J. P. et al. Potential applications and pitfalls of bayesian inference of phylogeny. *Systematic Biology*, v. 51, n. 5, p. 673–688, 2002. Disponível em: <http://sysbio.oxfordjournals.org/content/51/5/673.abstract>. Citado na página 37.

53 BISHOP, J.; DEAN, A.; MITCHELL-OLDS, T. Rapid evolution in plant chitinases: molecular targets of selection in plant-pathogen coevolution. *Proceedings of the National Academy of Sciences*, National Acad Sciences, v. 97, n. 10, p. 5322–5327, 2000. Citado na página 38.

54 FELSENSTEIN, J. Evolutionary trees from dna sequences: a maximum likelihood approach. *Journal of molecular evolution*, Springer, v. 17, n. 6, p. 368–376, 1981. Citado na página 39.

55 FELSENSTEIN, J.; FELENSTEIN, J. *Inferring phylogenies*. [S.l.]: Sinauer Associates Sunderland, 2004. v. 2. Citado na página 41.

56 BALL, F.; MILNE, R. K. Simple derivations of properties of counting processes associated with markov renewal processes. *Journal of Applied Probability*, JSTOR, p. 1031–1043, 2005. Citado na página 42.

57 MININ, V. N.; SUCHARD, M. A. Counting labeled transitions in continuous-time markov models of evolution. *Journal of mathematical biology*, Springer, v. 56, n. 3, p. 391–412, 2008. Citado na página 42.

58 PAN, V. Y.; CHEN, Z. Q. The complexity of the matrix eigenproblem. In: *Proceedings of the Thirty-first Annual ACM Symposium on Theory of Computing*. New York, NY, USA: ACM, 1999. (STOC '99), p. 507–516. ISBN 1-58113-067-8. Disponível em: <http://doi.acm.org/10.1145/301250.301389>. Citado na página 44.

59 CRAWLEY, M. J. The R book. In: . [S.l.]: John Wiley & Sons, 2012. p. vii. ISBN 978-0-470-97392-9. Citado na página 51.

60 O'MEARA, B. *Phylogenetics, Especially Comparative Methods*. 2015. <http://cran.r-project.org/web/views/Phylogenetics.html>. Citado na página 51.

61 FREE SOFTWARE FOUNDATION. *GNU General Public License*. Disponível em: <http://www.gnu.org/licenses/gpl.html>. Citado na página 51.

62 REVELL, L. J. phytools: an r package for phylogenetic comparative biology (and other things). *Methods in Ecology and Evolution*, Blackwell Publishing Ltd, v. 3, n. 2, p. 217–223, 2012. ISSN 2041-210X. Disponível em: <http://dx.doi.org/10.1111/j.2041-210X.2011.00169.x>. Citado 2 vezes nas páginas 51 e 59.

63 PARADIS, E.; CLAUDE, J.; STRIMMER, K. APE: analyses of phylogenetics and evolution in R language. *Bioinformatics*, v. 20, p. 289–290, 2004. Citado na página 51.

64 EDDELBUETTEL, D. et al. Rcpp: Seamless r and c++ integration. *Journal of Statistical Software*, v. 40, n. 8, p. 1–18, 2011. Citado na página 53.

65 EDDELBUETTEL, D.; SANDERSON, C. Rcpparmadillo: Accelerating r with high-performance c++ linear algebra. *Computational Statistics and Data Analysis*, v. 71, p. 1054–1063, March 2014. Disponível em: <http://dx.doi.org/10.1016/j.csda.2013.02.005>. Citado na página 53.

66 SANDERSON, C. Armadillo: An open source c++ linear algebra library for fast prototyping and computationally intensive experiments. NICTA, 2010. Citado na página 53.

67 VAZIRANI, S. *Algoritmos*. McGraw Hill Brasil, 2009. 157 p. ISBN 9788563308535. Disponível em: <https://books.google.com.br/books?id=LEXL_WaXoHMC>. Citado na página 56.

68 STALLMAN, R. *What is free software?* 2015. <http://www.gnu.org/philosophy/free-sw.html.en>. Citado na página 56.

69 RAYMOND, E. The cathedral and the bazaar. *Knowledge, Technology & Policy*, Springer, v. 12, n. 3, p. 23–49, 1999. Citado na página 57.

70 PARADIS, E. *Analysis of Phylogenetics and Evolution with R*. [S.l.]: Springer Science & Business Media, 2011. 313-330 p. Citado na página 59.

71 AMDAHL, G. M. Validity of the single processor approach to achieving large scale computing capabilities, reprinted from the afips conference proceedings, vol. 30 (atlantic city, nj, apr. 18–20), afips press, reston, va., 1967, pp. 483–485, when dr. amdahl was at international business machines corporation, sunnyvale, california. *Solid-State Circuits Society Newsletter, IEEE*, IEEE, v. 12, n. 3, p. 19–20, 2007. Citado na página 69.

72 OpenMP Architecture Review Board. *OpenMP Application Program Interface Version 3.1*. 2008. Disponível em: <http://www.openmp.org/mp-documents/OpenMP3.1.pdf>. Citado na página 69.

Apêndices

APÊNDICE A – Manual de uso da aplicação

O manual de uso da aplicação segue o padrão do R para desenvolvimento de documentação e acompanha a distribuição do pacote. O objetivo é enumerar e descrever as funções e *datasets*[1] presentes, citando argumentos para execução, formato do retorno e funções relacionadas, além de referências para mais informações e exemplos de uso.

[1] Conjuntos de dados disponibilizados junto com o pacote que são normalmente utilizados para testes e exemplos.

Package 'sfreemap'

January 11, 2016

Type Package

Title Simulation Free Stochastic Mapping

Version 1.0

Author Diego Pasqualin

Depends R (>= 3.1.0)

Imports Rcpp (>= 0.11.6), ape (>= 3.0-10), phytools (>= 0.3-93), reshape2(>= 1.4.1), ggplot2(>= 1.0.1), parallel (>= 3.2.1), phangorn (>= 1.99.14), seqinr (>= 3.1-3)

Suggests testthat (>= 0.10.0), knitr (>= 1.11)

Maintainer Diego Pasqualin <dpasqualin@c3sl.ufpr.br>

Description More about what it does (maybe more than one line).

License GPL (>= 2)

LazyData true

LinkingTo Rcpp, RcppArmadillo

VignetteBuilder knitr

NeedsCompilation yes

R topics documented:

`sfreemap-package` *Simulation Free Stochastic Mapping*

Description

More about what it does (maybe more than one line).

Details

The DESCRIPTION file:

Package:	sfreemap
Type:	Package
Title:	Simulation Free Stochastic Mapping
Version:	1.0
Author:	Diego Pasqualin
Depends:	R (>= 3.1.0)
Imports:	Rcpp (>= 0.11.6), ape (>= 3.0-10), phytools (>= 0.3-93), reshape2(>= 1.4.1), ggplot2(>= 1.0.1), parallel (>
Suggests:	testthat (>= 0.10.0), knitr (>= 1.11)
Maintainer:	Diego Pasqualin <dpasqualin@c3sl.ufpr.br>
Description:	More about what it does (maybe more than one line).
License:	GPL (>= 2)
LazyData:	true
LinkingTo:	Rcpp, RcppArmadillo
VignetteBuilder:	knitr

Index of help topics:

```
correlation             Creates an object that can be used to plot a
                        correlation matrix
describe.sfreemap       describe a  phylo  object modified by sfreemap
map_posterior_distribution
                        Analyses a set of trees regarding to a specific
                        tree and output an object that can be used by
                        many other functions of this package
plot.correlation        Plot a correlation matrix
plot_distribution_chart
                        Plot the distribution of states in a given node
plot_distribution_tree
```

```
                        Plot the dwelling times distribution of a state
                        in a given tree
pruning                 Prune a phylogetic tree
read_tips               Read the tip states and labels from a file
reorder.sfreemap        Reorder a  phylo  object modified by sfreemap
rescale                 Rescale branches of a  phylo  or  multiPhylo
                        object to a given height
sfreemap                Simulation free stochastic character mapping on
                        a phylogenetic tree
sfreemap-package        Simulation Free Stochastic Mapping
sfreemap.corals.tips    Tip data for sfreemap.corals.trees
sfreemap.corals.trees   Partial sequences of the 12S and 28S rRNA for
                        108 species of scleractinian corals and one sea
                        anemone
sfreemap.primates.dna.tips
                        Tip data for sfreemap.primates.trees
sfreemap.primates.summary.tree
                        Taxonomic sampling of Old World monkeys
sfreemap.primates.trees
                        Taxonomic sampling of Old World monkeys
```

Further information is available in the following vignettes:

```
sfreemap    sfreemap (source, pdf)
```

This package can be used to perform a stochastic mapping in one or more phylogenetic trees, providing as result the expected number of transitions of states and the dwelling times for each branch of the tree.

Afterwards it is possible to analyse the results through charts and plots from tools provided in this package.

The method implemented is based on an article written by Vladimir Minin and Marc Suchard, describing a way to perform analytical stochastic mapping, without requiring to simulations as most methods do. As a result of this, the execution of most functions is really fast.

Author(s)

Diego Pasqualin

Maintainer: Diego Pasqualin <dpasqualin@c3sl.ufpr.br>

References

Vladimir N Minin e Marc A Suchard. Fast, accurate and simulation-free stochastic mapping. Philosophical Transactions of the Royal Society B: Biological Sciences, 363(1512):3985-3995, 2008.

`correlation` *Creates an object that can be used to plot a correlation matrix*

Description

Creates an object that can be used to plot a correlation matrix using plot.correlation.

Usage

```
correlation(map, state, name)
```

Arguments

`map`	A `sfreemap` object, result of sfreemap execution;
`state`	The character state that will be used to calculate The correlation;
`name`	A string to uniquely identify this mapping.

Details

Correlation matrix is useful to check if different methods implies on similar results. To if you run sfreemap with different parameters you can easily check the correlation of the outcomes by summing up objects created with this function and plottig it using plot.correlation.

This function returns an object that can handle the plus operator. See examples.

Value

An object of class `correlation`

Author(s)

Diego Pasqualin <dpasqualin@inf.ufpr.br>

See Also

sfreemap, plot.correlation

Examples

```
r1 <- sfreemap(sfreemap.corals.trees[[1]], sfreemap.corals.tips, parallel=FALSE)
r2 <- sfreemap(sfreemap.corals.trees[[1]], sfreemap.corals.tips, method= mcmc ,
                  n_simulation=1, parallel=FALSE)

cor <- correlation(r1,  colonial ,  empirical ) + correlation(r2,  colonial ,  mcmc )
plot(cor)
```

describe.sfreemap *describe a* phylo *object modified by sfreemap*

Description

This function returns the expected number of transitions and dwelling times of for the states of the tree passed as argument.

If tree is a multiPhylo the result will be the mean value for the trees.

Usage

```
describe.sfreemap(tree, ...)
```

Arguments

tree	A result of a sfreemap execution, can be of types phylo or multiPhylo
...	Other arguments

Value

A named list with values transitions and dwelling_times for each state.

Author(s)

Diego Pasqualin <dpasqualin@inf.ufpr.br>

See Also

sfreemap

map_posterior_distribution *Analyses a set of trees regarding to a specific tree and output an object that can be used by many other functions of this package*

Description

This function will compared the trees against the base_tree and return an object that can be used with plot_distribution_chart to plot the posterior distribution for dwelling times and number of transistions on a particular node and plot_distribution_tree, which plots the posterior distribution in the entire tree.

Usage

```
map_posterior_distribution(base_tree, trees, scale.branches=TRUE
                                    , scale.trees=FALSE, parallel=TRUE)
```

Arguments

base_tree	A single tree, result of the function sfreemap;
trees	A multiPhylo object, result of sfreemap execution
scale.branches	Whether the function should scale the dwelling times to the branch length, resulting in a proportion of the time instead of an absolute value (the expected number of transitions is not scaled).
scale.trees	A value representing the maximum height to scale the tree, or FALSE if the trees should not be scaled.
parallel	Whether the function should run in parallel (defaults to TRUE)

Value

A named list with three items:

- emr for expected markov reward (the dwelling times for states) and lmt for the labelled markov transitions (the expected number of state transitions).

  Each criteria (emr and lmt) is composed by an array three dimentions, like this: $emr[trees, states, nodes]$.

  This structure represent the corresponding value for each node and state on each tree in trees when compared o base_tree. When a node from base_tree has no match, the corresponding state values will be set to NA.

- base_tree the original base_tree given as argument. This is useful for the functions that use the result of this function, cited in the description.

Author(s)

Diego Pasqualin <dpasqualin@inf.ufpr.br>

See Also

sfreemap

Examples

```
sm <- sfreemap(sfreemap.corals.trees, sfreemap.corals.tips, parallel=FALSE)
map <- map_posterior_distribution(sm[[1]], sm, parallel=FALSE)
p <- plot_distribution_chart(map, 160)
print(p)
```

`plot.correlation` *Plot a correlation matrix*

Description

Plot a correlation matrix.

Usage

```
## S3 method for class  correlation
plot(x, y=NULL, ...)
```

Arguments

`x`	A result of correlation function;
`y`	Not used;
`...`	Not used.

Details

Plot `obj`, result of correlation.

Author(s)

Diego Pasqualin <dpasqualin@inf.ufpr.br>

See Also

correlation

`plot_distribution_chart`
Plot the distribution of states in a given node

Description

This function plots the posterior distribution for a state (or all states) in particular node, highlighting the confidence interval desired calculated using the highest posterior density (HPD).

Usage

```
plot_distribution_chart(map, nodes=NULL, trees=NULL, states=NULL, conf_level=95
                        , number_of_ticks=20, type= emr )
```

Arguments

map	A result of a map_posterior_distribution execution;
nodes	A vector containing the nodes to filter, or NULL to not filter;
trees	A vector containing the trees to filter, or NULL to not filter;
states	A vector containing the states to filter, or NULL to not filter;
conf_level	An integer representing the confidence level desired, ranging from 0 to 100. Defaults to 95 percent;
number_of_ticks	The number of intervals in the x axis. Defaults to 20;
type	emr for expected markov reward (dwelling times) or lmt for labelled markov transitions (number of transitions). Defaults to emr;

Value

This function returns a ggplot object that shows the plot when printed. The print itself can be changed as the pleased, using regular ggplot layers functions.

Author(s)

Diego Pasqualin <dpasqualin@inf.ufpr.br>

See Also

map_posterior_distribution, sfreemap

Examples

```
sm <- sfreemap(sfreemap.corals.trees, sfreemap.corals.tips, parallel=FALSE)
map <- map_posterior_distribution(sm[[1]], sm, parallel=FALSE)
p <- plot_distribution_chart(map, 160)
print(p)
```

---

plot_distribution_tree

Plot the dwelling times distribution of a state in a given tree

---

Description

This functions plots the posterior distribution of a state in the tree used as base_tree on map_posterior_distribution.

The legend shows the colors representing the incidence of state given as parameter in the trees analysize in map. The NA values means that there are no sufficient data to compute the values for the branch given the confidence interval (conf_level parameter). In other words, the branch is not present in a sufficient number of the analysed trees.

Usage

```
plot_distribution_tree(map, state, type= emr , conf_level=95, number_of_ticks=20
                    , tip_states=NULL, fsize=0.7, ftype="i", lwd=3)
```

Arguments

map	A result of a map_posterior_distribution execution;
state	The state to be plotted
type	emr for expected markov reward (dwelling times) or lmt for labelled makov transistions (number of transitions). Defaults to emr;
conf_level	An integer representing the confidence level desired, ranging from 0 to 100. Defaults to 95 percent;
number_of_ticks	The number of intervals in which the data will be divided. Defaults to 20;
tip_states	tip states as provided to sfreemap;
fsize	relative font size for tip labels;
ftype	font type - options are "reg", "i" (italics), "b" (bold), or "bi" (bold-italics);
lwd	line width for three branches.

Value

Returns the tree plotted, with the data used in the plot stored in tree$maps.

Author(s)

Diego Pasqualin <dpasqualin@inf.ufpr.br>

See Also

map_posterior_distribution

Examples

```
sm <- sfreemap(sfreemap.corals.trees, sfreemap.corals.tips, parallel=FALSE)
map <- map_posterior_distribution(sm[[1]], sm, parallel=FALSE)
tree <- plot_distribution_tree(map, state= colonial , tip_states=sfreemap.corals.tips)
```

pruning *Prune a phylogetic tree*

Description

Returns t1 with only the tips that are in t2 too. Optionally reroot t1.

Usage

```
pruning(t1, t2, reroot=NULL)
```

Arguments

t1	The tree to be prunned (phylo object.
t2	The base tree (phylo object.)
reroot	Optional. The node that should represent the new root of t2

Value

A phylo object representing the prunned tree.

Author(s)

Diego Pasqualin <dpasqualin@inf.ufpr.br>

read_tips *Read the tip states and labels from a file*

Description

Read a CSV file given as argument and return a matrix representing the states of the tips and its labels (taxa names).

The CSV file must be in a specific format. The first column contain the taxa names, each one in a different row. Every subsequent column is a character, and the state of the character is specified by each taxa on each row, forming a matrix.

You can have as many characters (columns) as you want, but you must specify one to be read, using the character argument, which is an integer representing the column of the character (first character is in column 1).

When some taxon have an ambiguous state, put all states side-by-side, without an separator. For instance, if taxa X can be at both states 'a' and 'b', write 'ab' in the file.

Usage

```
read_tips(file, character=1, sep="\t")
```

Arguments

file	The path to the file containing the data.
character	If the file contains more than one character, you can specify which one to read by the column index.
sep	The csv separator, defaults to TAB.

Details

If you decide to use spaces to separate taxa names from characters make sure the taxa names doesn't have spaces. For instance: "A_grahamae_Agra_Cur" is correct, "A grahamae Agra Cur" is wrong.

Value

A matrix with the taxa label as the row names and a number of columns equal to the number of possible states of the character. The values of the matrix can be one, when the taxon can be at the state, and 0 otherwize.

Author(s)

Diego Pasqualin <dpasqualin@inf.ufpr.br>

---

reorder.sfreemap *Reorder a* phylo *object modified by sfreemap*

Description

This function reorders the edges (and all related objects) of an object phylo modified by sfreemap.

From reorder.phylo (package ape):

In the "cladewise" order each clade is formed by a series of contiguous rows. In the "postorder" order, the rows are arranged so that computations following pruning-like algorithm the tree (or postorder tree traversal) can be done by descending along these rows (conversely, a preorder tree traversal can be performed by moving from the last to the first row). The "pruningwise" order is an alternative "pruning" order which is actually a bottom-up traversal order (Valiente 2002). (This third choice might be removed in the future as it merely duplicates the second one which is more efficient.) The possible multichotomies and branch lengths are preserved.

Usage

```
## S3 method for class  sfreemap
reorder(x, ...)
```

Arguments

x	A tree, result of a sfreemap execution;
...	• order: The resulting order. Can be cladewise or pruningwise. • index.only: logical value indicating whether only an index should be returned. • other arguments, passed to reorder.phylo

Value

A phylogenetic tree of class phylo with its edges reordered.

Author(s)

Diego Pasqualin <dpasqualin@inf.ufpr.br>

See Also

sfreemap, reorder.phylo

---

rescale *Rescale branches of a* phylo *or* multiPhylo *object to a given height*

Description

Rescale a phylo or multiPhylo object to a given height.

Usage

```
rescale(tree, height=NULL, parallel=FALSE)
```

Arguments

tree	One or more trees, result of a sfreemap execution;
height	The desired height to scale branches to.
parallel	Whether computation should run in parallel. Only useful for multiPhylo

Value

A phylogenetic tree of class phylo or multiPhylo with its branches rescaled.

Author(s)

Diego Pasqualin <dpasqualin@inf.ufpr.br>

See Also

sfreemap

sfreemap *Simulation free stochastic character mapping on a phylogenetic tree*

Description

This function performs an analitic stochastic character mapping on a phylogenetic tree (algorithym proposed by Minin and Suchard).

It can be called with a combination of parameters, much like any vectorized function in R. In other words, calling it with N trees (multiPhylo object) and a single rate matrix Q will return N mapped trees. Calling sfreemap with a single tree (phylo object) and M Q matrices will result in M mapped trees, replicas of the single tree with the algorithm applied to it using all Q matrices. Same logic applies to prior as well. It is important to note though that it you pass on N > 1 trees and M > 1 rate matrices (or priors), M and N should be equal.

Usage

```
sfreemap(tree, tip_states, Q=NULL, type="standard", model="SYM"
    , method="empirical", ...)
```

Arguments

tree	a phylogenetic tree as an object of class "phylo", or a list of trees as an object of class "multiPhylo".
tip_states	Two formats are accepted: • A named vector containing the states of the nodes at the tips as values, and the taxa labels as names; • A matrix with characters as columns, tip labels as rows and the state as values.
Q	The transition rate matrix. Can be given as a matrix with state names on dimensons or estimated by the program. Options for estimation depend on argumentos method, model and type.
type	The type of the tip_states being analysed. It can be "standard", usually used for morphological characters, or "dna", for nucleotideos. Default to "standard".
model	By choosing to estimate Q user can then select a model. When type="standard" the available methods are: • "SYM" (default): symmetrical model, e.g, matric(c(0,1,2,1,0,3,2,3,0)); • "ER": equal rates model, for example matrix(c(0,1,1,0)); • "ARD": all rates different model, for example matrix(c(0,1,2,0)); When type="dna" the available methods are JC, F81, K80, HKY, TrNe, TrN, TPM1, K81, TPM1u, TPM2, TPM2u, TPM3, TPM3u, TIM1e, TIM1, TIM2e, TIM2, TIM3e, TIM3, TVMe, TVM, SYM and GTR.
method	The method argument is only used when type= standard and the available options are:

- `empirical (default)`: first it fits a continuous-time reversible Markov model for the evolution of x and then simulates stochastic character histories using that model and the tip states on the tree. This is the same procedure that is described in Bollback (2006), except that simulation is performed using a fixed value of the transition matrix, Q, instead of by sampling Q from its posterior distribution (phytools).
- `"mcmc"`: samples n_simulation Q matrices from the posterior probability distribution of Q using MCMC, then performs stochastic maps conditioned on each sampled value of Q.

`...` Optional parameters, listed below:

- `"prior"`: the prior distribution on the root node of the tree. Options are:
  - `"equal"` (default): root node is sampled from the conditional scaled likelihood distribution at the root;
  - `"estimated"`: the stationary distribution is estimated by numerically solving `pi*Q=0`;
- `"tol"` (default: 1e-8): the tolerance for zero elements in Q, elements less then tol will be set to tol;
- `"parallel"` (default: TRUE): when `tree` is of type `multiPhylo` we can run sfreemap in parallel. The number of processes created will be the same as the cores available in your machine.
- When `Q="mcmc"` some other parameters might be set:
  - `"n_simulations"` (default: 100): The number of Q matrices that will be generated;
  - `"burn_in"` (default: 1000): the burn in for the MCMC;
  - `"sample_freq"` (default: 100): number of generations for each sample taken;

Value

Returns a modified object of class "phylo", adding the following data:

`mapped.edge` a matrix containing the expected value for dwelling times for each state along each edge of the tree;

`mapped.edge.lmt` a matrix containing the expected number of labelled markov transitions for each state along each edge of the tree;

`Q` the given or estimated value of Q;

`logL` The likelihood of calculated for the given or sampled Q;

`prior` The prios given or calculated for the root node;

Author(s)

Diego Pasqualin <dpasqualin@inf.ufpr.br>

References

Vladimir N Minin e Marc A Suchard. Fast, accurate and simulation-free stochastic mapping. Philosophical Transactions of the Royal Society B: Biological Sciences, 363 (1512): 3985-3995, 2008.

`sfreemap.corals.tips` *Tip data for sfreemap.corals.trees*

Description

One binary character with states "solitary" "colonial"). Complete description on sfreemap.corals.trees

Usage

```
sfreemap.corals.tips
```

Format

A named vector containing the states as values and species as names

References

Barbeitos, Marcos S., Sandra L. Romano, and Howard R. Lasker. "Repeated Loss of Coloniality and Symbiosis in Scleractinian Corals." Proceedings of the National Academy of Sciences 107, no. 26 (2010): 11877-82.

`sfreemap.corals.trees` *Partial sequences of the 12S and 28S rRNA for 108 species of scleractinian corals and one sea anemone*

Description

This dataset comprises partial sequences of the 12S and 28S rRNA for 108 species of scleractinian corals and one sea anemone (used as outgroup) and one binary character (0 = solitary and 1 = colonial species). The posterior distribution of trees was sampled via MCMC using the CIPRES Science Gateway (http://www.phylo.org/sub_sections/portal/) implementation of BEAST (Drumond et al., 2007). Sequences were aligned with reference to a consensus secondary structure model (details in Barbeitos et al., 2010) and data were partitoned into loop and stems. Analyses were run under the Hasegwa Kishino Yano substitution model (HKY, Hasegawa et al., 1985) with rate heterogeneity among sites modeled by a gamma distribution discretized into 4 categories and a relaxed (uncorrelated) lognormal molecular clock. The chain was run for 10 million generations and sampled every 1,000 generations. Stationarity was assessed using Tracer (Rambaut et al., 2014) after the first 10% chain was discarded. In order to maximize independence among samples, the ensuing distribution of trees was "thinned" to 901 trees, i.e, sub-sampled until the integrated autocorrelated time (IACT) dropped to 0 (see Pagel et al., 2006 for details).

Usage

```
sfreemap.corals.trees
```

Format

A "multiPhylo" object containing 901 phylogenetic trees

References

Barbeitos, Marcos S., Sandra L. Romano, and Howard R. Lasker. "Repeated Loss of Coloniality and Symbiosis in Scleractinian Corals." Proceedings of the National Academy of Sciences 107, no. 26 (2010): 11877-82.

Drummond, Alexei, and Andrew Rambaut. "BEAST: Bayesian Evolutionary Analysis by Sampling Trees." BMC Evolutionary Biology 7 (2007): 214.

Hasegawa, M., H. Kishino, and T. Yano. "Dating the Human-ape Split by a Molecular Clock of Mitochondrial DNA." Journal of Molecular Evolution 22 (1985): 160-74.

---

sfreemap.primates.dna.tips

Tip data for sfreemap.primates.trees

Description

Complete description on sfreemap.primates.trees

Usage

```
sfreemap.primates.dna.tips
```

Format

A matrix containing the species as row names and characters as columns

---

sfreemap.primates.summary.tree

Taxonomic sampling of Old World monkeys

Description

Summary of sfreemap.primates.trees

Usage

```
sfreemap.primates.summary.tree
```

Format

A "phylo" object

`sfreemap.primates.trees`

Taxonomic sampling of Old World monkeys

Description

Taxonomic sampling of Old World monkeys (parvorder Catarrhini, which includes humans, gorillas, chimpanzees, orangutans, gibbons, baboons, macaques and langurs) used by Pagel et al. (2006), plus one New World monkey (the tufted capuchin, Cebus apella), which was used as outgroup. Complete and partial cytochrome b sequences were downloaded from GeneBank and aligned using the online version of MAFFT (Katoh et al., 2007). The alignment was converted to amino acid sequences and trimmed so that the reading frame starts at the first site. Data were partitioned by codon position and analyzed under the general-time reversible (GTR, Tavare, 1986) substitution model with rate heterogeneity among sites also modeled by a 4-category gamma distribution and rate variation across branches accommodated by a relaxed (uncorrelated) lognormal molecular clock. Four independent chains were run for 100 million generation and sampled every 100,000 generations. Stationarity and convergence among chains were assessed using Tracer (after a 10% burnin) and the posterior distributions of trees were also "thinned". These posteriors were subsequently combined using LogCombiner (Drummond 2006) into a single distribution of 3,600 trees.

Usage

```
sfreemap.primates.trees
```

Format

A "multiPhylo" object containing 3596 phylogenetic trees

References

Barbeitos, Marcos S., Sandra L. Romano, and Howard R. Lasker. "Repeated Loss of Coloniality and Symbiosis in Scleractinian primates." Proceedings of the National Academy of Sciences 107, no. 26 (2010): 11877-82.

Pagel, Mark, and Andrew Meade. "Bayesian Analysis of Correlated Evolution of Discrete Characters by Reversible-Jump Markov Chain Monte Carlo." Am Nat 167, no. 6 (2006): 808-25.

Katoh, Kazutaka, Kei-ichi Kuma, Hiroyuki Toh, and Takashi Miyata. "MAFFT Version 5: Improvement in Accuracy of Multiple Sequence Alignment." Nucleic Acids Research 33, no. 2 (2005): 511-18. doi:10.1093/nar/gki198.

Tavare, S. "Some Probabilistic and Statistical Problems in the Analysis of DNA Sequences." American Mathematical Society: Lectures on Mathematics in the Life Sciences 17 (1986): 57-86.

Drummond, Alexei, and Andrew Rambaut. "BEAST: Bayesian Evolutionary Analysis by Sampling Trees." BMC Evolutionary Biology 7 (2007): 214.

Index

APÊNDICE B – Exemplos de uso da aplicação

Os exemplos de uso da aplicação, mais conhecidos pela comunidade R como *vignettes*, visam demonstrar como o usuário final pode utilizar a ferramenta. Possui linguagem e formato mais informal que o manual do usuário e apresenta exemplos do tipo "pergunta-resposta", detalhando como o usuário pode responder perguntas específicas sobre seu conjunto de dados conectando diferentes funções do pacote ou analisando os retornos com a ajuda de outras ferramentas e pacotes.

Sfreemap

Diego Pasqualin, Marcos Barbeitos and Fabiano Silva

2016-01-10

Contents


```
## Loading required package: ape
```

1 Introduction

Put some description here so people don't lose their minds...

2 Installing *sfreemap*

To install the development version from github:

```
library(devtools)
install_github("dpasqualin/sfreemap")
```

The stable version can be installed from CRAN using:

```
install.packages(sfreemap)
```

Then, to load the package, use:

```
library("sfreemap")
```

3 New object classes

One new class of object extend existing data structure for phylogenetic trees: * **sfreemap:** complements "phylo" and "multiPhylo" classes from ape and phytools with *mapped.edge.lmt*, a matrix containing the expected value for the number of transitions among states.

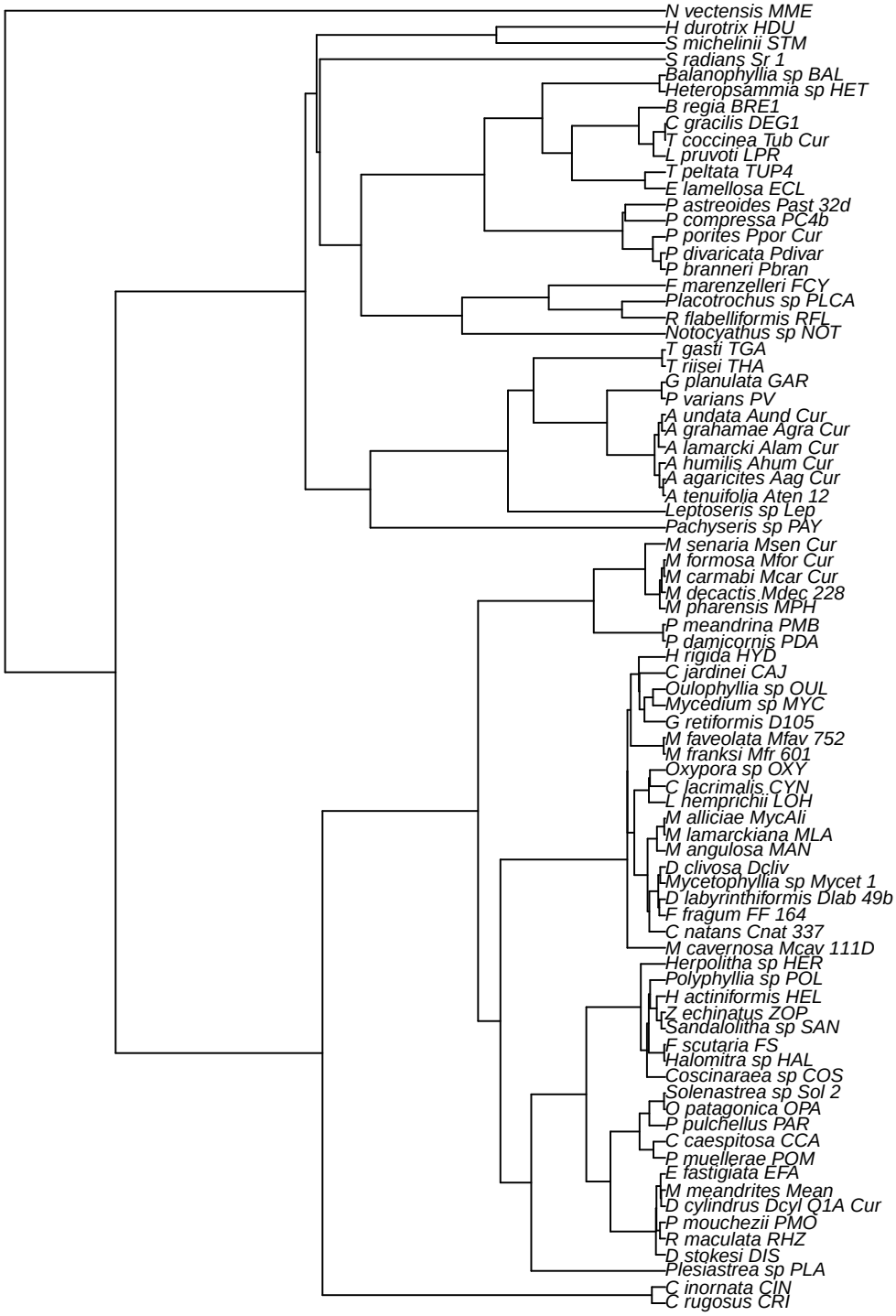
4 Simple stochastic mapping

4.1 Standard type

The program accepts the parameter *type=standard* which should be used when the character is of morphological type. The dataset *sfreemap.corals.trees* and its corresponding tip values *sfreemap.corals.tips* can be used as example here.

Just to have an idea of the dataset we are working on, let's have a look at the first tree:

```
plot.phylo(sfreemap.corals.trees[[1]], cex=0.7)
```



The command *sfreemap.map* runs by default with *method='empirical'* and *type='standard'*, so we don't have supply it now. Please check the package documentation for more details.

This function will run in parallel with a number of processes equal to the number of cores in your machine. If you want to disable parallel processing you can pass parallel=FALSE as a third parameter. Many other function of sfreemap presents this parameter.

```
data <- sfreemap.map(sfreemap.corals.trees, sfreemap.corals.tips, parallel=FALSE)
class(data)
```

```
## [1] "sfreemap"   "multiPhylo"
```

```
data
```

```
## 901 phylogenetic trees
```

All trees have the objects *mapped.edge* and *mapped.edge.lmt*, representing the expected dwelling times and expected number of transitions for each state on each branch of the tree. The following commands shows how to use these objects.

```
t <- data[[1]] # Let's work with the first tree
t
```

```
##
## Phylogenetic tree with 81 tips and 80 internal nodes.
##
## Tip labels:
##  A_lamarcki_Alam_Cur, H_durotrix_HDU, C_jardinei_CAJ, B_regia_BRE1, L_hemprichii_LOH, A_agaricites_A
##
## Rooted; includes branch lengths.
```

```
# Get the list of the first 10 edges
t$edge[1:10,]
```

```
##       [,1] [,2]
##  [1,]   82   83
##  [2,]   83   84
##  [3,]   84   85
##  [4,]   85   36
##  [5,]   85   48
##  [6,]   84   86
##  [7,]   86   87
##  [8,]   87   88
##  [9,]   88   51
## [10,]   88   89
```

```
# Get the dwelling times for the first edge, the one connecting node 82 with 83
t$mapped.edge[1,]
```

```
## colonial solitary
##    39.77    44.47
```

```
# Do the same thing using the branch name
t$mapped.edge['82,83',]
```

```
## colonial solitary
##    39.77    44.47
```

```
# Do the same thing for number of transitions
t$mapped.edge.lmt['82,83',]
```

```
## colonial,solitary solitary,colonial
##            0.2432            0.2161
```

4.2 DNA type

Use *type=dna* when working with nucleotides. The datasets *sfreemap.primates.trees* and *sfreemap.primates.dna.tips* can be used as example.

Again, we will run the program with only the first ten trees. The tips dataset has several characters, and again we are going to use just the first ten. What the program will do in this case is to run each tree against each character, in the order they appear in their respective objects. The result will be ten mapped trees. Same logic would apply for standard type.

```
sfreemap.map(sfreemap.primates.trees[1:10], sfreemap.primates.dna.tips[,1:10], parallel=FALSE)
```

```
## 10 phylogenetic trees
```

It is also possible to run the program passing a single tree as parameter and multiple characters. In this case each have ten trees as result, each one with a mapping that corresponds to a character.

```
sfreemap.map(sfreemap.primates.trees[[1]], sfreemap.primates.dna.tips[,1:10], parallel=FALSE)
```

```
## 10 phylogenetic trees
```

Of course, the user can also map a single character into multiple trees, like following:

```
res <- sfreemap.map(sfreemap.primates.trees[1:10], sfreemap.primates.dna.tips[,1], parallel=FALSE)
```

When using a single tree one can compute the mean value across trees for the number of transitions and dwelling times as follow:

```
# Using result from the last execution
mean.dt <- Reduce('+', lapply(res, function(x) x$mapped.edge)) / length(res)
mean.dt[1:10,] # just the first ten rows to give an idea..
```

```
##               -       a
## 60,61 6.013e-03 0.13635
## 61,62 2.616e-03 0.21027
## 62,63 6.944e-04 0.17085
## 63,64 2.599e-05 0.04004
## 64,65 3.463e-05 0.05752
## 65,36 1.301e-05 0.04059
## 65,66 1.202e-05 0.02736
## 66,37 4.384e-06 0.02853
## 66,38 2.289e-06 0.01787
## 64,18 1.713e-05 0.03332
```

Although the command above is quite typical for an R experienced programmer, it might not be as simple to remember and understand for a more regular user. Besides, what if we want the median instead of the mean? The command would be completely different. That is why we provide some tools to analyse the mappings produced by *sfreemap.map* command, which will be described in the next section.

5 Analysing mapped data with histograms

As we have shown, the object created by *sfreemap.map* can be analysed and manipulated to produce summaries and plots. In this section we will show some tools to make this task a lot easier.

For the examples we will map again the *sfreemap.corals.trees* dataset, but this time scaling the trees to have the same distance from the root to the terminals. This is important for analysing the number of transitions considering the same proportion on all trees.

```
trees <- sfreemap.rescale(sfreemap.corals.trees, height=1)
data <- sfreemap.map(trees, sfreemap.corals.tips)
```

Now we will run the function below, which analysis mapping results on all trees, creating an object with the dwelling times and the number of transitions for every node of every tree. It tries to match nodes using function *matchNodes* from *phytools* packages, which consider two nodes to be the same when they share the same taxa. If you haven't rescaled the trees before but want to do it now, just pass on *scale.trees=1* (or any other value) to the function below.

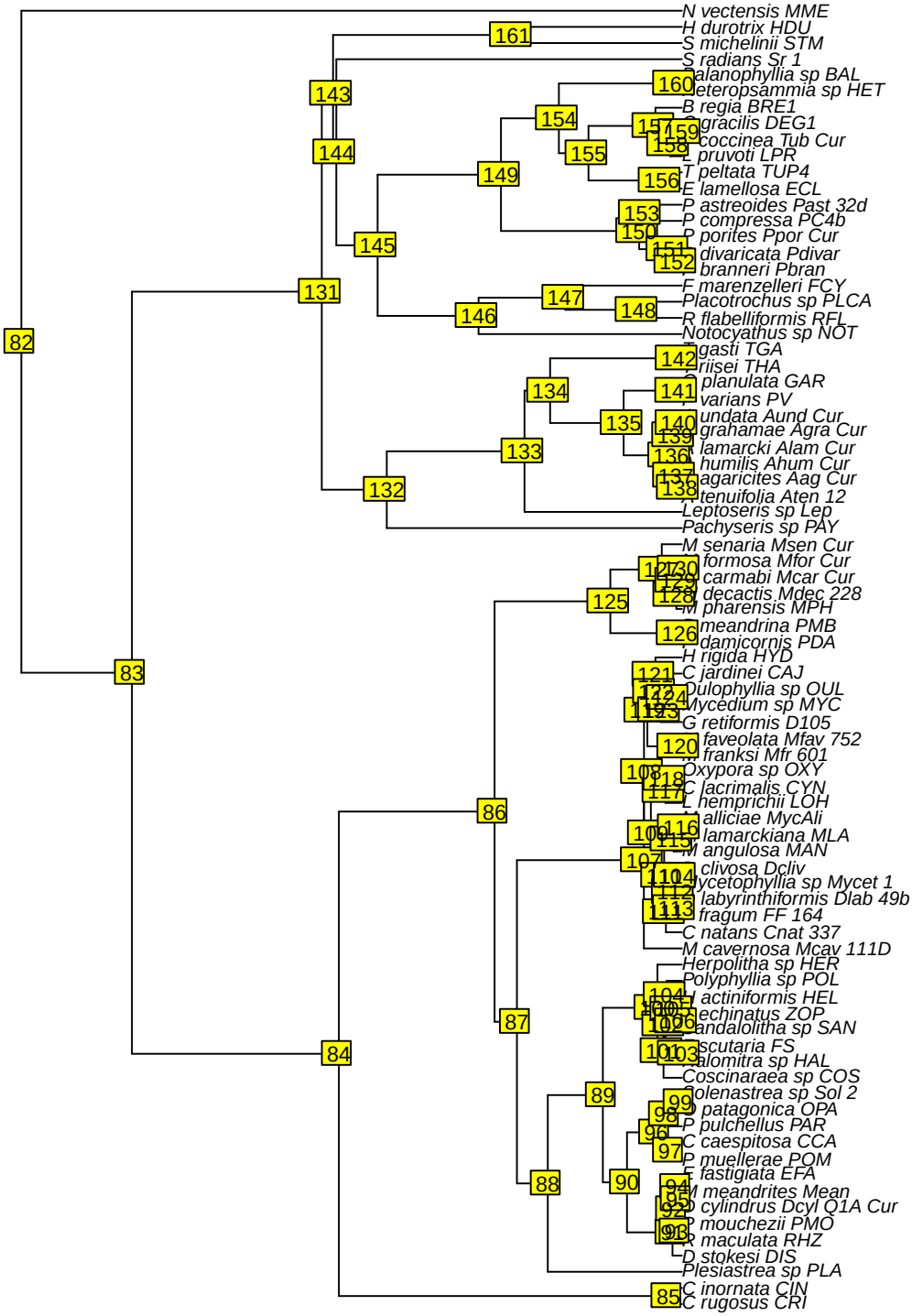
```
base_tree <- data[[1]]
mpd <- sfreemap.map_posterior_distribution(base_tree, data, scale.branches=TRUE, parallel=FALSE)
```

The first parameter of the function is the so called "base tree", the tree on which the branches of the other trees will be compared to. Second parameter is all other trees to be analysed, *scale.branches* is used to represent the dwelling times regarding the percentage of time spent in the branch, instead of the absolute value (the expected number of transitions is not scaled, as it doesn't make sense to do so).

5.1 Expected dwelling times for states of a branch across all trees

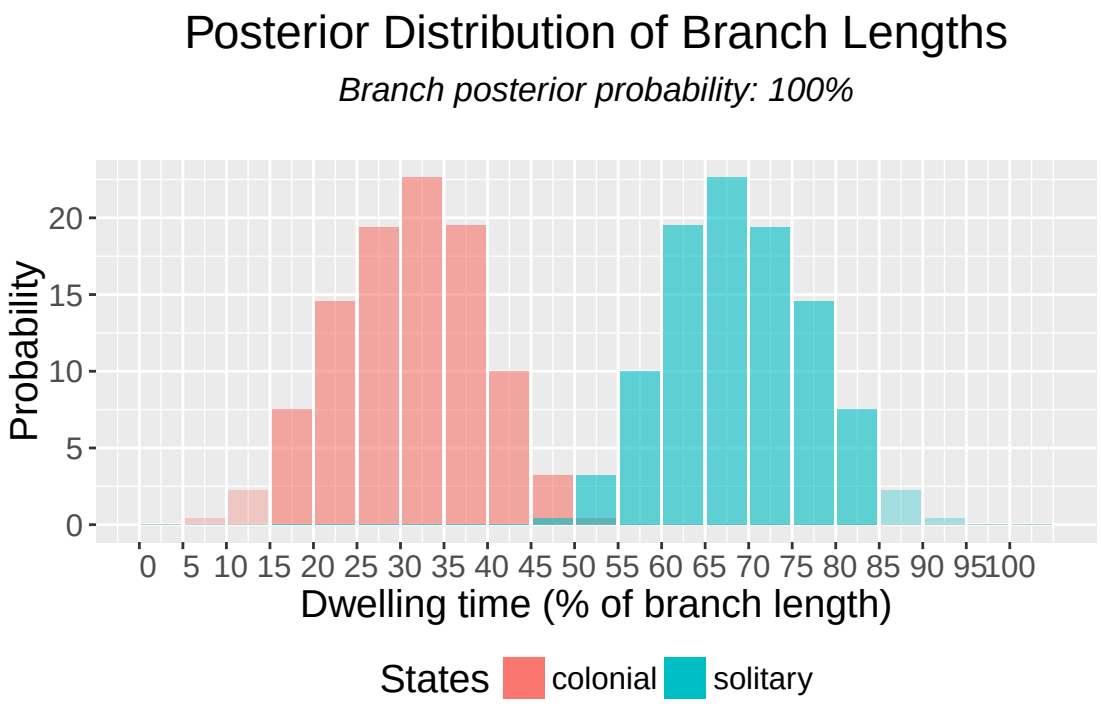
Let's suppose we want to check the posterior distribution of states on a specific branch of the tree. Branches don't usually have names, so we will define a branch by it's ending node. For example, the code below will plot our base tree and it's correspondent node numbers.

```
plot.phylo(data[[1]], cex=0.7)
nodelabels(cex = .75, bg = "yellow")
```



Picking one node, let's say *89*, we can see the posterior distribution for dwelling times on the branch that **ends** on node *89* by typing the following command (these are the default values for arguments *conf_level*, *number_of_ticks* and *type*, so you can omit it and get the same results):

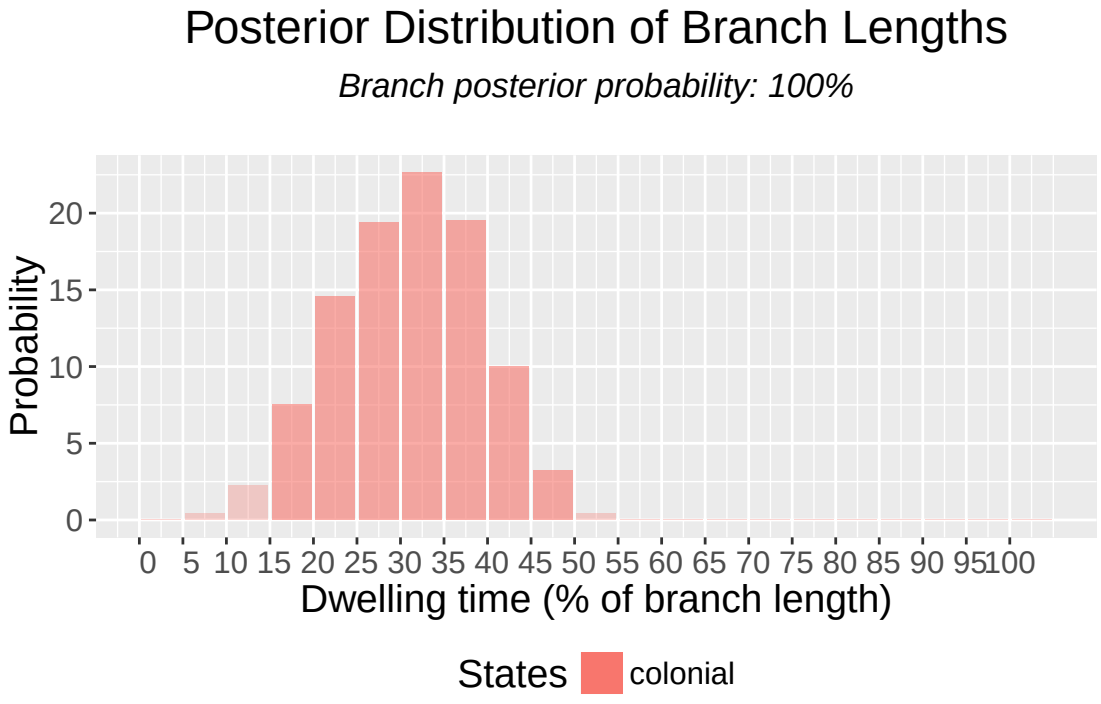
```
sfreemap.plot_distribution(mpd, 89, conf_level=95, number_of_ticks=20, type='emr')
```



The interpretation is as follows: the *colonial* state was present with 95% certainty around 10% and 45% of the time on the branch ending on node 89 and, with equal certainty, the state *solitary* was present during 55% to 90% of the time on this branch. At the top of the chart, the text *branch posterior probability: 100%* means that this particular branch was present on 100% of the trees given as argument to the function *sfreemap.map_posterior_distribution* and compared to out *base tree*.

It is also possible to plot only one of the states by supplying the *states* argument.

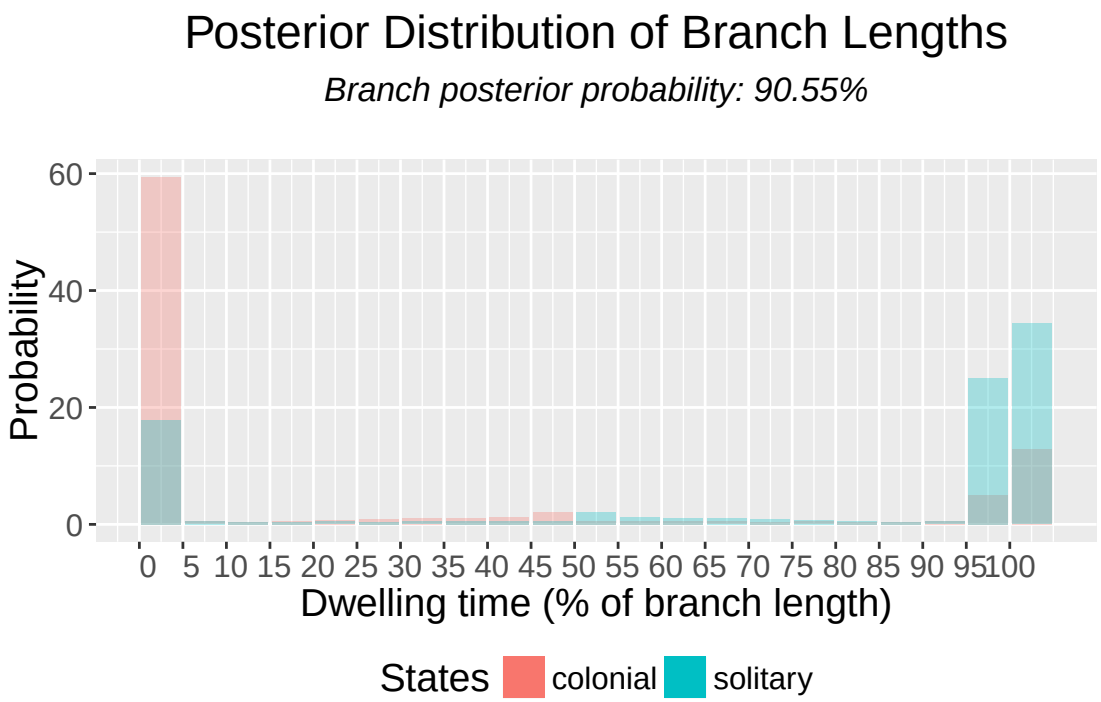
```
sfreemap.plot_distribution(mpd, 89, states='colonial')
```



Expected dwelling times for states for all branches across all trees

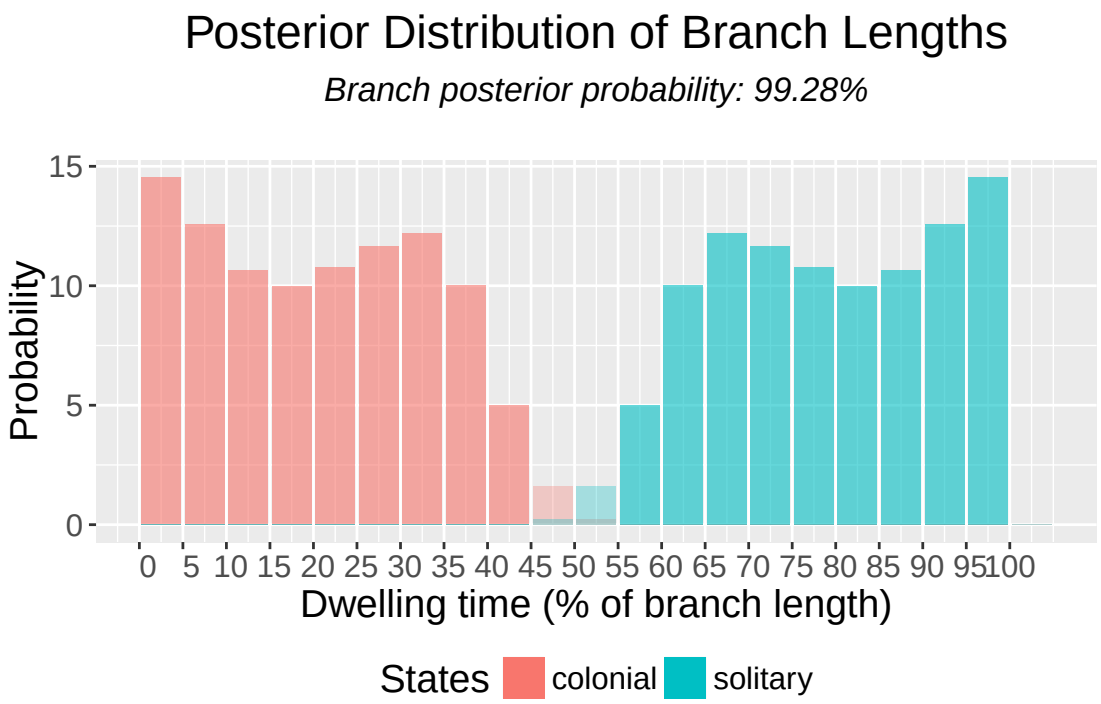
Quite easy to do that, just omit the *node* parameter and the function will consider the distribution over all nodes.

```
sfreemap.plot_distribution(mpd, type='emr')
```



If you want to plot the distribution over a specific set of nodes just pass it as a vector, like this:

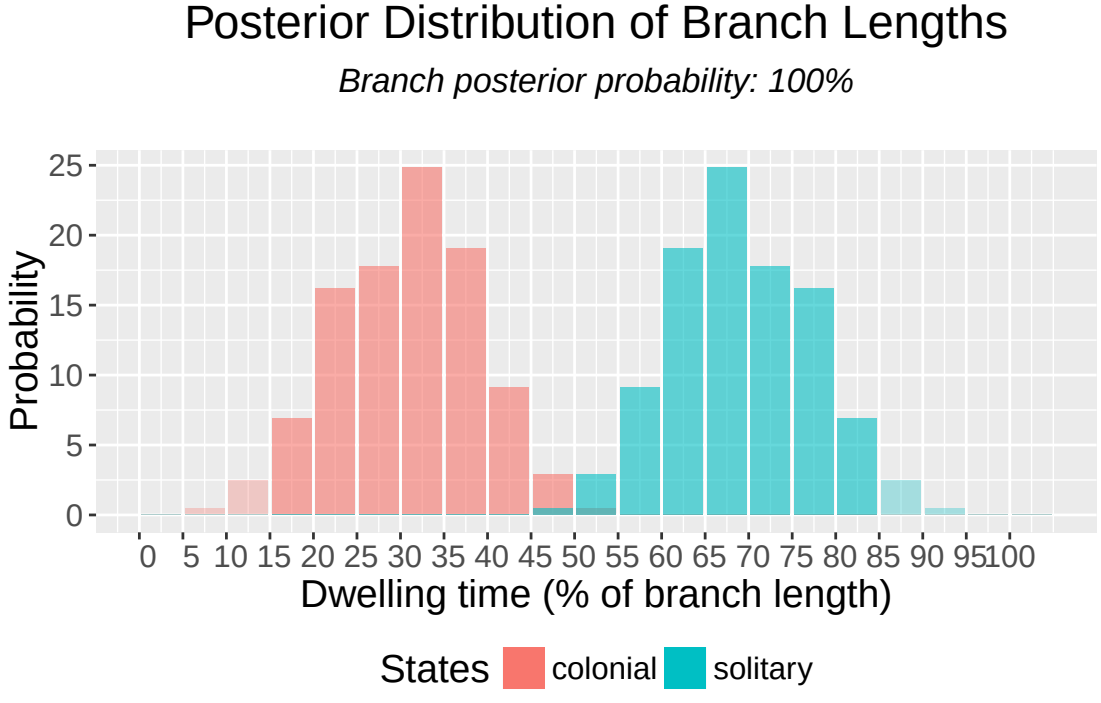
```
sfreemap.plot_distribution(mpd, nodes=c(89,90), type='emr')
```



Expected dwelling times for states for all branches on a group of trees

To filter or limit the distribution over a specific group of trees, or maybe a single tree, pass on the argument *trees*, which work in the same way as *nodes* and *states*.

```
# get the odd trees
trees <- seq(1, length(sfreemap.corals.trees), 2)
sfreemap.plot_distribution(mpd, nodes=89, trees=trees)
```

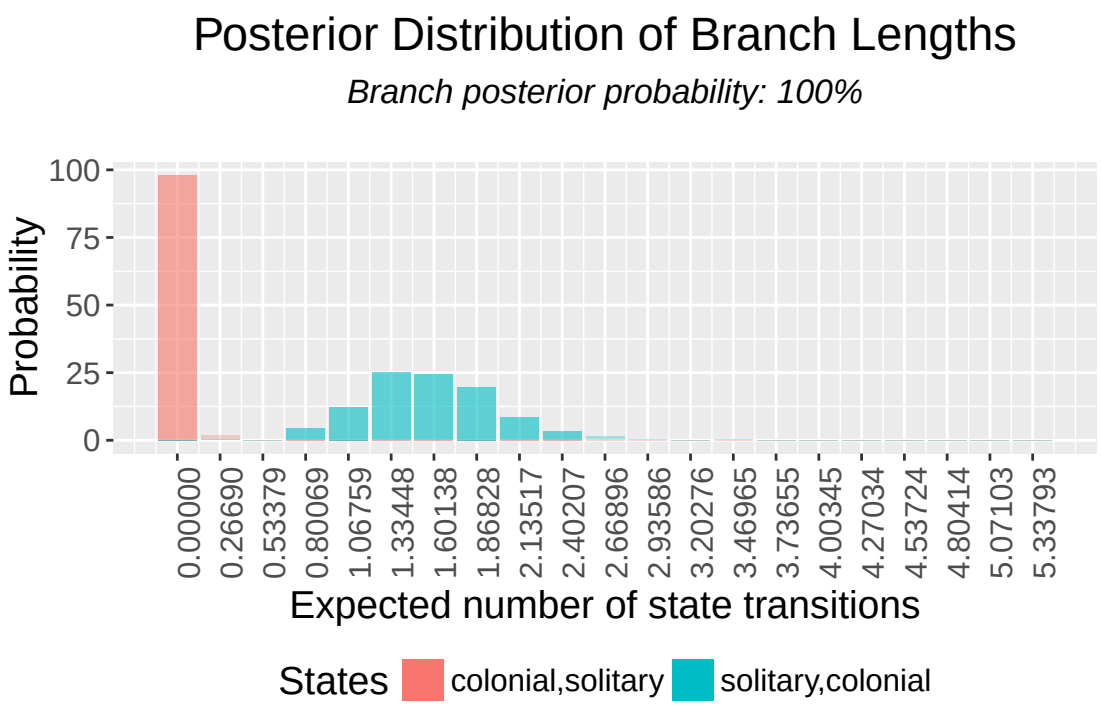


Expected number of transitions for states of one or more branches across all trees

Similar plots can be generated for the number of transitions using the same function and *mpd* object created before, just changing the parameter *type* from *emr* to *lmt* (labelled markov transitions).

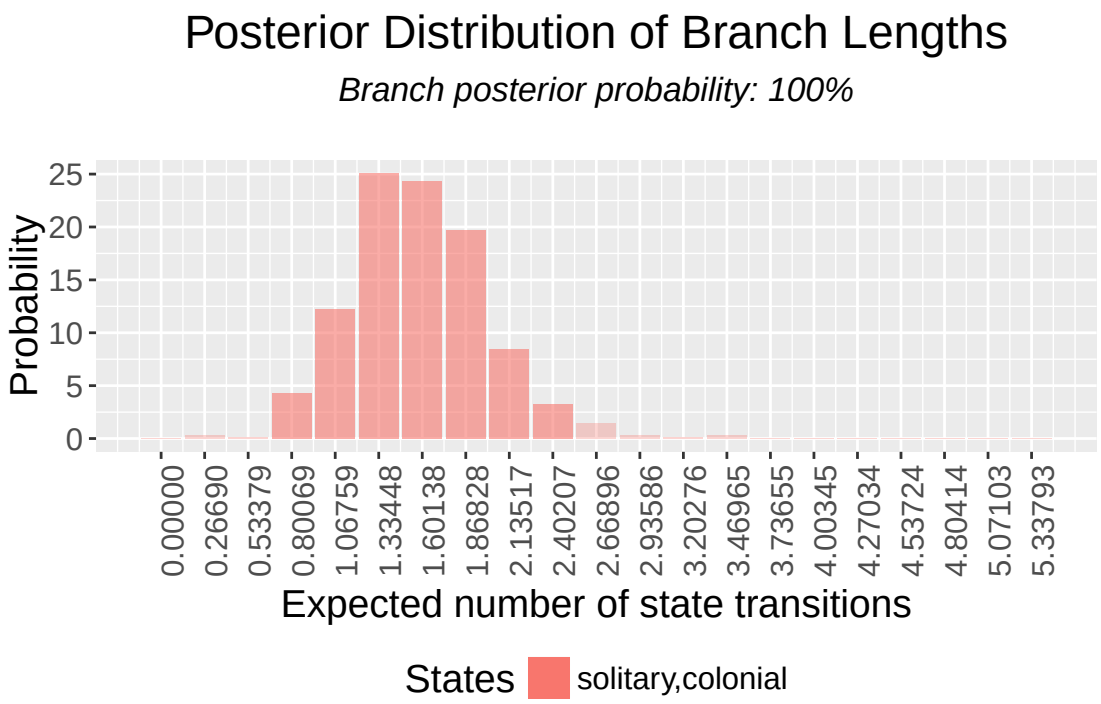
As stated before, it doesn't make sense to scale the number of transitions according to the branch length, so here the absolute values are represented in the x-axis.

```
sfreemap.plot_distribution(mpd, 85, type='lmt')
```



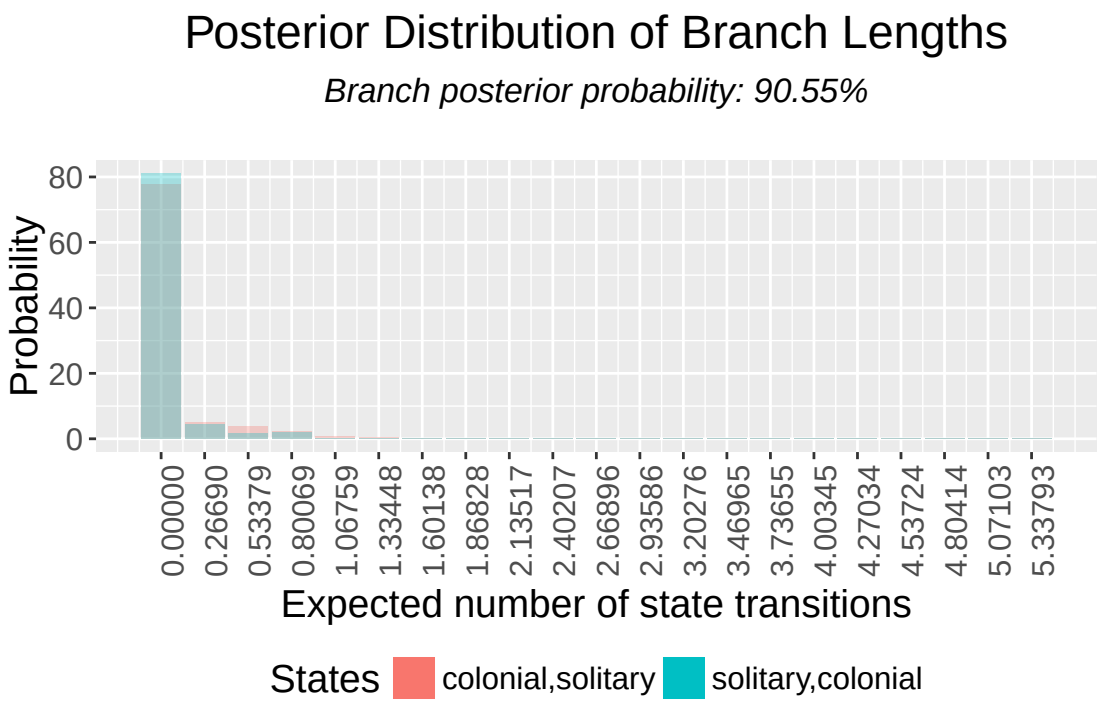
Needless to say, but you can plot a single state transition here:

```
sfreemap.plot_distribution(mpd, 85, states='solitary,colonial', type='lmt')
```



Example considering all nodes and all states:

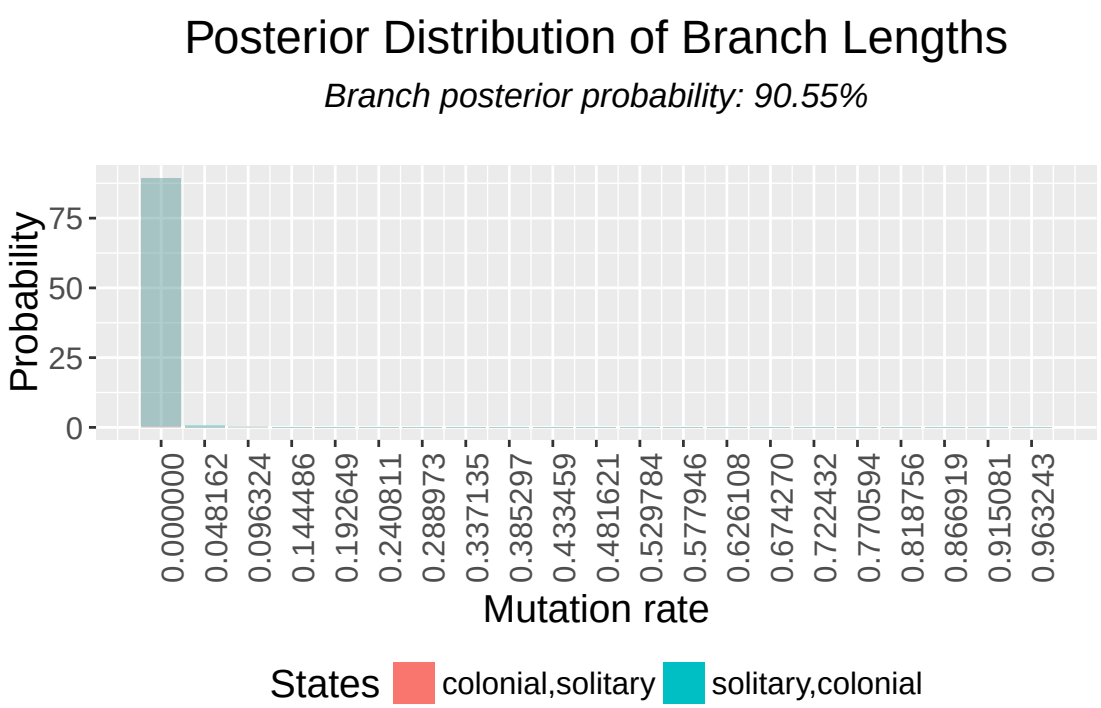
```
sfreemap.plot_distribution(mpd, type='lmt')
```



Expected mutation rate for states of one or more branches across all trees

As the last type of distribution plot, *mr (mutation rate)* shows the rate of mutation for states per branch. Parameters are very similar than before. It is possible to filter by trees, nodes and/or states.

```
sfreemap.plot_distribution(mpd, type='mr')
```



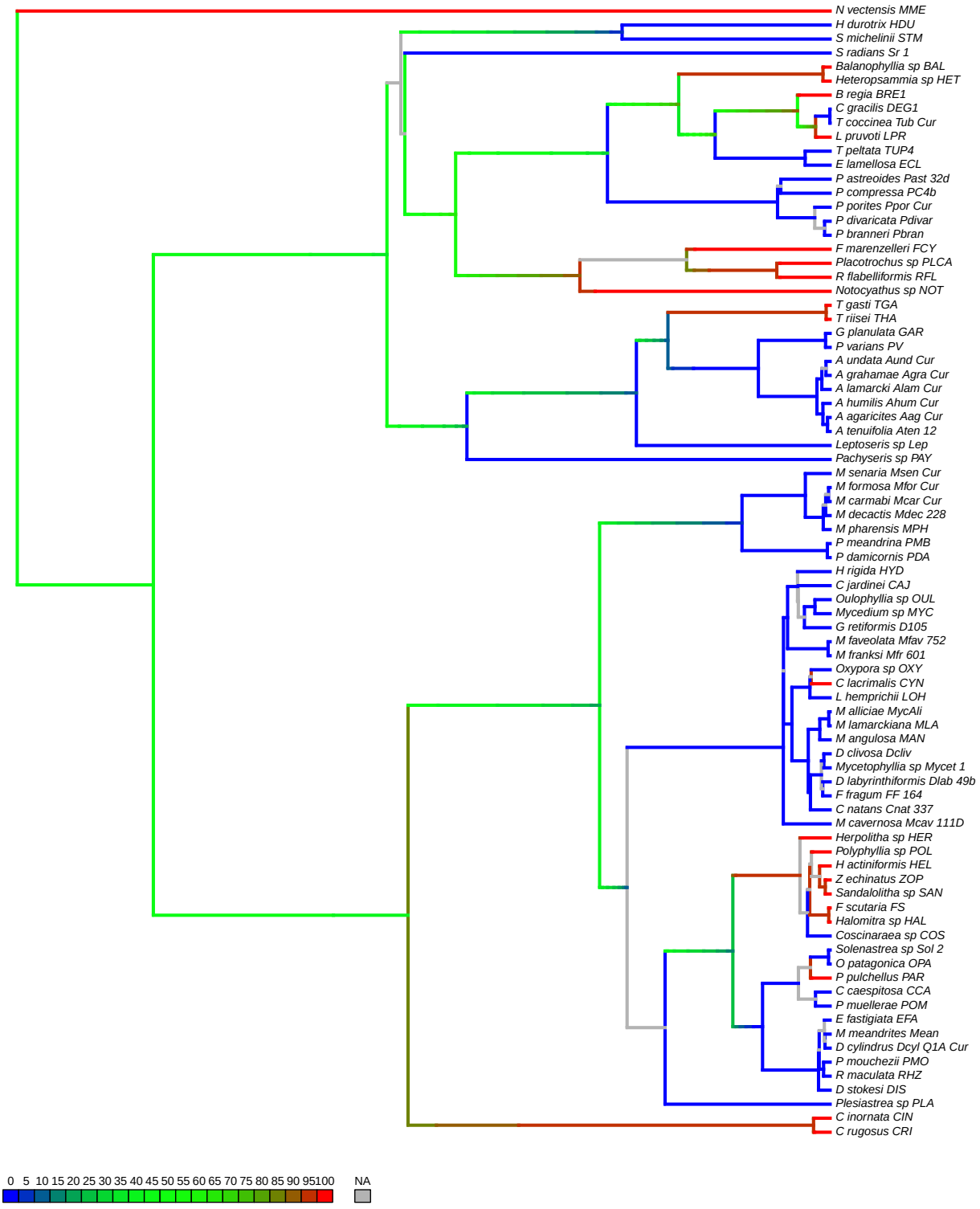
Analysing mapped data by plotting a tree

Sfreemap can show up mutation rate and dwelling times in a graphical representation of a tree. Fist let's generate a mapping for all trees of *sfreemap.corals.trees* dataset, and then map the postero distribution (as we did before).

```
data <- sfreemap.map(sfreemap.corals.trees, sfreemap.corals.tips, method='empirical')
mpd <- sfreemap.map_posterior_distribution(data[[1]], data)
```

Now we can use the function below to plot a tree showing the distribution of time spent by a particular state (second argument). As the legend says, red shifted colors indicates that the state was more present, and on the other side of the spectrum, blue indicates less time spent for the state on that particular branch.

```
sfreemap.plot_tree(mpd, 'colonial', type='emr', conf_level=95)
```

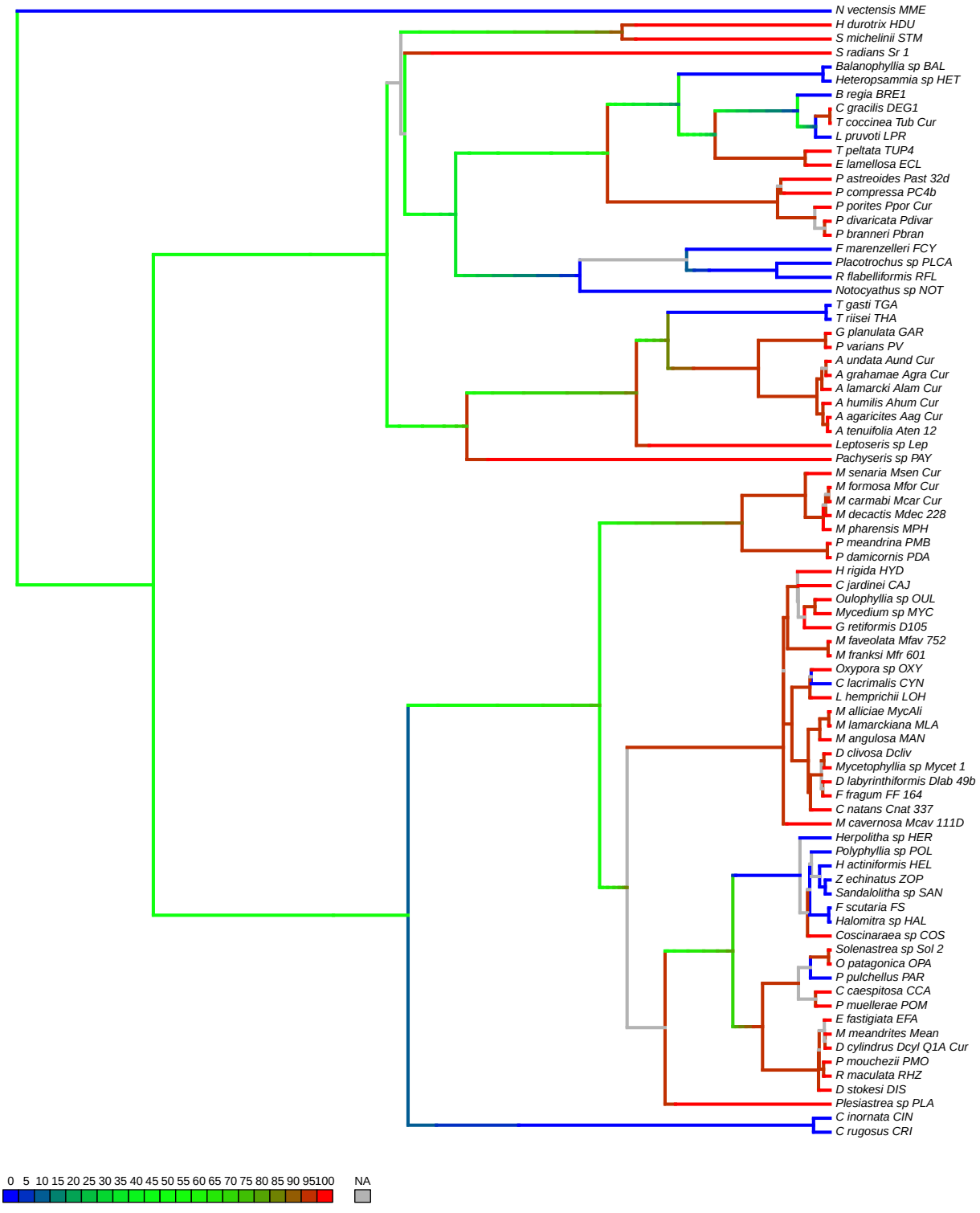



```
##
## Phylogenetic tree with 81 tips and 80 internal nodes.
##
## Tip labels:
```

```
##  A_lamarcki_Alam_Cur, H_durotrix_HDU, C_jardinei_CAJ, B_regia_BRE1, L_hemprichii_LOH, A_agaricites_A
##
## Rooted; includes branch lengths.
```

Now the same for the other state:

```
sfreemap.plot_tree(mpd, 'solitary', type='emr', conf_level=95)
```

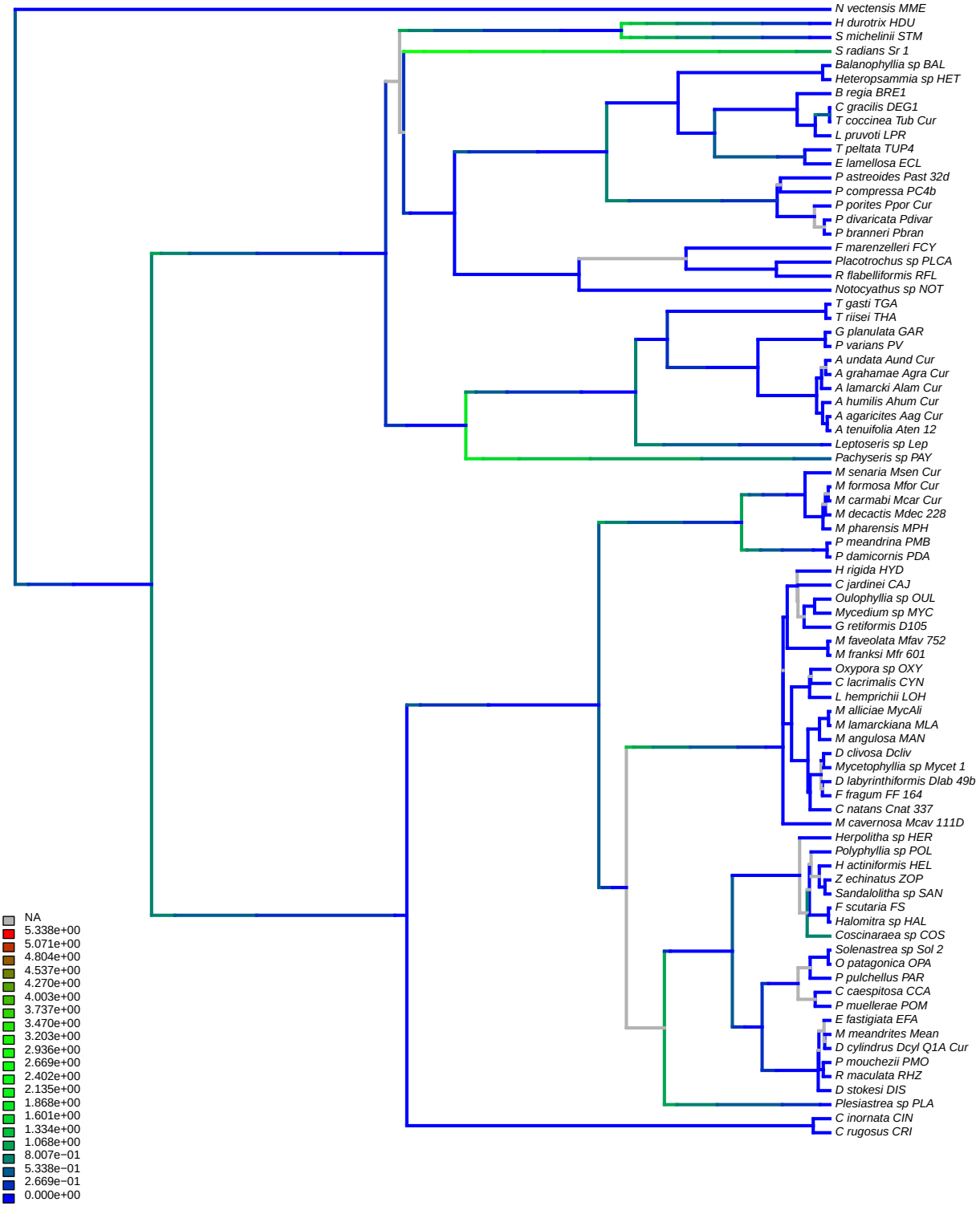



```
##
## Phylogenetic tree with 81 tips and 80 internal nodes.
##
## Tip labels:
```

```
##  A_lamarcki_Alam_Cur, H_durotrix_HDU, C_jardinei_CAJ, B_regia_BRE1, L_hemprichii_LOH, A_agaricites_A
##
## Rooted; includes branch lengths.
```

It is also possible to analyse the number of transitions on each branch by changing the `type'' parameter tolmt" and specifying a transition between two state instead of just one state (comma separated). In this case the legend doesn't show a percentage, but the absolute expected value.

```
sfreemap.plot_tree(mpd, 'colonial,solitary', type='lmt', conf_level=95)
```




```
##
## Phylogenetic tree with 81 tips and 80 internal nodes.
##
## Tip labels:
```

```
##  A_lamarcki_Alam_Cur, H_durotrix_HDU, C_jardinei_CAJ, B_regia_BRE1, L_hemprichii_LOH, A_agaricites_A
##
## Rooted; includes branch lengths.
```

6 Correlation matrix

It is possible to plot a correlation matrix to compare results provided by different parameters on *sfreemp.map* or even different programs, as long as they return a *phylo* object with mapping on *mapped.edge*.

Let's first create the object that will hold our data to plot the correlation matrix. The only parameter is the name of the state that will be used for the comparison.

```
cor <- sfreemap.correlation('colonial')
```

Now we can run the mapping and add the results in the correlation matrix. You can compare as many mappings as you with by just repeating the commands below. Note that *add.data* receives the object we created above, the mapping and, as the last paramater, the name we will in the resulting matrix to tell apart the different data we are comparing.

```
# Estimate Q using empirical method
data <- sfreemap.map(sfreemap.corals.trees[[1]], sfreemap.corals.tips, method='empirical')
cor <- add.data(cor, data, 'sfreemap empirical')

# Estimate Q using mcmc method
data <- sfreemap.map(sfreemap.corals.trees[[1]], sfreemap.corals.tips, method='mcmc', n_simulation=1)
cor <- add.data(cor, data, 'sfreemap mcmc')
```

Like we said before, it is possible to use data from different programs, as long as they return (or you adapt) something similar as what *sfreemap.map* returns. In fact, we based our return object on the return of the *make.simmap* function from *phytools* package, which are using running below to compare with *sfreemap.map*.

```
require(phytools)
```

```
## Loading required package: phytools
```

```
## Loading required package: maps
```

```
##
##  # ATTENTION: maps v3.0 has an updated 'world' map.        #
##  # Many country borders and names have changed since 1990. #
##  # Type '?world' or 'news(package="maps")'. See README_v3. #
```

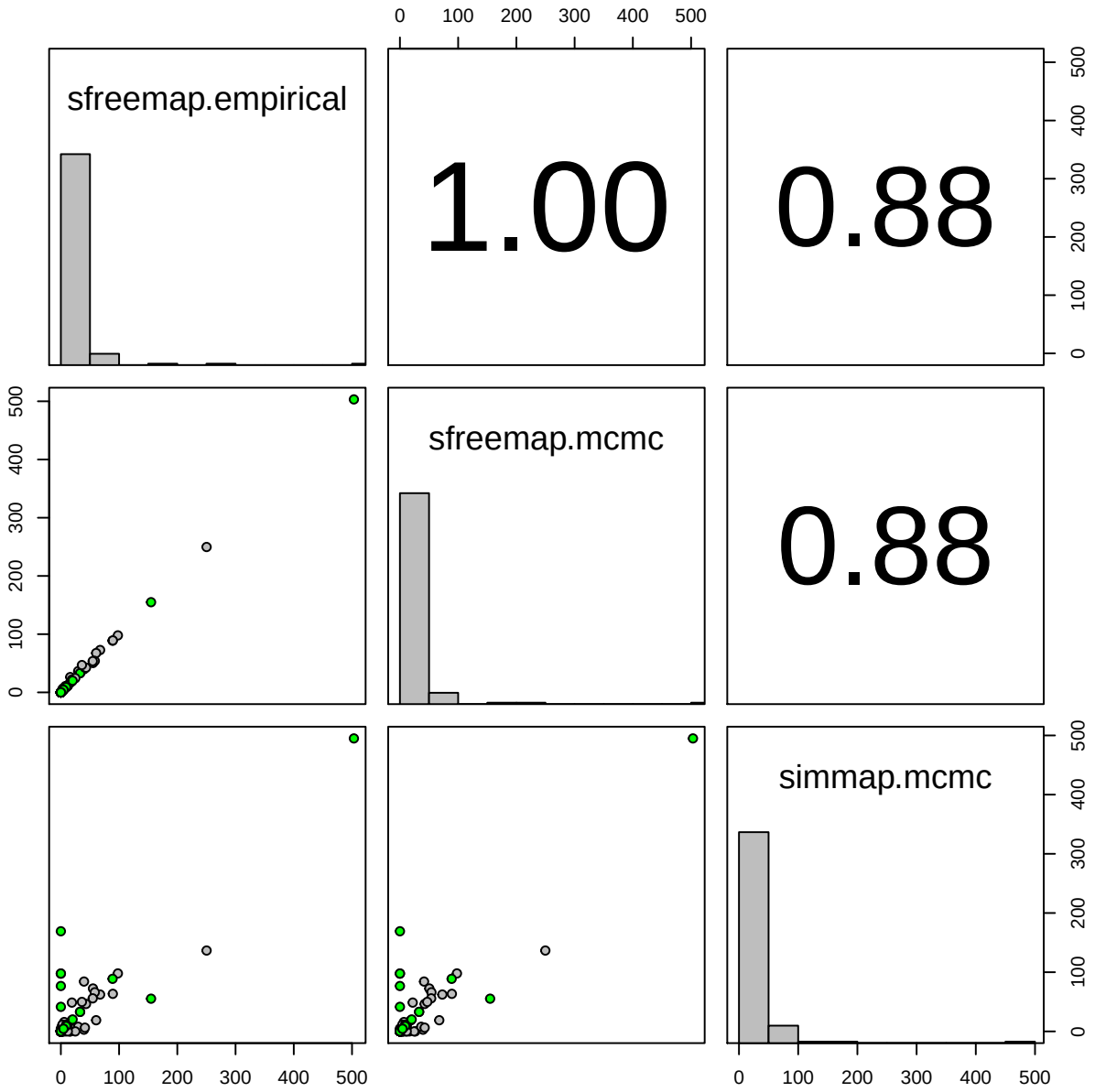
```
data <- make.simmap(sfreemap.corals.trees[[1]], sfreemap.corals.tips, Q='mcmc', n_simulation=1)
```

```
## Done.
```

```
cor <- add.data(cor, data, 'simmap mcmc')
```

At last, let's see the result in a nice image:

```
plot(cor)
```


sfreemap.empirical
1.00
0.88
sfreemap.mcmc
0.88
simmap.mcmc
0
100
200
300
400
500