Tutorial 13

Inferência por Verossimilhança Máxima: homologia estática e dinâmica

BIZ0433 - INFERÊNCIA FILOGENÉTICA: FILOSOFIA, MÉTODO E APLICAÇÕES.

Conteúdo

Objetivo **214**

13.1 Verrossimilhança: homologia estática em GARLI **215**

13.1.1 GARLI: Single model 217

13.1.2 GARLI: Partition model 220

13.2 Verrossimilhança: homologia dinâmica em POY **223**

13.2.1 Formas de implementação 225

13.2.2 Seleção de modelos de verossimilhança em POY 225

13.2.3 **MAL**: *Maximum Average Likelihood* em POY 229

13.2.4 **MPL**: *Maximum Parsimonious Likelihood* em POY 232

13.3 Considerações finais sobre análises de verossimilhança e homologia dinâmica **233**

13.4 Referências **233**

Objetivo

Este tutorial apresenta implementações de busca sub o critério de verossimilhança máxima (**ML**) dentro do contexto de homologia estática e dinâmica. O tutorial começa com a apresentação do programa GARLI e como ele pode ser utilizado para análises filogenéticas de homologia estática considerando um único modelo de substituição ou distintos modelos para diferentes partições de dados. Em seguida o tutorial apresenta o uso de POY em análises de **ML** dentro do contexto de homologia estática e dinâmica. Para POY, o tutorial explora a implementação de *Maximum Avarege Likelihood* (MAL) e *Most Parsimonious Likelihood* (MPL). Adicionamente, este tutorial ilustra como POY pode ser utilizado para seleção de modelos de verossimilhança, bem como os modelos implementados atualmente no programa. Este tutorial deve ser considerado como um elemento introdutório para os conceitos e implementações dos métodos abordados. Maiores detalhes sobre o uso deste critério em análises filogenéticas, a forma de implementação de estratégias heurísticas de buscas mais agressivas, e como aplicar o métodos em diversos caracteres genotípicos e fenotípicos podem ser encontrados na documentação de POY e na literatura referenciada no mesmo. Os arquivos associados a este tutorial estão disponíveis no GitHub. Você baixar todos os tutoriais com o seguinte comando:

```
svn checkout https://github.com/fplmarques/cladistica/trunk/tutorials/
```

13.1 Verrossimilhança: homologia estática em GARLI

No tutorial anterior exploramos os conceitos relacionados com estimativas de verossimilhança máxima de parâmetros, cálculo de verossimilhança para reconstruções e topologias e seleção de modelos. Desta forma, ficou evidente que o processo lógico de inferência filogenética é o mesmo dentro deste critério de otimalidade, ou seja, topologias são avaliadas quanto ao seu mérito por medidas de verossimilhança e a tolologia que maximiza a probabilidade de obter os dados é selecionada – da mesma forma que fazemos com o critério de parcimônia.

Neste componente do tutorial iremos explorar os aspectos práticos de inferência sob o critério de verossimilhança máxima em GARLI [1]. Há uma série de outros programas disponíveis para o mesmo propósito (*e.g.*, IQ-TREE, RAxML, PhyML, PAUP*, entre outros), mas considero o GARLI um dos mais versáteis disponível atualmente[1].

O versatilidade de GARLI está na forma como ele é executado. O programa requer um arquivo de configuração – usando a mesma lógica que você já usou quando executou o YBIRÁ – no qual você define os parâmetros de análise, bem como o arquivo de dados e sufixos de saída em um arquivo de configuração que é lido e executado pelo programa. O exemplo abaixo mostra o conteúdo do arquivo de configuração padrão de GARLI:

```
[general]
datafname = arquivo_de_dados.nex
ofprefix = minha_analise
streefname = stepwise
attachmentspertaxon = 10
constraintfile = none
searchreps = 100
outgroup = 1

outputeachbettertopology = 0
outputcurrentbesttopology = 0

enforcetermconditions = 1
genthreshfortopoterm = 5000
scorethreshforterm = 0.001
significanttopochange = 0.01

writecheckpoints = 0
restart = 0

randseed = -1
availablememory = 2000
logevery = 10
saveevery = 100
refinestart = 1
outputphyliptree = 0
outputmostlyuselessfiles = 0
collapsebranches = 1

#linkmodels means to use a single set of model parameters for all subsets.
linkmodels = 1
```

[1] Em comparação ao IQ-TREE, GARLI parece explorar mais adequadamente o espaço de topologias, o que pode ser desejável em alguns casos. No entanto, IQ-TREE permite explorar modelos mais complexos de substituição e particionamento de dados e apresentar tempo computacional menor. Caso tenham interesse por esse critério de otimização, sugiro a consulta da documentação do IQ-TREE.

Creative Commons Attribution-ShareAlike 4.0 International License

```
# subsetspecificrates means to infer overall rate multipliers for each data subset.
# This is equivalent to *prset ratepr=variable* in MrBayes.
subsetspecificrates = 1
# set for single model, different subset rates (like site-specific rates model in PAUP*)

[model0]
#JC
#number of free parameters: 0
datatype = nucleotide
ratematrix = (a a a a a a)
statefrequencies = equal
ratehetmodel = none
numratecats = 1
invariantsites = none

[master]
bootstrapreps = 0

nindivs = 4
holdover = 1
selectionintensity = 0.5
holdoverpenalty = 0
stopgen = 5000000
stoptime = 5000000

startoptprec = 0.5
minoptprec = 0.01
numberofprecreductions = 2
treerejectionthreshold = 20.0
topoweight = 0.01
modweight = 0.002
brlenweight = 0.002
randnniweight = 0.1
randsprweight = 0.3
limsprweight =  0.6
intervallength = 100
intervalstostore = 5

limsprrange = 6
meanbrlenmuts = 5
gammashapebrlen = 1000
gammashapemodel = 1000
uniqueswapbias = 0.1
distanceswapbias = 1.0

resampleproportion = 1.0
inferinternalstateprobs = 0
```

Este arquivo de configuração possui os parâmetros básicos de uma análise que considera um único modelo de substituição para sequências nucleotídicas. Os detalhes sobre cada uma dessas linhas não serão apresentados neste tutorial e o leitor deve consultar a documentação do programa para maiores detalhes. Para os propósitos deste tutorial, iremos comentar apenas algumas linhas, justamente aquelas que são frequentemente modificadas à cada análise:

i. Todos os arquivos de configuração do GARLI são divididos em três seções, `[general]`, `[model#]` e `[master]`.

ii. O comando "`datafname`", linha 2, especifica o arquivo de entrada. O GARLI aceita

Creative Commons Attribution-ShareAlike 4.0 International License

arquivos de dados no formato FASTA ou NEXUS (formato de PAUP*). No entanto, você verá que para algumas análises é necessário o uso do segundo formato. Neste caso, o uso do `sequencematrix` é recomendável, uma vez que esse programa tem uma opção para exportar seus dados o formato simplificado de NEXUS compatível com o GARLI (veja Tutorial 12).

iii. O comando "`ofprefix`", linha 3, define o prefixo dos arquivos que o GARLI escreverá no diretório de execução. Use um nome curto que lhe informe a análise que está fazendo.

iv. O comando "`searchreps`", linha 7, define o número de réplicas que você deverá fazer em sua análise.

v. Linhas 30 a 34 especificam como os modelos serão serão considerados pelo GARLI. Neste arquivo de configuração, GARLI irá considerar um único modelo para a partição com uma única taxa para o modelo.

vi. Linhas 36 a 44, especifica o modelo de substituição a ser utlizado; nesse caso o modelo JC69 [2]. Os modelos disponíveis em GARLI estão no arquivo `garli_models.txt` no diretório deste tutorial. O modelo deve ser selecionado anteriormente, veja seção 12.4.4 do Tutorial 12.

13.1.1 GARLI: SINGLE MODEL

Iremos iniciar nossas análises em GARLI considerando um único modelo para os dados, independentemente se eles contém uma ou mais partições. Uma vez editado o arquivo de configuração do GARLI, para iniciar a análise execute:

```
alan@turing:~$ garli garli.conf
```

Por *default*, o GARLI espera encontrar no diretório de execução um arquivo chamado "`garli.conf`". Se esse arguivo existe, basta executar:

```
alan@turing:~$ garli
```

No entanto, exceto em execuções paralelas, ou seja, utilizando computadores de auto desempenho ou multi-processados, é recomendável atribuir nomes distintos para os arquivos de configuração. No diretório deste tutorial por exemplo, há o arquivo `garli_single.conf`. A execução deste arquivo seria:

```
alan@turing:~$ garli garli_single.conf
```

Exercicio 13.1

Execute a linha de comando acima.

Creative Commons Attribution-ShareAlike 4.0 International License

A execução desta análise gera os seguintes arquivos de saida:

```
-rw-rw-r- 1 alan alan     4357 Jun  5 15:36 part1.best.all.tre
-rw-rw-r- 1 alan alan      766 Jun  5 15:36 part1.best.tre
-rw-rw-r- 1 alan alan   325906 Jun  5 15:36 part1.log00.log
-rw-rw-r- 1 alan alan   108822 Jun  5 15:36 part1.screen.log
```

O conteúdo destes arquivos são os seguintes:

1. `part1.best.all.tre`: Contém as topologias ótimas encontradas em cada uma das réplicas executadas.
2. `part1.best.tre`: Contém as melhores topologias encontradas na réplica que resultou na melhor topologia.
3. `part1.log00.log`: Contém o *log* da execução.
4. `part1.screen.log`: Contém o `sdt.err` do GARLI, ou seja, as informações de saida que o programa escreve durante a análise.

Este último arquivo possui algumas informações que devemos chamar sua atenção. Se você executar a linha de comando:

```
alan@turing:~$ head -n 50 part1.screen.log
```

Você deverá obter:

```
################################################
PARTITIONING OF DATA AND MODELS
GARLI data subset 1
CHARACTERS block #1 ("Untitled DATA Block 1")
Data read as Nucleotide data,
modeled as Nucleotide data
Summary of data:
       10 sequences.
       62 constant characters.
       48 parsimony-informative characters.
       10 uninformative variable characters.
       120 total characters.
       61 unique patterns in compressed data matrix.
Pattern processing required < 1 second
################################################
```

Os dados acima sumariam as informações de sua matriz de caracteres. Observe que dos 120 caracteres presentes em sua matriz, apenas 61 deles foram condiderados durante a análise. Se você executar a seguinte linha de comando:

Creative Commons Attribution-ShareAlike 4.0 International License

```
alan@turing:~$ tail -n 50 part1.screen.log
```

Você deverá obter:

```
Completed 10 replicate search(es) (of 10).

NOTE: Unless the following output indicates that search replicates found the
same topology, you should assume that they found different topologies.
Results:
Replicate 1 : -685.0223
Replicate 2 : -689.6789
Replicate 3 : -689.6789        (same topology as 2)
Replicate 4 : -686.7873
Replicate 5 : -684.7616        (same topology as 4)
Replicate 6 : -684.7616        (same topology as 4)
Replicate 7 : -689.6789        (same topology as 2)
Replicate 8 : -684.7039 (best)
Replicate 9 : -685.0222        (same topology as 1)
Replicate 10 : -689.6789       (same topology as 2)

Parameter estimates across search replicates:
```

	r(AC)	r(AG)	r(AT)	r(CG)	r(CT)	r(GT)	pi(A)	pi(C)	pi(G)	pi(T)	alpha	pinv
rep 1:	1	46.32	1	1	46.32	1	0.066	0.039	0.330	0.565	0.175	0.355
rep 2:	1	8.028	1	1	8.028	1	0.071	0.040	0.323	0.566	0.159	0.000
rep 3:	1	8.032	1	1	8.032	1	0.071	0.040	0.323	0.566	0.159	0.000
rep 4:	1	12.68	1	1	12.68	1	0.068	0.040	0.328	0.565	0.292	0.363
rep 5:	1	30.39	1	1	30.39	1	0.066	0.039	0.330	0.566	0.076	0.043
rep 6:	1	30.41	1	1	30.41	1	0.066	0.039	0.330	0.566	0.076	0.043
rep 7:	1	8.036	1	1	8.036	1	0.071	0.040	0.323	0.566	0.159	0.000
rep 8:	1	35.09	1	1	35.09	1	0.066	0.039	0.330	0.566	0.073	0.040
rep 9:	1	46.43	1	1	46.43	1	0.066	0.039	0.330	0.565	0.175	0.355
rep 10:	1	8.015	1	1	8.015	1	0.071	0.040	0.323	0.566	0.159	0.000

```
Treelengths:

TL

rep 1:  482.826

rep 2:  8.665

rep 3:  8.673

rep 4:  32.616

rep 5:  544.587

rep 6:  544.992

rep 7:  8.669

rep 8:  707.949

rep 9:  483.584

rep10:  8.662

Saving final trees from all search reps to part1.best.all.tre

Saving final tree from best search rep (#8) to part1.best.tre
```

Creative Commons Attribution-ShareAlike 4.0 International License

Estas linhas sumarizam seus resultados após 10 réplicas, dentre as quais a réplica 8[2] resultou na melhor solução com lnL -684.7039. Este arquivo também informa as estimativas dos parâmetros em cada uma das réplicas, tais como as taxas das diversas categorias de subistituição (*i.e.*, `r(AC)`, `r(AG)`, `r(AT)` e etc), as frequências das bases (*i.e.*, `pi(A)`, `pi(C)`, `pi(G)` e `pi(T)`), o parâmetro α da distribuição gama (Γ) e a porcentagem dos sítios invariáveis (I). Essa tabela é resultado dos parâmetros livres do modelo $HKY + I + \Gamma$ adotado na análise (veja arquivo de configuração).

Exercicio 13.2

Neste exercício você deverá fazer uma análise sob o critério de verossimilhança máxima em GARLI para os arquivos `partition1+partition2aln1.nex`, `partition1+partition2aln2.nex` e `partition1+partition2aln2.nex` de acordo com os modelos que você selecionou para seus respectivos dados concatenados no Tutorial 12. Para executar esse exercício você deverá:

1. Editar três os arquivos de configuração de GARLI, um para cada arquivo de dados.
2. Executar o GARLI para cada um dos arquivos de configuração.
3. Sumarizar seus resultados na Tabela 13.1, no qual n é o número de caracteres em seu alinhamento, k é o número de parâmetros livres[3] e o AIC_c deve ser computado de acordo com as equações 12.21 e 12.22 do Tutorial 12[4].

Tabela 13.1: Análise de modelo único em GARLI.

Dataset	lnL	n	k	AIC_c	Model
`partition1+partition2aln1.nex`					
`partition1+partition2aln2.nex`					
`partition1+partition2anl3.nex`					

13.1.2 GARLI: PARTITION MODEL

O GARLI permite que diferentes modelos sejam aplicados à distintas partições de dados. Embora isso seja mais uma das características que conferem versatilidade ao programa, considere que nem sempre – dentro dos critérios de seleção de modelos em estimativas de verossimilhança máxima – a adoção de diferentes modelos individuais confere melhores índices à sua análise. Este tutorial

[2] Seus resultados podem não ser idênticos a este, pois isso depende de aleatorização, mas devem obedecer a mesma lógica.
[3] Lembre-se que os parâmetros estimados pelo modelo compreende, a topologia, os comprimentos de cada um dos ramos desta topologia – cujo número é dado pela função $(2 * t) - 3$, no qual t é o número de terminais), e os parâmetros livres do modelo de substituição.
[4] O script `calculate_aicc.py` pode ser utilizado para obter essa métrica, basta executar *script*.

Creative Commons Attribution-ShareAlike 4.0 International License

irá explorar a implementação dessas análises de forma superficial. Maiores detalhes sobre esse tipo de análise podem ser obtidos em GARLI partition model documentation[5].

Neste tipo de análise, o GARLI requer que os dados estejam no formato NEXUS. Isso é necessário, pois estes arquivos deverão conter um bloco de instrução definindo as partições. Considere por exemplo o arquivo `partition1+partition2aln1.nex`. Ele é a junção dos arquivos `partition1.fas` + `partition2aln1.fas` do Tutorial 12. O primeiro arquivo possui 120 caracteres, ao passo que o alinhamentdo do segundo resultou em 99 caracteres. Portando, no arquivo concatenado `partition1+partition2aln1.nex`, os caracteres 1 a 120 referem-se à primeira partição, ao passo que os caracteres 121 a 219 referem-se à segunda partição. Sendo este o caso, a primeira etapa de preparação para a análise requer a edição do arquivo `partition1+partition2aln1_garli.nex` para que contenha as seguintes linhas:

Arquivo texto 13.1: partition1+partition2aln1_garli.nexlabel

```
BEGIN SETS;
        CHARSET p1 = 1-120;
        CHARSET p2 = 121-219;
        charpartition partitions = s1:p1, s2:p2;
END;
```

As linhas 1 e 4 são usadas para abrir e fechar o bloco que define as partições, respectivamente. São definidos dois conjuntos de caracteres, `p1` e `p2`, o primeiro de 1 a 120 e o segundo de 121 a 219, linhas 2 e 3. Na linha 4, definimos os `submodels` de GARLI, `s1` e `s2`, para as partições `p1` e `p2`, respectivamente. Note que GARLI usa sequencialmente `s1`, `s2`, `s3` e etc para definir os `submodels`. No entanto, no arquivo de configuração, a sequência de modelos inicia-se no 0; portanto, `[model0]`, `[model1]`, `[model2]` e etc. Este arquivo foi editado e renomeado como `partition1+partition2aln1_garli.nex`

Vejamos como ficaria o arquivo de configuração para este arquivo. Considere que, ao selecionar o modelo para os arquivos `partition1.fas` e `partition2aln1.fas`, o jModelTest selecionou o modelo de acordo com o AIC_c. Os modelos selecionados foram $HKY + I + \Gamma$ e $HKY + \Gamma$, respectivamente. Desta forma, esses modelos foram implementados no arquivo `garli_p1+2aln1.conf` – verifique o conteúdo deste arquivo. Observe que além de especificar dois modelos, as linhas 34 a 38 deste arquivo foram modificadas para que o programa considere modelos distintos para cada uma das partições.

Os resultados desta análise estão no diretório do tutorial. A topologia seleciona, com lnL -1475.1010, foi recuperada na réplica #4 e está escrita no arquivo `results_p1+p2aln1.best.tre`. Considere que o particionamento dos modelos confere a essa análise um novo atributo: seu modelo inclui dois modelos de substituição distintos. Para avaliar esse modelo, é necessário calcular o AIC_c considerando esses dois modelos de substituição. Recorde que o AIC_c é calculado pela seguinte equação:

[5] Veja https://molevol.mbl.edu/index.php/Garli_using_partitioned_models

Creative Commons Attribution-ShareAlike 4.0 International License

$$AIC_c = (-2l + 2k) + \frac{(2k(k-1))}{(n-k-1)}$$

Nesta equação, k é o termo que talvez seja menos intuitivo. Como já foi definido, o valor de k é a soma de todos os parâmetros estimados pelo modelo. Ele inclui, a topologia (1), os comprimentos dos ramos ($(2 * t) - 3$, onde t é o número de terminais; portando, $(2 * 10) - 3 = 17$), e o número de parâmetros livres dos modelos de substituição ($6 + 5 = 11$; veja comentários nos modelos implementados no arquivo `garli_p1+2aln1.conf`). Portanto, o valor de k para esta análise seria $1 + 17 + 11 = 29$.

Calculado o valor de k, o cálculo do AIC_c é feito da seguinte maneira:

$$AIC_c(p1 + 2aln1) = (-2 * -1475.1010 + 2 * 29) + \frac{(2*29(29-1))}{(219-29-1)}$$

$$AIC_c(p1 + 2aln1) = (2950.202 + 58) + \frac{1624}{192}$$

$$AIC_c(p1 + 2aln1) = 3008 + 8.4583 = 3016.6603$$

Exercicio 13.3

Neste exercício você deverá fazer uma análise sob o critério de verossimilhança máxima em GARLI para os arquivos `partition1+partition2aln2.nex` e `partition1+partition2aln3.nex` assumindo modelos distintos para cada partições de acordo com os modelos que você selecionou os conjuntos de dados individuais no Tutorial 12. Para execurar esse exercício você deverá:

1. Editar os arquivos de dados para implementar os blocos de partições.
2. Editar dois os arquivos de configuração de GARLI, um para cada arquivo de dados.
3. Executar o GARLI para cada um dos arquivos de configuração.
4. Sumarizar seus resultados na Tabela 13.2.
5. Responder as perguntas abaixo:

Tabela 13.2: Análise de modelo particionado em GARLI.

Dataset	lnL	n	k	AIC_c	Model
`partition1+partition2aln1.nex`	-1475.1010	219	29	3016.6603	$HKY + I + \Gamma / HKY + \Gamma$
`partition1+partition2aln2.nex`					
`partition1+partition2anl3.nex`					

Creative Commons Attribution-ShareAlike 4.0 International License

1. De acordo com o critério de AIC_c, qual análise você selecionaria? Justifique.

2. Seu resultado é dependente do critério de otimalidade? Justifique.

13.2 Verrossimilhança: homologia dinâmica em POY

Para dados sujeitos a alinhamento[6], a implementação de análises de homologia dinâmica também é possível. O programa POY permite a implementação dessas análises. A análise dos dados moleculares sob verossimilhança em POY, considerando dados pré-alinhados (homologia estática) ou não (homologia dinâmica), pode ser significativamente mais demorada do que análises sob o critério de otimalidade de parcimônia. Portanto, é necessário adotar algumas estratégias que visem reduzir o tempo computacional sob verossimilhança. Uma das estratégias mais efetivas – embora com algumas deficiências – é iniciar a análise com parâmetros de busca menos complexos, adotando até mesmo parcimônia como critério de otimalidade inicial, e coletar topologias que seriam refinadas sob o critério de verossimilhança no qual se adota procedimentos mais complexos de cômputo até que se atinja o nível de agressividade de busca heurística desejado. Veja abaixo alguns componentes – ou procedimentos – que podem ser manipulados durante sua análise dentro do contexto exposto acima:

i. **Topologias candidatas sob o critério de parcimônia:**

Esta etapa da análise começa com a construção e busca inicial (de complexidade arbitrária) sob parcimônia antes de adotar o critério de verossimilhança na escolha de modelos e diagnose de topologias. Este procedimento economiza tempo, evitando RAS e refinamento (*i.e.*, TBR, SPR, *tree-fusing*, entre outros) sob o critério de verossimilhança – o que demanda enorme quantidade de cálculo. Um problema potencial desta estratégia heurística é que uma ou mais topologia considerada ótima sob o critério de parcimônia pode estar longe no espaço de árvores de uma topologia considerada ótima sob o critério de verossimilhança. Por esse motivo, é recomendável que sejam adotadas estratégias de refinamento (via algoritmos de rearranjo – *e.g.*, TBR – ou até mesmo algoritmos genéticos – *e.g., tree-fusing*) após a adoção do critério de verossimilhança.

ii. **Estimativa aproximada (grosseira) de parâmetros:**

A granularidade da estimativa dos parâmetros do modelo pode ser manipulada durante a análise. Por exemplo, durante os rearranjos sob o critério de verossimilhança, todos os

[6]Texto baseado na documentação de POY [3]

Creative Commons Attribution-ShareAlike 4.0 International License

comprimentos de ramos (independentemente da distância entre a quebra e a reconexão) são re-otimizados, o mesmo ocorrendo para os parâmetros do modelo a cada rearranjo. O tempo de execução pode ser minimizado executando a otimização do modelo somente se o custo de um rearranjo está dentro de um número limite em comparação ao melhor custo encontrado até o momento, ou pela otimização do comprimento de ramos em determinadas condições – distância da quebra e da reconexão em cada rearranjo. Esta estratégia é adotada, por exemplo, pelo programa RAxML - Randomized Axelerated Maximum Likelihood [4] que inicia as buscas heurísticas adotando o modelo GTRCAT [7] e durante o processo de refinamento adota o modelo GTRGAMMA que otimiza mais agressivamente os parâmetros do modelo.

iii. **Granularidade na precisão de cálculos (*Floating point granularity*):**

A precisão das casas decimais pode ser diminuida durante os cálculos para limitar o tempo gasto em otimizar os valores dos parâmetros durante os rearranjos ou até mesmo durante a transformação dos caracteres em verossimilhança. A escolha da granularidade atua – limitando – na precisão dos cálculos e no número de iterações realizadas durante o processo de estimativa de parâmetros. Um problema potencial desta estratégia heurística é que a redução da precisão (*coarse granularity*) pode afetar negativamente as análises para as quais várias topologias e/ou comprimentos ramos são muito similares, ou até igualmente ótimos existam. Adicionalmente, os índices de verossimilhança – ou *likelihood scores* – sob baixa precisão não são comparáveis com aqueles obtidos em outros programas que adotam o mesmo critério de otimalidade. Neste caso, a otimização completa deve ser realizada na topologia final.

iv. **Cronograma de Otimização:**

Essa estratégia manipula a etapa na qual é aplicado maior rigor na otimização dos parâmetros do modelo. Por exemplo, sob o rearranjo tradicional sob o critério de verossimilhança, todos os comprimentos de ramo (independentemente da distância entre quebra e reconexão) e os parâmetros do modelo são recalculados para cada topologia. Pode-se diminuir o tempo computacional especificando quando tal rigor deve ser aplicado; otimizando os parâmetros do modelo somente se o custo de um rearranjo está dentro de um limite pré-especificado de acordo com a topologia ótima obtida até então, ou otimizando os comprimentos dos ramos dentro de uma distância específica entre quebra e reconexão para cada rearranjo. O problema potencial com esta estratégia heurística é definir a priori quando este rigor deve ser aplicado e é bem provável que estas propriedades são altamente dependente dos dados em mãos.

v. **Estratégias de rearranjo:**

O número de topologias visitadas durante a etapa de refinamento por algoritmos de rearranjo (que variam entre NNI e TBR) pode ser restringido nos estágios iniciais de busca. Esta

[7] Esse é um algoritmo de aproximação, veja http://sco.h-its.org/exelixis/resource/download/NewManual.pdf

Creative Commons Attribution-ShareAlike 4.0 International License

abordagem minimiza o número de cálculos durante os rearranjos, enquanto que o aumento da granularidade restringe o tempo gasto em um determinado cálculo. Uma vez que topologias foram selecionadas e supostamente estejam perto da solução ótima, pode-se proceder com buscas e estimativa de parâmetros mais agressivas. O problema potencial desta estratégia é similar ao uso de topologias iniciais estimadas pelo critério de parcimônia e o usuário pode ficar restrito a um ótimo local limitado pelo espaço explorado durante o refinamento por algoritmos de rearranjo.

Existem outras possibilidades de estratégias heurísticas, incluindo alternância entre os critérios de otimalidade em caracteres estáticos (transformações alternadas entre critérios de verossimilhança e parcimônia para caracteres estáticos), entre as variações de um mesmo critério de otimalidade (entre MPL e MAL), ou entre premissas para caracteres (entre quatro e cinco estados de caráter para um determinado modelo).

13.2.1 Formas de implementação

O POY permite o uso de dois critérios de verossimilhança em inferência filogenética. A primeira delas seria a *Maximum Average Likelihood* (MAL) na análise de dados quantitativos com alfabetos de qualquer tamanho (*i.e.*, sequências nucleotídicas, aminoácios, entre outros) para **sequências alinhadas**[8], considerando *gaps* como dados lacunares (*i.e., missing data*) ou como um quinto estado de caráter. A segunda seria a *Most Parsimonious Likelihood* (MPL) que pode ser aplicada aos mesmos tipos de dados, além de sequências não alinhadas (busca heurística MPL/DO). Adicionalmente, o POY avalia e permite a aplicação de uma série de modelos de substituição (14) além de considerar INDELs (*i.e.*, inserções e deleções) nesses modelos.

13.2.2 Seleção de modelos de verossimilhança em POY

Como discutido no Tutorial 12, este prodedimento define e seleciona o modelo estocástico de evolução (modelo de substituição) que otimiza a função de verossimilhança. POY, como na maioria dos programas que implementa esse critério de otimalidade, aplica um modelo homogêneo que estima as taxas de substituição utilizando uma única matriz de probabilidades ao longo de toda topologia. A adoção do critério de verosimilhança requer a adoção de um modelo de substituição de caracteres. Toda análise de verossimilhança requer justificativa para a adoção de modelos. Estas justificativas residem na aplicação de critérios (*e.g.*, LRT, AIC, AICc, BIC, entre outros) que visam avaliar a relação entre nível de parametrização do modelo e a otimização da função de verossimilhança [veja breve discussão e referências em 5; Tutorial 12].

Os modelos de substituição disponíveis em POY incluem JC69/Neyman, F81, K2P/K80, F84, HKY85, TN93, e GTR, todos sob os quais pode-se ainda implementar categorias de taxa heterogêneas de substituição com a distribuição Gama (Γ) (Figura 13.1). POY não considera a

[8] MAL pode ser aplicado em sequências não alinhadas, porém requer que os caracteres sejam transformados inicialmente em caracteres estáticos, veja abaixo. O cálculo de MAL sob homologia dinâmica seria computacionalmente impraticável (Wheeler, pers. comm.).

Creative Commons Attribution-ShareAlike 4.0 International License

proporção de sítios ivariáveis (I) como um parâmetro do modelo, uma vez que ele tecnicamente estaria contemplado na distribuição Gama (Γ). A seleção de modelos pode feita de forma automatizada, cabendo ao usuário a escolha de um critério dentre os três disponíveis em POY: AIC, AICc and BIC [para a definição de BIC, veja 5].

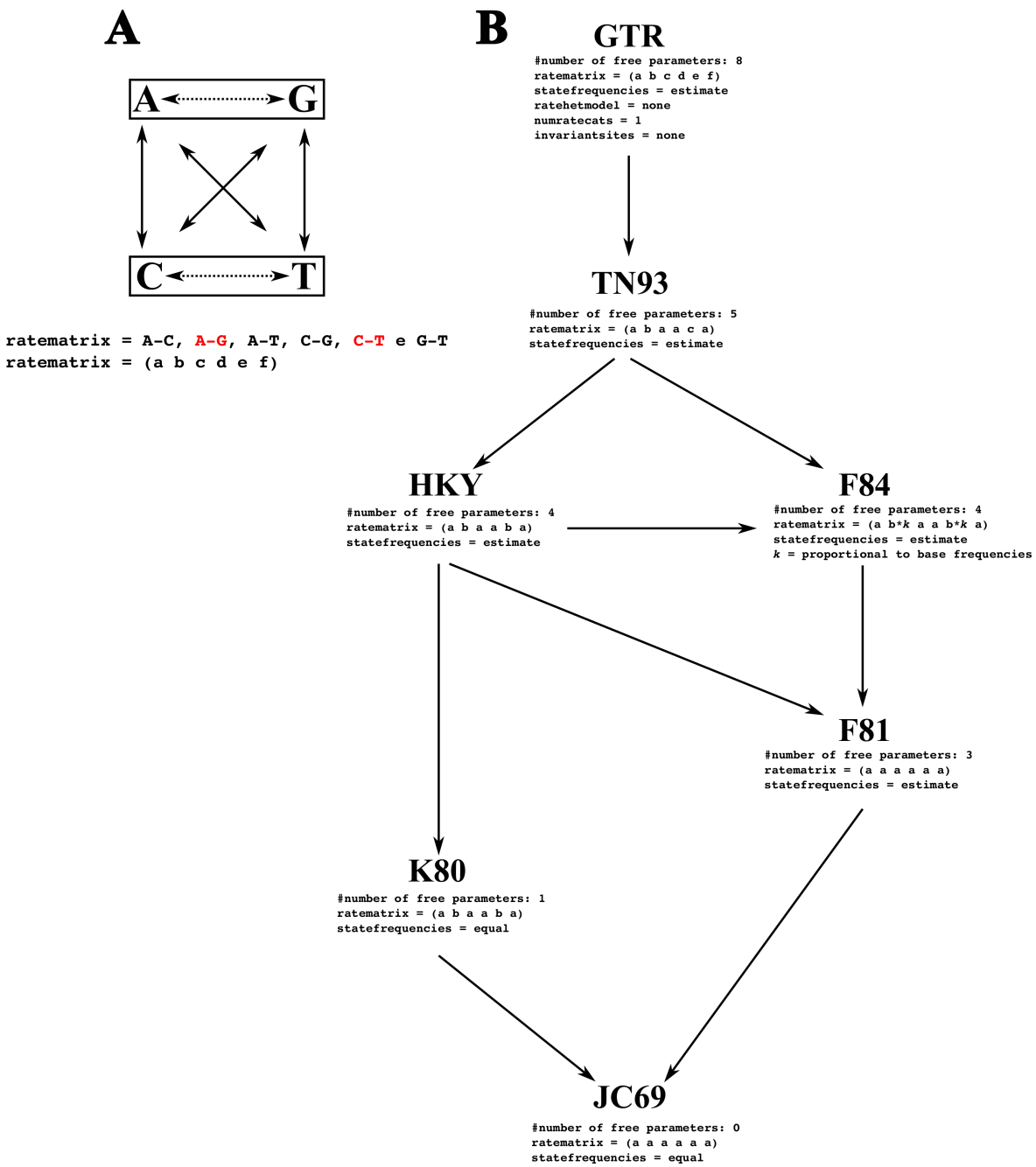

Figura 13.1: Modelos de substituição avaliados em POY. **A**. Seis substituições possíveis para nucleotídeos. Tranformações pontinhadas referem-se a transições, demais transformações são transversões. Estas seis transformações define os elementos das matrizes de substituições (`ratematrix` em Garli), cujas transformações em vermelho representam transições. A representação tradicional dessas transformações são expressas pelas letras *a–f*. **B**. Relação hierárquica dos modelos de substituição avaliados em POY. Para cada modelo é informado o número de parâmetros livres (*i.e.*, aqueles que são estimados durante a otimização da função de verossimilhança). Considere que para cada um desses modelos, POY considera taxas heterogêneas de substituição de acordo com 0 a 8 categorias da distribuição Gama (Γ).[9]

Creative Commons Attribution-ShareAlike 4.0 International License

Arquivo texto 13.2: `model_test_part1.poy`.

```
1 (*MAL Analysis: Partitions and Model Selection*)
2 read(prealigned:("partition1.fas",tcm:(1,1)))
3 build(100)
4 swap()
5 select(best:1)
6 transform(likelihood:(aicc:"part1_AICc.txt",rates:gamma:(4), priors: estimate,mal))
7 swap()
8 report("part1_LK.tre",trees:(branches),"part1_LK_lkm.txt",lkmodel)
9 exit()
```

A forma de implementação da seleção de modelos em POY é relativamente simples. Considere, por exemplo o *script* 13.2. Na linha 1, o *script* lê os dados do arquivo `partition1.fas` e atribui peso 1 para todas as transformações. Posteriormente, linha 2 e 3, ele faz 100 RAS+TBR, cuja a melhor topologia é selecionada na linha 5. Observem que as buscas e seleção foram feitas aqui sob o critério de parcimônia. A linha 6 transforma o critério de otimalidade para verossimilhança, mais precisamente *Maxinum Average Likelihood* (veja abaixo), e inicia o teste de modelos adotando AIC_c como critério de seleção. O modelo de verossimilhança considera 4 categorias de taxas de substituições, distribuição Gamma (Γ), e os parâmetros livres dos modelos deverão ser estimados durante a análise. Posteriormente, o POY adota o melhor modelo sob esse critério de otimalidade e faz uma busca por `swap` cujos resultados, topologias e modelo, são impressos em seus respectivos arquivos de saida. Observe que esse *script* executa a seleção de modelo e faz buscas – embora muito heurística – em uma única análise.

A execução do `model_test_part1.poy` resulta em 3 arquivos. O arquivo `part1_AICc.txt` contém uma tabela comparativa dos modelos analisados cujo resultado é apresentado na Tabela 13.3. O arquivo `part1_LK_lkm.txt` detalha o modelo de substituição resultado da busca. Finalmente, o arquivo `part1_LK.tre` contém a topologia com seus respectivos comprimentos de ramos resultado da análise.

Arquivo texto 13.3: `model_test_part2a1mi.poy`.

```
1  (*ML Analysis: Partitions and Model Selection*)
2  read("partition2a1.fas")
3  transform(tcm:(1,1))
4  search(max_time:0:0:02)
5  select(best:1)
6  transform(static_approx)
7  transform(likelihood:(aicc:"part2a1mi_AICc.txt",rates:gamma:(4), priors:estimate, gap:missing, mal))
8  swap()
9  report("part2a1mi_LK.tre",trees:(branches),"part2a1mi_LK_lkm.txt",lkmodel)
10 exit()
```

No exemplo anterior, os dados iniciais encontravam-se pré-alinhados. O *script* 13.3 executa as mesmas análise do *script* anterior para dados não alinhados. Se compararmos os dois *scripts* notaremos que o segundo (*script* 13.3) lê e transforma os dados iniciais de forma diferente (linhas 2 e 3). Posteriormente, na linha 4, ele faz uma busca pelo comando `search` por dois minutos e seleciona uma árvore ótima, linha 5. Na linha 7, os caracteres que até então eram tratados como homologia dinânica são transformados em caracteres estáticos (`transform(static_approx)`) – essa transformação é necessária pois o POY só faz MAL

Creative Commons Attribution-ShareAlike 4.0 International License

sob homologia estática. Em seguida, POY inicia a seleção de modelos como no *script* anterior, mas considerando gaps como dados lacunares (`gap:missing`) – da mesma forma que o GARLI. O POY permite três tratamentos diferentes para INDELs. Eles podem ser tratados como "*missing data*" e como tal não influenciam o cálculo de verossimilhança; podem ser tratados como caracteres (`gap:character`), neste caso inserções e deleções de A, C, G e T são tratadas individualmente como diferentes eventos e estimados independentemente; e finalmente, eles podem ser tratados de maneira conjugada (`gap:coupled`), e neste caso INDELs como um todo são tratados como um único parâmetro.

Tabela 13.3: Exemplo de arquivo de saída de POY para a seleção de modelos utilizando AIC_c mostrando os valores de verosimilhança para cada modelo avaliado. Colunas representam, modelo avaliado, *log* negativo da verossimilhança ($-L$), número de parâmetros estimados – inclui parâmetros livres do modelo + número de comprimentos de ramos (= $2t-3$, onde t é o número de terminais) + topologia – (K), número de caracteres (n), valores de AIC_c, diferença entre valores de AIC_c (Δ), pesos de Akaike (ω) e peso acumulado de Akaike (Cum(ω)). Neste exemplo, o modelo HKY+Γ possui o maior valor teórico de informação e seria o modelo selecionado.

Modelo	$-lnL$	K	n	AIC_c	$\Delta\ AIC_c$	ω	**Cum**(ω)
HKY+Γ	699.988278259	22	120	1454.40954621	0.	0.83608261736	0.83608261736
F84+Γ	701.73534071	22	120	1457.90367111	3.49412490153	0.145716795637	0.981799412997
TN93+Γ	702.530154326	23	120	1462.56030865	8.15076244343	0.0142014803771	0.996000893374
F81+Γ	707.445080336	21	120	1466.3187321	11.9091858917	0.00216871426416	0.998169607638
GTR+Γ	699.780583091	26	120	1466.65794038	12.2483941669	0.00183039236205	1.
F81	803.540920474	20	120	1655.56668943	201.157143224	1.74393601054e-44	1.
F84	812.286360077	21	120	1676.00129158	221.591745375	6.37108014702e-49	1.
K81+Γ	836.480712188	19	120	1718.56142438	264.151878168	3.65088083617e-58	1.
JC69+Γ	842.281419936	18	120	1727.3351171	272.925570891	4.54165865543e-60	1.
TN93	870.744628139	22	120	1795.92224597	341.512699759	5.80374993199e-75	1.
HKY	875.31326933	21	120	1802.05511009	347.64556388	2.70379754075e-76	1.
GTR	903.181048846	25	120	1870.19188493	415.782338718	4.32774342593e-91	1.
K81	944.292608696	18	120	1931.35749462	476.947948412	2.26109140474e-104	1.
JC69	955.127444482	17	120	1950.25488896	495.845342756	1.78156255514e-108	1.

Exercicio 13.4

Neste exercício você deverá fazer uma análise sob o critério de verossimilhança máxima em POY para o arquivo `partition2aln1.fas` que você gerou no Tutorial 12 implementando os três tratamentos de gaps possíveis no programa. Para executar esse exercício você deverá:

Creative Commons Attribution-ShareAlike 4.0 International License

Tabela 13.4: Análise de modelo em POY assumindo diferentes modelos para INDELs. Valores de k podem ser obtidos na Figura 13.1.

Script	lnL	n	k	AIC_c	Model
`model_test_part2a1ch.poy`					
`model_test_part2a1co.poy`					
`model_test_part2a1mi.poy`					

1. Executar os *scripts* `model_test_part2a1ch.poy`, `model_test_part2a1co.poy` e `model_test_part2a1mi.poy`.
2. Sumarizar seus resultados na Tabela 13.4.
3. Resposder as pertuntas abaixo:

1. De acordo com o critério de AIC_c, o tratamento de gaps influencia os índices de verossimilhança? Justifique.

2. Os direrentes tratamentos de gaps geram topologias distintas? Justifique.

3. Qual desses resultados será comparável com a análise que você fez utilizando o GARLI? Como elas se comparam?

13.2.3 **MAL**: *Maximum Average Likelihood* EM POY

Este componente do tutorial apresenta a análise de caracteres estáticos pelo critério de verossimilhança média máxima (MAL) e explora algumas estratégias que podem ser usadas para otimizar tempo de execução em POY. Esta análise é similar em intensidade às buscas feitas pelo programa PhyML [6]. Análises totalmente executadas sob este critério, como por exemplo em PAUP* [7] no qual as buscas (RAS+SWAP) são calculadas por verossimilhança, podem ser computacionalmente muito intensas. Desta forma, o exemplo abaixo, *script* 13.4, ilustra elementos de busca sob outro critério, parcimônia, na etapa de construção das topologias e posterior refinamento sob verossimilhança.

Creative Commons Attribution-ShareAlike 4.0 International License

Arquivo texto 13.4: mal_part1+2a1.poy.

```
1  read(prealigned:("partition1.fas",tcm:(1,1)))
2  read("partition2a1.fas")
3  transform(names:("partition2a1.fas"),(tcm:(1,1)))
4  set(root:"Taxon1")
5  search(max_time:00:00:01)
6  select()
7  transform(static_approx)
8  set(opt:coarse)
9  transform(likelihood:(tn93,rates:gamma:(4),priors:estimate,gap:coupled,mal))
10 swap(all:5,spr,optimize:(model:never,branch:never))
11 fuse(optimize:(model:never,branch:join_region))
12 select(best:1)
13 set(opt:exhaustive)
14 report("mal_p1+2a1.tre",trees:(branches),"mal_lkm_p1+2a1.txt",lkmodel)
15 exit()
```

No *script* 13.4, as três primeiras linhas tem a função de ler o arquivo pré-alinhado `partition1.fas` – pré-alinhado – e o arquivo `partition1.fas` – não-alinhado – atribuindo custos iguais as transformações. Posteriormente, o *script* implementa uma busca pelo comando `search` por um minuto e seleciona a(s) melhor(es) e única(s) topologia(s) desta análise de parcimônia (veja 13.2*iiii*). Após a seleção, os caracteres dinâmicos são transformados em caracteres estáticos (linha 7). Na linha 8 deste *script* é implementada a redução de precisão das casas decimais (*i.e., floating points*) durante as otimizações visando aumentar eficiência computacional (veja 13.2 *iiiiii*). Subsequentemente, linha 9, é implementado o critério de verossimilhança por MAL (`likelihood(...,mal)` sob o modelo de substituição `tn93`[10] considerando 4 categorias de taxas de substituição da distribuição Gama (`rates:gamma:(4)`) na qual a frequência das bases será estimada diretamente dos dados (`priors:estimate`) e os *gaps* serão considerados como quinto estado de caráter (`gap:coupled`) e tratados como um único parâmetro. Nas linhas 10 e 11 deste *script*, as estratégias de refinamento são implementadas sob a(s) topologia(s) coleta(s) durante a análise de parcimônia (veja 13.2*iiiv*). No entanto, observem que durante o *swap* tanto os parâmetros de modelo quando os comprimentos de ramos não serão otimizados (`optimize:(model:never,branch:never)`; veja 13.2*iiiiv*) e a estratégia de *swap* (`all:5,spr,`) é pouco agressiva (SPR – no qual rearranjo só ocorrerá dentro dos 5 ramos do ponto de quebra – ao invés de TBR que visitaria um maior número de topologias; veja 13.2*iiiv*). A etapa de *tree fusing* aumenta o rigor da otimização ao recalcular o comprimento de no máximo cinco ramos (`branch:join_region`, veja 13.2*iiiiv*). Após estas etapas de refinamento, uma única topologia ótima é selecionada, linha 12 (`select(best:1)`), e a precisão de *default* dos cálculos é re-estabelecida (`set(opt:exhaustive)`; veja 13.2*iiiiii*). Neste momento POY re-otimiza os parâmetro de verossimilhança. Finalmente, POY retorna a topologia encontrada com comprimentos de ramos (`"mal.tre",trees:(branches)`), o modelo de verossimilhança usado (`"mal_lkm.txt",lkmodel`) e termina a execução (`exit()`).

[10] esse modelo foi selecionado anteriormente em POY

Creative Commons Attribution-ShareAlike 4.0 International License

Exercicio 13.5

Neste exercício você deverá fazer uma análise sob o critério de verossimilhança máxima em POY concatenando os dados do arquivos `partition1.fas` com os três alinhamentos feitos no Tutorial 12 para as sequências no arquivo `partition2.fas` utilizando o *script* 13.4 e as modificações do mesmo. Os resultados destas análises deverão ser compilados na Tabela 13.5 – na qual você encontra os modelos que deverão ser assumidos os quais foram selecionados anteriormente. Após a análise, responda as seguintes perguntas:

Tabela 13.5: Análise sob Maximum Average Likelihood considerando diferentes alinhamentos.

Data	lnL	n	k	AIC_c	Model
`partition1+partition2aln1.nex`					TN93
`partition1+partition2aln2.nex`					TN93
`partition1+partition2aln3.nex`					TN93

1. De acordo com o critério de AIC_c, Qual análise você selecionaria? Justifique.

Exercicio 13.6

Os resultados do Exercício 5 não podem ser comparados com os resultados que você obteve analisando os mesmo dados em GARLI. Neste exercício, você deverá modificar os arquivos do exercício anterior para que esses resultados possam ser comparados. Após a análise, compile seus dados na Tabela 13.6, responda a seguinte pergunta:

Tabela 13.6: Comparação das análises de Maximum Average Likelihood em POY e GARLI.

Data	lnL	n	k	AIC_c	Model
`partition1+partition2aln1.nex`					F84
`partition1+partition2aln2.nex`					F84
`partition1+partition2aln3.nex`					F84

Creative Commons Attribution-ShareAlike 4.0 International License

1. Você poderia justificar a adoção de um desses programas em um estudo filogenético. Justifique.

_____

_____

_____

_____

_____

13.2.4 **MPL**: *Maximum Parsimonious Likelihood* EM POY

Como foi apresentado anteriormente, há diferentes sabores de verossimilhança. Este componente do tutorial apresenta a análise de caracteres dinâmicos pelo critério de verossimilhança máxima por parcimônia (MPL ou MPL/DO). Neste tipo de análise, alinhamento e busca de topologias são computados e examinados simultaneamente. Este processo é muito recente e pouco explorado. Tudo indica que dentro deste contexto, dado a complexidade do universo de soluções possíveis, análises heurísticas são insuficientes para obter resultados minimamente próximos da solução ótima. Desta forma, é possível que exceto em casos de banco de dados muito simples, análises dinâmicas sob MPL irão demandar recursos computacionais, incluindo paralelização de processos (*i.e., clusters*), além do que temos disponíveis para o dia a dia de nossas atividades.

Considere o *script* abaixo:

Arquivo texto 13.5: mpl_part1+2a1.poy.

```
1  read(prealigned:("partition1.fas",tcm:(1,1)))
2  read("partition2a1.fas")
3  transform(names:("partition2a1.fas"),(tcm:(1,1)))
4  set(root:"Taxon1")
5  search(max_time:00:00:01)
6  select()
7  set(opt:coarse)
8  transform(likelihood:(f81,rates:gamma:(4),priors:estimate,gap:coupled,mpl))
9  swap(all:5,spr,optimize:(model:never,branch:never))
10 fuse(optimize:(model:never,branch:join_region))
11 select(best:1)
12 set(opt:exhaustive)
13 report("mpl_p1+2a1.tre",trees:(branches),"mpl_lkm_p1+2a1.txt",lkmodel,"mpl_p1+2a1_ia.fas",fasta)
14 exit()
```

Este *script* apresenta a mesma estratégia do *script* 13.4 com algumas diferenças. Primeiro que não é necessária a transformação dos caracteres em homologia estática antes de submeter ao critério de verossimilhança. A transformação no entanto, define que o cálculo deve ser baseado em MPL. O comando `report` inclui como arquivo de saida o alinhamento implícito da análise.

Exercicio 13.7

Neste exercício você deverá executar o *script* `mpl_part1+2a1.poy`. Após a execução, examine os arquivos de saida e responda as seguintes perguntas:

Creative Commons Attribution-ShareAlike 4.0 International License

i. As topologias encontradas por MPL e MAL são as mesmas para esses dados? Comente.

ii. hà como comparar estas duas formas de Verossimilhança?

13.3 Considerações finais sobre análises de verossimilhança e homologia dinâmica

Há duas coisas a considerar ao optar pela análise de dados reais sob critérios de verossimilhança. A primeira delas é que o uso de homologia dinâmica em análises de verossimilhança é relativamente recente e pouco sabemos sobre o comportamento dessas análises. A segunda delas é a extrema complexidade computacional destes algoritmos tornando impeditivo sua aplicação em computadores convencionais em dados reais. Mesmo nos casos apresentados aqui, há muito espaço para melhorar a agressividade das análises. A leitura do manual do programa aponta alguns caminhos. Desta forma, este tutorial explorou de forma muito breve as estratégias de busca sob o critério de verossimilhança e algumas das estratégias heurísticas que podem ser implementadas com o intuito explorar melhor o espaço de soluções possíveis de análises sob estes critérios. Aos interessados em aplicar o critério de verossimilhança em seus dados, eu recomendaria a leitura atenda da documentação de POY. Nele você encontrará alguns conceitos que foram omitidos neste tutorial – visando simplicidade – e outras estratégias e ferramentas associadas a este tipo de análise.

Outro componente que não foi abordado neste tutorial é a inclusão de dados morfológicos em análises filogenéticas sob critérios de verossimilhança. A documentação de POY possuí um tutorial para este fim e o aluno interessado em conhecer um pouco mais sobre este tema deve consultar esse material.

13.4 Referências

1. Zwickl, D. J. Genetic algorithm approaches for the phylogenetic analysis of large biological sequence datasets under the maximum likelihood criterion. Tese de doutorado (The University of Texas, Austin, 2006).
2. Jukes, T. H. & Cantor, C. R. em *Mammalian protein metabolism.* ed. Munro, H. N. New York: Academic Press, 1969.
3. Varon, A.; Lucaroni, N.; Hong, L. & Wheeler, W. C. 2011–2014. POY version 4: phylogenetic analysis using dynamic homologies, version 5.0. New York, NY: American Museum of Natural History, 2011–2014.

Creative Commons Attribution-ShareAlike 4.0 International License

4. Stamatakis, A. 2014. RAxML Version 8: A tool for phylogenetic analysis and post-analysis of large phylogenies. *Bioinformatics* **doi: 10.1093/bioinformatics/btu033**:

5. Darriba, D. & Posada, D. jModelTest 2.0 v0.1.1. http://code.google.com/p/jmodeltest2/. 2015.

6. Guindon, S *et al.* 2010. New Algorithms and Methods to Estimate Maximum-Likelihood Phylogenies: Assessing the Performance of PhyML 3.0. *Systematic Biology* **56**(3): 307–321.

7. Swofford, D. 2003–2016. PAUP*. Phylogenetic Analysis Using Parsimony (*and Other Methods, Version 4.0a131). Sunderland, Massachusetts: Sinauer Associates, 2003–2016.

Creative Commons Attribution-ShareAlike 4.0 International License