
Laboratorio de
Sistemática &
Biogeografía

Sistemática filogenética

Introducción a la práctica.

Segunda Edición

Daniel Rafael Miranda-Esquivel,

Joan Salvador Arias,

Ivonne Janeth Garzón-Orduña

Versión: 2016.01.01 18 de abril de 2016

Índice general

Introducción general i

1. Selección de caracteres 1

2. Matrices de datos 14

3. Árboles 25

4. Búsquedas mediante parsimonia 36

5. Pesaje de caracteres 57

6. Alineamiento de ADN 66

7. Modelos de evolución 78

8. Máxima verosimilitud 91

9. Consensos **101**

10.Soporte en parsimonia y ML **111**

11.Soporte en ML **124**

12.Análisis Bayesiano **130**

A. Programas de cómputo **139**

A.1. Editores y manejadores de búsqueda 141

A.1.1. WinClada 141

A.1.2. MacClade & Mesquite 144

A.2. Búsqueda de árboles 149

A.2.1. TNT 149

A.2.2. NONA 151

A.2.3. PAUP* 153

A.2.4. MrBayes 155

B. Formatos de archivos **157**

B.1. Formatos de matrices 157

B.1.1. NONA 157

B.1.2. PAUP* 159

B.2. Formatos de árboles 161

B.2.1. NONA 161

B.2.2. PAUP* 162

C. Algunos comandos básicos para POY **164**

C.1. Análisis de sensibilidad con costos diferenciales . . . 164

Introducción general

Este libro ha sido pensado como uno de los tantos soportes posibles para las clases de sistemática de nivel básico y medio, además de servir de repaso a conceptos teóricos generales *pero* sobre la base empírica, y no como **reemplazo** de los libros básicos o avanzados sobre análisis filogenético.

¿Cómo está estructurado este manual?

El libro consta de 12 prácticas que van desde manejo de caracteres, pasando por editores de matrices y árboles a búsquedas tanto para análisis de parsimonia y máxima verosimilitud (ML), como para análisis bayesiano. En todos los casos se presentan las técnicas, la

metodología a seguir, los programas de cómputo a usar (y sus comandos), además de una serie de preguntas sobre la práctica o en general sobre la técnica. La literatura recomendada es una sugerencia de lecturas, desde el punto de vista de los autores, críticas, pero obviamente no cubre todos los artículos posibles; exploraciones constantes de revistas como *Cladistics*, *Systematic Biology*, *Zoologica Scripta* o *Molecular Phylogenetics and Evolution* y similares, actualizarían las perspectivas aquí presentadas. Al final se presenta un capítulo que trata sobre los programas usados, que se espera funcione como una guía rápida para el uso de los mismos pero que no reemplaza al manual.

A lo largo del libro se utiliza una tipografía consistente para indicar el nombre de los programas (v.g., TNT, Component), las instrucciones que deben ser escritas en los programas de línea de comando, por ejemplo:

```
> mult=replic 10;
```

y si se accede las instrucciones mediante un menú o un cuadro de diálogo en los programas de interfaz gráfica (v.g., **Analize/heuristics**).

El orden de las prácticas obedece a la estrategia de enseñanza se-

guido en la UIS pero puede usarse en otras secuencias, por ejemplo, las prácticas caracteres, excluyendo alineamiento, seguido de búsquedas incluida la búsqueda del modelo, consensos, por último soporte y al final discutir la práctica de alineamiento.

Por la misma naturaleza del análisis filogenético, la práctica más larga es la de búsquedas, es factible que todos los ejercicios no puedan completarse en una única sesión de laboratorio.

Usted podrá encontrar material adicional, los datos, algunos macros para los distintos programas y demás chismes en el sitio web del laboratorio de Sistemática & Biogeografía (LSB) de la UIS, en la dirección: ciencias.uis.edu.co/labsist.

Público objeto

Se espera que el usuario de este manual tenga conocimientos básicos, o que esté tomando un curso formal de sistemática filogenética a nivel de pre o posgrado. No se esperan características especiales en cuanto al dominio de computadores, pero el manejo básico de un editor de texto que permita grabar archivos sin formato es

muy recomendable, además de conocer el entorno de línea de comandos, el cual es usado en una gran cantidad de programas. Los usuarios pueden trabajar en cualquier plataforma desde Linux a MacOSX, pasando por Windows; en general se citan los programas apropiados para cada plataforma.

El formato de la bibliografía ha sido modificado para que en la medida de lo posible, muestre los DOI (Digital Object Identifier) para que la búsqueda de la bibliografía básica sea más fácil.

Programas eliminados

Algunos programas han sido eliminados, por ejemplo Piwe, Modeltest o component, y algunos se presentan al mínimo, en particular NONA, dado el poco uso actual. Modeltest y MrModeltest han sido superados por JModeltest y PAUP* en modo nativo, por lo que no es muy factible que se mantengan en uso, solo Modeltest es interesante para aprender sobre los comandos de PAUP*.

Programas adicionados

El área más activa en términos de programas ha sido análisis bajo Verosimilitud y Bayesiano, se han incluido los programas: RaxML/ExaML, PhyML, JModeltest y R; en especial para este último se dan ejemplos detallados para familiarizar al estudiante con la plataforma.

Agradecimientos

A la Universidad Industrial de Santander, en particular a la Escuela de Biología. A los estudiantes del programa de pregrado de la escuela de Biología de la UIS (Universidad Industrial de Santander) estudiantes profundización, a Viviana Romero quien reescribió casi por completo el capítulo de búsquedas.

.

Práctica 1

Selección de caracteres

Introducción

En un análisis filogenético cladístico es necesario identificar y usar caracteres homólogos, de hecho, la importancia de los caracteres es tal que Hennig (1968) se refería a los especímenes estudiados como semaforontes (**o los que llevan caracteres**). Aún a pesar del papel central del concepto de carácter, la definición no es tan fácil.

Richards (2003) señala que el término **carácter** no está bien definido y que su reconocimiento depende en gran parte del entrenamiento del taxónomo (Richards, 2002). Es por eso que la mayoría de las ocasiones se enfatiza en la importancia de un **análisis de caracteres** profundo y concienzudo (ver Hennig (1968); Neff (1986); Rieppel & Kearney (2001)). [1]

Las ideas de carácter y homología usadas aquí son independientes del tipo de caracteres usados, pero es mucho más sencillo presentar la discusión en términos de caracteres morfológicos. Más adelante se tratará directamente el tema de los caracteres moleculares.

Aunque no existe una fórmula mágica para reconocer caracteres homólogos, la principal idea es la similitud "especial", basada en la definición de homología como **similar dada por ancestría común** ((Rieppel & Kearney, 2001); *contra* Kluge (2003); Grant & Kluge (2004)). Por similitud no debe entenderse solo el parecido general, sino que existe una correlación estructural y de posición (la similitud y conjunción en el sentido de Patterson (1981)) del carácter en los taxa comparados.

[1] Este análisis de caracteres también implica revisar las afirmaciones cuantitativas o cualitativas hechas, ver Wiens (2001).

Por ejemplo, los brazos humanos, las alas de los murciélagos y las patas de los caballos presentan una estructura ósea similar, con diversos huesos colocados en las mismas posiciones, y en general el miembro completo está en la misma posición con respecto al cuerpo en los tres taxa, aunque sus formas son muy diferentes.

Otro factor importante a tener en cuenta son los "grados" o niveles de generalidad de homología. Es decir, que una estructura es homóloga con otra en un cierto grado, pero no comparable con ella en otro. Por ejemplo, artrópodos y vertebrados comparten la cefalización y el tener miembros pareados. Comparar directamente esas estructuras no es apropiado, aunque es posible que sean homólogas a nivel molecular, donde los mismos genes controlan el desarrollo de estas estructuras, pero la larga historia independiente de cada linaje hace imposible una comparación de las estructuras como homologías, aunque no prohíbe una comparación funcional.

Es importante recalcar que cuando alguien compara los caracteres de diferentes taxa, hace una inferencia fuerte sobre los caracteres: se trata del mismo carácter en los taxa; es decir, da una identidad histórica al carácter al hacerlo el mismo carácter por homología, aunque estén modificados los distintos estados (Hennig, 1968). Es-

to es lo que le da soporte al uso de la congruencia de caracteres para descubrir las sinapomorfías que le dan identidad histórica a los grupos supraespecíficos.

La selección y observación de caracteres, no solo morfológicos, es un trabajo que sólo se aprende mediante el trabajo continuo con los datos derivados de los especímenes en el laboratorio y con una lectura crítica de la literatura sobre el grupo. Es importante que usted pueda reconocer cuándo los caracteres usados en una descripción, una clave o un análisis filogenético cumplen o no con el requisito de la identidad histórica.

Algunos libros de texto sistemática, presentan distintos criterios de reconocimiento de caracteres (por ejemplo Hennig (1968); Wiley & Lieberman (2011)), aunque no es posible plantear reglas estrictas o situaciones de casos perfectos, eso no es un motivo para suponer que los caracteres morfológicos carecen de base conceptual y por ello deben ser excluidos del análisis (*contra* Scotland et al. (2003)).

Existen algunos criterios básicos aplicables en general a los caracteres: para cada carácter se reconocen diversos estados alternativos de ese carácter, los cuales son llamados estados de carácter o sim-

plemente estados. Es posible que un carácter no sea aplicable a todos los organismos de estudio, pero dentro de los organismos donde es aplicable, los estados de un carácter deben dividir el universo en al menos dos grupos mutuamente excluyentes, cada uno de ellos reconocible por un estado de carácter.

Dado que se tratan los caracteres como identidades históricas, se debe ser cuidadoso con **NO** crear estados de carácter que sirvan como "cajas de basura". Por ejemplo, en el carácter:

Color:

0. rojo
1. otro color

el estado "otro color" es ambiguo y puede contener individuos de colores tales como verde o amarillo, cuya única (falsa) similitud es que no poseen el color rojo, pero no existe una identidad histórica defendible, es decir el NO color rojo se ha generado desde múltiples ancestros. Todos los estados del carácter deben representar una unidad histórica, independientemente de si el estado es el apomórfico o el plesiomórfico. Siempre que encuentre un carácter donde uno de los estados está definido como negación (incluidas las

ausencias), debe revisar cuidadosamente que el estado "negativo" indique claramente una unidad.

Cuando los caracteres solo tienen dos estados, no es necesaria ninguna suposición sobre la "dirección del cambio"; sin embargo, cuando hay más de dos estados, la dirección es una pregunta importante. Algunos caracteres simplemente no dan posibilidad de escoger una dirección particular (por ejemplo, las bases nucleotídicas), por lo cual son caracteres no ordenados. Bajo una posición, uno puede asignar un posible orden a los caracteres que se leen como diversos grados de homología, sin asumir nada sobre el proceso evolutivo, por ejemplo, el carácter:

Pilosidad en la pata:

0. solo en el extremo anterior
1. hasta la mitad
2. toda la superficie
3. toda la pata y con presencia de pelos más largos en el extremo posterior

podría ordenarse desde 0 hasta 3 (hacerlo aditivo) para reflejar que existe un mayor grado de homología entre, por ejemplo, el estado

2 y 3, que 1 y 3. La otra posición no privilegiará ningún cambio con respecto a otro y la dirección se asignaría por la congruencia con otros caracteres (el carácter es no aditivo).

En otros casos, la identidad histórica es difícil de apreciar; por ejemplo:

transparentable con KOH:

0. si
1. no

el carácter como un todo parece ser más un artefacto de laboratorio que una propiedad heredable del organismo. En esa misma línea están caracteres como

número de componentes en PCA:

0. uno
1. dos

Al tratarse los estados de carácter como entradas alternativas, un mismo organismo no puede poseer dos estados o más estados del mismo carácter: para nuestra visión de homología de las alas y los miembros anteriores, no pueden existir ángeles o centauros, con

dos versiones diferentes del miembro anterior en el mismo organismo. Este es el criterio de **c**onjunción de Patterson; sin embargo, hay que tener en cuenta que existen homologías seriadas que **contradicen** esta idea, no solo en organismos segmentados como anélidos y artrópodos, sino en otros donde la segmentación no es clara (e.g., hojas de las plantas, pelos de los mamíferos, etc.). El criterio de conjunción de Patterson es un llamado al reconocimiento de las estructuras que se usan como caracteres en el análisis filogenético.

Técnicas

Use el dibujo al final de la práctica que corresponde a la figura 1 en Sokal (1983). Los "organismos" son caminálculos, seres hipotéticos creados por J. Camin para estudios sobre clasificaciones en fenética[2].

A partir de los dibujos:

1. Elabore una pequeña guía morfológica de los organismos,

[2]Copyright, Society of Systematic Biologists, se reproduce con permiso.

de modo que pueda diferenciar las diferentes estructuras en los diferentes organismos (en buen romance significa poner nombres a sus organismos y ligar tales nombres a estructuras fácilmente reconocibles; el ejercicio más cercano en un análisis para separar por **morfos** sus organismos).

2. Elabore un listado de los diferentes caracteres de los organismos, reconozca al menos 5 caracteres únicos y 5 compartidos. Anote en una hoja separada los organismos y sus estados.
3. Intercambie su listado (acompañado de la guía morfológica y los nombres de los organismos) con un compañero:
   a) Trate de reconocer los caracteres que su compañero describió, y anote cuales organismos poseen esos caracteres.
   b) En caso de ser necesario, re-escriba los caracteres de su compañero e indique la razón de sus cambios.
4. Compare la lista de caracteres por organismos que usted realizó con la que su compañero re-escribió.

Pregunta(s) general(es).

1. **Haga un ejercicio crítico de los caracteres de sus com-**

pañeros:

a) ¿están bien definidos?

b) ¿hay implícita una transformación?

c) ¿usaron de la misma forma la similaridad?

2. Trate de localizar los motivos de sus aciertos y fallas cuando trabajó con los caracteres de su compañero. ¿Son esos motivos consistentes con la crítica a los caracteres?
3. Evalúe la crítica de su compañero a sus caracteres: ¿es posible reescribir los caracteres para mejorarlos? Si así lo cree, reescríbalos; en caso contrario, dé argumentos para desechar el carácter o para mantener su definición actual.
4. Use la literatura de la que disponga sobre un grupo en particular, trate de evaluar los caracteres presentes y hágase preguntas similares a las que realizó cuando examinó los caracteres de sus compañeros.
5. Si es posible, trate de usar y contrastar un análisis filogenético con una descripción tradicional del mismo grupo.
6. Pleijel (1995) argumentó que los únicos caracteres válidos son las presencias, por lo que todos los caracteres son

una enumeración de presencia/ausencia (no hay estados de carácter), trate de identificar las posibles ventajas y falencias de esa aproximación.

7. **Lea el artículo de Goloboff et al. (2006) e identifique la forma como manejan los caracteres cuantitativos. ¿usan un esquema similar al usado para los caracteres discretos?**

Literatura recomendada

de Pinna (1991) [Una discusión sobre la formulación de los caracteres].

Goloboff et al. (2006) [Una visión a los caracteres cuantitativos].

Kluge (2003) [Una crítica al uso de la **similitud** como criterio de selección de caracteres].

Neff (1986) [Una normalización del análisis de caracteres].

Patterson (1981) [La **verdadera prueba de homología**].

Platnick (1979) [Un artículo clásico sobre la jerarquía -cladismo de patrón-, tanto de caracteres como de taxa].

Pleijel (1995) [Una visión al manejo de caracteres, diferente a la ofrecida aquí].

Rieppel & Kearney (2002) [Una visión crítica y extensa de la selección de caracteres, con ejemplos empíricos].

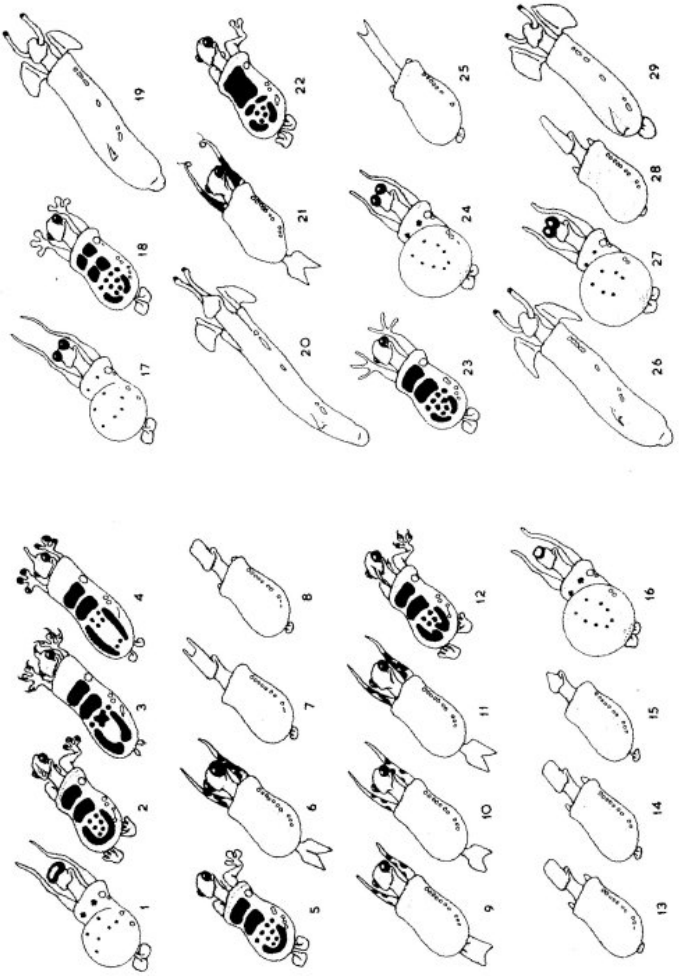


Figura 1.1: Los Caminálculos, tomado de Sokal (1983)

Práctica 2

Matrices de datos

Introducción

Una de las diferencias más importantes entre los trabajos taxonómicos con un enfoque **clásico** y los análisis filogenéticos es que estos últimos incluyen explícitamente una matriz de datos donde se pueden evidenciar los caracteres examinados, y cómo fueron interpretados.

Básicamente, la matriz de datos es una lista tabulada de las observaciones de los caracteres en los distintos taxa. Para facilitar su

lectura y su uso en programas de análisis filogenético, los caracteres y sus estados son codificados como números o como letras.

Una vez construida, la matriz es el punto de partida para la búsqueda de árboles y la manera más sencilla con la que otros investigadores pueden recuperar la información recopilada durante el **análisis de caracteres**.[1]

Tal y como se hizo en la práctica sobre selección de caracteres, lo que se busca es que el investigador sea consistente en la codificación de los caracteres, lo cual es importante en el manejo de caracteres inaplicables y no observados. Algunos autores los codifican de diferentes formas en la matriz (usualmente, con **-** y **?**) para facilitar la recuperación de información (puede encontrar una discusión más completa en Strong & Lipscomb (1999)), otros autores, y en general los programas, no reconocen diferencia entre unos y otros, mientras que programas como TNT, POY o PAUP* pueden reconocer los eventos de inserción-pérdida como un quinto carácter o como gaps (Giribet & Wheeler, 1999).

Cuando se usan caracteres con múltiples estados, es necesario cla-

[1] si desea puede ver nuestra visión ampliada en Miranda-Esquivel et al. (2004).

rificar cuáles fueron usados como aditivos y cuáles como no aditivos, y en el caso de haber sido recodificados, cómo se hizo esa codificación. Si se han codificado como aditivos, se debe indicar el ¿por qué? de tal codificación e incluir los argumentos que muestren que los estados de carácter se hallan anidados entre sí, la aditividad puede generar la estructura en la topología final, sin que haya realmente transformaciones que soporten los nodos.

Técnicas

Existen diferentes programas para manipular matrices de datos y árboles. Algunos de ellos permiten la interacción matriz-árbol (WinClada[2], Mesquite[3], MacClade[4], R[5] o Seaview[6]).

En su mayoría estos programas sirven de plataforma para manejar

[2] www.cladistics.com

[3] www.mesquite.org

[4] el programa no corre en versiones de MacOS X superiores a 10.6 (desde Lion en adelante)

[5] R es uno de los programas para análisis estadístico más versátiles, además de ser gratuito permite la implementación de múltiples procesos de cálculo desde estadistica básica hasta múltiples análisis en evolución; sirve de plataforma para *ape* Paradis et al. (2004), un módulo que permite distintas acciones con matrices y árboles, lo puede obtener desde http://cran.r-project.org.

[6] http://doua.prabi.fr/software/seaview

otros programas que realizan el análisis filogenético como tal. Muchos de los programas que manejan matrices están diseñados básicamente para algún tipo particular de datos (por ejemplo, ADN) y aquellos que tienen un marco amplio se quedan cortos para manejar cierta clase de información (por ejemplo, no interpretan la codificación IUPAC para polimorfismos de ADN).

Los programas pueden leer uno o varios formatos de archivos, pero solo algunos programas como Mesquite leen y escriben todos los formatos de datos; es importante revisar la compatibilidad de los programas para el manejo de archivos. En la mayoría de ellos es posible exportar entre los diferentes tipos de datos, o por lo menos entre los más usados. En general, la mayor parte de los programas trabaja bien con el formato NEXUS (Maddison et al., 1997), tanto para exportar como para importar, aunque el formato es en ocasiones muy diferente y algunos programas pueden no identificarlo correctamente. Otro formato importante es el de Hennig86 o NONA, pero en muchos programas, especialmente moleculares, su uso no está implementado.

Se han hecho algunos ejercicios de extender XML, un lenguaje

de marcas, para que sea posible usarlo en análisis filogenéticos[7], se busca en última instancia que los análisis filogéticos cumplan con los estándares de reproductibilidad (ver por ejemplo Cranston et al. (2014)).

Revise siempre la documentación del programa que desea usar y así podrá estar seguro si el programa que va a utilizar cumple con los requisitos que usted necesita y cual es el formato de las matrices.

Existen listas de programas para análisis filogenético que pueden ser consultadas en internet, por ejemplo:

http://evolution.genetics.washington.edu/phylip/software.html

o en

http://taxonomy.zoology.gla.ac.uk/software/

Abra el archivo de datos morfológicos para vertebrados:

"datosVertebrados.xls"

con un programa para hojas de cálculo y manipule las matrices con WinClada, Mesquite o TNT para Windows:

[7] www.phyloxml.org

1. Anote cuáles caracteres son multiestado y cuáles son binarios.
2. Identifique si hay o no caracteres aditivos.
3. A partir de los datos, construya una matriz nueva (en TNT para Windows use el menú **Data - edit data**).
4. Dado que los programas no cuentan con opciones de salvado automático, periódicamente salve la matriz.
5. Nomine los terminales y los caracteres y sus estados. Explore diferentes formas de llevar esa tarea a cabo.
6. Suponga que algunos autores consideran que las plumas son escamas modificadas. Aceptando esa información, recodifique la matriz y determine si el carácter es aditivo o no.
7. Seleccione el último carácter y colóquelo al principio de la matriz.
8. Introduzca un carácter nuevo en la posición 4.
9. Exporte la matriz en otros formatos, NEXUS si está usando WinClada, Nona si está usando Mesquite.
10. Verifique la compatibilidad de los datos, abriendo la matriz en el programa correspondiente.
11. Revise con un editor de texto los archivos que usted creó,

trate de identificar cuáles son las partes claves del formato[8].

12. Abra una de las matrices moleculares en cada programa y con un editor de texto y trate de identificar en qué se diferencia de la matriz morfológica.

13. en R:

    a) Instale en su ordenador la versión más reciente de R ($\geq$ 3.00-11).

    b) Carge las bibliotecas ***ape*** y ***phangorn*** [9]:

```
> library(phangorn)
```

    c) En caso de no tener las bibliotecas disponibles primero bájelas con la instrucción [10] :

[8] Aunque en general no se usan los editores de texto, este paso es crítico para después ser capaz de rastrear los problemas que pueden tener las matrices de datos.

[9] La biblioteca ***phangorn*** está diseñada para análisis filogenéticos que tiene como objetivo estimar árboles y redes, utilizando diferentes métodos como máxima verosimilitud, parsimonia, distancia y conjugación de Handamard, requiere las bibliotecas ***ape*** y ***rgl.*** Puede ser descargado desde http://cran.r-project.org/web/packages/phangorn/index.html

[10] La función install.packages("Nombre_paquete") permite hacer la descarga de los paquetes directamente del repositorio desde el entorno de R. También es posible hacer la descarga directamente de la página web e instalarlo desde cualquier directorio en el ordenador dando directamente la ruta a dicho sitio: install.packages ("../usuario/R/Packages/Paquete.tar.gz"), pero deberá descargar e instalar, independiente, todas las dependencias requeridas por el paquete. La instrucción library() le permitirá cargar el paquete en el entorno de R para

```
> install.packages(c("ape", "phangorn"), dependencies=T)
```

d) Abra la matriz de datos en formato de texto simple y asígnela a un objeto de R:

```
> datos <- read.table("matriz.txt")
```

e) Revise los nombres de las variables en la matriz con names(datos)

f) Revise los nombres de las terminales en la matriz con row.names(datos)

g) Abra la matriz de datos en formato phylip/newick y asígnela a un objeto de R:

```
> datos.Phylip <- read.phyDat("DNA1.phy", format="phylip", type="DNA")
```

h) Escriba en formato Nexus la matriz leída datos:

poder empezar a trabajar con todas sus utilidades.

```
> write.nexus.data(datos.Phylip, file="DNA1.nex")
```

i) Abra la matriz escrita con Winclada o Mesquite y revise la conversión.

14. En el directorio de datos existen dos matrices con distintos problemas "problema1.txt" y "problema2.txt", intente abrir las matrices, busque y corrija el error.[11]

Dependiendo de la plataforma que trabaje, usted tiene disponible distintos programas. Mesquite y R son programas gratuitos y válidos para todas las plataformas pero en general las búsquedas no son eficientes, aunque le permiten trabajar con varios tipos de datos e interactuar directa o indirectamente con programas como PAUP, TNT o PhyML. Sobre Windows usted cuenta con WinClada que funciona tanto en modo de manejo de edición de matrices (**windada**) como en modo de edición y manipulación de árboles. Además, le permite hacer búsquedas con NONA, también puede usar la versión de Windows de TNT que tiene interfaz gráfica. Si usa Mac una opción es McClade, el cual funciona

[11] tradicionalmente, las terminaciones de los archivos son .ss en WinClada, .nex con Mesquite o MacClade, pero **debe** recordar que la terminación del archivo no es necesariamente el formato.

como Mesquite pero es más veloz y eficiente, aunque es muy factible que no lo pueda usar con la versión actual de Mac OS X.

Revise la sección de Programas de cómputo para ver las instrucciones que utilizaría con un programa distinto a Winclada/NONA. En todos los casos familiarícese con los menús e instrucciones para abrir/cerrar y editar tanto matrices como árboles, tenga en cuenta que los manuales de los programas traen información adicional, por lo tanto su lectura es una muy buena opción.

Pregunta(s) general(es).

1. **Si desea transformar de un formato de matriz a otro usando exclusivamente editor de texto, ¿cuáles son los pasos a seguir? Ensaye la lista contruida con un ejemplo.**
2. **Haga el listado de los aspectos comunes a todos los formatos de matrices.**
3. **En el laboratorio anterior se insistió en la claridad de los caracteres y sus estados. A la luz de los resultados obtenidos usando árboles y la matriz:**
   a) **¿puede usted conectar la importancia del análisis de caracteres con la forma como se interpretan los**

cladogramas?

b) **Revise su bibliografía, y de ser posible compare trabajos con y sin matriz explícita. ¿Puede notar alguna diferencia?**

c) **¿Cree que es una ventaja incluir y publicar la matriz, o por el contrario es una desventaja?**

Literatura recomendada

Maddison et al. (1997) [Una introducción al formato NEXUS].

Maddison & Maddison (2014) [El programa más versátil para manipulación de matrices, por lo que la documentación debe ser leída].

Maddison & Maddison (2005) [Aunque el programa no esté en uso, el manual presenta gran cantidad de información básica].

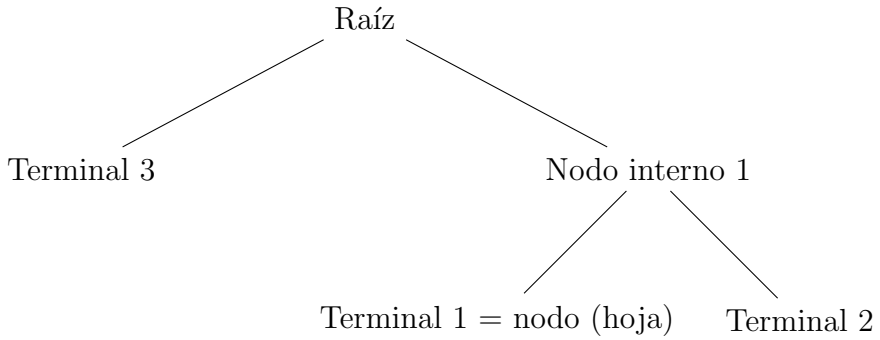
Neff (1986) [Una de las primeras discusiones sobre el manejo de pesos en caracteres].

Práctica 3

Árboles

Introducción

Una vez finalizado un análisis filogenético, el resultado es un agrupamiento que tradicionalmente se dibuja como un árbol donde los taxa usados estan como *hojas* o *terminales* del árbol. Las diferentes ramas se unen en *nodos* o *componentes*. Así, el agrupamiento ((Terminal 1, Terminal 2), Terminal 3) puede representarse con el árbol:



Es importante diferenciar entre el **contenido** del nodo, que son los terminales que forman el nodo (el nodo interno 1 tiene dos terminales: 1 y 2), y la **información** del nodo, que es cómo están organizados estos terminales en términos de grupos o clados anidados Nelson (1979). En general, para muchos autores un componente representa un grupo monofilético (i.e., definición *topológica* de monofilia).

En cladística, los árboles son también conocidos como **cladogramas** y en general, las dos palabras son usadas erroneamente como sinónimas.[1] En cladística (pero no en otras metodologías) los cladogramas deben tener transformaciones de caracteres en nodos (nodos soportados), en caso de no tener transformaciones, el

[1]Es necesario aclarar que todos los cladogramas son árboles, pero no todos los árboles son cladogramas

nodo no está soportado o es un nodo de costo cero, es decir, es un artefacto de los programas usados para obtener los árboles (véase Coddington & Scharff (1994); Goloboff (1998)). El número de transformaciones es un estadístico importante que hay que tener en cuenta y se usa como criterio para seleccionar entre diferentes cladogramas, se escoge la topología que sugiere el menor número de transformaciones. En los métodos estadísticos generalmente es importante la longitud de las ramas, esta puede representarse como una probabilidad o frecuencia de transformaciones y suelen presentarse los árboles con una escala que muestra la cantidad de cambio en términos de tiempo o de transformaciones; sólo la dimensión que va de las ramas a los terminales tiene importancia en ese caso, el "ancho" del árbol se usa para acomodar los diferentes terminales usados. En los cladogramas, las dimensiones no tienen ningún significado especial.

Técnicas

Una forma de expresar los árboles en formato de texto (por ejemplo para usar con programas o para el resumen de un artículo) es

la notación de paréntesis, donde los paréntesis limitan los nodos; dependiendo de los autores o los programas, los grupos hermanos son separados por espacios, comas o símbolos de suma, por ejemplo (a (b c)), es equivalente a (a+(b+c)) y a (a, (b, c)), tenga en cuenta que en los árboles filogenéticos el orden de los terminales, sin cambiar la topología, no altera el contenido del árbol, por ejemplo (a (b c)) es exactamente igual a ((c b) a) y a ((b c) a).

Existen diferentes programas para manipular y generar árboles; la mayoría de ellos permiten interacción matriz-árbol, mientras que otros solo utilizan árboles. En general, los programas de análisis filogenético permiten manipular árboles en un esquema gráfico rudimentario. Algunos programas permiten guardar información adicional a la topología en el mismo árbol como la longitud de la rama que conduce al nodo o terminal o una etiqueta. Los programas que pueden trabajar independientemente de la matriz suelen estar diseñados para la impresión o exportación gráfica de los árboles, soportando cambios de topología, longitud y etiquetas de las ramas, pero no transformaciones de caracteres.

Dependiendo de la plataforma que trabaje, usted tiene disponible

distintos programas: Mesquite, R[2] y Figtree[3] son válidos para todas las plataformas (adicionalmente son gratuitos). Mesquite es muy lento si su máquina es lenta (requiere de la máquina virtual de Java). Sobre Windows usted cuenta con WinClada (requiere la matriz de datos), su interfaz de impresión requiere mucha práctica (no es intuitiva). Otras alternativas, si usa Linux, pueden ser NJPlot o GNUplot. Para la impresión final del árbol para publicación, una opción puede ser R, que aunque es poco intuitivo al iniciar, logra resultados finales de mejor calidad que otros programas.

En general, la manipulación de árboles se activa con un menú (en WinClada **Edit/Mouse**, en FigTree directamente como barra de herramientas en la ventana; en Mesquite se usa el menú de herramientas de la ventana de edición de TreeView). Las acciones se realizan al seleccionar con el ratón la rama (WinClada) o arrastrando las ramas (para moverlas en Figtree). Más que comandos, en esta práctica es importante manipular el ratón.

1. Desde WinClada+NONA, Mesquite o TNT

[2]http://cran.r-project.org
[3]http://tree.bio.ed.ac.uk/software/figtree/

a) Revise los archivos de datos con un editor de texto y con el programa seleccionado abra el archivo que contenga tanto matrices como árboles.

b) Establezca la forma de obtener información sobre los árboles, en un principio los costos del árbol.

c) Cambie de posición algunos nodos o terminales de la topología, y observe cómo cambia el costo del cladograma y los estados asignados a los nodos o la forma como los caracteres son mapeados, pruebe también moviendo ramas completas.

2. En Mesquite y TNT (para Windows) usted puede mover ramas completas; en TNT (para Windows) use el menú **Trees - view** y seleccione con el botón izquierdo del ratón el clado a mover (el clado queda marcado en rojo), señale con el ratón el destino usando el botón derecho.
3. Desde WinClada o TNT (para Windows), haga una búsqueda de cladogramas para la matriz de vertebrados que contruyó en la práctica 2, use los parámetros por omisión (menú **Analize / heuristics**, **traditional search**).
4. En Mesquite o Figtree:

a) Cambie el orden de los terminales sin cambiar la topología (las relaciones entre terminales) del árbol.

b) Coloque etiquetas en los nodos, y salve el árbol.

c) Revise ese archivo en un editor de texto para determinar cómo se colocaron tanto las etiquetas como los costos.

d) Abra el árbol con longitud de ramas.

5. En Mesquite o WinClada abra el archivo "datosConarbol.dat":

   a) Indique la cantidad de árboles presentes y el costo de los mismos.

   b) Examine las agrupaciones obtenidas e identifique los caracteres que soportan los nodos.

   c) Mapee los caracteres en los árboles usando distintos tipos de optimizaciones:

      1) en WinClada ACCTRAN (=fast), DELTRAN (=slow) y no ambigua.

      2) en TNT (para Windows) (**Optimize** > **Character** > **Reconstructions** y seleccione un solo árbol y algunos caracteres).

      3) en Mesquite: Dollo e Irreversible (C-S).

revise cómo cambian las optimizaciones en los distintos nodos.

d) Revise de nuevo el efecto en el costo de los árboles con las siguientes modificaciones:

1) Cambie la raíz del árbol.
2) Cambie de posición nodos y terminales.
3) Seleccione un nodo y colapselo.

6. R:

a) Lea el archivo con árboles "arbolR.tre" use las instrucciones:

```
> library(ape); arbol < − read.tree("arbol.r.tre")
```

b) Grafique el objeto arbol use las instrucciones, que por defecto dibuja el árbol con longitud de ramas o como un filograma:

```
> plot(arbol)
  o
> plot.phylo(arbol)
```

c) Para graficar el árbol sin incluir la longitud de ramas

use las instrucciones:

```
> plot(arbol, use.edge.length=FALSE)
```

d) Para asignar las longitudes de las ramas como rótulos de los nodos y graficarlos use las instrucciones:

```
> arbol$node.label <- arbol$edge.length
> plot(arbol, show.node.label=TRUE)
```

e) Para graficar el árbol con título de gráfica ("Árbol 1"), colores de ramas y tipo cladograma use las instrucciones:

```
> plot(arbol, type="cladogram", main="Árbol 1",
edge.col=c("red","blue","cyan"))
```

f) Para obtener el costo del árbol, lea la matriz de datos **"**matrizR.phy**"**, asígnela a un objeto que se llame MatrizDatos y use la función parsimony():

```
> parsimony(arbol,MatrizDatos)
```

Pregunta(s).

a) **Al hacer una búsqueda por defecto ¿Usted y sus compañneros obtienen las mismas topologías y los mismos caracteres que soportan los grupos?.**

b) **Examine sus caracteres y cambie su aditividad-no aditividad. ¿Cambia esto la forma como mapean? Haga el ejercicio para varios caracteres, binarios y multiestado.**

c) **Si cambia la raíz, ¿cambia el costo? ¿Por qué cree que se presenta el resultado que obtuvo? ¿Es (y por qué) una ventaja o una desventaja?**

d) **Dibuje el siguiente cladograma:**
(Lampreas (Tiburón (Esturión Teleósteo) (Celacanto (Pez-pulmonado (Anfibio (Mamífero (Tortuga (Lagarto (Cocodrilo Ave)))))))))

e) **Si la raíz esta colocada entre Ave y Cocodrilo, ¿cómo es la topología resultante?**

f) **Convierta la topología dibujada a notación de paréntesis.**

Pregunta(s) general(es). Aparte de no tener un análisis

explícito, existe una gran diferencia entre los árboles filogenéticos actuales y sus "equivalentes" usados por algunos taxónomos (por ejemplo, Haeckel, Romer, etc.) ¿Es usted capaz de descubrirla? Clave: intente dibujar alguno de esos árboles al estilo actual.

Literatura recomendada

Page & Holmes (1998) [El capítulo 2 esta dedicado a los árboles y presenta muy buenas ilustraciones, También puede consultar la página de docencia de Rod Page (http://taxonomy.zoology.gla.ac.uk/teaching/index.html)].

Maddison & Maddison (2005) [Aunque el programa no esté en uso, el manual presenta gran cantidad de información básica]

Práctica 4

Búsquedas mediante parsimonia

Introducción

Un discusión constante en sistemática es como seleccionar cladogramas, por ejemplo, Hennig (1968) plantea relaciones mediante la agrupación por sinapomorfías, no es muy explícito a lo que refiere la obtención del cladograma. Camin & Sokal (1965) posiblemente fueron los primeros en sugerir el uso de la *parsimonia* como un

posible método para hacer esta selección Camin & Sokal (1965); desde entonces el cladograma seleccionado es aquel que minimiza la cantidad de transformaciones, es decir *el cladograma más parsimonioso*. Posteriormente, esta técnica fue generalizada usando diferentes tipos de optimización: unas de las más conocidas e implementadas son: *la optimización de Wagner y la optimización de Fitch* (Wagner (1961); Kluge & Farris (1969); Farris (1970); Fitch (1971)).

La búsqueda del cladograma más parsimonioso es más compleja a medida que se agregan terminales, por ello los algoritmos para búsquedas exactas solo son viables con pocos terminales (aprox. 20-30), después de este número el espacio de árboles es tan grande que es imposible una búsqueda exacta (dado que es un problema NP-Completo). Por esta razón se utilizan búsquedas heurísticas que permiten obtener respuestas generalmente cercanas al óptimo global, y pese a que estas soluciones no proporcionan con certeza la solución óptima, se obtienen resultados difíciles de superar.

La forma más sencilla de elaborar cladogramas es usando el algoritmo de Wagner: como el orden de entrada de los taxa afecta la topología, se realiza una aleatorización de tal secuencia de en-

trada, la cual puede estar seguida de permutaciones de ramas, sin embargo, para matrices muy grandes consume gran cantidad de tiempo; esto se debe repetir múltiples veces para evitar caer en "optimos locales". Con este método es posible una solución optima incluso para matrices de 80 a 100 terminales. Problemas más grandes requieren nuevas estrategias, algunas de ellas derivadas de la cristalización simulada, aceptando momentáneamente cladogramas subóptimos para iniciar desde ellos la permutación de ramas. Otros utilizan combinaciones, bien sea entre búsquedas exhaustivas o entre búsquedas sobre reducciones de la matriz.

Para las matrices cada vez más grandes (por ejemplo matrices moleculares con más de 1000 terminales), lo que se busca es tratar de identificar el acuerdo entre distintas réplicas de búsquedas parciales, en vez de buscar la solución óptima (ver Farris et al. (1996); Goloboff (1997); Goloboff & Farris (2001)).

La mejor opción para programas gratuitos es TNT; este es el programa más completo para análisis cladístico, está disponible para Windows, MacOSX y Linux, es bastante rápido, además de tener implementado ratchet y las nuevas búsquedas. NONA otra posible opción puede manejarse como buscador con WinClada (pa-

ra Windows), es buena idea que se familiarice con estos programas y la línea de comandos, las búsquedas son más eficientes desde la línea de comandos. PAUP* tambien está disponible como ejecutable en varias plataformas (Windows, MacOS X y Linux). PAUP* no solo usa parsimonia sino distancias y máxima verosimilitud, aunque para parsimonia es menos versátil que TNT. TNT está diseñado para búsquedas exhaustivas en matrices grandes, la velocidad y sistema de macros son sorprendentes; pero, por lo menos hasta el momento, no hace búsquedas mediante ML. Si está interesado en búsquedas bajo ML las opciones son PhyML dadas las múltiples opciones de optimización y modelos usados o RaxML/ExamML por su velocidad, y la posibilidad de trabajar con ellos en paralelo.

Técnicas

El algoritmo de Wagner es la base para las búsquedas actuales. Para evitar el problema del orden de entrada de los datos, los

taxa se seleccionan al azar (RAS[1], la mayor parte de los programas actuales tienen esta opción: inician con una **semilla** determinada para el generado de número aleatorio y aseguran que la búsqueda sea exactamente igual a otra que tenga el mismo de inicio (semilla del generador de números seudo-aleatorios). Una vez construido un cladograma, este suele ser sometido a permutación de ramas para mejorar su calidad. Básicamente se toma un nodo (subárbol) y es eliminado del cladograma principal, luego se prueba si al unirlo en diferentes lugares del cladograma principal disminuye el costo con respecto al cladograma original. Se puede permutar ramas de varias formas; las más comunes son unir el nodo a las diversas ramas del cladograma principal (subpoda y replantado, SPR por sus siglas en Inglés), o intentar otros puntos de unión dentro del subárbol y cambiar la topología (bisección y reconexión de árboles, TBR). En general, la mayor parte de los programas utilizan TBR, puesto que el tiempo de permutación entre ambas técnicas es casi igual y TBR es mucho más eficiente.

- Con NONA/Winclada, TNT, POY y PAUP*

[1]En algunos escenarios se puede comenzar con un árbol al azar que será mejor punto de inicio que un árbol de Wagner, ver Goloboff (2014)).

Construya una tabla donde pueda registrar la información de tiempos de búsqueda, número de cladogramas y costo del mejor cladograma, para cada una de las siguientes búsquedas:

1. Búsquedas por omisión

   Ejecute el archivo datos.chica.dat siguiendo los comandos:

   - NONA/Winclada

     File/File open

     Heuristic Search

   - TNT

```
> proc nombre_archivo;
> mult;
```

   - POY

```
> read( 'nombre_archivo')
> build()
> swap()
```

- PAUP*

```
> set criterion=parsimony
> exec nombre_archivo
> hsearch
```

Pregunta(s).

- **¿Qué tipo de información puede obtener cuando carga el archivo de datos?**
- **¿Cál es la búsqueda por omisión en cada programa utilizado?**

2. Búsquedas modificadas

   Con el archivo el datos datosChica.dat realice las siguientes búsquedas, modificando los comandos que sean necesarios:

   a) 5, 10 y 100 réplicas.

   b) reteniendo 1 o 100 árboles por réplica.

   El manual de cada programa especifica los comandos a modificar para hacer dichas búsquedas como por ejemplo: Para búsquedas con NONA, el comando más usa-

do es mult* para las búsquedas iniciales, max* para permutar ramas (requiere árboles) y nix* para ratchet. En TNT también se puede usar mult; la permutación de ramas es con bbreak. En PAUP* debe definir el criterio de búsqueda: parsimonia usando set criterion=parsimony, y la búsqueda con hsearch tanto para árboles de Wagner como para permutar ramas; en este último caso use hsearch start=current.

Para los archivos de macros de TNT use la instrucción run seguida del nombre del archivo; en este caso pauprat.run y los parámetros run pauprat.run 10 5; TNT usa pesos de 1 y 2 en el archivo de salida, pauprat.

Pregunta(s).

- **Utilice un manejador gráfico que le permita visualizar la tendencia en los datos obtenidos.**
- **¿Encontró alguna tendencia en términos de tiempos o costos, al aumentar el número de réplicas?**

3. Búsquedas con RATCHET

   Con el mismo archivo.dat, realice las siguientes búsquedas:

a) Número de Búsquedas continuas 1, 5 y 20.

b) Número de réplicas/Búsqueda 1, 5 y 100.

c) Número iteraciones en RATCHET 5, 50 y 100.

En problemas más complejos de 100 o más terminales, se requiere utilizar técnicas más sofisticadas para obtener resultados satisfactorios. La más sencilla es el rastrillo o piñón (*ratchet* en inglés) (Nixon, 1999; Quicke et al., 2001), la cual es una forma simple de implementar una cristalización simulada. El método consiste en usar un árbol ya elaborado (por ejemplo con Wagner + TBR), perturbar la matriz de datos (con eliminación o repesado de caracteres), permutar las ramas del árbol para obtener el óptimo de la nueva matriz, volver la matriz a su estado original y buscar el árbol óptimo con permutado de ramas (todo ese proceso es una iteración, la cual se repite **n** veces). El rastrillo es eficiente usando solo unos pocos árboles por iteración y permutando una cantidad intermedia de caracteres (entre 10-25 %), en general, mejora drásticamente el ajuste de los cladogramas en las primeras iteraciones (Nixon, 1999).

Para producir posteriores mejoras en el ajuste de cladogramas con números mayores a 500 terminales, los métodos más eficientes parecen ser: la "deriva de árboles", que es una implementación más explícita de la cristalización simulada (es decir aceptar soluciones ligeramente subóptimás con una determinada probabilidad, y a medida que el análisis avanza, se disminuye la probabilidad de aceptación de los subóptimos), y la fusión de árboles, que utiliza lo mejor de diferentes soluciones. Una revisión completa de estos métodos se puede consultar en Goloboff (1999).

Pregunta(s).

- **¿Cuál es y qué hace el comando que permite hacer búsquedas con la técnica RATCHET en cada programa?**
- **¿Cada búsqueda o réplica es independiente de otra?**
- **¿Hay algún efecto en el resultado al hacer varias búsquedas continuas con el mismo número de réplicas o es suficiente una sola búsqueda con**

múltiples réplicas? Realice búsquedas adicionales con parámetros diferentes que le permitan responder estas preguntas. Utilice un manejador gráfico donde pueda visualizar la tendencia de los datos obtenidos.

- Análisis Filogenético en R: bibliotecas y dependencias

  1. Instale en su ordenador la versión más reciente de R ($\geq$ 3.00-11).
  2. Carge, y de ser necesario instale, la biblioteca ***phangorn***

```
> install.packages("phangorn", dependencies=TRUE)
> library (phangorn)
```

  3. Lectura del alineamiento o matriz y formato de escritura

     a) Cargue el alineamiento o la matriz llamada chars2.txt y defina los argumentos requeridos para que esta pueda ser leída.

     b) Escríbala como formato NEXUS utilizando el nom-

bre primates.nex:

```
> primates <- read.phyDat ("chars2.txt",
  format="phylip", type="DNA")
> write.nexus.data (primates,
  file="primates.nex")
```

c) Revise la matriz primates.nex con un editor de texto, identifique las particularidades del formato en R y si este archivo es similar al generado por winclada.

La función read.phyDat() permite leer diferentes tipos de caracteres como "DNA", "AA", "CODON" o "USER", este último es definido por el usuario. Posteriormente, el vector denominado primates es escrito en otro formato diferente al formato phyDat; la función write.nexus.data() escribe un archivo formato NEXUS a partir de un vector de secuencias. Los argumentos de una función pueden ser consultados con el comando args(Nombre_función) o con el comando de ayuda ?Nombre_función.

Pregunta(s). ¿Qué otras funciones en otras bibliotecas permiten leer y/o escribir archivos tipo Nexus, Fasta, Phylip, Clustal, Sequential e Interleave?

4. Topologías iniciales

   Estime la matriz de distancia, realice el análisis de agrupamiento y grafique para un posterior análisis.

```
> matrizDistancia <- dist.dna(as.DNAbin(primates))
> arbolUPGMA <- upgma (matrizDistancia)
> arbolNJ <- NJ(matrizDistancia)
> plot (arbolUPGMA, main="UPGMA", cex = 0.8)
> plot (arbolNJ, "unrooted", main="NJ", cex = 0.5)
```

La función dist.dna() permite obtener una matriz de distancias por pares de secuencias de ADN, bajo un modelo evolutivo determinado. Actualmente es posible estimar esta matriz bajo 11 modelos evolutivos diferentes, además permite estimar la varianza entre dis-

tancia y el valor de gamma. Existen varios método de agrupamiento por distancia para obtener la topología inicial, entre ellos los algoritmos UPGMA (=Unweighted Pair Group Method with Arithmetic Mean), WPGMA (=Weighted Pair Group Method with Averaging), NJ (Neighbor Joining) y UNJ (Unweighted Neighbor Joining); en este caso las funciones upgma() y NJ() permiten construir árboles de distancia bajo sus características específicas, los cuales pueden ser visualizados con la función plot().

La función random.addition(), permite definir los árboles iniciales de los cuales parte el análisis de parsimonia, use:

```
> arbolRAS <- random.addition(primates,
  method="fitch")
> plot(arbolRAS, main="UPGMA", cex = 0.8)
```

para construir los árboles y graficarlos.

Pregunta(s).

- **¿En qué difiere cada árbol obtenido?**
- **¿En qué consiste el método de FICHT y el método de SANKOFF?**
- **¿Qué hacen los algoritmos de agrupamiento UPGMA, WPGMA, NJ y UNJ?**
- **¿En qué consiste el método de construcción de árboles de random.addition?**

5. Parsimonia y optimización

   A partir de la matriz de datos primates y las topologías construidas, calcule el costo de los árboles y obtenga el árbol de menor costo o score.

```
> parsimony(treeUPGMA, primates)
> parsimony(treeNJ, primates)
> parsimony(treeRAM, primates)
```

La función parsimony() permite obtener el árbol de menor costo utilizando el algoritmo del método SANKOFF o de FITCH, en este caso es una búsqueda por omisión dado que no se especifican los argumentos.

Optimice cada topología obtenida, utilizando el método

de optimización por omisión, optimización por SPR y optimización por NNI. Registre sus resultados en una tabla.

6. Para la optimización por omisión use:

```
> optParsUPGMA <- optim.parsimony(treeUPGMA, primates)
> optParsNJ <- optim.parsimony(treeNJ, primates)
> optParsRAM <- optim.parsimony(treeRAM, primates)
```

7. Para la optimización con rearreglo de ramas específico, use:

```
> optParsUPGMA_SPR <- optim.parsimony(treeUPGMA, primates, rearrangements="SPR")
> optParsUPGMA_NNI <- optim.parsimony(treeUPGMA, primates, rearrangements="NNI")
```

La función optim.parsimony() intenta encontrar el o los árboles más parsimoniosos utilizando los métodos de rearreglos NNI y SPR.

Pregunta(s).

- **¿En qué difieren los métodos de rearreglos NNI, SPR y TBR?**
- **¿Cuáles son los argumentos por omisión de las funciones** parsimony() **y** optim.parsimony()**?**

8. Parsimonia usando Ratchet

   Utilice el método Ratchet en parsimonia para hacer las búsquedas del los árboles de menor costo. Cree una tabla para registrar los resultados de las búsquedas con los árboles obtenidos en el proceso anterior, use los siguientes parámetros para las búsquedas.

9. Búsqueda con Ratchet utilizando los árboles iniciales y optimizando con el método de rearreglos NNI.

```
> pratchet(primates, start=treeUPGMA,
method="fitch", maxit=100,
k=5, trace=1, all=FALSE, rearrangements="NNI")
```

10. Búsqueda con Ratchet utilizando los árboles iniciales y optimizando con el método de rearreglos SPR.

```
> pratchet(primates, start=treeUPGMA,
method="fitch", maxit=100,
k=5, trace=1, all=FALSE, rearrangements="SPR")
```

11. Búsqueda con Ratchet utilizando los árboles optimizados con NNI y optimizando con el método de rearreglos NNI.

```
> pratchet(primates, start=optParsUPGMA_NNI,
method="fitch", maxit=100,
k=5, trace=1, all=FALSE, rearrangements="NNI")
```

12. Búsqueda con Ratchet utilizando los árboles optimizados con NNI y optimizando con el método de rearreglos SPR.

```
> pratchet(primates, start=optParsUPGMA_NNI,
method="fitch", maxit=100,
k=5, trace=1, all=FALSE, rearrangements="SPR")
```

13. Búsqueda con Ratchet utilizando los árboles optimizados con SPR y optimizando con el método de rearreglos NNI.

```
> pratchet(primates, start=optParsUPGMA_SPR,
method="fitch", maxit=100,
k=5, trace=1, all=FALSE, rearrangements="NNI")
```

14. Búsqueda con Ratchet utilizando los árboles optimizados con SPR y optimizando con el método de rearreglos SPR.

```
> pratchet(primates, start=optParsUPGMA_SPR,
method="fitch", maxit=100,
k=5, trace=1, all=FALSE, rearrangements="SPR")
```

pratchet() hace búsquedas usando el método de Ratchet, estas búsquedas son hechas a partir de un árbol inicial ya sea optimizado o no, aunque es preferible partir de un óptimo ya dado. También puede iniciar haciendo una búsqueda para obtener el árbol o los árboles de partida, aplicar el método de ratchet y óptimizar.

Pregunta(s).

- **¿Hay diferencias entre las búsquedas con ratchet en términos de costos o tiempo?**
- **Escriba y ejecute la línea de código que le permitiría realizar una búsqueda con: 70 iteraciones en Ratchet, Metodo SANKOFF, rearreglo SPR, SIN especificar el árbol inicial.**

Pregunta(s) general(es).

- **De los diferentes programas usados, ¿Cuál estima usted que es el óptimo? Explique las razones de su selección.**
- **¿Cuál cree usted que sería(n) el(los) criterio(s) para seleccionar entre los diferentes programas?**
- **Elabore una tabla usando sus resultados y los de sus compañeros.**
- **Para cada matriz:**
  - **¿En qué clase de búsqueda se obtuvo el mejor resultado?**
  - **¿Cuánto demoró para obtener dicho resultado?**

- **Dado que con una técnica heurística existe el riesgo de no obtener el árbol más corto ¿Cómo justificaría usted la búsqueda realizada?**

Un última recomendación, en este laboratorio solo se utilizaron algunos tipos de búsquedas posibles y solo algunos comandos para cada programa. Revise otros comandos de búsqueda en estos programas u otros parámetros para los comandos usados en la práctica.

Literatura recomendada

Swofford et al. (1996) [Una descripción muy completa de la forma como se calculan las versomilitudes y los métodos de verosimilitud para encontrar árboles].

Práctica 5

Pesaje de caracteres

Introducción

Cuando se habla de pesaje de caracteres en cladística, se refiere a que algunos caracteres sugieren más información que otros. El tema es controversial; la idea de pesaje es central en autores como Neff (1986), mientras que para autores como Kluge (1997) es regresar a la subjetividad de la época de la taxonomía clásica, al imponer a la filogenia un prejuicio de "cómo" es la evolución Kluge (1997); otros autores (por ejemplo, Goloboff (1993, 1995)) defien-

den el uso de pesado, argumentando que es claro tras un análisis filogenético que diferentes caracteres poseen distintos niveles de información; lo cual se hace evidente al multiplicar cualquier cuantificador de homoplasia del carácter por el peso inicial que se le asignó, en la lógica de pesaje sucesivo Kluge & Farris (1969).

Existen dos formas de hacer pesado de caracteres y una tercera que es un criterio de búsqueda, que no es excluyente del pesaje de caracteres. El primero es el pesado *a priori*, antes de empezar el análisis.

En la actualidad la forma más común es disminuir el peso del tercer codón en los análisis moleculares. Aunque se han propuesto muchas formas de encontrar a partir de los caracteres un peso inicial, quienes practican pesado del tercer codón no están muy preocupados e insisten que lo importante es diferenciar un tipo de codón de los otros, por ejemplo: Springer et al. (2001); Strugnell et al. (2014).

En el pesado *a posteriori* el peso se asigna basándose en un análisis inicial de los datos; se estima la *confianza* del carácter, usando por ejemplo, el índice de consistencia, o el de retención, y con base

en esos pesos se reinicia el análisis (Farris, 1969). En general es el esquema de pesaje más usado. El pregunta clave aquí es: ¿cómo se hace este primer análisis? Y después, ¿cómo se termina?. Otro de los problemas de esta forma de pesado radica en la comparación de los cladogramas, puesto que diferentes juegos de pesos pueden producir diferentes respuestas que no son comparables (a nivel de los estadísticos de ajuste, como costos).

La tercera forma de "pesado" no es ni *a priori*, ni *a posteriori*, ya que en realidad es un criterio de búsqueda. Esta forma es conocida como pesado implícito (Goloboff, 1993). Este procedimiento utiliza la confianza del carácter (con una función cóncava de homoplasia) y utiliza ese valor como criterio óptimo; el problema de este método, de hecho el de cualquier método, es cómo escoger entre las diferentes funciones. El pesado sucesivo es muy popular para "reducir" la cantidad de árboles más parsimoniosos (Carpenter, 1988), aplicación que no es correcta ya que es una metodología con su propia lógica, definir los pesos de acuerdo al comportamiento en las búsquedas previas, por lo que los resultados no tienen que coincidir con los de pesos iguales, ni en topología ni en número de soluciones (implícito en Goloboff (1995)).

Para iniciar las rondas de pesado, Farris (1969) propuso comenzar basándose en la compatibilidad de caracteres (que no se contradicen entre sí). En la actualidad la forma más común es iniciar con un árbol de pesos iguales. Kjer et al. (2001) propusieron comenzar con el mejor resultado de varios árboles producto de *bootstrap* (u otro método que produzca pseudoréplicas).

Dado que se espera que el análisis sea autoconsistente (Farris, 1969), dos réplicas consecutivas deben generar los mismos árboles (y por lo tanto los mismos pesos). La autoconsistencia se puede ver afectada por el hecho de tener gran cantidad de árboles óptimos, por lo que es importante limitar el número de iteraciones. Kjer et al. (2001) proponen no iterar y solo mantener los pesos dados por los árboles de las pseudoréplicas.

Farris (2001) buscó solucionar simultáneamente ambos problemas. Propuso comenzar con un árbol donde se ha hecho *jackknife* con probabilidad de 0.5, restaurar luego todos los caracteres y comenzar a partir de los estadísticos del árbol producto de la permutación e iterar; el proceso se repite varias veces. Farris (2001) argumenta que si los pesos son independientes del punto inicial, las diferentes réplicas producirán aproximadamente los mismos resul-

tados (los resultados se muestran como un árbol consenso de la mayoría de las diferentes réplicas).

El método de Goloboff (1993) sigue la lógica de parsimonia tradicional, y aunque Goloboff (1993) veía su método como un refinamiento de pesado sucesivo, este pesaje se puede usar igualmente en coordinación con pesado implícito. La forma común de pesado implícito es usar una función cóncava de la forma $\frac{k}{k+h}$, donde k es la constante de concavidad y h la homoplasia del carácter. Cuanto mayor sea la constante de concavidad, menor es la diferencia entre los caracteres muy homoplásicos y los poco o no homoplásicos, por lo que es equivalente a parsimonia lineal.

Técnicas

Para la matriz de datos (datos.pesado.dat).

1. En PAUP* se pueden definir juegos de caracteres usando el código X - .\ N, donde X es el número del carácter donde se empieza y N el número de posiciones en los que se vuelve a aplicar la opción. Por lo tanto la instrucción weights 3:all;

todos los caracteres tendrán peso de 3 y con weights 1: 3 - .\ 3; cada tercer carácter, a partir del carácter 3 será pesado con 1. Usted puede chequear esto usando cstatus full=yes; que le mostrará el peso de todos los caracteres y deben verse en la secuencia 3 3 1 3 3 1..., también tiene implementado una opción para tomar pesos a partir de los árboles en memoria. Esa opción es reweight en la que se pueden modificar el tipo de pesado, qué valor se va a utilizar y la escala de los pesos. ¡Use la ayuda en línea para manipular estos valores!.

a) Abra la matriz y realice una búsqueda simple.

b) Repita la búsqueda pesando diferencialmente el tercer codón (por ejemplo asignandole un peso de 0), primero creamos la partición:

```
> charset terceraPosicion=3-.\ 3;
```

y luego se asigna el peso

```
> weight 0 :terceraPosicion;
```

Compare los resultados con los de la matriz sin pesado diferencial.

c) Coloque todos los pesos iguales y realice una búsqueda con pesado sucesivo, itere un máximo de 10 veces. Chequee si los pesos se estabilizaron revisando el costo de los árboles usando pscores;.

2. En TNT el pesado implícito se activa utilizando piwe=N; donde N es el valor de concavidad a usar. Con PAUP* se activa usando

```
> pset Goloboff=yes gk=(N-1);
```

La notación de N-1 es necesaria ya que PAUP* comienza con 0 y suma uno (técnicamente es como comenzar desde 1). Tanto en TNT como en PAUP* el límite es de 32000 (lo cual es prácticamente equivalente a parsimonia lineal). Además TNT y PAUP* usan todos los valores decimales, por lo cual es recomendable recurrir a alguna medida de soporte para los nodos, con lo que se evita la sobre-resolución. A diferencia de PAUP*, TNT usa una función a minimizar[1]; para comparar los reportes del ajuste generados en PAUP*, puede obtener el ajuste usando fit*;. En PAUP*

[1]el inverso de la usada en PIWE: $\frac{h}{k+h}$

use

```
> pscores gfit=yes;
```

TNT permite definir su propia función de concavidad usando piwe [A B C...;, donde A es el fit para 0 pasos extra, B para 1, C para 2 y asi sucesivamente.

3. Tanto en TNT como en PAUP*:
   a) Con los caracteres con pesos iguales, active los pesos implícados con k=1, y haga una búsqueda.
   b) Revise el soporte de los nodos usando *jackknife* (use pocas réplicas, máximo 100).
   c) Repita los análisis con valores de k de 3, 6 y 10.

Pregunta(s) general(es).

- **Compare sus resultados (topologías y caracteres en los nodos) con todos los métodos de pesado utilizados. ¿En qué se parecen y en qué se diferencian los resultados?**
- **Usted debió definir una forma de pesaje del tercer codón. Defienda su esquema de pesado y compárelo con el de sus compañeros.**

- **Dados los resultados que encontró al medir el soporte de los nodos, ¿cree usted que hay sobreresolución en los datos usados?**
- **Escriba un ensayo corto (de media página) donde presente su posición con respecto al pesado de caracteres, tenga en cuenta las siguientes preguntas:**
  - **¿Está a favor o en contra? ¿por qué?**
  - **¿Cuáles son las ventajas y problemas que ofrece su posición?**
  - **¿El pesaje contradice la lógica de agrupamiento por homologías?**

Literatura recomendada

Farris (1969) [La idea original de pesar usando la información derivada de los cladogramas].

Goloboff (1995) [Una defensa de pesado implícito].
Goloboff et al. (2008a) [Un análisis empírico que muestra que pesaje es un mejor predictor que parsimonia lineal].
Kluge (1997) [Un ataque a todas las formas de pesaje].

Práctica 6

Alineamiento de ADN

Introducción

Aunque los datos moleculares pueden proporcionar grandes cantidades de caracteres, presentan retos en la asignación de homologías al tratarse exactamente de las mismas bases en todas las secuencias; además en muchas ocasiones las secuencias tienen un tamaño desigual. Para resolver estos problemas se han ideado los métodos de alineamiento, que buscan recuperar las posibles homologías dentro de diferentes secuencias, al utilizar una matriz de

transformaciones, entre las cuatro bases y los gaps o InDels (inserción/eliminación); así, colocando gaps las dos secuencias son equivalentes en costos y posiciones para las bases; por ejemplo, compare la secuencia 1x con cualquiera de los alineamientos 2. [1]

1x: t T C C g A A T T g g c t a c T T C C g A A t T T g G

2x: t C G A T T G C C A C T C G A T T G

2a: t - - C g A - - T g c c t a c T - C - g A - t T - g G

2b: t - C - g A - T - g c c t a c - T - C g A - t - T g G

2c: t - C - g - A T - g c c t a c - T - C g - A t - T g G

Ya que se necesitan más de tres secuencias para un análisis filogenético, es necesario el alineamiento simultáneo de múltiples secuencias. El primer método usado es el alineamiento múltiple que usa un árbol **guía** con las secuencias en los terminales; a partir de este árbol se hace el primer alineamiento para el par más cercano[2], luego optimiza los gaps, posteriormente se alinea con la siguiente terminal que sea la más cercana y el proceso se repite

[1] Ejemplo modificado de Siddall, http://research.amnh.org/ siddall/methods.

[2] En realidad es el par más similar, ya que se usa una técnica de distancia para obtener el árbol guía.

hasta llegar a la base; la idea es parecida a la optimización en un análisis filogenético.

Una nueva idea, usada principalmente por cladistas, es la optimización directa (Wheeler, 1996), donde el objetivo no es construir el alineamiento, sino directamente el cladograma; en este caso, se hacen los alineamientos locales (como en el alineamiento múltiple) y los gaps son considerados transformaciones, es decir se colocan temporalmente en los nodos. Wheeler (1996) argumenta que implementa una idea de homología dinámica acorde con el análisis filogenético. En la optimización de estados fijos (Wheeler, 1999) se usa la secuencia completa, y la matriz de transformaciones sólo sirve para asignar costos, pero no estados en el nodo. Aunque es intuitivamente muy interesante, no se ha usado en la práctica. Cualquiera que sea la idea para realizar los análisis moleculares, los resultados dependen de la matriz de transformación inicial; Wheeler (1995) desarrolló una metodología para comparar los parámetros de las transformaciones, conocida como "análisis de sensibilidad". Bajo esta idea, se hacen diferentes análisis con diferentes matrices de transformación, y se selecciona aquella que maximice algún criterio previamente seleccionado (ya sean las topologías, como el

costo de las transformaciones); hasta ahora, esta idea solo se ha utilizado en el contexto de análisis con múltiples genes o entre genes y morfología simultáneamente.

Técnicas

Si desea hacer un alineamiento múltiple rápido use preferencialmente MUSCLE sobre CLUSTAL, pero si el objetivo es la filogenia sugerida por las secuencias es mejor usar POY, ya que tiene en cuenta el árbol final. Puede hacer una exploración en CLUSTAL y posteriormente llevar sus marcos como archivos separados para ser analizados con POY.

Algunos programas (como CLUSTAL) usan un solo árbol guía derivado del cálculo de la distancia entre las secuencias; como todos los métodos basados en distancias, es dependiente del orden de entrada de los datos; MUSCLE aunque parte de un árbol guía de distancia, revisa el resultado del alineamiento, primero a nivel goblal y luego a nivel local; POY construye distintos árboles y prefiere el alineamiento que sugiere el árbol de menor costo. Aunque sea lento, POY produce mejores resultados (en cuanto a

la calidad del alineamiento con miras a evaluar la filogenia) que CLUSTAL. Giribet & Wheeler (1999) recomiendan que si se usa CLUSTAL, se prueben diferentes secuencias de entrada de los datos, es importante recalcar que el árbol usado por CLUSTAL es sólo un árbol para optimizar las secuencias, no un árbol filogenético como tal; mientras que el árbol generado por POY es un árbol de alineamiento y a la vez es un árbol filogenético. Dado que el objetivo de POY es el análisis simultáneo, a lo largo de este libro se usará POY como otro programa más de análisis. Aun así, POY puede producir los alineamientos implícitos sugeridos por cada árbol (así se lo utilizará en esta práctica), por lo que es posible usar este resultado como entrada para programas de búsquedas que usen secuencias alineadas, de usar los alineamientos, los costos usados para el alineamiento deberían ser los mismos usados para el análisis filogenético.

Al asignar los costos, hay que tener en cuenta la desigualdad triangular, es decir, que el costo de una transformación nunca puede ser mayor al de una transformación equivalente, pero que tome otros estados, si las transiciones tienen un costo de 3 y las transversiones de 1, la transformación A→ G, tendría un costo de 3,

pero si se hace de la forma A→ T→ G, el costo sería de 2 (dos transversiones). Esto haría que a los nodos se les pudiesen asignar estados no observados en los terminales.

Para facilitar el alineamiento como las asignaciones de homología, al usar POY se pueden (y preferencialmente se deben) dividir las secuencias analizadas en pequeñas secuencias o marcos; definidos mediante iniciadores universales, estructura secundaria o motivos conservados. Cuando las secuencias son muy desiguales en estos segmentos, muchas veces se prefiere eliminarlas a analizarlas (o se hace un análisis exploratorio que incluya esas secuencias).

1. En CLUSTAL/MUSCLE:
    a) Abra con un editor los archivos datos.ou.dat.
    b) Ejecute CLUSTAL con los parámetros predefinidos cual es la relación transición/tranversión definida?
    c) Modifique los parámetros de tal forma que obtenga una relación de ts/tv de (a) 1/2 (b) 1/5.
    d) Ejecute nuevamente CLUSTAL con estas modificaciones en los parámetros.
    e) Cambie los parámetros dando a los gaps un costo del doble del costo de las transversiones.

f) Ejecute MUSCLE con los parámetros predefinidos. ¿cuál es la relación transición/tranversión definida?

g) Repita el paso 5 y ejecute nuevamente MUSCLE con los nuevos parámetros.

h) Guarde el alineamiento en un archivo de texto en formato FASTA, PHYLIP o NEXUS.

2. POY cuenta con una interfaz de usuario que es relativamente rápida de dominar. Para los diferentes tipos de costos, el comando más importante es transform(). Por ejemplo, con transform(tcm(1,2)) se da peso de 1 a las sustituciones y de 2 a los gaps o a los datos morfológicos (estáticos). Matrices de transformación más complejas pueden elaborarse y luego leerse con ese mismo comando. Una de las grandes ventajas de POY es que puede ser usado en un *cluster* de computadoras.
3. En POY:

a) Abra los datos en POY usando

```
> read("datos.fas.dat")
```

coloque todos los costos iguales y realice la búsqueda

con:

```
> transform (tcm(1,1)); build( 100); select( unique);swap();select()
```

b) Transforme sus secuencias de homología dinámica a estática, use transform(static_aprox).

c) Guarde el alineamiento en un archivo de texto usando

```
> report("alin.txt",phastwinclad)
```

d) Revise el alineamiento con Winclada.

e) Repita los pasos anteriores incluyendo la matriz de datos morfológicos.

f) Repita los pasos anteriores con un alineamiento tipo FSO use transform(fixed_states).

g) Pese la matriz morfológica 2 veces el valor de los datos moleculares usando transform((static,weight:2)).

h) Cambie los parámetros de tal manera que los gaps valgan el doble que las sustituciones.

i) Elabore una matriz de costos para obtener relaciones de ts/tv de 1/2 o 1/5, léala usando transform((all, tcm:("costos.txt"))

y genere el alineamiento.

j) Haga un análisis de sensibilidad usando las matrices datos.fas.dat y datos2.fas.dat usando los mismos costos para los dos conjuntos de datos y con costos diferenciales para cada tipo de datos, revise el apéndice C, página 164.

k) Haga un análisis tradicional (build, select, swap, select) y compárelo con un análisis sin seleccionar los árboles para swap(=build, swap, select).

Pregunta(s). ¿Qué efecto tiene el select **intermedio?**

l) La secuencia analizada divídala en al menos tres marcos y repita el proceso, evalúe los tiempos usados en cada uno de los procesos.

Pregunta(s). ¿Puede hacer el análisis del punto anterior con algún comando de POY?, búsquelo y una vez lo haga repita el proceso pero fraccionando la secuencia usadas.

m) Repita los tres pasos anteriores usando ML.

Pregunta(s).

- **¿Cómo hace POY la reconstrucción bajo ML? revise el manual**
- **¿Las topolgías obtenidas vía parsimonia son iguales (o no) a las generadas por ML?**
- **¿Los alineamientos implícitos son equivalentes? use los costos de los alineamientos y los caracteres informatívos evaluados en winclada.**

n) Repita todo el proceso usando FSO.

Pregunta(s). ¿Es posible hacer FSO usando ML?

Pregunta(s) general(es).

- **Compare los alineamientos de MUSCLE y CLUSTAL, al modificar los distintos parámetros, ¿Son similares los resultados?**
- **Compare los árboles de MALIGN, POY y los obtenidos en un análisis cladístico con el programa de su preferencia. ¿Son similares los resultados?**
- **Compare sus resultados con los de sus compañeros. ¿Cúales son los costos de los árboles (reportados por POY) y las topologías?**

- **Dados los diferentes costos usados en el análisis simultáneo de morfología y datos moleculares, ¿cuál cree usted que es el resultado óptimo y por qué?**
- **Existe una disputa sobre una relación entre los juegos de costos y el soporte. Dados sus conocimientos, escriba un pequeño ensayo donde indique sus ideas, su posición y sus argumentos en esta discusión.**

Literatura recomendada

Edgar (2004) [Compara los alineamientos derivados de MUSCLE y CLUSTAL].

Frost (2001) [Un artículo empírico sobre el análisis de sensibilidad].

Giribet (2003) [Revisa la exploración de los resultados del análisis de sensibilidad vs. soporte de clados].

Giribet & Wheeler (1999) [Uno de los pocos artículos que discute explícitamente el tema de los gaps].

Wheeler (1995) [La propuesta del análisis de sensibilidad para la asignación de parámetros en los alineamientos].

Wheeler et al. (2006) [Una guía completa para POY].

Práctica 7

Modelos de evolución

Introducción

En sistemática molecular se enfrenta un conjunto de datos distinto a los caracteres morfológicos: un mismo tipo de estados (ACGT/GAP) se encuentra repetido a lo largo de toda la matriz de datos, para este tipo de datos moleculares, existen aproximaciones estadísticas, dado que el origen de muchos de los análisis filogenéticos moleculares es en biología molecular o en genética de poblaciones. Bajo la estimación estadística se asume que un modelo genera las se-

cuencias de ADN y a partir de tal modelo se estima qué tanto se ajustan los datos a las hipótesis filogenéticas. Este procedimiento se conoce como **máxima verosimilitud** (*Maximum Likelihood* en la literatura en inglés).

En los modelos moleculares se asignan tanto la probabilidad de transformar una base cualquiera en otra base, incluida esa misma base, como las frecuencias de bases en el equilibrio. El cálculo de esas probabilidades está influenciado por diferentes parámetros como por los criterios de selección; en principio, los parámetros deberían ser estimados a partir de los datos, aunque algunos autores usaron en un inicio modelos predefinidos de acuerdo a la opinión del investigador.

El modelo con el menor número de parámetros es conocido como JC o JC69, por los autores y el año en que fue propuesto (Jukes & Cantor, 1969), en este modelo se asume que la probabilidad de una base para transformarse en otra es siempre la misma y que la frecuencia de las bases es igual. A partir de este modelo se pueden agregar más parámetros que hacen más complejo el modelo en diferentes direcciones, ya sea al diferenciar entre los tipos de base (ya sea químicamente AG-CT, o cada una por separado),

diferenciar las probabilidades basados en la proporción de cada base, tener o no en cuenta o la variación dada por cada sitio al asumir que algunos sitios son más propensos a cambiar que otros (función Γ) y, finalmente, si existe o no un reloj molecular.

La información básica se puede consultar en Felsenstein (2004) o Swofford et al. (1996), este último es una referencia básica para comprender el desarrollo de los distintos modelos. Page & Holmes (1998) ofrecen un capítulo ilustrativo sobre el tema y si desea un tratamiento avanzado debe consultarse Yang (2006). La idea general de modelos también se aplica para la evolución proteíca, pero dado que casi siempre se tiene tanto la secuencia de aminoácidos como la nucleotídica, esta última es la preferida al explotar más directamente la información (al menos a nivel de variación).

Dado que los valores de las probabilidades de los datos son muy pequeñas, y que es más fácil sumar los logaritmos que multiplicar los números iniciales, para facilitar se utiliza el negativo del logaritmo natural de la verosimilitud, que es el valor tradicionalmente reportado por los programas así como en la literatura.

Debido al aumento en el número de los parámetros, los modelos

puedan explicar mejor los datos, lo que no implica que los modelos sean ***más*** realistas, por lo que la selección de un modelo no puede basarse en el aumento neto del valor de verosimilitud, sino si la mejora es (o no es) estadísticamente significativa, dado el número de parámetros usados en el modelo.[1]

Existen dos aproximaciones básicas para este problema. Dado que la mayor parte de los modelos son una especialización de otros modelos más sencillos, es decir son modelos anidados, en los cuales el modelo más sencillo es sólo un caso especial del modelo más complejo, se puede hacer una prueba jerárquica de verosimilitud (hLRT[2]) que compara la proporción en la que se incrementa la verosimilitud al agregar el parámetro; se asume una distribución χ^2 para la proporción, tomando el número de parámetros extra como los grados de libertad. El problema de esta prueba es que no es claro si la distribución χ^2 es válida, y solo permite comparar modelos anidados; la prueba es un tanto conflictiva al comparar GTR

[1]Es buena idea familiarizarse con algunas pruebas de hipótesis usando verosimilitud, por lo que puede consultar la descripción en la wiki.

[2]

$$\delta = -2ln\frac{Max[L_0(modelo - nulo|datos]}{Max[L_1(modelo - alternativo|datos]} = 2(lnL_0 - lnL_1) \qquad (7.1)$$

con GTR+I o GTR+Γ; se asume que la forma como se agregan los parámetros no sesga el resultado, lo cual no es válido; usted puede revisar esta afirmación usando programas como JModelTest o PAUP*.

Otra forma de selección del modelo óptimo está basada en la medida de la información, usando el criterio de Akaike[3], o la cantidad de información bayesiana[4]. En estos casos se calcula la cantidad de información que sugiere la topología dado el modelo y los parámetros de este. La ventaja es que se puede hacer la comparación entre modelos sin tener que tomar una secuencia particular y sin importar si los modelos están anidados.

Para los cálculos, se puede usar JModeltest, o PAUP*, o R + ape + PhyML o JModeltest + PhyML.

JModeltest es un programa de Java por lo que funciona de la misma forma en todas las plataformas de computo, la salida es en modo de texto / tablas dentro del mismo programa; el programa

[3]AIC= -2lnL + 2N, donde:
lnL es el ajuste del modelo reportado y
N el número de parámetros libres

[4]BIC=-2lnL + Nlnn, donde:
lnn, es el logaritmo natural de el costo de la secuencia

le permite mayor control sobre la forma de iniciar el cálculo, tanto desde el árbol de inicio hasta la forma de implementar el test jerárquico, desde JC hacia GTR (*forward*) o en sentido contrario (*backward*); puede usar hasta 203 modelos y de manera automática corre los análisis en múltiples procesadores, lo que hace más eficiente que PAUP* o R, además puede ejecutarlo desde la línea de comandos con modelos diferenciales lo que facilita su uso en clusters de computadores; R en conjunto con ***ape*** y PhyML no solo puede realizar el cálculo del mejor modélo de evolución molecular sino que puede graficar los resultados. Es posible que obtenga distintos modelos dependiendo de la forma como estructure su análisis.

Técnicas

Par obtener el modelo con programas como PAUP* o R debe tener un archivo de datos con las secuencias alineadas, revise la práctica 6.

En JModeltest2 + PhyML

1. Abra en un editor la matriz de datos "DNA1.phy". Revise las principales características del formato PHYLIP
2. Abra la matriz de datos en el programa JModelTest.
3. Calcule los valores de likelihood (*likelihood scores*), a partir de un árbol base de BIONJ.
4. Calcule el valor de AIC (criterio de elección del modelo de mayor ajuste a la matriz). El valor de AIC será mejor cuanto más **pequeño** sea.
5. Revise la salida de AIC y anote cuál es el modelo de mayor ajuste para la matriz de datos, así como los parámetros de este modelo.
6. Repita el análisis a partir del cálculo de criterio bayesiano (BIC). Compare sus resultados, tanto para AIC como para BCI con los obtenidos por sus compañeros.
7. Repita 3 pero use una búsqueda de FIXED BIONJ+JC y evalúe el resultado del test jerárquico.
8. Repita desde 3 con un árbol base de ML optimizado y evalúe el modelo.

Pregunta(s). ¿Existe alguna diferencia entre los mode-

los sugeridos por cada enfoque?

En R+ape+PhyML

1. Abra R y revise si tiene instalada la biblioteca ***ape*** , si es necesario, instalela.
2. Coloque como directorio activo el directorio donde tenga los datos y PhyML
3. Cree un archivo con las instrucciones[5]:

```
> library(ape) # cargamos la libreria ape
> DNA < − read.dna("alineado.phy") # leemos datos alineados en formato tipo phylip
> table(unlist(lapply(DNA, length)))## listado de tamaño de las secuencias ##Test de modelos de evolución con ape/R y modelos via phyml
> library(ape) ## cargamos la libreria ape
## con phymltest probamos 28 modelos en phyml
## ape + R hacen todo el proceso
## en Linux cambie a
```

[5]La instrucción en R para comentarios es # y por lo tanto no es ubna instrucción del programa

```
## execname = "./phyml", si es local o
## execname = "phyml", si es global
##
## en windows use:
> modelo <- phymltest("alineado.phy",
format = "sequential", itree = NULL, exclude = NULL,
execname = "phyml.exe", append = FALSE)
## use format = "interleaved" si aplica a su matriz
> print(modelo) ## con print puede revisar los valores
de AIC
> summary(modelo) ## con summary puede tener un
resumen de los valores del test jerárquico
> plot(modelo, main = 'test de modelos para ML, usando
PhyML') ## grafica con 'plot' los valores de AIC
```

4. Compare el modelo sugerido por AIC y por el test jerárquico.
5. Cargue cada instrucción por línea de comandos desde un editor de texto y si tiene todas las instrucciones escritas en un archivo "modelos.R", puede usar este archivo y ejecutar todas las instrucciones desde la línea de comandos:

```
> R -f modelos.R
```

Tanto JModeltest como R+ape usan PhyML para el cálculo de los valores de likelihood; los dos programas permiten evaluar el modelo de una manera más amigable que la línea de comandos y tener o una salida gráfica (R+ape) o una salida de texto fácilmente leíble.

En PAUP*:

1. Abra la matriz de datos "DNA1.nex".
2. use

```
> help automodel;
```

   para obtener ayuda sobre la instrucción y las opciones en uso.

3. construya un árbol tal y como lo hizo en la sección 4, use un árbol de NJ con distancia de Jukes-Cantor como inicio:

```
> set criterion=dist;
> dset distance=JC;
```

```
> hsearch;
```

Este es el árbol de inicio para hacer los cálculos para la selección del modelo.

4. use automodel para hacer una búsqueda por defecto, use:

```
> automodel AIC=yes AICc=no BIC=no modelset=j3
lset=AIC invarSites=no gammaRates=no;
```

para hacer una búsqueda con el equivalente a 3 modelos en jmodeltest; compare los resultados con la búsqueda anterior.

5. Calcule el valor de AIC para 4 diferentes modelos: JC, K80, GTR y uno de su elección. El valor de AIC será mejor cuanto más **pequeño**.
6. Para los mismos modelos y el escogido por el hLRT en JModelTest, calcule el criterio de información bayesiano. Compare sus resultados, tanto para AIC como para BIC con el de sus compañeros, y determine cual es o son los mejores modelos para esos criterios. Ordene los modelos del mejor al peor.

Pregunta(s).

- **¿Es el modelo escogido por el criterio de Akaike igual al escogido en el test jerárquico? ¿Usted esperaría que lo fuesen?**
- **¿Es igual el resultado en las distintas exploraciones realizadas? ¿Usted esperaría que fuesen iguales ó desiguales?**
- **¿Son iguales los resultados en AIC y BIC?**
- **De usar los tres programas ¿usted espera la misma respuesta?**

Pregunta(s) general(es).

- **¿Cómo se relaciona el concepto de caracteres homólogos usado en los primeros laboratorios, con el concepto de los modelos?**

- **¿Prefería usted usar siempre el mismo modelo (por ejemplo, JC)? Argumente su respuesta.**

Literatura recomendada

Posada & Crandall (2001) [presentan de manera completa cómo seleccionar modelos basándose en la maximización del ajuste].

Swofford et al. (1996) [es una referencia básica para comprender el desarrollo de los distintos modelos y los cálculos asociados].

Práctica 8

Máxima verosimilitud

Introducción

Algunos autores han sugerido que parsimonia es un modelo de evolución, pero que este no es explícito; por ejemplo que la tasa de transformaciones es baja (ver Swofford et al. (1996); Felsenstein (2004)), e independientemente de si el argumento es o no correcto (vea una posición en contra en Farris (1983) y la argumentación de Steel (2002)), estos autores sugieren que deben usarse modelos explícitos de evolución como los que se vieron en la práctica 7

sobre selección de modelos (ver página 78).

En este caso se estima el árbol (la hipótesis filogenética) que maximice la probabilidad de los datos actuales, dado el modelo evolutivo sugerido para las secuencias a analizar (existen intentos de generalizar los modelos de evolución a datos morfológicos, ver Lewis (2001)).

Uno de los principales problemas con la estimación de verosimilitud es que el valor depende de el costo de las ramas. En la actualidad, inicialmente se ignora el valor de las diferentes ramas, y luego de encontrar un árbol específico, se calcula el mejor valor de verosimilitud para una rama a la vez. Swofford et al. (1996) ofrecen una explicación extensa de los cálculos relacionados con el método.

Técnicas

Los métodos para buscar y seleccionar árboles usados en verosimilitud son básicamente los mismos usados en parsimonia: a un árbol inicial se lo mejora con permutación de ramas (ver práctica

4, página 36). Existen diferentes formas de encontrar un árbol inicial en verosimilitud. Algunos investigadores suelen empezar por un árbol de distancia (NJ: *neighbor joining* o BIONJ). El problema de este inicio es que para una configuración particular, solo existe un árbol de NJ[1]. Otra forma es la descomposición de estrella, donde se tiene una politomia basal y se trata de resolverla desde la raíz. Este método es muy lento, puesto que a diferencia de un árbol de Wagner, la magnitud del problema es grande desde el inicio. Una alternativa adicional es comenzar con un árbol al azar, pero estos pueden ser subóptimos, con lo cual la fase de permutación es muy lenta. Una última forma, es utilizar árboles de Wagner, pero basados en verosimilitud, o si se desea una respuesta más rápida, un árbol obtenido por parsimonia y luego permutar las ramas usando verosimilitud.

1. en PAUP*
    a) Antes que nada estime el mejor modelo para la matriz, use los procedimientos de la práctica 7.

[1]tal y como lo hemos recalcado previamente para los análisis de distancia el orden de entrada influye en el resultado, la topología resultante puede cambiar si se cambia el orden de entrada a la matriz.

Pregunta(s). Argumente qué criterio utilizó para seleccionar el modelo.

b) La búsqueda inicial en PAUP* hágala bajo el criterio de parsimonia, y posteriormente pase al criterio de ML y haga una pequeña búsqueda sobre los árboles generados en el análisis con parsimonia; recuerde salvar los árboles de cada modelo (incluida el costo de las ramas). Para las búsquedas puede utilizar los mismos comandos que se usaron en parsimonia, con el comando

```
> set criterion=likelihood
```

se coloca a PAUP* en modo de verosimilitud; con el comando Lset usted puede modificar los diferentes parámetros de los modelos (función Γ, distribución de bases, tipos de cambios, invariantes). Abra la matriz de datos, busque el árbol más parsimonioso.

c) Permute las ramas de ese árbol, usando el modelo obtenido en $1a$. Anote el valor de la verosimilitud (-lnL) reportado.

d) Pruebe alternativamente los siguientes modelos: JC, HKY,

F81 y TRM, usando como frecuencia para las bases la estimación empírica y sin el parámetro Γ. Reporte los valores de verosimilitud para cada modelo.

Pregunta(s). ¿Las topologías obtenidas son similares?

e) Repita el ejercicio hecho en $1b$, pero use la estimación empírica del parámetro Γ para HKY y GTR. Reporte los valores de verosimilitud para cada modelo.

f) Repita el ejercicio hecho en $1b$, pero use invariantes. Reporte los valores de verosimilitud para cada modelo.

Pregunta(s).

- **¿Las topologías obtenidas para todos los modelos son similares?**
- **¿como puede evaluar si los datos se ajustan o no al reloj molecular?**

2. en PhyML

PhyML es un programa mucho más rápido que PAUP* y por lo tanto debería ser una de sus primeras opciones, el programa cuenta con dos esquemas de instrucciones: el primero por línea de commandos (que se usa de manera indirecta al

obtener el modelo óptimo con R+ape o con JModelTest) y el segundo es una interfaz similar a la de PHYLIP, por la que puede navegar entre las distintas opciones (en la forma de menú de texto), y donde puede modificar los parámetros de búsqueda: *nni*, *spr* o una mezcla de *nni+spr* o los parámetros ligados a modelo, con tipos de sustituciones y optimizaciones de invariantes o Γ.

Para un archivo de entrada de datos denominado *archivo*, el programa genera dos archivos de salida:

archivo.phy_phyml_stats.txt (estadísticos) y *archivo*.phy_phyml_tree (árbol en notación Newick). Estos dos archivos en caso de existir pueden ser sobreescritos (R) o la salida actual puede ser añadida a una salida previa (O).

Inicialmente solo debe dar el nombre de la matriz de datos (en formato PHYLIP) y el programa le irá guiando por las decisiones a tomar. Puede avanzar al siguiente menú con +, regresar con -, o iniciar el análisis con Y.

a) Haga una corrida por defecto en PhyML.

Pregunta(s). ¿qué parámetros usa por defecto el

programa?

b) Estime el mejor modelo para la matriz, use los procedimientos de la práctica 7. Argumente qué criterio utilizó para seleccionar el modelo.

c) Busque el árbol de ML usando el modelo obtenido en 2*b*, haga una corrida con *nni*, Anote el valor de la verosimilitud (-lnL) reportado y la topología obtenida.

d) Repita 2*c* cambiando la búsqueda a *spr* y a *nni+spr*.

e) Pruebe alternativamente los siguientes modelos: JC, HKY, y GTR, para los tres modelos con y sin el parámetro Γ. Reporte los valores de verosimilitud para cada modelo.

Pregunta(s). Compare la topología y el costo de las ramas de los árboles de cada análisis.

Actualmente se puede realizar un análisis de evidencia total con DNA usando una evaluación del ML para un conjunto de datos particionado, donde cada partición puede tener su propio modelo (en realidad variaciones de GTR).

RaxML es un programa mucho más rápido que PhyML y por lo tanto debería ser una de sus primeras opciones, el

programa cuenta con un solo esquema de instrucciones: por línea de commandos.

3. en RaxML

   a) Haga una corrida en RaxML, con 10 búsquedas ras de ML.

      1) Dado un archivo "dna.phy", para usar el programa debe acceder a la línea de comandos y usar las instrucciones:

```
> raxml -m GTRGAMMA -p 12345 -s dna.phy -#
20 -n ML0
```

      Para 20 búsquedas de adición al azar de terminales, a partir de un archivo de datos "dna.phy" y con un prefijo de salida de ML0, usando el modelo GTRGAMMA.

   b) Haga una corrida en RaxML, con 10 búsquedas ras de ML para un análisis particionado.

      1) use las instrucciones:

```
> raxml -m GTRGAMMA -p 12345 -q particion-
SimpleDNA.txt -s dna.phy -n ML01
```

donde los límites de cada partición se encuentran en el archivo particionSimpleDNA.txt.

2) si prefiere hacer el análisis particionado pero estimando las frecuencias de bases para cada partición de manera independiente debe usar:

```
> raxml -M -m GTRGAMMA -p 12345 -q sim-
pleDNApartition.txt -s dna.phy -n ML02
```

Pregunta(s).

- **Compare sus resultados con los de sus compañeros. ¿Cúal fue el mejor valor de verosimilitud, independientemente del modelo?**
- **¿Influye en sus resultados la selección del mecanismo de reorganización ramas?**
- **¿Cómo considera los tiempos de ejecución comparados con otros métodos como parsimonia? ¿Difieren los resultados con los de parsimonia?**

Pregunta(s) general(es).

- **Se ha propuesto que se pueden usar modelos para el análisis morfológico. Analice los puntos favorables o desfavorables e indique su punto de vista sobre el tema.**
- **¿Qué implicaciones tiene el uso de modelos en el concepto de homología y carácter presentado anteriormente?**

Literatura recomendada

Swofford et al. (1996) [Una descripción muy completa de la forma como se calculan las versomilitudes y los métodos de verosimilitud para encontrar árboles].

Lewis (2001) [Introduce el uso de modelos en morfología].

Práctica 9

Consensos

Introducción

Por las prácticas anteriores y por la literatura consultada, usted ha notado que en muchas ocasiones se produce más de un cladograma como respuesta, para resumir la información contenida en los diferentes cladogramas se puede usar un consenso, que de información sobre los agrupamientos en los diferentes árboles iniciales. Swofford (1991) y Nixon & Carpenter (1996) ofrecen una discusión extensa sobre los árboles de consenso con dos visiones

diferentes. Es importante recalcar que la topología del consenso es un resumen de los cladogramas y **no** es una **hipótesis de filogenia**.

En el consenso estricto, solo se incluyen los nodos compartidos por todos los cladogramas, este método es el más conservativo; otros métodos presentan la información de algunos nodos, como el consenso de la mayoría, que presenta los nodos que se encuentran por encima del valor de corte. Otros dos tipos de consensos, que en general no son usados, son el de Bremer y de Nelson, sus respuestas pueden ser similares al consenso estricto y sólo difieren de este en casos muy particulares.

El consenso de Adams se basa en operaciones de conjuntos entre los nodos; es muy útil para mostrar cuáles son los taxa que producen inestabilidad en el cladograma, pero puede producir nodos que no se encuentran en ninguno de los cladogramas originales. Kearney (2002) ofrece una buena discusión sobre cómo combinar los resultados del consenso estricto y el de Adams.

Los consensos de la mayoría son muy populares, especialmente en los análisis moleculares, aunque su uso es **muy** discutible (Sharkey

& Leathers (2001) vea también Goloboff & Pol (2005)). En este tipo de consenso se hace un conteo de las veces que el nodo aparece en los diferentes árboles: si el nodo aparece en al menos la mitad de los árboles, o en el valor de corte seleccionado, el nodo es incluido en el consenso, es posible hacer cortes más estrictos que el 50 %. Es una convención colocar en forma de porcentaje la cantidad de veces en las que el nodo apareció.

Como se mencionó en la introducción de la práctica de búsquedas, los métodos de consenso también se usan para dar respuestas parciales, como búsquedas de *jackknife*, o el doble consenso de Goloboff & Farris (2001); o como resultados en el caso de los análisis bayesianos que usan el árbol obtenido en el consenso de la mayoría (Huelsenbeck & Ronquist, 2001).

Técnicas

En la mayoría de los programas, los consensos estricto y de la mayoría están implementados (por ejemplo, TNT, PAUP*, POY o WinClada). En algunos casos, la implementación del consenso de la mayoría es sólo hasta el 50 %, y el usuario decide que tan

estricto hace el corte, eliminando los nodos que estén por debajo del valor de corte (un consenso estricto sólo retendría los nodos con soporte del 100 %).

En otros casos, los programas pueden elaborar el consenso de la mayoría, pero no reportan los porcentajes de aparición en los nodos (por ejemplo en TNT), o simplemente los visualizan pero no salvan directamente el árbol con los porcentajes incluidos (TNT). Es importante tener en mente que si va a trabajar en TNT, debe usar algunos macros para recuperar los reportes de frecuencia.

Finalmente hay que alertar un poco acerca de la resolución de los cladogramas usados para generar los consensos. La mayor parte de los programas usa los árboles perfectamente dicotómicos, por lo que pueden incluirse ramas no soportadas; así, al hacer un consenso es necesario eliminar tales árboles con ramas no soportadas. La mayoría de los programas actuales incluyen opciones para controlar la salida: incluir o no incluir árboles con nodos de costo cero (por ejemplo TNT, NONA, WinClada, MacClade y PAUP*), otros no (Component).

A partir de la matriz de datos: datos.consenso.dat, en Win-

Clada, PAUP*, POY o TNT:

1. Abra la matriz de datos, y realice una búsqueda convencional (ver práctica 4). Guarde los árboles encontrados.
2. Realice y salve un consenso estricto, un consenso de la mayoría al 50 %, y en caso de que el programa lo permita, incluya 75 % y 90 % de corte. En todos los casos reporte el número de nodos presentes en el árbol, y compare los grupos encontrados. Recuerde anotar las frecuencias de los grupos.
3. Para WinClada use los menús correspondientes en la sección de **winclados**; el menú **Trees** tiene la entrada **consensus compromise**, donde hay la posibilidad de hacer consenso estricto, consenso estricto eliminando nodos no soportados (nelsen, que no debe confundirse con el consenso de Nelson) y consenso de la mayoría[1]. Para salvar los árboles siga las instrucciones previas usadas en la práctica 3.
4. En PAUP*:
    - Siga el mismo esquema, realice una búsqueda y calcule

[1] una vez calculado el consenso de la mayoría, WinClada tiene algunos problemas posteriores en en la forma como los grafíca, por ejemplo, los *Hashmarks* no se pueden activar (no son dibujados), y modificar la topología produce cambios inesperados en las frecuencias de los nodos.

primero el consenso por defecto y posteriormente los consensos estricto y de Adams; guárdelos en un archivo, use la instrucción

```
> contree
```

- En caso de duda, use el nombre del comando y el signo **?** (por ejemplo, contree ?) para que evalúe las opciones del comando; si desea guardar el consenso directamente en un archivo de árboles **debe** hacerlo desde esta instrucción.

5. En TNT use:

```
> nelsen
```

- para obtener el consenso estricto, si desea que el consenso sea el último árbol, use nelsen*
- Para el consenso de la mayoría, use la instrucción majority y majority* respectivamente.
- Para guardar los árboles de consenso directamente, después de calculados use

```
> save {strict}
```

o

```
> save {majority}
```

- En todos los casos **debe** haber abierto previamente el archivo de árboles (tsave* archivoArboles.tre). En la versión de menú puede usar **Trees** y la entrada **consensus** y escoger los diferentes tipos de consenso.

6. En POY, se pueden realizar consensos con la instrucción

```
> report(consensus)
```

- para el consenso estricto y report(consensus:x) para consensos de la mayoría, donde x es un entero mayor que 50.
- Si desea guardar los resultados en un archivo debe colocar el nombre entre comillas dobles. Por ejemplo:

```
> report ("mayoria.txt", consensus:75)
```

guarda el consenso de la mayoría al 75 % en el archivo

mayoria.txt.

7. En R, cargue primero las bibliotecas: ***ape*** y ***jrich*** .

   - En PAUP* haga un búsqueda y guarde los árboles iniciales en formato Newick en un archivo arboles.phy con las instrucciones:

```
> hsearch; savet file=arboles.phy format=Newick;
```

   - Asigne los árboles salvados a un objeto denominado arboles:

```
> arboles <- read.tree("arboles.phy")
```

   - Si lo desea, puede graficarlos con el comando plot.
   - construya y grafique el consenso de la mayoría y el consenso de la mayoría pesado Sharkey et al. (2013).

```
> plot(wconsensus(arboles,collapse=F));
plot(wconsensus(arboles,collapse=T))
```

   - compare las dos topologías y mire los grupos que fueron colapsados.

Pregunta(s) general(es).

- **Compare las topologías de los árboles encontrados: ¿difieren los resultados entre los programas? Revise si las frecuencias de los nodos comunes son iguales en los diferentes resultados.**
- **Al asignar las transformaciones (sinapomorfías) de cada nodo sobre el consenso, ¿cómo lo haría usted? Compare su aseveración con la implementada en los programas (recuerde la práctica de matrices).**
- **¿Recomendaría usted el uso de consensos de mayoría como herramienta para resumir la información de los árboles iniciales?**

Literatura recomendada

Goloboff & Farris (2001) [Presenta las técnicas de consensos rápidos].

Miyamoto (1985) [presenta una crítica hacia la interpretación de los consensos en clasificaciones].

Nixon & Carpenter (1996) [Una discusión clásica sobre consensos].

Sharkey & Leathers (2001) [Una crítica al uso y abuso del consenso de mayoría].

Práctica 10

Soporte en parsimonia y ML

Introducción

Luego de obtener los cladogramas, puede ser importante saber qué tan fuerte es la evidencia que da soporte a un nodo. Es necesario diferenciar la **evidencia que soporta** el nodo, es decir las transformaciones (sinapomorfías), de la **fuerza del soporte** es decir la diferencia entre las agrupaciones encontradas con posibles

agrupaciones alternativas. Aquí se usará soporte en ese segundo sentido y para el primer acercamiento, referirse a la práctica 4 en página 36.

Existen diferentes métodos para medir el soporte, unos basados en remuestreo al azar de los caracteres y otros en el uso de árboles subóptimos. En los métodos de remuestreo se toma la matriz original, se perturba, y luego se analiza. Al final, se hace un consenso de la mayoría, donde la frecuencia de aparición de cada nodo (reportada como porcentaje) indica la cantidad de evidencia favorable para este. Los métodos de remuestreo más populares, especialmente en los análisis moleculares, son el **Bootstrap**, donde se construye una matriz del mismo tamańo de la original usando caracteres de la matriz original tomados al azar y **Jackknife** donde cada carácter puede ser borrado independientemente (jac y sim) o repesado (sim). En estos dos métodos, la probabilidad de seleccionar cada carácter es independiente de los demás y cada carácter sólo es muestreado una vez. Aunque eso permite que los caracteres no informativos y los autapomórficos no afecten el resultado, estos se hacen dependientes de la probabilidad usada para muestrear cada carácter.

Otra forma de medir el soporte es utilizar árboles subóptimos (Bremer, 1994). En este método se hace un consenso estricto con esos árboles subóptimos, y eventualmente, cuanto mayor sea la diferencia entre los árboles óptimos y subóptimos, mayor será el número de nodos encontrados en los árboles óptimos que desaparecerán. El objetivo es contar la diferencia de costo necesaria para que el nodo colapse.

Las mediciones del soporte con remuestreo (usando consenso de la mayoría) o con cladogramas subóptimos (usando soporte de Bremer previamente descrito) sólo tienen en cuenta la cantidad de evidencia favorable, y aunque un nodo puede parecer bien soportado, es posible que nodos que aparecen con alto valor de soporte, tengan muchos caracteres a favor, pero también muchos en contra. Para resolver este problema, en remuestreo se ha elaborado el índice GC Goloboff et al. (2003), que es la diferencia de la cantidad de veces que aparece el grupo (el soporte usual) y el número de veces que aparece el mejor grupo alternativo. Para el soporte de Bremer relativo Goloboff & Farris (2001) la idea es similar: comparar el costo de los árboles que favorecen el grupo y los que lo contradicen.

El concepto de soporte puede extenderse al análisis de particiones. En este caso, se mide qué tan soportado está un nodo en una parte de los datos (una partición, que puede ser, por ejemplo, uno de los diferentes genes analizados). Si el nodo esta presente en el consenso de los árboles más parsimoniosos de la partición, el soporte se halla de la forma convencional; si por el contrario el nodo no está presente, se busca el árbol más parsimonioso que contenga el nodo, y la diferencia de pasos entre este árbol y el árbol más parsimonioso de la partición es el soporte de Bremer negativo del nodo (es negativo, indica qué tan contradicho es el nodo). Los soportes por particiones permiten detectar cuáles son las particiones que sugieren el nodo y cuales lo contradicen. De esta forma, no solo se mide el soporte, sino que es posible observar la congruencia por nodos entre las particiones. Aunque al sumar los valores de Bremer de las particiones el resultado puede ser igual al valor del soporte de Bremer, esto no siempre sucede.

La medición de soporte utiliza conjuntos de árboles, así que el problema en este caso es cómo conseguir ese conjunto de árboles. Las perturbaciones de la matriz están implementadas en casi todos los programas, por lo que el proceso puede realizarse de forma au-

tomática. En algunos casos los parámetros de la perturbación son difíciles de aclarar. En *bootstrap* no existe problema, puesto que se toma de la matriz un carácter al azar hasta completar una matriz del mismo tamaño de la original, así no existe diferencia entre las diferentes fuerzas de permutación. El problema es que en las matrices hay caracteres no informativos que pueden sesgar la matriz producto de la permutación. Otro inconveniente es que la mayoría de los datos moleculares son secuencias sucesivas de caracteres, y los morfológicos no es fácil entender cómo están relacionados. El método exige una distribución homogénea de la información y que el muestreo sea al azar.

Con *jackknife* y permutación simétrica, cada carácter es alterado independiente de los demás, con lo que autapomorfías y caracteres no informativos no influyen en la distribución final. Además, los métodos basados en permutaciones exigen que los resultados de la matriz permutada sean confiables. Es decir, búsquedas estrictas. Esto hace que los métodos consuman mucho tiempo. Este problema se ha solucionado usando muchas búsquedas muy superficiales, y usando el consenso estricto (o uno muy fuerte) de las búsquedas independientes (ver Farris et al. (1996); Goloboff & Farris (2001)),

aunque eso podría disminuir la frecuencia de los nodos con soportes bajos (en el dado caso de que esos sean de interés).

Es importante notar que cuando se hacen remuestreos con POY, en el caso de los datos moleculares, los fragmentos completos de ADN son usados como caracteres, y no cada base particular (como se hace tradicionalmente con una matriz previamente alineada), por lo que jackknife puede eliminar fragmentos completos de datos moleculares.

El soporte de Bremer utiliza árboles subóptimos. Ya que muchos programas no lo tienen definido explícitamente, es necesario recurrir a técnicas externas para poder hacer la medición. Además, la mayoría de árboles subóptimos se buscan usando como fuente uno de los árboles óptimos y luego permutando las ramas y reteniendo los cladogramas que cumplan con el costo máximo especificado. Esto genera dos problemas: el primero, es que la mayor parte de los árboles pueden pertenecer a un mismo **vecindario** o **isla** de árboles; el segundo es que muchos de los cladogramas que pueden colapsar el nodo, en realidad son cladogramas que no presentan evidencia de agrupamiento, es decir, no hay sinapomorfías en los nodos. Otra forma de realizar el soporte de Bremer es hacer

búsquedas del árbol más parsimonioso que no contenga el grupo al cual se desea medir el soporte. Una ventaja de esto, es que proporciona una medición directa, aun para nodos no presentes en el árbol más parsimonioso. La desventaja más notoria es que hay que realizar muchas búsquedas para medir el soporte de cada nodo.

Técnicas

1. **En WinClada, PAUP*, POY y TNT:**
   a) Abra la matriz de datos datos.soporte.pars.dat, realice **bootstrap** y **jackknife** (con los valores por omisión y usando corte del 36 %). Haga búsquedas simples para obtener los resultados durante la práctica (50 réplicas). TNT usa el comando resample para los diferentes métodos de soporte basados en permutación de la matriz. Si obtiene soportes relativos superiores a 100 %, eso indica que los árboles subóptimos no son suficientes para evaluar el nodo, por lo que debe repetir el cálculo pero con un subop más grande; repita el proceso al menos tres veces para evitar que los árboles subóptimos sean

muestreos de unas pocas "islas" de árboles.

b) Compare sus resultados con los de sus compañeros.

2. **En TNT y NONA:**

a) Abra la matriz de datos, búsque el árbol más parsimonioso.

b) Retenga 1000 árboles y acepte subóptimos hasta 5 pasos más largos.

c) Haga 500 réplicas de Wagner sin permutar ramas, reteniendo solo un árbol por réplica. Luego llene la pila de árboles haciendo permutación de ramas, use

```
> bbreak
```

en TNT, y en NONA:

```
> max*;
```

En NONA y TNT el soporte de Bremer se obtiene con

```
> bsupport
```

debe haber calculado previamente los árboles subóptimos con

```
> subop X;mult
```

Si usa la instrucción con un asterisco bsupport*
el soporte calculado es el relativo en forma de porcentaje; también puede usar bsupport [; o bsupport];

d) Con los árboles obtenidos calcule el soporte de Bremer, absoluto y relativo.

e) Repita el proceso, pero reteniendo árboles 7 pasos más largos.

3. **En POY:**

a) Calcule el soporte de Bremer con la instrucción:

```
> calculate_support()
```

usando los árboles más cortos que no contengan ca-

da clado. Con diferentes parámetros se implementan el bootstrap y el jackknife; en esos casos es bueno incluir el tipo de búsqueda,en general simple, durante el remuestreo, por ejemplo:

```
> calculate_support(bootstrap:100, build(10),swap())
```

calcula 100 réplicas de bootstrap, y en cada una se construyen 10 arboles de Wagner, mejorados con permutado de ramas, y

```
> calculate_support(jackknife(resample:100, remove:33.3),build(10),swap())
```

calcula 100 réplicas de jackknife eliminando el 33.3 % de los caracteres (por omisión se elimina 36 %) y las búsquedas son iguales a las del ejemplo de bootstrap.

Pregunta(s). **POY** **Al remover fragmentos completos de ADN, ¿puede influenciar los cálculos de soporte usando métodos de remuestreo?**

4. **En** **RaxML:**

a) Abra la matriz datos.soporte.pars.dat en un editor de texto y conviertala en un formato que RaxML pueda leer, use winclada/mesquite de ser necesario.

b) Para un análisis no particionado use los comandos:

```
> ./raxml −f i −s datos.phy −n resultados −m
GTRGAMMA −b 1234 −# 20
```

donde:

- -f i indica que será un análisis de boostrapping tradicional
- -s es el archivo de datos
- -n es el nombre de los resultados
- -m es el modelo
- -b es el número usado para el generador de números al azar
- -# es el número de réplicas.

c) Al finalizar revise los archivos llamados:

RAxML_info.resultados

RAxML_bootstrap.resultados

d) Para un boostrapping rápido use la línea de comandos:

```
> ./raxml −f a −s datos.phy −n fastboot −m
GTRGAMMA –x 1234 −# 100
```

donde los cambios de las instrucciones para un bootstrapping tradicional son -f a -x en vez de -f i -b.

Para un análisis particionado solo debe adicionar a la línea de comandos la instrucción –q archivo-particiones.

5. En PAUP*

   a) En este programa solo se hacen permutaciones sobre la matriz; con bootstrap se configura y ejecuta bootstrap, mientras que con jackknife se hace lo propio para jackknife; recuerde que PAUP* es más lento que TNT, por lo que los comandos de búsqueda y el número de réplicas deben ser acordes con sus tiempos.

6. **En TNT:**

   a) Abra la matriz "datosParticiones.dat" en un editor de texto. Note al final la presencia de la instrucción

```
> blocks
```

esta instrucción le permite definir grupos de caracteres

en TNT.

b) Abra la matriz con TNT, mire cuantos bloques de datos están definidos y ejecute el macro brempart.run, con el número de particiones definido (por ejemplo para 3 particiones, run brempart.run 3;). El macro le reporta para cada nodo el soporte de Bremer para los datos completos, y para cada partición definida. Para ver los valores abra la matriz y el árbol de salida (BremPart.txt) y escriba

```
> ttags;
```

Compare los valores.

Pregunta(s).

- **Compare con sus compañeros sus resultados de soporte basado en permutaciones y en soporte de Bremer.**
- **¿Los nodos con mayor soporte son los mismos?**
- **¿Existe correlación entre los diferentes métodos?**
- **¿Usted sugeriría (o no) el uso de soporte relativo? Explique las razones de su escogencia.**

Práctica 11

Soporte en ML

Introducción

Tal y como se hizo para el análisis de parsimonia (práctica 10), bajo ML es importante conocer la evidencia de cada nodo, máxime cuando en ML no hay transformaciones, por lo que los valores de soporte son indicadores de la **evidencia** del nodo. Los métodos para medir el soporte están basados en remuestreo de los caracteres; así, tanto *bootstrap* como *jackknife* son aplicables a datos moleculares, aunque el método preferido es el *bootstrap*; dado que

no se usa una distribución en particular se denomina ***bootstrap* no paramétrico** en contraposición al **paramétrico**, que se basa en la simulación de los datos dado un modelo particular. Algunas evaluaciones empíricas han mostrado que el análisis depende fuertemente del modelo calculado y del árbol obtenido a partir de tal modelo (Felsenstein, 2004; Antezana, 2003).

Técnicas

La medición de soporte usando *boostrap* no paramétrico ya fue discutida en la sección de soporte en parsimonia; el *boostrap* paramétrico es una prueba de simulación que depende del modelo encontrado como estimador de la evolución de las moléculas (Felsenstein, 2004); la prueba busca evaluar que pasaría si obtenemos más datos a partir del mismo modelo, el cual es *cierto*. Con esta idea se simulan datos a partir del árbol que se obtiene con los datos **reales**; a partir de estos datos simulados se estiman los árboles, los cuales en conjunto son resumidos con un consenso de la mayoría que sirve como estimador de los soportes de los nodos. Las secuencias simuladas se obtienen con los mismos parámetros y con

el mismo modelo que la hipótesis inicial.

1. Con JModeltest:
   a) Para la matriz datos.param.dat calcule el mejor modelo.
2. En PAUP*:
   a) Obtenga el mejor árbol con cualquiera de los programas usados para análisis de ML, preferencialmente PAUP*; para salvar un árbol en formato Newick con los costos de las ramas, use la instrucción:

```
> savet format=phylip brlens=yes;
```

   b) Calcule el soporte de boostrap no paramétrico para el árbol generado, use 100 réplicas.
3. Seq-Gen no esta compilado para ningún sistema operativo, puede obtener el código fuente desde http://tree.bio.ed.ac.uk/software/seqgen/ y compilarlo. Revise las prácticas anteriores y la sección de programas (página 139), Con Seq-Gen para obtener 10 réplicas -n10 de secuencias de 27 bases -l27, use HKY -mHKY con los parámetros de Ts/Tv y frecuencias

de base que trae por defecto el programa[1], con una distribución Γ con un valor α de 1.6985 (-a1.6985) en formato NEXUS, a partir del árbol salvado como arbol.tre; use la línea de comandos:

> $seq\text{-}gen - mHKY - n10 - l27 - a1.6985 < arbol.tre > sec.nex$

Escoja el archivo de árboles que obtuvo con PAUP* y especifique la salida.

Usted puede tener un archivo de instrucciones e insertarlo despues de cada secuencia con la instrucción -xInstruc.txt en Seq-Gen.

a) En Seq-Gen simule 10 secuencias con el árbol producto del análisis bajo ML[2] y con el modelo generado en $1a$.

4. Con PAUP* o con el programa que uso en $2a$:

a) Analice el archivo de simulaciones obtenido en 3, usan-

[1]En seq-gen no existe el modelo JC, por lo que debe usar HKY con ts=tv y frecuencias iguales.

[2]También puede usar un árbol con constricción, es decir forzando la monofilia de un grupo en particular, si la pregunta está limitada a un grupo particular.

do los mismos comandos que uso en $2a$, recuerde que tiene 10 réplicas, por lo que debe incluir los comandos al final de cada matriz, usando un bloque de PAUP*; dado que Seq-Gen genera un archivo Newick/Phylip, en PAUP* use la instrucción

```
> tonex from=archivo.phy to=archivo.nex format=relphylip
```

para convertir entre formatos, revise con tonex ? las opciones requeridas.

b) Obtenga el consenso de la mayoría para los árboles generados.

Pregunta(s) general(es).

- **Compare con los valores obtenidos para la misma matriz usando soporte de *bootstrap* paramétrico y no *paramétrico*, ¿Son equivalentes los resultados?**
- **Repita la comparación pero con los resultados de sus compañeros.**
  - **Dado el número de réplicas usado, ¿obtienen resul-**

tados similares de soporte?

- **¿Variará el soporte al aumentar el número de réplicas?**

- **Dados los métodos de soporte usados, ¿usted qué tipo de soporte recomienda?**

Literatura recomendada

Antezana (2003) [Una visión crítica al uso del boostrap paramétrico].

Felsenstein (2004)[Tiene un capítulo descriptivo de los análisis de boostrap].

Práctica 12

Análisis Bayesiano

Introducción

Aunque tiene una larga tradición en análisis de probabilidades, el análisis bayesiano es la forma más reciente de generar filogenias. El análisis está basado en el teorema de Bayes (1763), donde se asignan probabilidades posteriores (en este caso la probabilidad de que el nodo en el árbol sea correcto) a determinadas soluciones de los datos (en este caso hipótesis filogenéticas) partiendo de probabilidades asignadas *a priori* y dependiendo de la frecuencia de

recuperación de una solución, la cual es asimilada a la probabilidad posterior en el teorema de Bayes.

A pesar de que la asignación de las probabilidades *a priori* es un tema crítico dentro del cálculo de las probabilidades posteriores y son difíciles de defender, generalmente se asume que todos los árboles son igualmente posibles (una inferencia dura, dado que algunas soluciones podrían ser biológicamente imposibles, o algunos nodos pueden no estar soportados) y la verosimilitud de las soluciones se calcula bajo algún modelo de evolución (los mismos modelos que se utilizan en máxima verosimilitud)(Huelsenbeck & Ronquist, 2001).

El análisis bayesiano examina múltiples soluciones que pueden ser obtenidas dados los datos, en tiempos considerablemente cortos con respecto a otras optimizaciones como ML, en las cuales las búsquedas heurísticas son la única opción cuando se trata de matrices datos con un gran número de taxa. Aunque algunas de las ventajas del análisis bayesiano pueden parecer atractivas, se debe tener en cuenta que tanto la asignación de los priores, la selección de modelos de evolución y la selección final de los árboles obliga a hacer afirmaciones muy fuertes acerca de los procesos.

Por ejemplo, algunas de las afirmaciones que justifican el uso de MCMCMC como muestreo parecen no cumplirse (e.g., que las probabilidades de cada grupo sean equivalentes a las halladas en un *bootstrap* paramétrico o no paramétrico), una implementación apropiada continúa siendo tema de discusión dentro del análisis filogenético (Alfaro et al., 2003; Alfaro & Holder, 2006). Otro argumento en contra del análisis bayesiano es que la consistencia estadística, que argumentan los defensores de maxima verosimilitud, no parece ser aplicable a la estimación bayesiana Goloboff & Pol (2005).

Técnicas

Dado que el análisis bayesiano debe explorar las posibles soluciones de los datos; esto es, calcular las probabilidades posteriores de todos los árboles, necesita métodos de cálculo muy poderosos. Para resolver esta exigencia, el análisis bayesiano utiliza una cadena de Markov-Montecarlo bajo el algoritmo de Metropolis-Hastings (resumido todo como MCMCMC o MCMC). El algoritmo MCMC funciona básicamente seleccionando y desechando so-

luciones (árboles filogenéticos), siguiendo una caminata aleatoria; inicialmente, al escoger un árbol determinado este es perturbado (modificado) azarosamente, lo que genera un nuevo árbol, que es rechazado o aceptado dependiendo de su probabilidad, la cual se calcula por medio del algoritmo de Metropolis & Hastings; si este árbol es aceptado, sufre perturbaciones adicionales. Uno de los problemas que enfrenta el método de MCMC es que la frecuencia con que se encuentran determinados árboles es dependiente de sus probabilidades posteriores, lo que genera que las cadenas queden estacionadas en determinadas zonas del espacio de soluciones.

El ajuste de los priores por omisión es una distribución plana de *Dirichlet* donde todos los valores son 1.0. Este ajuste es apropiado si se desean estimar los parámetros desde los datos sin asumir un conocimiento *a priori* acerca de los valores de estos parámetros. Sin embargo, MrBayes permite ajustar los valores de las frecuencias nucleotídicas como iguales; por ejemplo, en el caso de estar usando modelos como JC o SYM, se puede ajustar un prior específico que ponga más énfasis sobre la igualdad de las frecuencias nucleotídicas de la que está por omisión.

Un problema con MrBayes es que durante el muestreo no se exa-

mina el soporte de las ramas y dado que la respuesta es un consenso de la mayoría, las ramas que no tienen soporte pueden aparecer con probabilidades posteriores altas (Goloboff & Pol, 2005).

Para leer los datos MrBayes utiliza execute y el nombre del archivo, al igual que PAUP*, Mrbayes reconoce los comandos con las tres o cuatro primeras letras, por ejemplo, puede abrir los datos sólo con exe. Con mcmc MrBayes inicia la búsqueda en automático utilizando los ajustes que estén predefinidos. Cambie el número de cadenas nchains, el número de generaciones ngen y la temperatura temp seguiendo con el comando mcmcp; por ejemplo,

```
> mcmcp ngen=1000
```

para ajustar el número de generaciones a 1.000.

Para ajustar la frecuencia de muestreo la cadena, use

```
> mcmcp samplefreq
```

Por omisión, la cadena es muestreada, cada 100 generaciones, este será también el mismo número en que se reportarán los resultados en el archivo de salida.

Si desea ajustar frecuencias iguales para modelos como JC o SYM, use el comando

```
> prset statefreqpr=fixed(equal)
```

.

Si desea ajustar un prior específico que enfatice la igualdad de las frecuencias nucleotídicas, puede usar

```
> prset statefreqpr=dirichlet (10,10,10,10)
```

o un mayor énfasis aún con

```
> prset statefreqpr=dirichlet (100,100,100,100)
```

Con showmodel puede ver en cualquier momento cómo esta configurado el modelo de sustitución nucleotídica a usar; help muestra una lista de los comandos disponibles en MrBayes; al igual que otros de los programas usados con help comando, se obtiene una información de ayuda sobre este comando en particular. Con help lset se obtiene una lista similar al obtenido con help, pero al final una tabla muestra los ajustes actuales y predefinidos del modelo

molecular. Para ver los ajustes predefinidos de las búsquedas use help MCMC.

1. A partir de las secuencias alineadas use JModelTest para evaluar cuál es el mejor modelo de evolución del gen (ver práctica 7), revise el modelo propuesto y busque la equivalencia para MrBayes en la página: http://mrbayes.sourceforge.net/.
2. Realice una corrida en MrBayes con los modelos JC y con el (los) modelos propuestos por JModelTest, inicialmente con pocas generaciones (1000), pocas cadenas (2) y la "temperatura" por defecto.
3. MrBayes produce dos salidas, un archivo .p donde se imprimen los parámetros del modelo de evolución (este archivo está delimitado por tabuladores y es fácilmente exportable a programas para gráficas) y un archivo .t donde se reportan los árboles y los costos de las ramas. Grafique el archivo de salida .p con el programa Tracer[1]:
   a) A partir de la gráfica decida cúantos árboles desea eli-

[1]Este es un programa en Java, baje desde http://tree.bio.ed.ac.uk/software/tracer/ el ejecutable para su sistema operativo.

minar antes de hacer el consenso de la mayoría como resumen.

Pregunta(s). ¿Cree que el consenso estricto de los árboles obtenidos serviría? Argumente su respuesta.

b) Aumente el número de generaciones a 100.000 y repita el análisis.

4. Repita los procesos al menos 3 veces, compare las topologías resultantes y los valores del consenso de la mayoría obtenidos para el análisis bayesiano.

Pregunta(s) general(es).

- **¿Podría dejar que MrBayes seleccione el mejor modelo?**
- **Se ha propuesto que en el análisis bayesiano se sobreestiman las ramas sin soporte al generar topologías muy resueltas. ¿Cómo solucionaría usted tal problema?**

Literatura recomendada

Archibald et al. (2003) [Un "primer" de fácil lectura]

Goloboff & Pol (2005) [Una crítica muy completa al análisis bayesiano].

Huelsenbeck & Ronquist (2001) [Una muy buena presentación del análisis bayesiano por sus principales defensores].

Apéndice A

Programas de cómputo

Hoy en día los análisis filogenéticos usan decenas, cientos a miles de terminales y de caracteres, por lo que se requiere gran cantidad de cálculos. Aunque es posible hacerlos a mano, esto se hace imposible para más de 10 taxa y una veintena de caracteres. Esta sección busca familiarizar al estudiante con los programas más comunes, aparte de los usados durante las distintas prácticas. Es casi un truismo que la selección de programas depende de la(s) plataforma(s) utilizada(s), el problema a resolver, y en menor grado de los fondos disponibles, gratis o "por unos pocos dólares" se puede tener un repertorio apropiado de programas, la condición central a tener en cuenta es la **velocidad** y la

eficiencia. La **facilidad de manejo** en la gran mayoría de casos es secundaria; aunque la curva de aprendizaje puede ser lenta y tortuosa, las habilidades ganadas hacen que los análisis en el largo plazo sean más fáciles de ejecutar; a eso se suma la posibilidad de manejar "lenguajes" por casi todos los programas. Es posible que al final del proceso todo sea asunto de un par de instrucciones, una vez que ha perfeccionado sus habilidades y que posee archivos de instrucciones preajustadas a sus gustos y necesidades.

En la medida de lo posible, familiarícese con la línea de comandos, casi todos los programas interactuan de esa manera y lo aprendido le permitirá analizar los datos en un menor tiempo ya que podrá hacer análisis a medida, más alla de lo que permita la interfaz gráfica y podrá hacer los cálculos en paralelo, algo que no es fácil o posible desde la interfaz gráfica.

A.1. Editores y manejadores de búsqueda

A.1.1. WinClada

Autor: Nixon, 2002.

Plataforma: Windows 9x o superior [Funciona MUY bien en wine dentro de Linux].

Disponibilidad: Shareware (30 días de prueba), tiene un costo de US 50.00, http://www.cladistics.com.

El programa require de NONA para realizar las búsquedas (PIWE o Hennig86 son deseables, pero no obigatorios).

Aunque es un programa obsoleto, es uno de los programas más útiles, tanto para principiantes como para iniciados. El programa tiene dos entornos para manejar matrices y árboles. El manejo e impresión de árboles permite que se tengan imágenes listas para publicación con pocos pasos y sin necesidad de retoque posterior. Tiene múltiples "conflictos" con el manejo de imágenes que pueden ser incómodos; además, el usuario debe tener presente que el programa numera por defecto (taxa, caracteres y árboles) desde cero (la manera lógica para los programa-

dores), o desde uno, por lo que se debe tener cuidado al escribir el texto y referenciar la gráfica.

WinDada **Matrix**: Le permite cambiar algunos aspectos de la matriz (crearlas, cambiar tamaño, salvarlas, fusionarlas).

Edit: Lo más importante es que permite que los datos sean o no modificables. Además, permite adicionar polimórfismos.

Terms/Chars: Estos menú le permiten modificar cosas específicas de terminales y caracteres, tales como nombre y orden, seleccionarlos, borrarlos, adicionar.

View: Opciones de presentación. Además, puede ver estadísticos de los caracteres, aditivos, activos, pasos mínimos y máximos; y le permite intercambiar modo numérico o IUPAC (para DNA).

Output: Le permite exportar en formatos diferentes, o información variada sobre los caracteres.

Key: Si tiene los caracteres nominados, funciona como clave interactiva.

Analize: Tiene diferentes entradas para búsquedas y cálculo de soporte. En **heuristics**, es posible configurar la cantidad máxima de árboles a retener, la cantidad de árboles por cada réplica (árbol de

Wagner+TBR) (i.e. estrategias NONA/PAUP) y otros detalles de la búsqueda, como especificar una semilla para los números aleatorios (0 es el tiempo interno de la computadora). En **ratchet**: se puede modificar la cantidad de iteraciones de ratchet, el número de árboles a retener por iteración y el número de búsquedas de ratchet, bien sean secuenciales o simultáneas. Mientras que con **bootstrap/jackknife/CR with NONA** permite manipular las diferentes opciones para hacer soporte con remuestreo. Recuerde borrar todos los árboles antes de ejecutar la búsqueda de remuestreo.

CPanel Para llenar la matriz. **Mode**: Cambia el modo de acceso de los caracteres.

Winclados Para manejar el árbol. En general, la mayor parte de estas acciones son accesibles desde la barra de herramientas con el ratón.

Las teclas F2-F12 tienen (con y sin SHIFT) diversas funcionalidades de visualización, como engrosar/adelgazar el ancho de las ramas, expandir/contraer el costo, moverse entre árboles.

Trees: opciones para visualizar y manipular el árbol sin modificar su topología (forma, consensos, ir a un árbol determinado).

Nodes: Para ver frecuencias de nodos, desafortunadamente su funcionalidad es muy inestable, y a veces produce resultados inesperados.

HashMarks: Para ver las transformaciones en los nodos.

Edit/Mouse modes: Modifica las acciones posibles del ratón para modificar el árbol, como mover nodos, colapsarlos, cambiar la raíz.

Diagnoser: Para mapear caracteres.

A.1.2. MacClade & Mesquite

Autores: Maddison & Maddison (2005)

Plataforma (McClade): Macintosh (MacOS X, PPC y Classic 68k. Esta última funciona bien con emuladores como BasiliskII).

Plataforma (Mesquite): Cualquiera, requiere Java virtual machine.

Disponibilidad (McClade): Gratuito pero sin actualizaciones, se puede acceder desde el sitio de los autores: macclade.org.

Disponibilidad (Mesquite): Gratuito, se puede acceder desde el sitio mesquiteproject.org.

MacClade es un programa cuyas cualidades gráficas son impre-

sionantes, su interfaz es agradable, sencilla y no transmite miedo a los novatos. Dadas sus magnificas propiedades gráficas, es **muy** recomendable para la edición de árboles y matrices, pero principalmente para la optimización de caracteres, la cual permite distintos tipos de cambio de los mismos (v.g., Dollo, ACTRAN, DELTRAN) y adicionalmente tiene una característica única de mapeo, el "*equivocal ciclying*" [Maddison & Maddison, 1992].

En MacClade la edición de los datos y el manejo de los árboles se llevan a cabo en dos interfaces separadas para árboles y matrices, las cuales son accesibles en la ventana **Windows**. Dado que Macclade está diseñado para ser igualmente funcional con datos morfológicos y moleculares, existen múltiples herramientas de edición.

Data Editor En esta interfaz se construye y edita la matriz. Sus ventanas son:

- **Edit**: se pueden duplicar caracteres o taxa, editar bloques de comandos que corran directamente en PAUP*.
- **Utilities**: se pueden buscar secuencias particulares dentro de la secuencia total, reemplazar caracteres por otros, reemplazar datos ausentes por *gaps* y viceversa, importar alineamientos desde

el *genbank* y finalmente se puede lograr que la matriz hable por sí sola ¡sí, que hable! Un lector automático lee en forma descendente los taxa de su matriz facilitándole un poco el trabajo (puede usar inglés británico, si esto lo hace más feliz).

- **Characters**: permite adicionar caracteres, incluir los estados, ver una lista de los caracteres que se han incluido, determinar el formato de los caracteres que están en la matriz (si son proteínas, nucleótidos, o estándar, que es el definido para simbología de números en las casillas); también permite determinar el tipo de cambio que se permitirá para los caracteres (cuando se está haciendo una optimización), el peso de los caracteres.
- **Taxa**: permite hacer cosas semejantes a **characters**, pero para el manejo de los taxa, incluir taxa nuevos, crear listas de taxa o reordenarlos.
- **Display**: finalmente esta ventana maneja toda la configuración gráfica de la matriz, el tipo de letra, su tamaño, el color de los caracteres, el ancho de columna, etc.
- **Windows** le permite moverse entre la interfaz de los datos y el árbol y llamar a una caja de herramientas que por omisión siempre está presente en la parte inferior izquierda de la ventana. Esta caja ofrece herramientas como llenar estados, expandir co-

lumnas, cortar, entre otras. En esta ventana **notes about trees** y **note file** permiten incluir comentarios acerca de los árboles y sobre la matriz.

Tree Window Esta interfaz maneja y modifica los árboles y posee las ventanas básicas del **data editor**, no obstante, incluye otras nuevas, específicas para el mapeo de caracteres y manejo de los árboles. Solo se describen aquellas nuevas ventanas:

- **Trees**: Esta ventana permite cambiar, abrir archivos externos para incluir árboles, crear una lista de los árboles que esta incluidos a la matriz, guardar los árboles, exportarlos en formato de hennig86 o Phylip (ver anexo de formatos para más información) y manejar las politomías como blandas o duras (Coddington & Scharff, 1994).
- Σ: esta ventana incluye el manejo de todos los estadísticos del árbol, como el costo, el índice de consistencia, índice de retención, índice de consistencia escalonado, el número de cambios en el árbol y finalmente, puede generarse el archivo de comandos para correr el índice **decay**, ó soporte de Bremer en PAUP* (vea la práctica 10).

- **Trace**: aquí se maneja todo lo relacionado con el mapeo de los caracteres; puede mapear todos los caracteres a la vez, o escoger un determinado carácter para ser mapeado. También se escoge el tipo de cambio de los caracteres (**resolving options**) el cual puede ser ACTRAN o DELTRAN, y escoger el tipo de mapeo para los caracteres continuos, MacClade y Mesquite son los únicos programas que permiten mapear tal tipo de caracteres.
- **Chart**: En esta ventana se pueden generar cuadros de estadísticas de los caracteres *vs* pasos y árboles (characters states/etc.) o de los cambios de caracteres, la ocurrencia de los estados a través de todos los caracteres. Finalmente puede comparar dos árboles resolviendo sus politomías o tal y como son.
- **Display**: finalmente en esta ventana se manejan los aspectos gráficos de la resentación de los árboles tales como el tipo de letra, el tamaño y el estilo de la letra, el estilo y forma del árbol, numeración de las ramas, el tamaño del árbol, la configuración de los colores del mapeo y la simbología del los taxa (como números o nombres).

Aunque con menos capacidades que McClade, Mesquite es una herramienta poderosa, teniendo en cuenta que es gratuito. Su principal falla es que no permite ver las transformaciones en los nodos de to-

dos los caracteres simultáneamente. Su interfaz es muy similar a la de McClade y la distribución de ventanas y comandos es más o menos equivalente.

A.2. Búsqueda de árboles

A.2.1. TNT

Autor: Goloboff et al. (2008b), plataforma: Windows, MacOS, Unix.

Disponibilidad: Gratis, es posible descargarlo desde http://www.lillo.org.ar/phylogeny/tnt/.

Para parsimonia, es el programa más rápido que se ha desarrollado; incluye varios tipos de búsquedas especializadas, como la deriva y la fusión de árboles. Al igual que NONA, proc le permite abrir la matriz de datos o archivos de instrucciones. Para las búsquedas puede configurar los diferentes métodos usando ratchet, drift y mult; al usar ? puede obtener los parámetros, y con = puede modificarlos. Mult puede usarse como la "central de búsqueda", definiendo un número de réplicas para una búsqueda de Wagner y las posteriores mejoras con ratchet y drift (según como estén configurados). Para correr simplemente escri-

ba mult: replic X; para ejecutar el número de réplicas que desee (X), con tsave* nombre se abre el archivo "nombre" para guardar árboles. Guárdelos con save y al final no olvide cerrar el archivo: tsave/;.

El programa cuenta con ayuda en línea que puede ser consultada usando help o escribiendo help comando para un comando específico. Este es único programa que tiene directamente implementado el remuestreo simétrico, además tiene implementado el soporte de Bremer, el soporte relativo de Bremer, soporte de Bremer dentro de los límites del Bremer absoluto y puede calcular la cantidad de grupos soportados-contradichos (FC), con resample usted puede configurar la permutación. Con subop (también en NONA) usted puede indicar el costo de los subóptimos, en diferencia de pasos con respecto al árbol óptimo. Bsupport hace el cálculo del índice de Bremer, con un * calcula el valor relativo, o puede usar Bsupport] para calcular el soporte relativo dentro de los límites del absoluto. Los resultados se pueden almacenar usando algunas herramientas del lenguaje de macros[1].

Versión de menú En Windows es bastante similar a winClada, pero esta mucho mejor organizada. Algunas funciones son:

[1] www.lillo.org.ar/phylogeny/tnt/scripts/General_Documentation.pdf

- **Settings**: Además de las diferentes opciones de macros, manejo de memoria, también contiene los parámetros usados en el colapsado de ramas, consensos, y pesado implícito.
- **Analyze**: Contiene los diferentes tipos de búsquedas y sus parámetros, el manejo de árboles subóptimos, y los remuestreos.
- **Optimize**: Permite ver y dibujar sinapomorfías, mapear caracteres, y revisar estadísticos de caracteres y árboles (por ejemplo long o peso).
- **Trees**: Allí se encuentran las entradas para el dibujo de árboles, el cálculo soporte de Bremer, realizar consensos y super-árboles, así como árboles al azar, y el manejador de etiquetas de los nodos (*tags*).
- **Data**: Aparte de la configuración de caracteres y terminales, posee un editor básico de datos, que aunque muy simple, es extremadamente fácil de manejar, y más directo que los editores basados en mostrar la matriz (Como winClada, o Mesquite).

A.2.2. NONA

Autor: Goloboff, 1998.

Plataforma: Linux, Mac o Windows (9x o superior).

Disponiblilidad: Gratuito en conjunto con otros programas en www.lillo.org.ar/phylogeny/Nona-PeeWee/.

Este programa es una buena opción que existe para búsquedas bajo parsimonia, dado que es gratuito, tiene lenguaje de macros y es veloz, pero dados los costos, velocidad y operatibidad de TNT, este último es la mejor opción.

Básicamente los comandos de NONA son los mismos de TNT. Una diferencia importante (aparte de la velocidad y algoritmos implementados) es que en NONA los comandos son ejecutados (en vez de poder configurar sin ejecutar). A diferencia de PAUP*, no puede desactivar terminales.

Al igual que en TNT, el comando simple de búsqueda es mult. Para permutar ramas se utiliza max. En ambos casos debe colocarse un asterisco mult*; max* para que utilice TBR como permutación, en caso contrario usará SPR.

Para salvar el árbol, el comando es sv. La primera vez que se llama, solo abre el archivo (Debe darse como parámetro o el programa solicita un nombre). Con sv* salva todos los árboles en memoria. Usando ksv se guardan los árboles colapsados, de lo contrario siempre se guardan

dicotómicos. Para salvar el consenso es necesario usar inters.

El único método de soporte explícitamente implementado es el soporte de Bremer bsupport: con asterisco bs* muestra los soportes relativos, ó de lo contrario muestra el valor absoluto. Pero el programa viene acompañado de varios programas y macros que permiten calcular *jackknife* y *bootstrap*, así como hacer medidas basadas en frecuencias relativas (FC).

A.2.3. PAUP*

Autor: Swofford, 2002.

Plataforma: MWindows, MacOSX, Unix.

Disponibilidad: el programa para la interfaz gráfica tiene un costo entre US 80.00 a US150.00 dependiendo del sistema operativo y es distribuido por Sinauer, mientras que para línea de comandos es gratuito, www.sinauer.com/detail.php?id=8060.

PAUP* es uno de los programas más usados en cuanto a búsquedas mediante ML, o si el sistema operativo es Mac. Dados los costos es muy posible que la interfaz dominante llegue a ser por línea de comandos; la interfaz para Windows y MacOSX en modo gráfico tiene el mismo

acercamiento; como NONA y TNT tiene la opción de manejar un gran número de parámetros que permiten hacer una búsqueda **sobre pedido**, adicionalmente realiza todos los consensos directamente sin necesitar un segundo programa. Su gran desventaja es su velocidad y la falta de un lenguaje de macros.

set:esta función define la configuración básica de algunos parámetros. Son importantes increase=no, para que no le pregunte si desea aumentar el número de árboles, y maxtrees=N para dar como opción que guarde un número de árboles (N).

exec: para abrir la matriz de datos o archivos de instrucciones.

hsearch: es la función de búsqueda de cladogramas. Las opciones de este comando son addseq=random, lo que asegura que la entrada de datos para el árbol de Wagner es al azar.

swap=tbr/spr: es el tipo de permutación.

nreps=X: indica el número de réplicas que se hará en la búsqueda.

nchuck=X ckuckscore=1: le permite usar la estrategia NONA, indicando cúantos árboles desea por réplica. Si usted desea hacer solo permutación de ramas del árbol previamente construido, la serie de instrucciones sería search start=current chuckscore=no.

Para guardar los árboles utilice el comando savetre.

Con el comando contree se generan los árboles de consenso. Estos

deben ser guardados cuando se ejecuta el comado (parámetro treefile=nombre, donde nombre es el achivo.). Los métodos de medición de soporte implementados son el *bootstrap* con el comando bootsptrap y el *jackknife* con jackknife.

En caso de dudas, el programa posee una ayuda en línea, consúltela usando help o escribiendo help comando para un comando específico; para ver las opciones del comando use comando ?.

A.2.4. MrBayes

Autor: Ronquist et al., 2005.

Plataforma: Windows, MacOS, Unix.

Disponibilidad: gratuito, http://mrbayes.net

Es el programa más comúnmente usado para análisis bayesiano. Es gratuito y su interfaz es muy similar a la de PAUP*. Los principales comandos del programa son descritos en el capítulo de análisis bayesiano. El tutorial incluido es muy completo y viene dentro del programa. Una de las características más interesantes es la implementación de los "modelos" usados para caracteres morfológicos (Lewis, 2001), así como el modelo 'parsimonia' tal y como fue descrito por Tuffley & Steel (1997) que se activa usando

```
> lset parsmodel=yes;
```

Apéndice B

Formatos de archivos

B.1. Formatos de matrices

B.1.1. NONA

Es válido para NONA, TNT, Hennig86 (formato de morfología de POY) y WinClada; comienza con xread, luego el número de caracteres y el número de taxa, los polimórficos entre paréntesis angulares y los desconocidos con - o ?. Al final, un punto y coma y si se desea la aditividad de los caracteres (comenzando desde 0). Termina con p/ o p-, que los programas interpretan como fin del archivo.

```
xread 'Matriz ejemplo' 5 5
out 00000
alpha 10-20
beta 1102[01]
gamma 1?111
lamda 11111
;
cc -0.2 +3 -4;
p/;
```

Para ADN (en NONA) se usa dread, con la claúsula gap seguida de ? si se quieren asumir los *gaps* como desconocidos, o con ; si quiere que sean un quinto estado. Se usa codificación tipo IUPAC.

```
dread gap ; match . 'DNA' 5 5
out ACGTC
alpha AT-CG
beta RTAAC
gamma CGAY-
lamda TCNCC
```

;

cc -.;

p/;

B.1.2. PAUP*

Se inicia con la claúsula #nexus, y luego con el bloque data. se puede definir si los caráctertes son morfológicos, ADN o proteínas. Los polimórfismos se colocan entre paréntesis redondos. ADN en formato IUPAC.

#nexus

begin data;

dimensions ntax=5 nchar=5;

format missing=? gap=- symbols="0 1 2";

matrix

out 00000

alpha 10-20

beta 1102(01)

gamma 1?111

lamda 11111

;

```
end;
begin assumptions;
typeset tipoUno=unord:1-3 5, ord:4;
end;
begin paup;
 [Aqui puede colocar instrucciones específicas de paup, por ejemplo
búsquedas]
 hsearch add=random;
end;

#nexus
begin data;
dimensions ntax=5 nchar=5;
format missing=? gap=- datatype=dna;
matrix
out ACGTC
alpha AT-CG
beta RTAAC
gamma CGAY-
lamda TCNCC
;
```

end;

B.2. Formatos de árboles

B.2.1. NONA

NONA, Hennig86, WinClada y TNT usan este formato; con * indican que hay más árboles y con ; que es el último árbol. Nótese el espacio para separar los terminales. El primer taxon es 0. Al igual que en las matrices, p- o p/ indican el final de lectura del archivo. Winclada puede incluir al incio la lista de nombres, pero no es compatible con otros programas.

```
 tread 'tres arboles'
(0 (1 (2 (3 4 ))))*
(0 (1 (2 3 4 )))*
(0 ((1 2 )(3 4 )));
p/;

 tread 'solo un arbol'
(0 (1 (2 (3 4 ))));
```

p/;

B.2.2. PAUP*

En PAUP* el árbol está embebido en el archivo de la matriz o en un formato aparte. Los grupos son separados por comas y el primer taxon es 1.

```
#nexus

begin trees;
translate
1 lamda,
2 alpha,
3 beta,
4 gamma,
5 out
;
tree *primero=(5,(1,(2,(3,4))));
tree politomico=(5,(1,(2,3,4)));
tree tercero=(5,((1,2),(3,4)));
end;
```

El orden no altera los árboles en los programas. Así:

(0,(1,(2,(3,4))))

es igual a

((1,((3,4),2)),0)

Apéndice C

Algunos comandos básicos para POY

C.1. Análisis de sensibilidad con costos diferenciales

(* tomado de urlgroups.google.com/group/POY4/ *)

(* dados dos archivos de datos *)

(* 1.fas y 3.fas *)

```
> read("1.fas", "3.fas")
> transform((names:("1.fas"), tcm:"412.txt"),
  (names:("3.fas"), tcm:"121.txt"))
> store("misdatos")
> build(5, trees:2)
> select(best:1)
> transform((all, static_approx))
> report("todo13.mtr", phastwinclad) (* Resultados en el ar-
  chivo Sensitivity_results.txt *)
> echo ("412+121 Resultados",
  output:"Sensitivity_results.txt")
> report ("Sensitivity_results.txt", treestats)
  (* Deseche los árboles *)
> select(best:0)
> use("misdatos")
> select(characters,names:("1.fas"))
> build(5)
> select()
> echo("1.fas Resultados", output:"Sensitivity_results.txt")
```

```
> report("Sensitivity_results.txt", treestats) (* Deseche los
  árboles *)
> transform((all, static_approx))
> report("matriz1.mtr", phastwinclad)
> use("misdatos")
> select(best:0)
> select(characters,names:("3.fas"))
> build(5)
> select(best:1)
> echo ("3.fas Resultados", output:"Sensitivity_results.txt")
> report ("Sensitivity_results.txt", treestats)
> transform((all, static_approx))
> report ("matriz3.mtr", phastwinclad)
> quit()
```

Bibliografía

Alfaro, M. E. & M. T. Holder. 2006. The posterior and the prior in bayesian phylogenetics. Annual Review of Ecology, Evolution, and Systematics Pages 19–42.

Alfaro, M. E., S. Zoller, & F. Lutzoni. 2003. Bayes or bootstrap? a simulation study comparing the performance of bayesian markov chain monte carlo sampling and bootstrapping in assessing phylogenetic confidence. Molecular Biology and Evolution 20:255–266.

Antezana, M. 2003. When being "most likely" is not enough: examining the performance of three uses of the parametric bootstrap in phylogenetics. Molecular Evolution 56:198–222. [doi:10.1007/s00239-002-2394-1].

Archibald, J. K., M. E. Mort, & D. J. Crawford. 2003. Bayesian inference of phylogeny: a non-technical primer. Taxon Pages 187–191.

Bayes, T. 1763. An essay towards solving a problem in the doctrine of chances. Philosophical Transactions Pages 370–418. [doi:10.1098/rstl.1763.0053].

Camin, J. & R. Sokal. 1965. A method for deducing branching sequence in phylogeny. Evolution 19:311–326. [doi:10.2307/2406441].

Carpenter, J. M. 1988. Choosing among multiple equally parsimonious cladograms. Cladistics 4:291–296. [doi:10.1111/j.1096-0031.1988.tb00476.x].

Coddington, J. & N. Scharff. 1994. Problems with zero-length branches. Cladistics 10:415–423. [doi:10.1111/j.1096-0031.1994.tb00187.x].

Cranston, K., L. Harmon, M. O'Leary, & C. Lisle. 2014. Best practices for data sharing in phylogenetic research. PLOS Currents Tree of Life [doi:10.1371/currents.tol.bf01eff4a6b60ca4825c69293dc59645].

de Pinna, M. C. C. 1991. Concepts and tests of homology in the cladistic paradigm. Cladistics 7:367–394. [doi:10.1111/j.1096-0031.1991.tb00045.x].

Edgar, R. C. 2004. MUSCLE: multiple sequence alignment with high accuracy and high throughput. Nucleic Acids Research 32:1792–1797.

Farris, J. 1969. A successive approximations approach to character weighting. Systematic Biology 18:374–385.

Farris, J. 1970. Methods for computing wagner trees. Systematic Zoology 19:83–92.

Farris, J. 1983. Advances in Cladistics 2 cap. The logical basis of phylogenetic analysis, Pages 7–36. Columbia Univ. Press, New York.

Farris, J. 2001. Support weighting. Cladistics 17:389–394. [doi:10.1111/j.1096-0031.2001.tb00133.x].

Farris, J., V. Albert, M. Kallersjo, D. Lipscomb, & A. Kluge. 1996. Parsimony jackknifing outperforms neighbor-joining. Cladistics 12:99–24.

Felsenstein, J. 2004. Inferring phylogenies. Sinauer Associates, USA.

Fitch, W. 1971. Toward defining the course of evolution: minimum change for a specific tree topology. Systematic Zoology 20:406–416.

Frost, D. 2001. A Molecular Perspective on the Phylogeny of the Girdled Lizards (Cordylidae, Squamata). American Museum novitates American Museum of Natural History.

Giribet, G. 2003. Stability in phylogenetic formulations and its relationship to nodal support. Systematic Biology Pages 554–564.

Giribet, G. & W. C. Wheeler. 1999. On gaps. Molecular Phylogenetics and Evolution 13:132 – 143. [doi:dx.doi.org/10.1006/mpev.1999.0643].

Goloboff, P. 1993. Estimating character weights during tree search. Cladistics 9:83–91. [doi:10.1111/j.1096-0031.1993.tb00209.x].

Goloboff, P. 1995. Parsimony and weighting: A reply to turner and zandee. Cladistics 11:91–104. [doi:10.1111/j.1096-0031.1995.tb00006.x].

Goloboff, P. 1997. Self-weighted optimization: Tree searches and state reconstructions under implied transformation costs. Cladistics 13:225–245. [doi:10.1111/j.1096-0031.1997.tb00317.x].

Goloboff, P. 1998. Tree searches under sankoff parsimony. Cladistics 14:229–237. [doi:10.1006/clad.1998.0068].

Goloboff, P. & J. Farris. 2001. Methods for quick consensus estimation. Cladistics 17:S26–S34.

Goloboff, P. & D. Pol. 2005. Parsimony and Bayesian phylogenetics. 148-162 Oxford.

Goloboff, P. A. 1999. Analyzing large data sets in reasonable times: Solutions for composite optima. Cladistics 15:415 – 428. [doi:dx.doi.org/10.1006/clad.1999.0122].

Goloboff, P. A. 2014. Hide and vanish: data sets where the most parsimonious tree is known but hard to find, and their implications for tree search methods. Molecular Phylogenetics and Evolution Pages–. [doi:dx.doi.org/10.1016/j.ympev.2014.06.008].

Goloboff, P. A., J. M. Carpenter, J. S. Arias, & D. R. Miranda-Esquivel. 2008a. Weighting against homoplasy improves phylogenetic analysis of morphological data sets. Cladistics 24:758–773. [doi:10.1111/j.1096-0031.2008.00209.x].

Goloboff, P. A., J. S. Farris, M. Källersjö, B. Oxelman, M. J. Ramırez, & C. A. Szumik. 2003. Improvements to resampling measures of group support. Cladistics 19:324 – 332. [doi:dx.doi.org/10.1016/S0748-3007(03)00060-4].

Goloboff, P. A., J. S. Farris, & K. C. Nixon. 2008b. Tnt, a free program for phylogenetic analysis. Cladistics 24:774–786. [doi:10.1111/j.1096-0031.2008.00217.x].

Goloboff, P. A., C. I. Mattoni, & A. S. Quinteros. 2006. Continuous characters analyzed as such. Cladistics 22:589–601. [doi:10.1111/j.1096-0031.2006.00122.x].

Grant, T. & A. Kluge. 2004. Transformation series as an ideographic character concept. Cladistics 20:23–31.

Hennig, W. 1968. Elementos de una sistemática filogenética. Eudeba, Buenos Aires.

Huelsenbeck, J. & F. Ronquist. 2001. Mrbayes: Bayesian inference of phylogenetic trees. Bioinformatics 17:754–755.

Jukes, T. & C. R. Cantor. 1969. Evolution of protein molecules. Mammalian protein metabolism 3:21–132.

Kearney, M. 2002. Fragmentary taxa, missing data, and ambiguity: mistaken assumptions and conclusions. Systematic Biology 51:369–381.

Kjer, K. M., R. J. Blahnik, & R. W. Holzenthal. 2001. Phylogeny of trichoptera (caddisflies): characterization of signal and noise within multiple datasets. Systematic Biology 50:781–816.

Kluge, A. 1997. Sophisticated falsification and research cycles: Consequences for differential character weighting in phylogenetic systematics. Zoologica Scripta 26:349–360. [doi:10.1111/j.1463-6409.1997.tb00424.x].

Kluge, A. & J. Farris. 1969. Quantitative phyletics and the evolution of anurans. Systematic Zoology 18:1–32. [doi:10.1093/sysbio/18.1.1].

Kluge, A. G. 2003. The repugnant and the mature in phylogenetic inference: atemporal similarity and historical identity. Cladistics 19:356–368. [doi:10.1111/j.1096-0031.2003.tb00379.x].

Lewis, P. 2001. A likelihood approach to estimating phylogeny from discrete morphological character data. Systematic Biology 50:913–925.

Maddison, D. R. & W. P. Maddison. 2005. MacClade 4: Analysis of phylogeny and character evolution. Version 4.08a. Sinauer Associates Incorporated.

Maddison, D. R., D. L. Swofford, & W. P. Maddison. 1997. NEXUS: an extensible file format for systematic information. Systematic Biology 46:590–621. [doi:10.1093/sysbio/46.4.590].

Maddison, W. P. & D. R. Maddison. 2014. Mesquite: a modular system for evolutionary analysis. (mesquiteproject.org).

Miranda-Esquivel, D., I. Garzón, & J. Arias. 2004. ¿taxonomía sin historia? Entomólogo 32:4–7.

Miyamoto, M. M. 1985. Consensus cladograms and general classifications. Cladistics 1:186–189. [doi:10.1111/j.1096-0031.1985.tb00421.x].

Neff, N. 1986. A rational basis for a priori character weigthing. Systematic Zoology 35:110–123.

Nelson, G. 1979. Cladistic analysis and synthesis: Principles and definitions, with a historical note on adanson's familles des plantes (1763–1764). Systematic Biology 28:1–21. [doi:10.1093/sysbio/28.1.1].

Nixon, K. & J. Carpenter. 1996. On consensus, collapsibility, and clade concordance. Cladistics 12:305–321.

Nixon, K. C. 1999. The parsimony ratchet, a new method for rapid parsimony analysis. Cladistics 15:407–414. [doi:10.1111/j.1096-0031.1999.tb00277.x].

Page, R. & E. Holmes. 1998. Molecular Evolution: A Phylogenetic Approach. Evolutionary Biology Series Wiley.

Paradis, E., J. Claude, & K. Strimmer. 2004. APE: analyses of phylogenetics and evolution in R language. Bioinformatics 20:289–290. [doi:10.1093/bioinformatics/btg412].

Patterson, C. 1981. Significance of fossils in determining evolutionary relationships. Annual Review of Ecology and Systematics 12:195–223.

Platnick, N. I. 1979. Philosophy and the transformation of cladistics. Systematic Biology 28:537–546. [doi:10.2307/sysbio/28.4.537].

Pleijel, F. 1995. On character coding for phylogeny reconstruction. Cladistics 11:309–315. [doi:10.1111/j.1096-0031.1995.tb00092.x].

Posada, D. & K. A. Crandall. 2001. Selecting the best-fit model of nucleotide substitution. Systematic Biology 50:580–601. [doi:10.1080/10635150118469].

Quicke, D. L. J., J. Taylor, & A. Purvis. 2001. Changing the landscape: A new strategy for estimating large phylogenies. Systematic Biology 50:60–66. [doi:10.1080/10635150119012].

Richards, R. 2002. Kuhnian valures and cladistic parsimony. Perspectives on Science 10:1–27.

Richards, R. 2003. Character individuation in phylogenetic inference. Philosophy of Science 70:264–279.

Rieppel, O. & M. Kearney. 2001. The origin of snakes: limits of a scientific debate. Biologist 48:110–114.

Rieppel, O. & M. Kearney. 2002. Similarity. Biological Journal of the Linnean Society 75:59–82. [doi:10.1046/j.1095-8312.2002.00006.x].

Scotland, R. W., R. Olmstead, & J. Bennett. 2003. Phylogeny reconstruction: the role of morphology. Systematic Biology 52:539–548.

Sharkey, M. & J. Leathers. 2001. Majority does not rule: the trouble with majority-rule consensus trees. Cladistics 17:282–284.

Sharkey, M. J., S. Stoelb, D. R. Miranda-Esquivel, & B. J. Sharanowski. 2013. Weighted compromise trees: a method to summarize competing phylogenetic hypotheses. Cladistics 29:309–314. [doi:10.1111/cla.12000].

Sokal, R. 1983. A phylogenetic analysis of the caminalcules. i. the data base. Systematic Zoology 32(2):159–184.

Springer, M. S., R. W. DeBry, C. Douady, H. M. Amrine, O. Madsen, W. W. de Jong, & M. J. Stanhope. 2001. Mitochondrial versus nuclear gene sequences in deep-level mammalian phylogeny reconstruction. Molecular Biology and Evolution 18:132–143.

Steel, M. 2002. Some statistical aspects of the maximum parsimony method. Pages 125–139 en Molecular Systematics and Evolution: Theory and Practice (R. DeSalle, W. Wheeler, & G. Giribet, eds.) vol. 92 de EXS 92. Birkhäuser Basel.

Strong, E. & D. Lipscomb. 1999. Character codding and inaplicable data. Cladistics 15:363–371.

Strugnell, J. M., M. D. Norman, M. Vecchione, M. Guzik, & A. L. Allcock. 2014. The ink sac clouds octopod evolutionary history. Hydrobiologia 725:215–235. [doi:10.1007/s10750-013-1517-6].

Swofford, D. 1991. When are phylogeny estimates from molecular and morphological data incongruent? Phylogenetic Analysis of DNA Sequences. Oxford.

Swofford, D., G. Olsen, P. Waddell, & D. Hillis. 1996. Phylogenetic Inference cap. 11, Pages pp. 407–514. Sinauer Associates, Sunderland, Massachusetts.

Tuffley, C. & M. Steel. 1997. Links between maximum likelihood and maximum parsimony under a simple model of site substitution. Bulletin of mathematical biology 59:581–607.

Wagner, W. H. 1961. Problems in the classification of ferns. Recent advances in botany 1:841–844.

Wheeler, W. 1996. Optimization alignment: The end of multiple sequence alignment in phylogenetics? Cladistics 12:1–9.

Wheeler, W. 1999. Fixed character states and the optimization of molecular sequence data. Cladistics 15:379–385.

Wheeler, W., L. Aagesen, C. P. Arango, J. Faivovich, T. Grant, C. D'Haese, D. Janies, W. L. Smith, A. Varón, & G. Giribet. 2006. Dynamic homology and phylogenetic systematics: a unified approach using POY.

Wheeler, W. C. 1995. Sequence alignment, parameter sensitivity, and the phylogenetic analysis of molecular data. Systematic Biology 44:321–331. [doi:10.1093/sysbio/44.3.321].

Wiens, J. 2001. Character analysis in morphological phylogenetics: problems and solutions. Systematic Biology 50:689–699.

Wiley, E. & B. Lieberman. 2011. John Wiley & Sons.

Yang, Z. 2006. Computational molecular evolution. Oxford University Press, USA.