

## Índice general

<b>Introducción general</b>	<b>i</b>
<b>1. Selección de caracteres</b>	<b>1</b>
<b>2. Matrices de datos</b>	<b>10</b>
<b>3. Árboles</b>	<b>18</b>
<b>4. Búsquedas mediante parsimonia</b>	<b>26</b>
<b>5. Consensos</b>	<b>41</b>
<b>6. Pesaje de caracteres</b>	<b>48</b>
<b>7. Soporte en parsimonia y ML</b>	<b>54</b>
<b>8. Modelos de evolución</b>	<b>64</b>
<b>9. Máxima verosimilitud</b>	<b>73</b>
<b>A. Programas de cómputo</b>	<b>81</b>
A.1. Editores de matrices y manejadores de búsqueda . . . . .	82
A.1.1. WinClada . . . . .	82

A.1.2. MacClade & Mesquite . . . . .	86
A.2. Búsqueda de árboles . . . . .	91
A.2.1. TNT . . . . .	91
A.2.2. NONA . . . . .	95
A.2.3. Poy . . . . .	98
A.2.4. PAUP* . . . . .	102
A.2.5. MrBayes . . . . .	104
<b>B. Formatos de archivos</b>	<b>106</b>
B.1. Formatos de matrices . . . . .	106
B.1.1. NONA . . . . .	106
B.1.2. PAUP* . . . . .	108
B.1.3. MALIGN . . . . .	110
B.1.4. POY . . . . .	112
B.2. Formatos de árboles . . . . .	113
B.2.1. NONA . . . . .	113
B.2.2. PAUP* . . . . .	114

## Introducción general

Este libro ha sido pensado como uno de los tantos soportes posibles para las clases de sistemática de nivel básico y avanzado, 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 XXYYXX secciones 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 las instrucciones que se acceden mediante un menú o un cuadro de dialogo en los programas de interfaz gráfica (v.g., **Analyze/heuristics**).

El orden de los capítulos obedece a la estrategia de enseñanza de la UIS pero el manual puede ser seguido en otros órdenes, por ejemplo todo sobre caracteres, excluyendo alineamiento, seguido de búsquedas incluida la búsqueda del modelo, consensos y por último soporte y al final discutir alineamiento. 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: <http://tux.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 a dominio de computadores, pero el manejo de un editor de texto que permita grabar archivos sin formato es *muy* recomendable, además de manejar 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.

## Programas eliminados

Algunos programas han sido eliminados, por ejemplo Piwe, Modeltest o component, y alguno han sido sustraído al mínimo (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

RaxML/ExaML PhyML JModeltest

## 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 por reescribir 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 señala que el término **carácter** no está bien definido (Richards, 2003) 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 (v.g., Hennig, 1968; Neff, 1986; Rieppel & Kearny, 2001)<sup>1</sup>.

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

---

<sup>1</sup>Este análisis de caracteres también implica revisar las afirmaciones cuantitativas o cualitativas hechas, ver Wiens 2001

como **similar por ancestría común** (Rieppel & Kearny, 2001; *contra* Kluge 2002, 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.

Por ejemplo, los brazos humanos, las alas de los murciélagos y las patas de los caballos presentan una estructura ósea muy 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; Kluge, 2002). Esto 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 simplemente 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: (a) rojo; (b) otro color", el estado (b) es ambiguo y puede contener individuos de diferentes colores, cuya única (falsa) similitud sea 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. total a 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

”transparentable con KOH: (a) sí, (b) no”, suponiendo que el estado 1 no es problemático, 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: (a) uno; (b) dos”. Al tratarse los estados de carácter como 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 ”conjunción” 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 el cual corresponde a la figura 1 en Sokal (1983). Los ”organismos” son camináculos, seres hipotéticos creados por J. Camin para estudios sobre clasificaciones en fenética.<sup>2</sup> A partir de los dibujos:

1. Elabore una pequeña guía morfológica de los organismos, de modo que pueda diferenciar las diferentes estructuras en los diferentes organismos (en buen romance significa poner nombres a sus organismos y ligar tales

---

<sup>2</sup>Copyright, Society of Systematic Biologists, se reproduce con permiso

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 cuáles 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.

Preguntas generales.

1. Haga un ejercicio crítico de los caracteres de sus compañeros:

- 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 se manejan los caracteres cuantitativos. ¿usan un esquema similar al usado para los caracteres discretos?

### Literatura recomendada

de Pinna (1991) [Ofrece una visión a la formulación de los caracteres].

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

Neff, 1986 [Una normalización del análisis de caracteres].

Patterson, 1982 [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 & Kearny, 2002 [Una visión crítica y extensa de la selección de caracteres, con ejemplos empíricos].

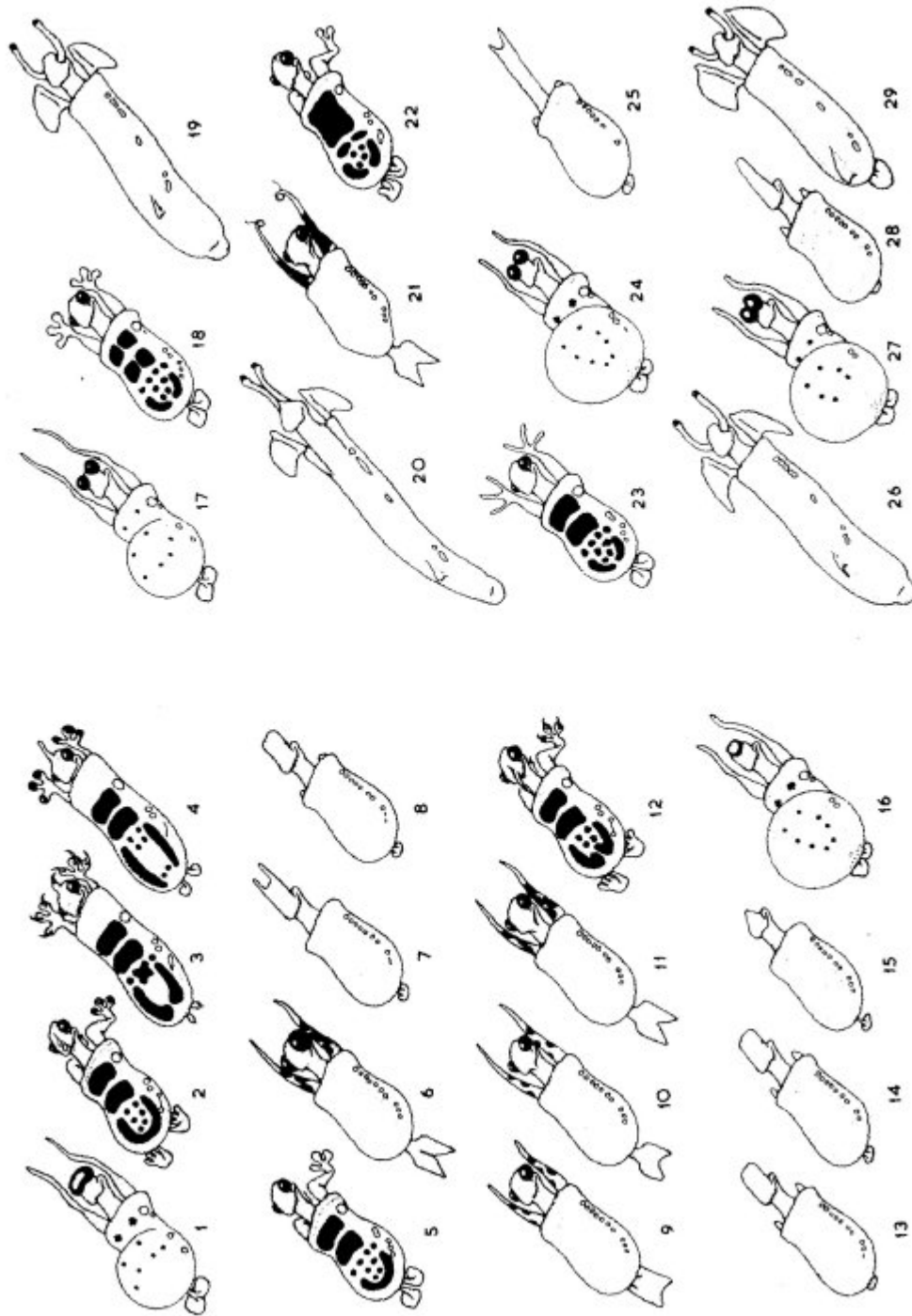


Figura 1.1: Los Camináculos, tomado de Sokal (1983)

Copyright *Society of Systematic Biologists*, se reproduce con permiso.

## 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 puede 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**<sup>1</sup>.

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,

---

<sup>1</sup>si desea puede ver nuestra visión ampliada en Miranda et al., 2004.



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 o Poy pueden reconocer como un quinto carácter los eventos de inserción -perdida o gaps (Giribet & Wheeler, 1999).

Cuando se usan caracteres con múltiples estados, es necesario clarificar cuáles fueron usados como aditivos y cuáles como no aditivos, o en el caso de haber sido recodificados, como 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 al 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<sup>2</sup>, Mesquite<sup>3</sup>, MacClade<sup>4</sup>, R<sup>5</sup> o Seaview<sup>6</sup>). En su mayoría estos programas sirven de plataforma para manejar programas que realizan el análisis filogenético como tal. Muchos de los programas que manejan matrices están diseñados básicamente

---

<sup>2</sup>[www.cladistics.com](http://www.cladistics.com)

<sup>3</sup>[www.mesquite.org](http://www.mesquite.org)

<sup>4</sup>el programa no corre en versiones de MacOS X superiores a 10.6 (desde Lion en adelante)

<sup>5</sup>R es uno de los programas para análisis estadístico más versátiles del momento, además de ser gratuito permite la implementación de múltiples procesos de cálculo, de tal manera que es posible realizar distintos tipos de operaciones y cálculos desde estadística básica hasta análisis en evolución; sirve de plataforma para ape (Paradis et al., 2008), un módulo que permite distintas acciones con matrices y árboles [cran.r-project.org](http://cran.r-project.org)

<sup>6</sup>[doua.prabi.fr/software/seaview](http://doua.prabi.fr/software/seaview)

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. El otro formato importante es el de Hennig86 o NONA, pero en muchos programas, especialmente moleculares, su uso no está implementado.

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/>

## Técnicas

Abra el archivo de datos morfológicos para vertebrados **”datos.vertebrados.xls”** con un programa para hojas de cálculo y manipule las matrices con Win-

Clada, Mesquite o TNT para Windows:

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<sup>7</sup>.

---

<sup>7</sup>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.

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:

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

- Cargue las bibliotecas *ape* y *phangorn*<sup>8</sup>:

```
> library(ape)
```

```
> library(phangorn)
```

- En caso de no tenerlas disponibles primero bajelas con la instrucción:

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

9

- Abra la matriz de datos en formato de texto simple y asignela a un objeto de R:

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

---

<sup>8</sup>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", y puede ser descargado desde <http://cran.r-project.org/web/packages/phangorn/index.html>

<sup>9</sup>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 pagina web e instalarlo desde cualquier directorio en el ordenador dando directamente la ruta a dicho sitio:

```
> install.packages("../usuario/R/Packages/Paquete.tar.gz")
```

Tenga en cuenta que de este modo deberá descargar e instalar todas las dependencias requeridas por el paquete de manera independiente. El comando `library()` le permitirá cargar el paquete en el entorno de R para poder empezar a trabajar con todas sus utilidades.

- Revise los nombres de las variables en la matriz con

```
> names(datos)
```

- Revise los nombres de las terminales en la matriz con

```
> row.names(datos)
```

- Abra la matriz de datos en formato **phylip** y asignela a un objeto de R:

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

- Escriba en formato **Nexus** la matriz leída anteriormente:

```
> write.nexus.data(datos.Phylip, file="NuevaDNA1.nex")
```

- 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.<sup>10</sup>

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 inreractuar directa o indirectamente con programas como PAUP, TNT o PhyML. Sobre Windows usted cuenta

---

<sup>10</sup>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.

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 interface gráfica. Si usa Mac una opción es McClade, el cual funciona 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 los comandos 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. Preguntas generales.

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].

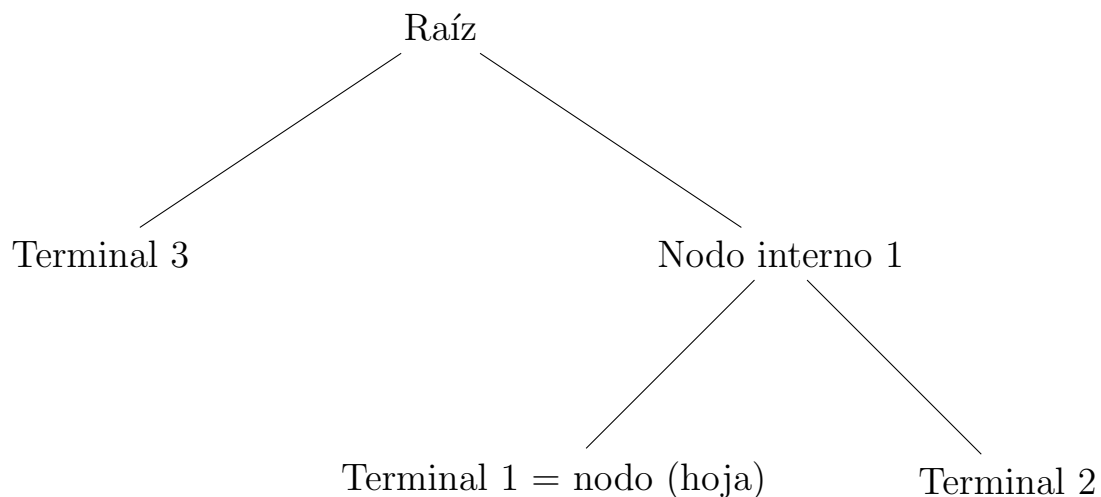
cita mesquite y sus bibliotecas

cita a NN? hennig? por que las matrices son importantes?

## 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



terminos 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 el término es usado sin distinción. 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 nodo no está soportado o es un nodo de longitud 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 suele representarse como una probabilidad o frecuencia de transformaciones y suelen presentarse los árboles con una escala que muestra la longitud de ramas; 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, longitudes y etiquetas de las ramas, pero no transformaciones de caracteres.

Dependiendo de la plataforma que trabaje, usted tiene disponible distintos programas: Mesquite, R<sup>1</sup> y Figtree<sup>2</sup> 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 alternativa, 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 arboles 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

---

<sup>1</sup>[cran.r-project.org](http://cran.r-project.org)

<sup>2</sup>[tree.bio.ed.ac.uk/software/figtree/](http://tree.bio.ed.ac.uk/software/figtree/)

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

- a) Revise los archivos de datos con editor de texto y con el programa seleccionado abra una matriz de datos que contenga tanto datos como árboles.
- b) Establezca la forma de obtener información sobre los árboles, en un principio los costos o longitud del árbol.
- c) Cambie de posición algunos nodos o terminales de la topología, y observe cómo cambia la longitud 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 rat

ón 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 de matrices, use los parámetros por omisión (menú **Analyze / 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 las longitudes.
  - d) Abra el árbol con longitud de ramas.
5. En Mesquite o WinClada abra el archivo **"datos.conarbol.dat"**:
- a) Indique la cantidad de árboles presentes y la longitud 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: en WinClada ACCTRAN (=fast), DELTRAN (=slow) y no ambigua, En TNT (para Windows) (**Optimize > Character > Reconstructions** y seleccione un solo árbol y algunos caracteres) y en Mesquite: Dollo e Irreversible (C-S); revise cómo cambian las optimizaciones en los distintos nodos.
  - d) Revise de nuevo el efecto en la longitud de los árboles de 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 **"arbol.r.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, 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 la longitud del árbol, lea la matriz de datos "**matriz.r.phy**" a un objeto que se llame **MatrizDatos** y use la función **parsimony()**:

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

### Preguntas.

- a) Al hacer una búsqueda por defecto ¿Usted y sus compañeros obtienen las mismas topologías y los mismos caracteres que soportan los grupos?.
- b) Al cambiar la aditividad/no aditividad de un caracter ¿Cambia esto la forma como se mapean? Haga este ejercicio para varios caracteres, tanto binarios como multiestado.
- c) Si cambia la raíz, ¿cambia la longitud? ¿Por qué cree que se presenta el resultado que obtuvo? ¿Es (y por qué) una ventaja o una desventaja?
- d) Examine sus caracteres y cambie su aditividad-no aditividad. ¿Cambia esto la forma como mapean? Haga este ejercicio para varios caracteres, binarios y multiestado.
- e) Si cambia la raíz, ¿cambia la longitud? ¿Por qué cree que se presenta el resultado que obtuvo? ¿Es (y por qué) una ventaja o una desventaja?
- f) Dibuje el siguiente cladograma:  
*(Lampreas (Tiburones (Esturión Teleósteos) (Celacanto (Peces-pulmonados (Anfibios (Mamíferos (Tortugas (Lagartos (Cocodrilos Aves))))))))))*
- g) Si la raíz esta colocada entre Aves y Cocodrilos, ¿cómo es la topología resultante?
- h) Convierta la topología dibujada a notación de paréntesis.

Preguntas generales. 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 ([taxonomy.zoology.gla.ac.uk/teaching/index.html](http://taxonomy.zoology.gla.ac.uk/teaching/index.html))].

## 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 sinapomorfias, 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; 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 mas conocidas e implementadas son: *la optimización de Wagner y la optimización de Fitch* ([Wagner, 1961](#); [Kluge & Farris, 1969](#); [Farris, 1970](#); [Farris et al., 1970](#); [Fitch, 1971](#)).

La búsqueda del cladograma mas parsimonioso es más compleja a mediada 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 entrada (Dayo, 1969), la cual puede estar seguida de permutaciones de ramas. Sin embargo, este ultimo paso para matrices muy grandes consume gran cantidad de tiempo; esto se debe repetir múltiples veces para evitar caer en "óptimos locales". Con este método es posible una solución optima incluso para matrices de 80 a 100 terminales. Problemas mas 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.

Una nueva ventana para las matrices cada vez más grandes (por ejemplo matrices moleculares con más de 1000 terminales), es tratar de identificar el acuerdo entre distintas réplicas de búsquedas parciales, en vez de buscar la solución óptima (e.g. Farris et al., 1996; Farris, 1997; Goloboff, 1997b; Goloboff & Farris, 2001).

En terminos de programas de computo, 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 (para 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 clusters.

### 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<sup>1</sup>), la mayor parte de los programas actuales tienen esta opción: inician con una **semilla** determinada para el generador de número aleatorio y aseguran que la búsqueda sea exactamente igual a otra que tenga el mismo de inicio (semilla del generador de números pseudo-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 la longitud 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).

---

<sup>1</sup>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).

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\*

Construya una tabla (Apéndice 1) 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
```

### Preguntas.

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

## 2. Búsquedas modificadas

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

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

Preguntas.

- 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 replicas?

### 3. Búsquedas con RACHET

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

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; Quickle, 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 nuevas 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

subóptimas 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 \(1999\)](#).

Preguntas.

- ¿Cuál es y que hace el comando que permite hacer búsquedas con la técnica RACHET 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 replicas? Realice búsquedas adicionales con parámetros diferentes que le permitan responder estas preguntas. Utilice un manejador grafico 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) Escribala como formato NEXUS utilizando el nombre `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`.

Preguntas.

- ¿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.

```
> dm - dist.dna (as.DNAbin(primates))

> treeUPGMA - upgma (dm)
```

```
> treeNJ - NJ(dm)
```

```
> plot (treeUPGMA, main="UPGMA", cex = 0.8)
```

```
> plot (treeNJ, "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 distancia y el valor de gamma. Existen varios métodos 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()`.

```
> treeRAM - random.addition(primates, method="fitch")
```

```
> plot (treeRAM, main="UPGMA", cex = 0.8)
```

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

Preguntas.



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

## 5. Parsimonia y optimización

A partir de la matriz de dataos primtes.nex y las topologías construidas, calcule la longitud 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 longitud 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 en el punto C, utilizando el método de optimización por omisión, optimización por SPR y optimización por NNI. Registre sus resultados en el Apéndice 2.

## 6. Optimización por omisión

```
> optParsUPGMA - optim.parsimony (treeUPGMA, primates)
```

```
> optParsNJ - optim.parsimony (treeNJ, primates)
```

```
> optParsRAM - optim.parsimony (treeRAM, primates)
```

## 7. Optimización con rearrreglo de ramas específico

```
> 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 mas parsimoniosos utilizando los métodos de rearrreglos NNI y SPR.

Preguntas.

- ¿En que difieren los métodos de rearrreglos NNI, SPR y TBR?
- ¿Cuales son los argumentos por omisión de las funciones `parsimony()` y `optim.parsimony()`?

## 8. Parsimonia usando Rachet

Utilice el método Rachet en parsimonia para hacer las búsquedas del o los árboles de menor costo. Complete la tabla del Apéndice 3 creando nuevas líneas de código para hacer las búsquedas con los árboles obtenidos en el proceso anterior, use los siguientes parámetros para las búsquedas.

## 9. Búsqueda con Rachet utilizando los árboles iniciales y optimizando con el método de rearrreglos NNI.

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

10. Búsqueda con Rachet utilizando los árboles iniciales y optimizando con el método de rearrreglos SPR.

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

11. Búsqueda con Rachet utilizando los árboles optimizados con NNI y optimizando con el método de rearrreglos NNI.

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

12. Búsqueda con Rachet utilizando los árboles optimizados con NNI y optimizando con el método de rearrreglos SPR.

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

13. Búsqueda con Rachet utilizando los árboles optimizados con SPR y optimizando con el método de rearrreglos NNI

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

14. Búsqueda con Rachet utilizando los árboles optimizados con SPR y optimizando con el método de rearrreglos 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 optimizar.

Preguntas.

- ¿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.

Preguntas generales.

- 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?, ¿cuál fue el tiempo en que se obtuvo 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 algunos de los posibles comandos para

cada programa. Trate de encontrar otros comandos de búsqueda en estos programas u otros parámetros para los comandos usados en la práctica.

Número de réplicas	árboles retenidos/réplica
5	1
5	100
10	1
10	100
100	1
100	100

Número de Búsquedas continuas	Número de réplicas/Búsqueda	Número iteraciones en RACHET
1	5	50
1	5	100
1	10	50
1	50	50
5	1	5
20	1	5

## Práctica 5 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 da 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, solamente los nodos compartidos por todos los cladogramas son incluidos en el resumen final. Este método es el más conservativo, otros métodos presentan la información de algunos nodos, como el consenso de la mayoría, presenta solo 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 son muy 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. Kearny (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 el valor de corte seleccionado, el nodo es incluido en el consenso. Es posible hacer cortes más estrictos. 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 et al. 2001, 2002).

### Técnicas

En la mayor parte de los programas los consensos estrictos y de la mayoría están implementados (por ejemplo, TNT, PAUP\*, POY, WinClada o Component). 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 el árbol con los porcentajes in-



cluidos (TNT). Es importante que si va a usar esta clase de consenso, evalúe cómo recuperar los reportes de frecuencias si usa TNT.

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 longitud cero (por ejemplo TNT, NONA, WinClada, MacClade y PAUP\*), otros no (Component).

A partir de la matriz de datos: datos.consenso.dat,  
en WinClada, PAUP\*, POY o TNT:

1. Abra la matriz de datos, y realice una búsqueda convencional (ver práctica XXX poner vinculo). 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<sup>1</sup>. Para salvar los árboles siga las instrucciones previas usadas en la práctica XXXX.

---

<sup>1</sup>una vez calculado el consenso de la mayoría, WinClada tiene algunos problemas posteriores en el

#### 4. En PAUP\*:

- Siga el mismo esquema, realice una búsqueda y calcule 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 el nombre del comando y el signo ?: `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.
- También puede usar  

```
> save {strict}
```

  
o

---

manejo de los árboles en la forma como los gráficos. Por ejemplo, los *Hashmarks* no se pueden activar (no son dibujados), y modificar la topología produce cambios inesperados en las frecuencias de los nodos.

```
> save {majority}
```

para guardar los árboles de consenso directamente, después de calculados.

- En todos los casos **debe** haber abierto previamente el archivo de árboles. 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.

- En R, cargue primero las bibliotecas: ***ape*** y ***jrich*** .
  - Asigne haga un búsqueda en PAUP y guarde los árboles iniciales en formato Newick
  - ```
> hsearch; savet file=arboles.phy format=Newick;
```
  - Asigne los árboles a un objeto de nombre 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 cita Sharkey et al 2013 cladistics.
- `> plot(wconsensus(arboles, collapse=F))`
- `> plot(wconsensus(arboles, collapse=T))`
- compare las dos topologías y mire los grupos que fueron colapsados.

### Preguntas generales.

- 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 6 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 NANCY NEFF, 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; otros autores (por ejemplo, Goloboff, 1993, 1995) defienden 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 (buscar cita farris y kluge 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 su 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 codon no están muy preocupados e insisten que lo importante es diferenciar un tipo de codon de los otros (Swofford et al., 1996), por ejemplo:

”Third codon positions of mt protein-coding genes are known to be saturated, or nearly so, at deep phylogenetic levels ... . We therefore examined the effect of downweighting third positions by either assigning zero weight or employing transversion parsimony.” cite Springer2001

” Omission of the 3rd codon positions of the protein-coding genes from the analysis was justified due to significant base composition heterogeneity. Base composition heterogeneity can cause species to group by compositional similarity rather than evolutionary history (Lockhart et al., 1994).

In addition, sequence saturation of the 3rd codon positions of these genes (as was detected in Guzik et al. (2005) and Strugnell et al. (2005) can also produce spurious phylogenetic signal (e.g. Phillips Penny, 2003) and therefore removing these data from the analyses will have also removed some noise. This is likely to be of greater importance in the deeper divergences and more distant relationships where there has been a greater length of time for saturation to accumulate. ” cite Strugnell2014

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 difer-

entes juegos de pesos pueden producir diferentes respuestas que no son comparables (a nivel de los estadísticos de ajuste, como longitud).

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 (en vez de la longitud pesada); 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 (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, 2002) propusieron comenzar con el mejor resultado de varios árboles producto de *bootstrap* (u otro método que produzca pseudoréplicas). Para detener el procedimiento, Farris (1969) propuso esperar hasta que los resultados fueran autoconsistentes, es decir que se produjeran los mismos árboles (y por lo tanto los mismos pesos). Como la autoconsistencia puede verse afectada por el hecho de tener gran cantidad de árboles óptimos, es importante asegurarse de limitar el número de iteraciones. Kjer et al. (2001, 2002) proponen no iterar y solo mantener los pesos dados por los árboles de las pseudoréplicas.



Farris (2001) trató de solucionar simultaneamente 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 argumenta que si los pesos son independientes del punto inicial, las diferentes réplicas producirán aproximadamente los mismos resultados (los resultados se muestran como un árbol consenso de la mayoría de las diferentes réplicas).

El método de Goloboff es prácticamente igual a parsimonia tradicional. Es importante recalcar que aunque Goloboff 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, menos diferencia habrá entre los caracteres muy homoplásicos y los poco o no homoplásicos.

## Técnicas

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

- a) 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!

- 1) Abra la matriz y realice una búsqueda simple.
- 2) Repita la búsqueda pesando diferencialmente el tercer codón (por ejemplo asignandole un peso de 0), primero creamos la partición

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

y luego se asigna el peso

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

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

- 3) 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 la longitud de los arboles usando

```
> pscores;
```

b) 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);
```

. Nóte que escribimos N-1, pues 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 (el inverso de la usada en PIWE:  $\frac{h}{k+h}$ ); para comparar los reportes del ajuste generados en PAUP\*, puede obtener el ajuste usando

```
> fit*;
```

. En PAUP\* use

## Práctica 7 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 26.

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 tamaño 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 longitud 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 la longitud 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 está 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 cuáles 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 automática. 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 morfológi-

cos 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. El problema es que muchos programas no lo tienen definido explícitamente, por lo que 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 la longitud máxima especificada. 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, 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, búsquese 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 [; 0 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 cada 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 replicas 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 replicas 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.

### 4) En TNT:

*a'* Abra la matriz datos.particiones.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.

#### 5) **En RaxML:**

*a'* Abra la matriz `datos.soporte.pars.dat` en un editor de texto y conviértala en un formato que RaxML pueda leer, use winclada 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 bootstrapping 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 la azar
- `-#` es el número de réplicas.

*c'* Al finalizar revise los archivos llamados:

`RAXML_info.resultados`

`RAXML_bootstrap.resultados`

*d'* Para un bootstrapping 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.

#### 6) 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 NONA/TNT, por lo que los comandos de búsqueda y el número de réplicas deben ser acordes con sus tiempos.

WinClada, PAUP\*, POY y TNT.

En NONA y TNT el soporte de Bremer se obtiene con

POY implementa el soporte de Bremer con

Preguntas

Práctica

Compare con sus compañeros sus resultados de soporte basado en permutaciones y en soporte de Bremer. ¿Usted sugeriría (o no) el uso de soporte relativo? Explique las razones de su escogencia.

¿Los nodos más soportados son los mismos? ¿Existe correlación entre los diferentes métodos?

Generales Escriba un pequeño ensayo con sus puntos de vista sobre los diferentes métodos de medición de soporte, destacando tanto los puntos positivos como los negativos de cada metodología.

Indique cuáles métodos, y por qué, preferiría si trabaja: (1) solo con datos morfológicos, (2) solo con datos de ADN, (3) una matriz combinada.

¿Cree usted que pueden existir métodos más eficaces para medir el soporte? De ser así, escriba un pequeño ensayo sobre las cualidades esperables teniendo en cuenta los distintos tipos de datos.

¿Qué es lo que usted desearía y esperaría de un método que mida el soporte?

POY Al remover fragmentos completos de ADN, ¿puede influenciar los cálculos de soporte usando métodos de remuestreo?

#### Literatura recomendada

Goloboff & Farris, 2001 [Presentación del soporte relativo de Bremer].

Goloboff et al., 2003 [Propuesta del muestreo simétrico así como de diversas medidas de soporte basadas en remuestreo].

Grant & Kluge, 2003 [Una crítica extensa a los diferentes formas de medición de soporte].

Ramírez, 2005 [Excelente revisión de los diferentes métodos para medir el soporte]

## Práctica 8 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 se originan en biología molecular o en genética de poblaciones. Bajo la estimación estadística se asume que un modelo genera las secuencias 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 asigna 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 y 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  $\Gamma$ ) y, finalmente, si existe o no un reloj molecular.

La información básica se puede consultar en citar Felsenstein () Swofford et al. (1996) es una referencia básica para comprender el desarrollo de los distintos modelos. Page & Holmes (1998) ofrecen un capítulo ilustrativo sobre el tema. La idea general de modelos también se aplica para la evolución proteica, 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 de valor de verosimilitud, sino si la mejora es (o no es) estadísticamente significativa, dado el número de parámetros usados en el modelo.

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<sup>1</sup>) que compara la proporción en la que se incrementa la verosimilitud al agregar el parámetro; se asume una distribución  $\chi^2$  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  $\chi^2$  es válida, y solo permite comparar modelos anidados; la prueba es un tanto conflictiva al comparar GTR con GTR+I o GTR+ $\Gamma$ ; 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<sup>2</sup>, o la cantidad de información bayesiana<sup>3</sup>. En estos casos se calcula cuanta información contiene el modelo dados los parámetros de este. La ventaja es que se

---

<sup>1</sup>

$$\delta = -2\ln \frac{Max[L_0(modelo - nulo|datos)]}{Max[L_1(modelo - alternativo|datos)]} = 2(\ln L_0 - \ln L_1) \quad (8.1)$$

<sup>2</sup>AIC= -2lnL + 2N, donde lnL es el ajuste del modelo reportado y N el número de parámetros libres

<sup>3</sup>BIC=-2lnL + Nlnn, donde lnn, es el logaritmo natural de la longitud de la secuencia



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 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 en modo de texto, con modelos diferenciales lo que facilita su uso en clusters de computadores; es posible que obtenga distintos modelos dependiendo de la forma como estructure su análisis. R en conjunto con *ape* y PhyML puede realizar el cálculo del mejor modelo de evolución molecular y graficar los resultados.

## Técnicas

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

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 cual 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 BIC con los obtenidos por sus compañeros.
- 7) Repita 6b3 pero use una búsqueda de FIXED BIONJ+JC y evalúe el resultado del test jerárquico. ¿Existe alguna diferencia entre los modelos sugeridos por cada enfoque?
- 8) Repita desde 6b3 con un árbol base de ML optimizado y evalúe el modelo. ¿Existe alguna diferencia?

En R+ape+PhyML

- 1) Abra R y revise si instalada la biblioteca **ape**, en caso de no tenerla, instálela.
- 2) Coloque como directorio activo el directorio donde tenga los datos y PhyML
- 3) Cree un archivo con las instrucciones:

```
## cargamos la libreria ape
```

```
> library(ape)
```

```
##
## leemos datos alineados en formato tipo phylip
> DNA - read.dna("alineado.phy")

## listado de tamaño de las secuencias
> table(unlist(lapply(DNA, length)))

## Test de modelos de evolución con ape

## cargamos la libreria ape
> library(ape)

## con phymltest probamos 28 modelos en phyml
## ape + R hacen todo el proceso
## en linux cambie a
## execname = "./phyml", si es local o
##
## execname = "phyml", si es global
##
## en windows
##
> modelo-phymltest("alineado.phy", format = "sequential", itree = NULL, exclude
= NULL, execname = "phym1.exe", append = FALSE)

##
## use format = "interleaved" si aplica a su matriz
##
```

```
## con print puede revisar los valores de AIC
##
> print(modelo)
##
## con summary puede tener un resumen de los valores del test
jerárquico
##
> summary(modelo)
##
## grafique con 'plot' los valores de AIC
##
> plot(modelo, main = 'test de modelos para ML, usando PhyML')
```

- 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 leible.

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 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

5) 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.

6) 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**.

7) Para los mismos modelos y el escogido por el hLRT en JModel-Test, calcule el criterio de información bayesiano. Compare sus resultados, tanto para AIC como para BIC con el de sus com-

pañeros, y determine cuál es o son los mejores modelos para esos criterios. Ordene los modelos del mejor al peor.

Preguntas.

- ¿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?

Preguntas generales.

- ¿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].

## Práctica 9 Máxima verosimilitud

### Introducción

Algunos autores han sugerido que la parsimonia es un modelo de evolución que asume que la tasa de transformaciones es baja (por ejemplo, Swofford et al., 1996; Felsenstein, 2004). 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 8 sobre selección de modelos (ver página 64). 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 (aunque han habido intentos de generalizar los modelos de evolución a datos morfológicos, ver Lewis (2002)).

Uno de los principales problemas con la estimación de verosimilitud es que el valor depende de la longitud de las ramas. En la actualidad, se ignora al principio 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 cómo se hacen los cálculos relacionados con estos

métodos.

### 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 4, 26). Existen diferentes formas de encontrar un árbol inicial en verosimilitud. Los sistemáticos moleculares que usan modelos suelen empezar por un árbol de distancias (NJ: neighbor joining). El problema de este inicio es que para una configuración particular, solo existe un árbol de NJ<sup>1</sup>. Otra forma es la descomposición de estrella, donde se tiene una politomía 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 suelen 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 del capítulo 8.

Preguntas. *Argumente qué criterio utilizó para seleccionar el modelo.*

---

<sup>1</sup>tal y como lo hemos recalado 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.



*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 producto del análisis con parsimonia; recuerde salvar los árboles de cada modelo (incluida la longitud 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  $\Gamma$ , 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 *6b1a*. Anote el valor de la verosimilitud ( $-\ln L$ ) 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  $\Gamma$ . Reporte los valores de verosimilitud para cada modelo.

Preguntas. ¿Las topologías obtenidas son similares?

*e'* Repita el ejercicio hecho en *6b1d*, pero use la estimación empírica del parámetro  $\Gamma$  para HKY, F81 y TRM. Reporte los valores de verosimilitud para cada modelo.

*f'* Repita el ejercicio hecho en *6b1d*, pero use invariantes. Reporte los valores de verosimilitud para cada modelo.

Preguntas. ¿Las topologías obtenidas son similares?

Preguntas. ¿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 comandos (que se usa de manera indirecta al obtener el modelo óptimo con R+ape o con JModelTest) y el segundo es una interfase 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  $\Gamma$ .

Para un archivo de entrada de los datos denominado *archivo*, el programa genera dos archivos de salida: *archivo.phy\_phyml\_stats.txt* (salida con los estadísticos) y *archivo.phy\_phyml\_tree.txt* (salida con el árbol en notación Newick o PHYLIP). Estos dos archivos en caso de existir pueden ser sobrescritos (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 ira 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.

Preguntas. ¿qué parámetros usa por defecto el programa?

*b'* Estime el mejor modelo para la matriz, use los procedimientos

del capítulo 8. Argumente qué criterio utilizó para seleccionar el modelo.

*c'* Busque el árbol de ML usando el modelo obtenido en 6b2b, haga una corrida con *nni*, Anote el valor de la verosimilitud ( $-\ln L$ ) reportado y la topología obtenida.

*d'* Repita 6b2c 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  $\Gamma$ . Reporte los valores de verosimilitud para cada modelo.

**Preguntas.** Compare la topología y la longitud 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 comandos.

3) en RaxML

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

*b'* 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 tipo ras a partir de un archivo de datos "**dna.phy**" y con un prefijo de salida de ML0, usando el

modelo GTRGAMMA.

*c'* Haga una corrida en RaxML, con 10 búsquedas ras de ML para un análisis particionado.

*d'* use las instrucciones:

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

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

*e'* 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 simpleDNApartition.txt -s dna.phy  
-n ML02
```

### Preguntas.

- Comparé sus resultados con los de sus compañeros. ¿Cuál 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?

Preguntas generales. Recientemente 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].

## Tipografía

Para

texto principal Verdana para mejorar legibilidad en el texto  
leído en pantalla, como se supone que será el uso del manual.

texto mono espaciado usamos **inconsolata**,

para texto "bold" las opciones son

vintage [www.dafont.com/vintage.font](http://www.dafont.com/vintage.font)

ubuntu bold mono / ubuntu latin

## Práctica 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. Dado que "por unos pocos dólares" se puede tener un repertorio apropiado de programas, la condición central a tener en cuenta es la **velocidad**. 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 aprendizaje 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.

## A.1 Editores de matrices 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 requiere de NONA para realizar las búsquedas (PIWE/Hennig86 son deseables, pero no obligatorios).

Bajo Windows, este 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 un par de "conflictos" con el manejo de imágenes que pueden ser incómodos; además, el usuario debe tener presente que el programa puede numerar (taxa, caracteres y árboles) desde cero (la manera lógica para los programadores), 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 polimorfismos.

**Terms/Chars:** Estos menús 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.

**Analyze:** 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 la longitud, 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, requiera Java virtual machine.

Disponibilidad (McClade): Descontinuado pero puede acceder desde el sitio de los autores para el programa que funciona en OS X (hasta 10.6):

<http://macclade.org/index.html>.

Disponibilidad (Mesquite): Gratuito.

<http://mesquiteproject.org>.

MacClade es un programa cuyas cualidades gráficas son impresionantes, su interfase es agradable, sencilla y no transmite miedo a los novatos. Dadas sus magnificas propiedades gráficas, es **muuy** 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 esta 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 si sola ¡sí, que hable! Un lector automático lee en forma descendente los taxa de su matriz facilitándote un poco

el trabajo (puede usar de hecho 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.

La última ventana en el menú es **Windows**, el cual permite moverse entre las interfaces 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 columnas,

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 politomias como politomias blandas o duras (Coddington & Scharff, 1994).

**$\Sigma$ :** esta ventana incluye el manejo de todos los estadísticos del árbol, como la longitud, 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 el capítulo sobre programas).

**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 politomias o tal y como son.

**Display:** finalmente en esta ventana se manejan los aspectos gráficos de la representación 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 de los taxa (como números o nombres).

Aunque con menos capacidades que McClade, Mesquite es una herramienta poderosa, teniendo en cuenta que es gratu-



ito. Su principal falla es que no permite ver las transformaciones en los nodos de todos 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., 2007.

Plataforma: MS-DOS, Windows, MacOS, Unix.

Disponibilidad: es posible descargar un demo funcional, para 10 corridas. El programa tiene un costo de US80.00.

<http://www.zmuc.dk/public/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 ver cuáles son los parametros, 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 escriba

```
> mult: replic X;
```

para ejecutar el número de replicas 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, el soporte de Brtemer dentro de los límites del Bremer absoluto y puede calcular la cantidad de grupos soportados-contradichos (FC).

Con

```
> resample
```

usted puede configurar como quiere la permutación. Con

> subop

(también en NONA) usted puede indicar la longitud 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. Infortunadamente los resultados no se almacenan en ninguna parte, por lo que debe generar una salida (en formato de texto).

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

**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 superárboles, así como árboles al azar, y el manejador de etiquetas de los nodos (*tags*).

**Data:** Aparte de la configuración [U+FFFD] 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).

Usted puede encontrar un manual para el lenguaje de macros en la dirección:

<http://www.zmuc.dk/public/phylogeny/TNT/scripts/>

### A.2.2. NONA

Autor: Goloboff, 1998.

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

idad: Gratuito en conjunto con otros programas en  
<http://www.zmuc.dk/public/phylogeny/Nona-PeeWee>.

Este programa es la mejor opción que existe para búsquedas bajo parsimonia, dado que es gratuito, tiene lenguaje de macros y es veloz. Si está interesado en búsquedas usando concavidad, puede usar PIWE; si el interés son las búsquedas de Sankoff entonces PHAST/SPA son los programas requeridos. Todos estos son similares a NONA en cuanto a comandos se refiere, y están incluidos con la distribución de NONA.

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. Poy

Autor: Varón et al., 2007.

Plataforma: Linux, MacOS y Windows.

Disponibilidad: gratuito.

<http://research.amnh.org/scicomp/projects/poy.php>

Es el único programa disponible que realiza un análisis simultáneo sin utilizar alineamiento previo de las secuencias moleculares. También puede usarse para realizar alineamientos, o para hacer búsquedas convencionales de datos previamente alineados o morfológicos. La versión actual posee únicamente una implementación de parsimonia, pero se planean versiones que también realicen búsquedas bajo máxima verosimilitud.

El programa posee cuatro ventanas: una de salida, donde



se imprimen salidas pedidas por el usuario (por ejemplo, un árbol, o la ayuda). Una ventana de comandos, donde se escriben las ordenes al programa, una donde muestra que clase de tarea se esta realizando y en la última donde se muestra el progreso de las búsquedas.

Las matrices y arboles son abiertos usando

```
> read('archivo')
```

, donde archivo es el archivo a leer. Puede leer multiples conjuntos de datos, bien sea abriendo uno por uno, o varios en la misma instrucción como

```
> read('arch1','arch2')
```

. Los datos se van agregando. Para limpiar la memoria se utiliza

```
> wipe()
```

.

Se buscan árboles con

```
> build(x)
```

, que realiza x arboles de Wagner. Para mejorar dichos árboles hay que usar

```
> swap()
```

. Notese que las instrucciones están separadas. Es posible hac-

er búsquedas más sofisticadas usando nuevas tecnologías, como ratchet implementado en

```
> perturb(iterations: x, ratchet())
```

, donde se harían x iteraciones de rathcet, con

```
> swap(drift)
```

o

```
> swap(annealing)
```

se hace deriva de árboles y cristalización simulada, y

```
> fuse()
```

hace fusión de árboles.

El comando para manejar costos es

```
> transform(tcm())
```

. Se puede modificar los costos de transformacion para datos "estáticos" como morfología con

```
> transform(static, weight:2)
```

que daría un peso de 2 a cada transformación de datos morfológicos. Con

```
> transform(tcm(1,2))
```

se da un peso de 1 a las sustituciones y de 2 a los gaps y

caracteres estáticos, ese es el valor que viene por omisión.

Con

```
> transform(fixedstates)
```

se implementa el método de estados fijos de Wheeler.

Es posible calcular soportes con

```
> calculate_support()
```

que incluye *bootstrap*, *jackknife* y soporte de Bremer (por omisión). Las salidas se realizan con

```
> report('archivo')
```

pueden ser cladogramas en formato parentical, dibujados en *ascii*, o en *postscript*. También es útil para conocer estadísticas de los árboles, las optimaciones de los nodos y la integridad de los datos.

El programa funciona desde computadoras de escritorio hasta *clusters*. Puede ejecutarse directamente en la línea de comando o en procesos de lotes.

#### A.2.4. PAUP\*

Autor: Swofford, 2002.

Plataforma: MWindows, MacOS, Unix.

Disponibilidad: el programa para l interface 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.

<http://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 interface dominante llegue a ser por línea de comandos; la interface para Windows y MacOSX en modo gráfico tiene el mismo acercamiento; como NONA y TNT tiene la opción de manejar un gran número de parametros 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 cuántos 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 comando (parámetro `treefile=nombre`, donde nombre es el archivo.). 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.2.5. 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 incluido 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 de 'parsimonia'tal y como fue descrito por Tuffley & Steel (1997) que se activa usando

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



## Práctica 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 clausula

```
> 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 clausula *#nexus*, y luego con el bloque *data*. se puede definir si los caractertes son morfológicos, ADN o proteínas. Los polimórfismos se colocan entre paréntesis re-

dondos. 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.1.3. MALIGN

El esquema de MALIGN es similar al de GenBank.

```
> SequenceA
```

```
1 CAGCAGCACG CAAATTACCC ACTCCCGGCA CGGGAGGGTA GTGACGAAAA ATAACAATAC
```

61 CCGTC

SequenceB

1 CAGGCACGCA AATTACCCAC TCCCGGCAGA GGTAGTGACA AAAAATAACG ATACGGGACT

61 CCGTCAC

SequenceC

1 GGCACGGAGG TAGTGACGAA AAATAACGAT ACGGGACTCA TCCGAGGCC CGTAATCGGA

SequenceD

1 AAATTACCCA CTCCCGGCAC GGAGGTAGTG ACGAAAAATA ACGATACGGG ACTCA

SequenceE

1 GAGGTAGTGA CGAAAAATAA CAATACAGGA CTCATATCCG AGGCCCTGTA ATT

(Para proteínas, el asterisco "\*" fuerza la interpretación como proteína)

> SequenceF

1 ILAVEELVI SLIVES

SequenceG\*

1 AAYVTTTCC KKYK

#### **B.1.4. POY**

POY y Clustal, utilizan el formato de FASTA (la primera línea tiene un > seguido por el nombre y comentarios de la secuencia; la siguiente línea comienza la secuencia como tal).

> taxonA Comentarios

aaacgt

aac

taxonB

aaacgt

## B.2 Formatos de árboles

### B.2.1. NONA

NONA, Hennig86, WinClada y TNT usan este formato; con

```
> *
```

indican que hay mas 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)
```