CAPÍTULO 1

Soporte en parsimonia y ML

Introducción

Luego de obtener los cladogramas, para algunos es importante saber qué tan fuerte es la evidencia que da soporte a un nodo. Es importante 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.

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) da una idea de 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. En el **Jackknife** y en la **remuestreo simétrico** 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 más subóptimos sean los árboles, más nodos de los encontrados en los árboles óptimos desaparecerán. La idea, entonces, es contar la diferencia de longitud necesaria para que el nodo colapse.

Las mediciónes del soporte con remuestreo (usando consenso de la mayoría) o con cladogramas subóptimos (usando soporte de Bremer) sólo tienen en cuenta la cantidad de evidencia favorable. Un nodo puede estar bien soportado (por un carácter no contradicho), y es posible que nodos que aparecen con mejor 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 laternativo. 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 esta presente en el consenso de los árboles más parsimoniosos de la partición, el soporte se halla de la forma convencional; si por el contrario el nodo no está presente, se busca el árbol más parsimonioso que contenga el nodo, y la diferencia de pasos entre este árbol y el árbol más parsimonioso de la partición es el soporte de Bremer negativo del nodo (es negativo, indica qué tan contradicho es el nodo). Los soportes por particiones permiten detectar cuáles son las particiones que sugieren el nodo y cuales lo contradicen. De esta forma, no solo se mide el soporte, sino que es posible observar la congruencia por nodos entre las particiones. Aunque al sumar los valores de Bremer de las particiones el resultado puede ser igual al valor del soporte de Bremer, esto no siempre sucede.

1.1. Técnicas

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 parametros de la perturbación son difíciles de aclarar. En *bootstrap* no existe problema, puesto que se toma de la matriz un carácter al azar hasta completar una matriz del mismo tamaño de la original, así no existe diferencia entre las diferentes fuerzas de permutación. El problema es que en las matrices hay caracteres no informativos que pueden sesgar la matriz producto de la permutación. Otro inconveniente es que la mayoría de los datos moleculares son secuencias sucesivas de caracteres, y los morfológicos no es fácil entender cómo están relacionados. El método exige una distribución homogénea de la información y que el muestreo sea al azar.
Con *jackknife* y permutación simétrica, cada carácter es alterado independiente de los demás, con lo que autapomorfías y caracteres no informativos no influyen en la distribución final. Además, los métodos basados en permutaciones exigen que los resultados de la matriz permutada sean confiables. Es decir, búsquedas estrictas. Esto hace que los métodos consuman mucho tiempo. Este problema se ha solucionado usando muchas búsquedas muy superficiales, y usando el consenso estricto (o uno muy fuerte) de las búsquedas independientes (ver Farris et al., 1996; Goloboff & Farris, 2001), aunque eso podría disminuir la frecuencia de los nodos con soportes bajos (en el dado caso de que esos sean de interés).
Es importante notar que cuando se hacen remuestreos con POY, en el caso de los datos moleculares, los fragmentos completos de ADN son usados como caracteres, y no cada base particular (como se hace tradicionalmente con una matriz previamente alineada), por lo que jackknife puede eliminar fragmentos completos de datos moleculares.
El soporte de Bremer utiliza árboles subóptimos. 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.

1.2. Materiales

Matriz de datos (datos.soporte.pars.dat). Matriz de datos (datos.particiones.dat). Macro para TNT (brempart.run)

1.3. Métodos

En WinClada, PAUP*, POY y TNT:
(1) 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). Compare sus resultados con los de sus compañeros.
En TNT y NONA:
(2) Abra la matriz de datos, búsque el árbol más parsimonioso.
(3) Retenga 1000 árboles y acepte subóptimos hasta 5 pasos más largos. 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*;`.
(4) Con los árboles obtenidos calcule el soporte de Bremer, absoluto y relativo.
(5) Repita (3) y (4), pero reteniendo árboles 7 pasos más largos.
En POY:
(6) Calcule el soporte de Bremer.
En TNT:
(7) 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.
(8) 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.
En RaxML:
(9) Abra la matriz datos.soporte.pars.dat en un editor de texto y conviertala en un formato que RaxML pueda leer, use winclada de ser necesario.
(10) Para un análisis no particionado use los comandos:
./raxml –f i -s datos.phy -n resultados –m GTRGAMMA –b 1234 -# 20
donde:

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

(11) Al finalizar revise los archivos llamados:
RAxML_info.resultados RAxML_bootstrap.resultados.

(12) Para un boostrapping rápido use la línea de comandos

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

donde los cambios frente a 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.

1.3.1. Programas

WinClada, PAUP*, POY y TNT.

1.3.2. Comandos

En NONA y TNT el soporte de Bremer se obtiene con `bsupport`, **debe** haber calculado previamente los árboles subóptimos con `subop` y la respectiva búsqueda. Si usa la instrucción con un asterisco `bsupport*`, el soporte calculado es el relativo en forma de porcentaje; también puede usar `bsupport [;` o `bsupport ];`. 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.
POY implementa el soporte de Bremer con `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 búsquedaen general simpledurante 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.
En PAUP* solo se hacen permutaciones sobre la matriz; con `bootstrap` se configura y ejecuta el *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.

1.4. Preguntas

1.4.1. 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?

1.4.2. 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?

1.5. 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, incluyendo los métodos más recientes].

CAPÍTULO 2

Modelos de evolución

Introducción

Los sistemáticos moleculares se enfrentan ante un conjunto de datos distinto en algunos aspectos del que enfrentan los morfólogos: un mismo tipo de estados (ACGT) se encuentra repetido a lo largo de toda la matriz de datos. En general, las ideas desarrolladas para analizar los datos moleculares tienen una aproximación estadística, dado el origen de muchos de los análisis moleculares en biología molecular o en genética de poblaciones. Bajo la estimación estadística se asume un modelo que genera las secuencias de ADN y con base en el modelo se estima qué tanto se ajustan las hipótesis filogenéticas a los datos. A ese procedimiento se la conoce como **máxima verosimilitud** (*maximum likelihood* en la literatura en inglés).
Los modelos moleculares se basan en la asignación de la probabilidad de transformar una base cualquiera en otra base, incluida esa misma base. El cálculo de esas probabilidades está influenciado por diferentes parámetros, que, en principio, deberían ser estimados de los datos, pero también se han usado parámetros predefinidos, que quizás no la mejor opción.
El modelo con le menor número de parámetros es conocido como JC ó JC69, por los autores y el a no en que fue propuesto (Jukes y Cantor, 1969); se asume que la probabilidad de una base para transformarse en otra es siempre igual y que la frecuencia de las bases es igual (al menos al inicio de la evolución de los organismos). De ahí en adelante pueden agregarse más parámetros que hacen más complejo el modelo en diferentes direcciones. Puede asumirse diferencia entre cada tipo de base (ya sea químicamente AG-CT, o cada una por separado), diferencia de las probabilidades basada en la proporción de cada base, tener en cuenta o no los sitios invariantes, asumir que algunos sitios son más propensos a cambiar que otros (función Γ) y, finalmente, si existe o no un reloj molecular.
Swofford et al. (1996) es una referencia básica para comprender el desarrollo actual de los modelos. Page & Holmes (1998) ofrecen un capítulo ilustrativo sobre el tema. La idea general de modelos también se aplica para la evolución proteíca, pero dado que casi siempre se tiene tanto la secuencia de aminoácidos como la nucleotídica, esta última es la preferida al explotar más directamente la información (al menos a nivel de variación).
Es de notar que las probabilidades de los datos (la verosimilitud) suelen ser muy peque nas, por lo que para facilitar los cálculos se utiliza el negativo del logaritmo natural de la verosimilitud, que es el valor reportado por los programas así como en la literatura.

2.1. Técnicas

Debido al aumento en el número de los parámetros,los modelos puedan explicar más satisfactoriamente los datos (pero esto no implica que los modelos sean ***más*** *realistas*); por lo que la selección de un modelo no puede basarse en que el modelo mejora la probabilidad de los datos, 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[1]) que compara la proporción en la que se incrementá la verosimilitud al agregar el parámetro; se asume una distribución χ^2 para la distribución de esa proporción, tomando el número de parámetros extra como los grados de libertad. El problema de esta prueba es que no es claro si la distribución χ^2 es válida, y solo permite comparar modelos anidados; la prueba es un tanto conflictiva al comparar GTR con GTR+I o GTR+Γ; se asume que la forma como se agregan los parámetros no sesga el resultado; usted puede revisar esta afirmación usando programas como `ModelTest`.
La otra forma está basada en medidas de información, como el criterio de Akaike, o la cantidad de información bayesiana. En estos casos se calcula cúanta información contiene el modelo dados la cantidad de parámetros que posee. La ventaja es que se puede hacer la comparación entre modelos sin tener que tomar una secuencia particular.

2.2. Materiales

Matrices de datos formato:
`NEXUS` para `ModelTest` (datos.modelo.dat).
`PHYLIP` para `JModelTest` (datos.oualin.dat).
Matriz de instrucciones para R:
`model.R`

2.3. Métodos

2.3.1. Modeltest+PAUP

1. Abra la matriz de datos y ejecute el archivo *modelblock3* en `PAUP*`.

2. Use la salida producida por `PAUP*`: model.scores. como archivo de entrada para `Modeltest`.

3. Siga paso a paso el test jerárquico presentado. Intente realizar otro test jerárquico comenzando con otro juego de parámetros (note que la salida de `ModelTest` incluye

[1]
$$\delta = -2log\frac{Max[L0(modelo - nulo|datos]}{Max[L1(modelo - alternativo|datos]} \tag{2.1}$$

los ajustes de los 56 modelos explorados).

4. 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 **peque no**[2].

5. Para los mismos modelos y el escogido por el hLRT en `ModelTest`, calcule el criterio de información bayesiano[3]. Compare sus resultados, tanto para AIC como para BIC con el de sus compa neros, y determine cúal es o son los mejores modelos para esos criterios. Ordene los modelos del mejor al peor.

2.3.2. JModeltest+PhyML

1. Abra en un editor la matriz de datos. Identifique cuales son 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**.
5. Revise la salida de AIC y anote cual es el modelo de mayor ajuste para la matriz de datos, asi como los parámetros de este modelo.
6. Repita el análisis a partir del cálculo de criterio bayesiano (BIC). Compare sus resultados, tanto para AIC como para BCI con los obtenidos por sus compañeros.
7. Repita 3 pero use una búsqueda de `FIXED BIONJ+JC` y evalue el resultado del test jerárquico. ¿Existe alguna diferencia entre los modelos sugeridos por cada enfoque?
   .
8. Repita desde 3 en la casa con un árbol base de ML optimizado y evalue el modelo. ¿Existe alguna diferencia?

2.3.3. R+ape+PhyML

1. Revise si su computador tiene `R` instalado; en caso contrario baje el binario para windows (http://cran.r-project.org/bin/windows/base/) e instalelo; una vez abra el programa baje `ape` y cualquier otro modulo solicitado.
2. Abra `R` y coloque como directorio activo el directorio donde tenga los datos y `PhyML`

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

[3]BIC=-2lnL + Nlnn, donde lnn, es el logaritmo natural de la longitud de la secuencia

3. En R use las instrucciones del apéndice ?? en la página ?? (`model.R`), o cargue cada instrucción por línea de comandos desde un editor de texto.

4. Compare el modelo sugerido por AIC y por el test jerárquico.

2.3.4. Programas

En general se puede usar Modeltest o MrModeltest, que es una modificación de Modeltest pero calcula un menor número de modelos; en caso que de usar MrModeltest, tenga en mente que **no** ha evaluado todos los modelos.
Si desea un cálculo más cuidadoso entonces la elección es 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 loa forma de implementar el test jerárquico, desde JC hacia GTR (*forward*) o en sentido contrario (*backward*); es posible que obtenga distintos modelos dependiendo de la forma como estructure su análisis.
R es uno de los programas para anális estadístico más versatiles del momento, además de ser gratuito permite la implementación de múltiples procesos de álculo, de tal manera que es posible realizar distintos tipos de operaciones y cálculos desde estadistica básica hasta análisis en evolución. R sirve de plataforma para ape (faltan citas:Paradis et al., 2008), un módulo de R que permite distintas acciones con matrices y árboles. En conjunto con PhyML, ape puede realizar el cálculo del mejor modélo de evolución molecular.

2.3.5. Comandos

Modeltest usa par sus cálculos de likelihood a PAUP*, el programa se ejecuta a través de la línea de comandos, o usando un archivo por lotes (archivo .bat) tal y como lo hizo en la práctica de alineamiento con POY (vea la página ??); la secuencia básica de instrucciones desde la línea de comandos es:

```
modeltest <model.scores >salida.txt
```

donde salida.txt es su archivo de salida con la terminación texto (y en formato de texto).

En R Puede usar un archivo de instrucciones en una manera similar a las instrucciones en POY con la línea de comandos:

```
R -f model.R
```

si todas sus intrucciones estan en el archivo `model.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 facilmente leible.

2.4. Preguntas

2.4.1. Práctica

¿Es el modelo escogido por el criterio de Akaike igual al escogido en el test jerárquico? ¿Usted esperaría que lo fuesen?
Dada la exploración en (3), ¿es igual el resultado para las dos exploraciones? ¿Usted esperaría que fuesen iguales ó desiguales?
Para las listas obtenidas en (5), ¿son iguales las listas en AIC y BIC?
De usar los tres programas ¿usted espera la misma respuesta?

2.4.2. 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

2.5. Literatura recomendada

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

CAPÍTULO 3

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 2 sobre selección de modelos (ver página 6).
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 ha habido intentos de generalizar los modelos de evolución a datos morfológicos, 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.

3.1. 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. 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[1]. 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

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

en verosimilitud, o si se desea una respuesta más rápida, un árbol obtenido por parsimonia y luego permutar las ramas usando verosimilitud. 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 relaidad variaciones de GTR).

3.2. Materiales

Matrices de datos (datos.like.dat, dna.phy). Particiones para dna.phy (simpleDNApartition.txt).

3.3. Métodos

3.3.1. en PAUP

1. Antes que nada estime el mejor modelo para la matriz, use los procedimientos del capítulo 2. Argumente qué criterio utilizó para seleccionar el modelo.
2. Abra la matriz de datos, busque el árbol más parsimonioso.
3. Permute las ramas de ese árbol, usando el modelo obtenido en 1. Anote el valor de la verosimilitud (-lnL) reportado.
4. Pruebe alternativamente los siguientes modelos: JC, HKY, F81 y TRM, usando como frecuencia para las bases la estimación empírica y sin el parámetro Γ. Reporte los valores de verosimilitud para cada modelo.
5. Repita el ejercicio hecho en 4, pero use la estimación empírica del parámetro Γ para HKY, F81 y TRM. Reporte los valores de verosimilitud para cada modelo.
6. Repita el ejercicio hecho en 4, pero use invariantes. Reporte los valores de verosimilitud para cada modelo.
7. Estime si los datos se ajustan o no al reloj molecular.

3.3.2. en PhyML

1. Haga una corrida por defecto en PhyML, ¿qué parámetros usa por defecto el programa?
2. Estime el mejor modelo para la matriz, use los procedimientos del capítulo 2. Argumente qué criterio utilizó para seleccionar el modelo.
3. Busque el árbol de ML usando el modelo obtenido en 2, haga una corrida con `nni`, Anote el valor de la verosimilitud (-lnL) reportado y la topología obtenida.
4. Repita 3 cambiando la búsqueda a `spr` y a `nni+spr`.
5. Pruebe alternativamente los siguientes modelos: JC, HKY, y GTR, con y sin el parámetro Γ en los tres modelos. Reporte los valores de verosimilitud para cada modelo.
6. Compare los árboles resultado de cada salida, no solo por topología sino tambien por longitud de las ramas

3.3.3. en RaxML

1. Haga una corrida en RaxML, con 10 búsquedas RAS de ML.

2. Haga una corrida en RaxML, con 10 búsquedas RAS de ML para un análisis particionado.

3.3.4. Programas

PAUP*, PhyML, PHYLIP, RaxML o PAML; entre otros[2].

3.3.5. Comandos

PAUP*

La búsqueda inicial en PAUP* hágala bajo el criterio de parsimonia, y posteriormente pase al criterio de ML y haga una peque na 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 Γ, distribución de bases, tipos de cambios, invariantes).
Puede consultar el `model.block` que acompa na a ModelTest (y usado en la práctica sobre obtención del modelo óptimo) para que vea cómo se implementa cada uno de los modelos. Recuerde que ModelTest usa explícitamente los 56 modelos; también puede usar la salida de ModelTest para cambiar el modelo. Las instrucciones generadas por ModelTest pueden ser incluidas **dentro** del archivo de datos, para lo cual puede usar un editor de texto, Mesquite o MacClade, pero **no** WinClada.

PhyML

PhyML es un programa mucho más rápido que PAUP* y por lo tanto debería ser una de sus primeras opciones, el programa cuenta con dos esquemas de instrucciones: el primero por línea de commandos (que se usa de manera indirecta al obtener el modelo óptimo con R+ape o con JModelTest) y el segundo es una 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 Γ.
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 sobreescritos (R) o la salida actual puede ser añadida a una salida previa (O).
Inicialmente solo debe dar el nombre de la matriz de datos (en formato PHYLIP) y el programa le ira guiando por las decisiones a tomar. Puede avanzar al siguente menú con +, regresar con -, o iniciar el análisis con Y.

[2]puede consultar http://evolution.genetics.washington.edu/phylip/software.html

RaxML

RaxML es un programa mucho más rápido que PhML y por lo tanto debería ser una de sus primeras opciones, el programa cuenta con un solo esquema de instrucciones: por línea de commandos.

Para usarlo 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.

Si lo que desea es un análisis particionado debe usar:

```
raxml -m GTRGAMMA -p 12345 -q simpleDNApartition.txt -s dna.phy -n ML12a
```

Los límites de cada partición se encuentran en el archivo simpleDNApartition.txt.

si prefiere hacer el análisi 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 ML12a
```

3.4. Preguntas

3.4.1. Práctica

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

3.4.2. 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?

3.5. Literatura recomendada

Swofford et al., 1996 [Una descripción muy completa de la forma como se calculan las verosimilitudes y los métodos de verosimilitud para encontrar árboles].
Lewis, 2001 [Introduce el uso de modelos en morfología].

CAPÍTULO 4

Soporte en ML

Introducción

Tal y como se hizo para el análisis de parsimonia (práctica 1), bajo ML es importante conocer la evidencia de cada nodo, máxime cuando en ML no hay transformaciones, por lo que los valores de soporte son indicadores de la **evidencia** del nodo. Los métodos para medir el soporte están basados en remuestreo de los caracteres; así, tanto *bootstrap* como *jackknife* son aplicables a datos moleculares, aunque el método preferido es el *bootstrap*; dado que no se usa una distribución en particular se denomina ***bootstrap* no paramétrico** en contraposición al **paramétrico**, que se basa en la simulación de los datos dado un modelo particular. Algunas evaluaciones empíricas han mostrado que el análisis depende fuertemente del modelo calculado y del árbol obtenido a partir de tal modelo (Antezana, 2003; Felsenstein, 2004).

4.1. Técnicas

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

4.2. Materiales

Matriz de datos (datos.param.dat).

4.3. Métodos

En Modeltest / PAUP*:
(1) Para la matriz calcule el mejor modelo y obtenga el mejor árbol en PAUP*, y sálvelo en formato Phylip con las longitudes de las ramas.
(2) Calcule el soporte de boostrap para el árbol generado en (1); use 100 réplicas.
En Seq-Gen:
(3) Ejecute Seq-Gen con el árbol producto de ML[1] y el modelo calculado; simule 10 secuencias.
En PAUP*:
(4) Analice el archivo de simulaciones obtenido en (3), usando los mismos comandos que uso en (1)[2].
(5) Obtenga el consenso de la mayoría para los árboles generados en (5).

4.3.1. Programas

Modeltest, PAUP*, Seq-Gen.
Seq-Gen esta compilado para todas las plataformas.

4.3.2. Comandos

Revise las prácticas anteriores y la sección de programas (página ??). En PAUP* para convertir de un archivo tipo Phylip a un archivo tipo NEXUS, use la instrucción `tonex`; use primero `tonex ?` para ver las opciones requeridas. Para salvar un árbol en formato Phylip con las longitudes de las ramas, use la instrucción `savet format=phylip brlens=yes`.
Con Seq-Gen para obtener 10 réplicas `-n10` de secuencias de 27 bases `-l27` con JC, use HKY `-mHKY` con los parámetros de Ts/Tv y frecuencias de base que trae por defecto el programa[3], con una distribución Γ con un valor α de 1,6967 `-a1.6967` en formato NEXUS, a partir del árbol salvado como arbol.tre; use la línea de comandos

```
seq-gen -mHKY -n10 -l27 -a1.6967 <arbol.tre >sec.nex
```

Escoja el archivo de árboles que obtuvo con PAUP* y especifique la salida.
Usted puede tener un archivo de instrucciones e insertarlo despues de cada secuencia con la instrucción `-xInstruc.txt`.

4.4. Preguntas

4.4.1. Práctica

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

[1] También puede usar un árbol con constricción, es decir forzando la monofilia de un grupo en particular, si la pregunta está limitada a un grupo particular.
[2] Recuerde que tiene 10 réplicas, por lo que debe incluir los comandos al final de cada matriz, usando un bloque de PAUP*.
[3] En seq-gen no existe el modelo JC, por lo que debe usar HKY con Ts=tv y frecuencias iguales.

usado, ¿obtienen resultados similares de soporte? ¿Variará el soporte al aumentar el número de réplicas?

4.4.2. Generales

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

4.5. Literatura recomendada

Antezana, 2003 [Una visión crítica al uso del boostrap paramétrico].
Felsenstein, 2004 [Tiene un capítulo descriptivo de los análisis de boostrap muy útil].