tcb@footnote@i0 tcb@footnote@ii1 tcb@footnote@iii2 tcb@footnote@iv3

ón

Índice general

PRÁCTICA 4

Búquedas 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 ms 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 agre-

gan 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 bsqueda 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 solucin 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 aleatorizacin 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 mtodo es posible una solucin 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 bsquedas 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, usted lo encuentra disponible para Windows/DOS, 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, Mac[1] y Linux). PAUP* no solo usa parsimonia sino distancias y máxima verosimilitud, aunque para parsimonia es menos versátil que TNT. TNT está diceñado 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.

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[2]), la mayor parte de los programas actuales tienen esta opción: inician con una **semilla** determinada para el generado de número aleatorio y aseguran que la búsqueda sea exactamente igual a otra que tenga el mismo de inicio (semilla del generador de números seudo-aleatorios). Una vez construido un cladograma, este suele ser sometido a permutación de ramas para mejorar su calidad. Básicamente se toma un nodo (subárbol) y es eliminado del cladograma principal, luego se prueba si al unirlo en diferentes lugares del cladograma principal disminuye 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). 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.

[1]debe revisar la compatibilidad con las últimas versiones de Mac OS X, ya que el programa no se ha actulizado en los últimos años

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

- 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

tcb@footnote@cnt@i0

Pregunta

- **¿Qué tipo de informacin puede obtener cuando carga el archivo de datos?**
- **¿Cúal es la búsqueda por omisin en cada programa utilizado?**

2. Búsquedas modificadas

Ejecute el archivo el archivo de datos "datos.chica.datz 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.

tcb@footnote@cnt@i0

Pregunta

- **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).

tcb@footnote@cnt@i0

Pregunta

- **¿Cúal 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 algn 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) Escríbala 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 funcin *write.nexus.data()* escribe un archivo formato NEXUS a partir de un vector de secuencias. Los argumentos de una funcin 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 método de agrupamiento por distancia para obtener la topología inicial, entre ellos los algoritmos UPGMA (=Unweighted Pair Group Method with Arithmetic Mean), WPGMA (=Weighted Pair Group Method with Averaging), NJ (Neighbor Joining) y UNJ (Unweighted Neighbor Joining); en este caso las funciones *upgma()* y *NJ()* permiten construir árboles de distancia bajo sus características especificas, los cuales pueden ser visualizados con la función *plot()* .

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

Otra forma de obtener árboles a partir de secuencias es usando la función *random.addition(* esta 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 construccin de árboles de random.addition?

5. Parsimonia y optimizacin

   A partir de la matriz de dataos primtes.nex y las topologas 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 busqueda por omisión dado que no se especifican los argumentos.

Optimice cada topología obtenida en el punto C, utilizando el método de optimizacin por omisin, optimizacin por SPR y optimizacin por NNI. Registre sus resultados en el Apéndice 2.

6. Optimizacin por omisin

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

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

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

7. Optimizacin con rearreglo 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 rearreglos NNI y SPR.

Preguntas

¿En que difieren los métodos de rearreglos 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 rearreglos 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 rearreglos 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 rearreglos 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 rearreglos 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 rearreglos 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 rearreglos SPR

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

pratchet() hace búsquedas usando el método de Rachet, 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 rachet y óptimizar.

Preguntas02

¿Hay diferencias entre las búsquedas con rachet en términos de costos o tiempo?

Escriba y ejecute la línea de código que le permitiría realizar una búsqueda con:

70 iteraciones en Rachet

Metodo SANKOFF

rearreglo SPR

Sin especificar el árbol inicial

PREGUNTAS GENERALES

De los diferentes programas usados, ¿Cúal estima usted que es el óptimo? Explique las razones de su selección.

¿Cúal 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?, ¿cúal 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?

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