Práctica con el ordenador 3. Filogenia mitocondrial

Preparación del ordenador

En esta práctica utilizaremos los paquetes `DECIPHER` (Wright, 2016) y `phangorn` (Schliep, 2011) de `R`. Asegúrate de haberlos instalado, si no los tienes instalados ya, puedes hacerlo ejecutando los comandos siguientes en la consola de `R`:

> `install.packages('phangorn')`
>
> `install.packages('BiocManager')`
>
> `BiocManager::install('DECIPHER')`

A continuación carga los paquetes.

In [1]:
```
# La función "supressMessages()" nos ahorra información innecesaria.
suppressMessages(library('phangorn'))
suppressMessages(library('DECIPHER'))
```

Objetivo

El objetivo de esta práctica es reproducir la filogenia de los primates catarrinos a partir de las secuencias mitocondriales completas de 13 individuos de 9 especies diferentes, tal y como aparecen en la figura siguiente, procedente de la web PhyloTree: http://www.phylotree.org/resources/mtDNA_human_relatives.htm.

EL procedimiento incluye los pasos siguientes:

1. Descargar las secuencias nucleotídicas de los genomas mitocondriales.
2. Alinearlos con el método implementado en el paquete `DECIPHER`.
3. Obtener la filogenia mediante algún método rápido como el *Neighbor-Joining*.

Ejercicio 1. Descarga de las secuencias

La tabla siguiente indica los **números de acceso** de las secuencias que necesitamos y la especie a la cual pertenece cada una.

Número de acceso	Especie
Y18001	*Papio hamadryas*
AJ309865	*Macaca sylvanus*

Número de acceso	Especie
AY612638	*Macaca mulata*
X99256	*Hylobates lar*
D38115	*Pongo pygmaeus*
D38114	*Gorilla gorilla*
X93347	*Gorilla gorilla*
GU189661	*Pan paniscus*
D38113	*Pan troglodytes*
X93335	*Pan trglodytes verus*
AM948965	*H. sapiens neanderthalensis*
J01415	*Homo sapiens* (Cambridge)
AF347015	*Homo sapiens* (Yoruba)

Puedes descargar las secuencias en formato FASTA de una en una siguiendo los enlaces de la tabla anterior. Es necesario guardarlas todas en el mismo archivo de texto plano, respetando el formato FASTA. Alternativamente se pueden utilizar los comandos siguientes para descargar en la carpeta de trabajo un archivo FASTA con las 13 secuencias:

In [ ]:
```
Numeros <- c('Y18001', 'AJ309865', 'AY612638', 'X99256', 'D38115',
             'D38114', 'X93347', 'GU189661', 'D38113', 'X93335',
             'AM948965', 'J01415', 'AF347015')
API     <- 'https://www.ebi.ac.uk/ena/browser/api/fasta/'
URL     <- paste0(API, paste(Numeros, collapse = ','))
URL
download.file(URL, destfile = 'primates.fasta')
```

Ejecuta el código anterior y comprueba que en la carpeta de trabajo se ha descargado un archivo llamado `primates.fasta`. Puedes abrir el archivo para inspeccionarlo, pero no es recomendable editar manualmente los archivos de datos.

Ejercicio 2. Alineamiento

Para alinear las 13 secuencias necesitamos haber cargado el paquete `DECIPHER`, leer las secuencias y guardarlas en un objeto dentro de la sesión de `R`, y aplicar la función `AlignSeqs()` para crear un nuevo objeto con las secuencias alineadas. Comenzamos por leer las secuencias y guardarlas en un *objeto* o variable de `R`:

In [ ]:
```
sinAlinear <- readDNAStringSet('primates.fasta')
sinAlinear
```

Ahora, el objeto `sinAlinear` contiene 13 secuencias mitocondriales. Observa como al invocar el nombre del objeto nos aparece un resumen de su contenido. Si ejecutas `names(sinAlinear)` en un bloque de código, observarás los nombres de las

secuencias y verás que son innecesariamente largos. Antes de alinear, es conveniente acortar los nombres, cosa que se puede hacer reasignando el valor de los nombres respetando el orden en el cual están las secuencias:

In [ ]:
```
names(sinAlinear)
```

In [7]:
```
names(sinAlinear) <- c('Papio hamadryas', 'Macaca sylvanus', 'Macaca mulatta',
           'Hylobates lar', 'Pongo pygmaeus', 'Gorilla gorilla',
           'Gorilla gorilla gorilla', 'Pan paniscus', 'Pan troglodytes',
           'Pan troglodytes verus', 'H. sapiens neanderthalensis',
           'H. sapiens (Cambridge)', 'H. sapiens (Yoruba)')
```

In [ ]:
```
names(sinAlinear)
```

Una vez comprobado que los nombres de las secuencias son los adecuados procedemos al alineamiento:

In [ ]:
```
alineadas <- AlignSeqs(sinAlinear, verbose = FALSE)
alineadas
```

Observa que la función `AlignSeqs()` **no** modifica el objeto original (`sinAlinear`), sino que crea otro. Por eso necesitamos *assignar* (`<-`) el resulado de la función a un nuevo objeto (`alineadas`) para retener en la memoria del trabajo el resultado del alineamiento.

Observa también que el nuevo objeto `alineadas` tiene un aspecto diferente al de `sinAlinear` . Para ver el alineamiento completo ejecuta el comando siguiente, con el cual se crea y abre el archivo `alineadas.html` en una pestaña nueva de tu navegador.

In [10]:
```
BrowseSeqs(alineadas, htmlFile = 'alineadas.html', openURL = TRUE)
```

2.1 ¿Por qué crees que algunas secuencias parecen tener fragmentos adicionales al principio o al final del alineamiento?

2.2 Con la función `readDNAStringSet()` carga el alineamiento contenido en el archivo `primates2.fasta` en un nuevo objeto, explóralo con la función `BrowseSeqs()` y determina en qué se diferencia del alineamiento que has creado tú previamente.

In [11]:
```
# Introduce aquí los comandos que necesites, basándote en los
# bloques anteriores.
alineadas2 <- readDNAStringSet('primates2.fasta')
BrowseSeqs(alineadas2, htmlFile = 'alineadas2.html', openURL = TRUE)
```

2.3 Decide cuál de los dos alineamientos, el creado por ti o el que ha cargado a partir del archivo `primates2.fasta` ,

es el más adecuado para utilizarlo en el paso siguiente.

Ejercicio 3. Reconstrucción filogenética

Una vez disponemos de un alineamiento, podemos inferir las relaciones filogenéticas entre las 13 secuencias. Hay muchos métodos diferentes: máxima parsimonia, máxima similitud, UPGMA, etc. Utilizaremos el método de Neighbor-Joining, basado en la matriz de distancias genéticas entre las secuencias, porque es un método rápido.

Distancias

El primer paso es obtener la **matriz de distancias**, lo cual podemos hacer con la función `dist.dna()` (del paquete `ape`, cargado automáticamente junto a `phangorn`). Ahora bien, el alineamiento de partida es un objeto de clase `DNAStringSet`, mientras que la función `dist.dna()` requiere que el alineamiento sea un objeto de clase `DNAbin`. Por tanto, se debe utilizar la función `as.DNAbin()` para transformar el objeto. Edita el bloque de código siguiente para poder utilizar el alineamiento elegido en el ejercicio 2.3.

In [ ]:
```
# Sustituye "alineamiento" por el nombre del objeto donde guardas
# el alineamiento que has decidido utilizar en el ejercicio 2.3.

aln <- as.DNAbin(alineamiento)
aln
```

Las distancias son una estimación del número medio de sustituciones nucleotídicas acumuladas entre dos secuencias desde el momento que divergieron por cada posición de su alineamiento. Una distancia genética de 1 implicaría que *en término medio* todas las posiciones han sufrido una mutación. Como es una media entre todas las posiciones del alineamiento, generalmente es posible reconocer la homología de dos secuencias y alinearlas incluso con ditancias mayores de 1: en algunas posiciones se han producido más de una mutación a lo largo de los dos linajes, mientras que en muchas otras posiciones no se ha producido ninguna.

Para poder estimar estas distancias es necesario adoptar un **modelo de evolución molecular**, el cual nos permite suponer cuantas sustituciones más de las observadas deben de haberse producido, a partir del número de cambios observados. En el bloque siguiente, sugerimos el modelo *K81* (Kimura, 1981). Para ver qué otras opciones tienes, consulta la ayuda de la función `dist.dna()` ejecutando la orden `help(dist.dna)` en un bloque de código.

In [ ]:
```
distancias.K81 <- dist.dna(aln, model = 'K81')
as.matrix(round(distancias.K81, 3))
```

Neighbor-Joining

El método de *neighbor-joining* produce un árbol no *ultramétrico* y no *enraizado* a partir de las distancias. Podemos aplicar este método a un objeto de clase *dist* con la función

`NJ()` del paquete `phangorn`:

In [ ]:
```
NJ.K81 <- NJ(distancias.K81)
plot(NJ.K81)
```

Podemos *enraizar* el árbol con la función `root()`, porque sabemos que el *clado* formado por *Papio hamadryas*, *Macaca mulatta* y *Macaca sylvanus* es un **outgroup** de todos los demás. Es decir, pertenecen a un linaje que divergió antes que los demás linajes que divergen entre ellos.

In [ ]:
```
NJ.K81.enraizado <- root(NJ.K81,
      outgroup = c('Papio hamadryas', 'Macaca mulatta', 'Macaca sylvanus'))
plot(NJ.K81.enraizado)
```

Bootstrap

Si te queda tiempo, ejecuta los bloques siguientes para añadir a las ramas del árbol su soporte de *bootstrap*. El soporte de *bootstrap* es una medida (entre 0 y 100) del grado de soporte que los datos (el alineamiento) otorgan a cada rama.

In [ ]:
```
# Sustituye "alineamiento" por el nombre del objeto donde guardas
# el alineamiento que has decidido utilizar en el ejercicio 2.3.

funcion <- function(x) NJ(dist.dna(as.DNAbin(x), model = 'K81'))
bootstrap <- bootstrap.phyDat(as.phyDat(as.DNAbin(alineamiento)), funcion, bs =
plotBS(NJ.K81, bootstrap)
```

Bibliografía

- Kimura, M. (1981) Estimation of evolutionary distances between homologous nucleotide sequences. *Proceedings of the National Academy of Sciences USA*, 78, 454–458.
- Schliep, K.P. (2011) *phangorn*: phylogenetic analysis in R. *Bioinformatics* 27 (4): 592-593. 10.1093/bioinformatics/btq706
- Wright, E.S. (2016) Using DECIPHER v2.0 to analyze big biological sequence data in R. *The R journal* 8 (1): 352-359. 10.32614/RJ-2016-025