# PRÁCTICA 3

**(1) Busque en NCBI la secuencia del ADN (en formato nucleótidos) del *Varicella-zoster virus*** (código RefSeq NC_001348.1). **Descargue esta secuencia en un fichero FASTA** (en el directorio "Mis Documentos") **Usando** read.fasta() **del paquete** seqinr **descargue su contenido en una variable llamada** list.dna **(una lista) y la secuencia de ADN en forma de vector de caracteres (nucleótidos) en una variable llamada** vector.dna

**(1.1) Acceso a vectores y listas usando el operador [ ].** Consulte las transparencias y ficheros R del tema 3 si tiene dudas sobre cómo usar los corchetes.

(1.1.1) Usando solo operador [ ] cambie las timinas ("t") por uracilos ("u") en vector.dna, escribiendo el resultado en la variable vector.rna, sin cambiar la variable vector.dna. Ayuda: donde haya una "t" ponga una "u".

(1.1.2) Transforme el vector vector.rna en una lista llamada list.rna (compruebe que ambos tienen la misma longitud).

(1.1.3) Un elemento de una lista se extrae con [[ ]] y una sublista se extrae con [ ]. Extraiga una sublista de 20 nucleótidos de la lista list.rna a partir de la posición 100*<código> y escríbala en la variable sublist.rna (compruebe con typeof() que es una lista y con length() que tiene 20 elementos).

(1.1.4) Convierta la sublista sublist.rna en un vector de 20 nucleótidos en la variable subvector.rna (compruebe con typeof() que es un vector).

(1.1.5) Convierta los caracteres de subvector.rna (que están en minúsculas) en caracteres en mayúsculas usando la función toupper() . Use la ayuda para saber cómo funciona este comando.

(1.1.6) Convierta el vector resultado de (1.1.5) en una cadena de caracteres llamada subvector.RNA usando el comando c2s().

**Respuestas: Todos los comandos usados y el resultado de** subvector.RNA (que es una cadena de caracteres aunque se llame subvector)

**(1.2) Uso de factores y table()**. Extraiga un vector de 123 nucleótidos de vector.rna partir de la posición 123*<código> y escríbala en la variable mi.vector.rna (compruebe con length() que tiene 123 elementos).

(1.2.1) Convierta dicho vector en un factor llamado mi.factor.rna (use el comando factor y la ayuda para saber qué hace y cómo funciona)

(1.2.2) Use el comando table() para calcular el porcentaje de uracilos en mi.factor.rna y el porcentaje gc en mi.vector.rna. Use la ayuda para aprender qué hace y cómo funciona este comando.

(1.2.3) Usando el cambio de etiquetas (labels) en un factor (consulte las transparencias si tiene dudas), cambie los uracilos por timinas en mi.factor.rna dando lugar a la variable mi.factor.dna. Usando table() calcule el porcentaje de timinas en mi.factor.dna.

(1.2.4) Traduzca a aminoácidos con translate() la variable mi.vector.rna y convierta el resultado en un factor llamadomi.factor.aaa que tenga como niveles los 20 aminoácidos.

(1.2.5) Use el comando table() para determinar si falta algún aminoácido en mi.factor.aaa . ¿Cuáles faltan?

(1.2.6) Cambien las etiquetas (labels) para que todos los aminoácidos de mi.factor.aaa estén escritos en letra minúscula; convierta el resultado en una cadena de caracteres llamada string.aaa .

(1.2.7) Busque en la web (vía internet) qué aminoácidos son hidrófugos (a veces se escribe *hidrofóbicos*). Calcule el porcentaje de dichos aminoácidos en mi.factor.aaa .

**(1.3) Manejo de funciones básicas**. Descarga el genoma anotado del *Varicella-zoster virus* en un fichero fasta y léelo en R. Aplica el comando length() y calcula un vector con las longitudes de sus genes. Calcule el valor mínimo, el valor máximo, la media y la mediana de dicho vector con los comandos **min(), max(), mean()**, y **median().**

(1.3.1) Usando los comandos **which.max(), which.min()** determine la posición del mínimo y del máximo. ¿Están repetidos estos valores? ¿Cuántas veces? Escriba comandos en R que respondan a dichas cuestiones.

(1.3.2) Ordene dichos números en orden ascendente con **sort()** en la variable ascendente y en orden descendente en la variable descendente (use la ayuda si necesita saber cómo hacerlo).

(1.3.3) Use el comando **unique()** para obtener un vector sin elementos repetidos. ¿Cuántos elementos tiene?

**Respuestas: Comandos usados y sus resultados.**

**(2) Manejo del comando which() y los vectores lógicos con [ ]. Buscaremos el codón "gau" en** mi.vector.rna **(del apartado 1.2). Lea con cuidado las instrucciones que se le dan (es más fácil de lo que parece).** Vamos a usar la función rot = function (v) c(v[-1],v[1]).
Ejecute rot(1:5) y rot(rot(1:5)) y rot(rot(rot(1:5)))explique qué es lo que hace la función rot().

(2.1) Encuentre las posiciones donde se encuentre una "g" en el vector mi.vector.rna usando el comando which() y guarde el resultado en el vector mi.pos.g.

(2.2) Encuentre las posiciones donde se encuentre una "a" en el vector rot(mi.vector.rna) usando el comando which() y guarde el resultado en el vector mi.pos.a.

(2.3) Encuentre las posiciones donde se encuentre una "u" en el vector rot(rot(mi.vector.rna)) usando el comando which() y guarde el resultado en el vector mi.pos.u. ¿Cómo usaría los tres comandos/vectores anteriores para encontrar las posiciones del codón "gau" en el vector mi.vector.rna ? Hágalo e indique en qué posiciones se encuentra dicho codón.

**(2.4) REPITA LO MISMO** para el codón de la metionina (búsquelo en wikipedia) en el vector.dna en lugar de mi.vector.rna

**Respuestas: Comandos usados y posiciones donde se encuentra dicho codón ("M") en vector.dna.**

**(3) Busque en NCBI la secuencia del ADN (en formato nucleótidos) del virus Heterosigma akashiwo RNA virus (HaRNAV), código NC_005281.** Puedes usar el comando where.is.this.acc("NC_005281") de seqinr para ver en qué bases de datos está este virus de ARN. **Descarga la secuencia en un fichero FASTA** (en el directorio "Mis Documentos") Usando read.fasta(). Guarda a secuencia de ADN en forma de vector de caracteres (nucleótidos) en una variable llamada vector.dna . Usando factor() cambia el formato de ADN a ARN y guárdalo en una variable vector.rna. **Respuesta: Comandos usados.**

**(3.1) Manejo de matrices.** Escriba un comando que construya una matriz matrix.rna de tres columnas rellenada **por filas** con el contenido de vector.rna (consulte las transparencias si tiene dudas). **Respuesta: Comandos usados. Extrae en forma de vector la fila cuyo ordinal es <código> * 10. Extrae en forma de matriz de tres columanas las filas con número ordinal <código> * (40:43). Escribe un comando que calcule cuántas adeninas hay en esta (sub)matriz.**

**(3.2) Más sobre manejo de matrices.**Escriba un comando que construya una matriz rna.matrix de tres filas rellenada por columnas con el contenido de vector.rna (consulte las transparencias si tiene dudas). **Respuesta: Comandos usados. Extrae en forma de vector la columna<código> * 11. Extrae**

**en forma de matriz de tres filas las columnas con número ordinal <código> * (33:37). Escribe un comando que calcule cuántas adeninas hay en esta (sub)matriz.**

**(4) Manejo elemental del comando apply(). No hemos visto este comando aún, pero es muy poderoso y merece la pena conocerlo.**

**(4.1) El comando apply().** Use la ayuda de R y/o busque en Google cómo funciona el comando apply() sobre matrices. **Aplique por filas** a la matriz matrix.rna la función c2s() (como resultado obtendrá un vector de codones en forma de cadenas de tres caracteres). Busque con which la cadena "gau" en dicho vector. Obtendrá muchas posiciones del codón, ¿cuál es el que está en la posición <código> en el resultado del which? Haga lo mismo con el vector.dna y la matrix matrix.dna para buscar todos los codones de las metioninas (páselo a una matriz, aplique c2s con apply y busque con which su posición). **Respuesta: Comandos usados y posición en la que se encuentra el <código>-ésimo codón "M" en vector.dna.**

**(4.2) Más apply().Aplique por columnas** a la matriz rna.matrix la función c2s() (como resultado obtendrá un vector de codones en forma de cadenas de tres caracteres). Busque con which la cadena "gau" en dicho vector. Obtendrá muchas posiciones del codón, ¿cuál es el que está en la posición <código> en el resultado del which? Haga lo mismo con el vector.dna y la matrix dna.matrix para buscar todos los codones de las metioninas (páselo a una matriz, aplique c2s con apply y busque con which su posición). **Respuesta: Comandos usados y posición en la que se encuentra el <código>-ésimo codón "M" en vector.dna.**

**(5) Divida su <código> por 10 y determine el <resto> de la división (un número entre 0 y 9). Busque en NCBI el genoma anotado del ebolavirus zaire con código genbank KM23309<resto>.1 ( que será un código entre KM233090.1, KM233091.1, KM233092.1, ... KM233099.1 )**. Guarde la secuencia del genoma anotado en un fichero **FASTA de aminoácidos** y léala en R usando el comando read.fasta() en una variable llamada virus.

**(5.1) La variable virus contiene una lista con el genoma**. El último gen será ultimo=virus[[length(virus)]]. Usa attr(ultimo,"Annot") para ver el nombre del gen y cuál es la proteína que codifica. Usa el comando attr( ,"name"). Mira location para saber en qué nucleótidos está codificada esta proteína. **Respuesta: Selecciona el gen número tres. ¿Cuántos aminoácidos tiene? Mira su anotación (atributo "Annot"). ¿Cómo se llama? ¿Qué proteína codifica? ¿Qué nucleótidos lo codifican? ¿Cuántos nucleótidos lo codifican?**

**(5.2) Calcula un vector numérico con el número de aminoácidos** de todos los genes del virus. Calcula un vector de cadenas de caracteres con el nombre de todas las proteínas. Calcula una lista de cadenas de caracteres con el nombre de todas las proteínas. **Respuesta: Comandos usados.**

**(5.3) Busca en la wikipedia cuáles son los aminoácidos** esenciales y no esenciales. Usando el comando which() determina cuántos aminoácidos esenciales y no esenciales tienen todos los genes del virus. Busca en la wikipedia cuáles son los aminoácidos no polares (o hidrófugos) y los aminoácidos polares no cargados. Usando el comando which() determina cuántos aminoácidos no polares y polares tienen todos los genes del virus. Determina una matriz (usando el comando matrix) con cuatro columnas y tantas filas como genes tiene el virus con los resultados que has obtenido. Sus columnas se llamarán con colnames() con los nombres c("esenc","noese","nopol","polar"). Ponle a las filas de la matriz el nombre de las proteínas usando rownames(res)=codigos. **Respuesta: Comandos usados y respuestas obtenidas.**

**(5.4) ¿Cómo usaría el comando write.table() para grabar en un fichero llamado "mivirus.dat" en el directorio "Mis Documentos"? ¿Cómo usaría el comando read.table() para leer dicho fichero y copiar los datos en una variable llamada datos? Consulte la ayuda si es necesario para saber cómo funcionan estos comandos.**

# PRÁCTICA 4

**(1) Busque en NCBI la secuencia del genoma completo del Mycoplasma pneumoniae (NCBI Reference Sequence: NC_000912, que tiene 816394 bp).** Descargue esta secuencia de nucleótidos en un fichero FASTA (en el directorio "Mis Documentos") y guarde su secuencia de nucleótidos en una variable tipo vector de caracteres llamada dna.

**(1.1) Uso del comando for/in para buscar un nucleótido.** Consulte las transparencias del tema 3 para estudiar/entender cómo funcionan el comando if (Condicion) {EjecutaComandos} y el comando for ( Algo in Vector) {EjecutaComandosConAlgo}. Escriba en la variable mi.dna los 100 nucleótidos de dna que se encuentran a partir de la posición <código>*123 (compruebe con length que tiene 100 nucleótidos). Ejecute el comando donde.g.which = which(mi.dna=="g"). El comando which se puede simular usando un for y un if de la siguiente forma:

```
donde.g = numeric(0)

for ( n in 1:100 )

{  if ( mi.dna[n] == "g" )

  {  donde.g = c(donde.g,n) }}
```

Comprueba que donde.g y donde.g.which son iguales. A partir de este código, escriba un nuevo algoritmo que usando un único for y cuatro comandos if , escriba un algoritmo que calcule las posiciones dónde se encuentran los cuatro nucleótidos (a, c, g, t) en mi.dna y los introduzca en las variables donde.a, donde.c, donde.g, y donde.t. Comprueba usando which que su código funciona correctamente para los cuatro nucleótidos. **Respuesta: Comandos de R usados y el resultado de la verificación de su resultado (no es necesario el resultado como tal).**

**(1.2) Uso del comando for/in para buscar un codón.** Escriba en la variable otro.dna los 932 nucleótidos de dna que se encuentran a partir de la posición <código>*1230 (compruebe con length() que tiene 932 nucleótidos). Ejecute el comando t.otro.dna = translate(otro.dna); donde.M.which = which(t.otro.dna=="M"). Este comando which() se puede simular usando un for y un if de la siguiente forma:

```
donde.M = numeric(0)

for ( nnn in seq(1,932,3) )

{ if ( all ( otro.dna[nnn+(0:2)] == c("a","t","g") ) )

  {   donde.M = c(donde.M,nnn)} }
```

¿Cómo puede comprobar que donde.M y donde.M.which han dado el mismo resultado? ¿Para qué sirve el comando tablecode()? Usando este comando mire los codones que codifican el aminoácido "Lys" y escriba un código usando un único for y tantos if como sean necesarios, para encontrar la posición del primer nucleótido de dichos aminoácidos en otro.dna. Compare su resultado con el obtenido usando translate y which. ¿Funciona bien su código? Arregle el código si fuera necesario. **Respuesta: Comandos de R usados y resultado de la búsqueda de Lys.**

**(1.3) Uso del comando for/in para buscar un codón.** Escriba un código en R para encontrar el aminoácido "Leu" en la variable otro.dna (como en el ejercicio (1.2) use un único for y tantos if como sean necesarios). Compare el resultado con el obtenido con translate y which. ¿Funciona bien su código? **Respuesta: Comandos de R usados y resultado de la búsqueda de Leu.**

**(2.1) Uso del comando while para buscar el codón de la metionina en cualquier marco abierto de lectura (ORF).** Escriba en la variable adn los 3000 nucleótidos de dna que se encuentran a partir de la posición <código>*3210 (compruebe con length que tiene 3000 nucleótidos). Podemos buscar el codón de la metionina con los 3 marcos abiertos de lectura +1, +2, y +3 usando los comandos for e if de la siguiente forma:

```
donde.Met = numeric(0)

for ( donde in 1:length(adn) )

{ if ( all ( adn[ donde+(0:2) ] == c("a","t","g") ) )

   { donde.Met = c(donde.Met,donde) }}
```

Compara donde.Met con el resultado de 3*which("M"==translate(adn)), ¿qué valores son iguales? Mira en la wikipedia el concepto de "Marco abierto de lectura". ¿A qué marco abierto de lectura corresponden los valores que son iguales? ¿Qué relación hay entre el resultado de donde.Met%%3 y el marco de lectura? Escribe una función ORF.codon = function(posiciones) { **<comandos>** }que a partir de un vector de posiciones del primer nucléotido de una serie de codones devuelva el marco de lectura +1, +2, y +3 asociado a cada uno de ellos. Aplica dicha función a donde.Met. **Respuesta: Comandos de R usados, función ORF.codon y respuesta a las preguntas.**

**(2.2) Uso de los comandos for/if para buscar los codones de parada "Stp".** Mire usando tablecode() el código genético de los codones de parada (Stop o "Stp"). Escriba un código R usando for y tantos if como sean necesarios, similar al del apartado (2.1), que determine la posición de los codones de parada en la variable adn y los introduzca en la variable donde.Stp. Aplique la función ORF.codon y determine a qué marco de lectura pertenecen. **Respuesta: Comandos de R usados y respuesta a las preguntas.**

**(3.1) Uso de los comandos for/if para determinar la secuencia complementaria.** Escriba una función llamada mi.comp que use un for para calcular la secuencia complementaria de una secuencia de ADN de entrada. Aplique dicho código a adn[1:10] y compruebe que el resultado es correcto con comp(adn[1:10]). ¿Cuántos comandos if ha necesitado? Ejecute adn.comp = mi.comp(adn). Busque el codón de la metionina en dicha secuencia complementaria usando el siguiente código:

```
donde.M.comp = numeric(0)

for ( pos in length(adn):3 )

{  if ( all ( adn.comp[pos-(0:2)] == c("a","t","g") ) )

   {  donde.M.comp = c(donde.M.comp,pos)}  }
```

Compara donde.Met.comp con el resultado de translate aplicado de forma adecuada a la secuencia adn.comp. ¿qué valores son iguales? ¿A qué marco abierto de lectura corresponden los valores que son iguales? ¿Qué relación hay entre el resultado de donde.Met%%3 y el marco de lectura? Escribe una función ORF.codon.comp = function(posiciones) que a partir de un vector de posiciones del primer nucléotido de una serie de codones en la cadena complementaria (sin invertir) devuelva el marco de lectura -1, -2, y -3 asociado a cada uno de ellos. Aplica dicha función a donde.Met.comp. **Respuesta: Comandos de R usados, función ORF.codon.comp y respuesta a las preguntas.**

**(3.2) El comando while() permite simular un comando for(). En concreto, for (algo in inicio:final) { <comandos> } se puede escribir como algo=inicio; while (algo<=final) { <comandos> ; algo=algo+1}. Use el comando while() para buscar los codones de parada "Stp" en la secuencia complementaria (sin invertir). Lea lo siguiente atentamente antes de hacer nada.** Escriba un código R usando while() y tantos if como sean necesarios, similar al del apartado (3.1), pero que en lugar de usar for use while, es decir, similar al apartado (2.1), que determine la posición de los codones de parada en la variable adn.comp y los introduzca en la variable donde.Stp.comp. Aplique la función ORF.codon.comp y determine a qué marco de lectura pertenecen. **Respuesta: Comandos de R usados y respuesta a las preguntas.**

# PRÁCTICA 5

**(1) Búsqueda del ADN completo de la mitocondria humana (Complete record).**

**(1.1)** Busca el ADN completo (nucleótidos) de la **Mitocondria humana** (Homo sapiens mitochondrion). Descarga la secuencia en la variable **mitoseq**. ¿Cuántos nucleótidos tiene? Verifica que son 16 569 bp en un ADN circular. ¿Qué código GenBank tiene? Verifique que es NC_012920.1 . Usando el comando write.fasta() grabe la secuencia de adn en un fichero FASTA llamado **mito.fasta** . Lea el fichero FASTA en R usando **read.fasta()** en una variable **mitoseq.prueba** y compruebe con un comando que es idéntica a **mitoseq**. **Respuesta: Escribe los comandos que has introducido en R (solo el comando). ¿Cuántos nucleótidos tiene el genoma?**

**(1.2)** El código genético de la mitocondria de los vertebrados difiere del código genético estándar. Seleccione 237 nucleótidos de la mitocondria a partir de la posición <código>*237 y escríbalos en la variable mi.mitoseq. Compare la traducción de mi.mitoseq usando el código genético estándar translate(mi.mitoseq,numcode=1) y usando el código genético de la mitocondria translate(mi.mitoseq,numcode=2). Escriba un bucle while() que recorra ambas traducciones y que busque las diferencias, obteniendo una lista de listas con la posición de los "aminoácidos" que son diferentes , así como sus valores; por ejemplo, si la segunda diferencia está en la posición 142 con una traducción "M" estándar y una "I" mitocondrial, la lista contendrá lista[[2]]=list(pos=142,codStd="M",codMit="I") . **Respuesta: Escribe los comandos que has introducido en R (solo los comandos). ¿Cuántas diferencias hay entre ambas traducciones? ¿Cuál es la lista con las diferencias?**

**(1.3)** Compare a vista el resultado de los comandos tablecode(2) y tablecode(1). ¿Cuáles son las diferencias que observa a vista? Escriba un bucle while() que recorra los 64 codones words(3) y busque las diferencias entre la traducción con translate() usando ambos códigos genéticos. El resultado será una lista de listas que contendrá las diferencias en forma de codón (vector de tres caracteres), traducción estándar y traduccción mitocondrial en un formato similar al apartado (1.2), es decir, tipo lista[[1]]=list(codon=c("a","t","a"),traStd="I",traMit="M"). **Respuesta: Escribe los comandos que has introducido en R (solo los comandos). ¿Cuántas diferencias hay entre ambos códigos genéticos? ¿Cuál es la lista con las diferencias?**

**(1.4) Gráficos de sectores (pie) en PNG.** Ejecuta los comandos pie(**table(mitoseq));pie(table(mi.mitoseq))**. ¿Qué diferencia hay entre ambos? ¿Por qué aparece una "n" como nucleótido en el primero? ¿Cómo eliminas la "n"? Dibuja un diagrama de tipo **pie()** de mitoseq sólo con los nucleótidos (a,c,g,t), sin que aparezca la "n". El comando png("fig1.png"); pie(table(dna)); dev.off() te permite grabar la figura en un fichero de tipo PNG con nombre fig1.png. **Respuesta: Código en R y figura insertada en tu fichero RTF (en WordPad usa el Menú "Imagen" y selecciona el fichero PNG que has obtenido).**

**(1.5) Gráficos de sectores (pie) en JPG.** Dibuja un diagrama de tipo pastel con la proporción de parejas de nucleótidos usando el comando pie (count(mitoseq,2)). El comando jpeg("fig2.jpg"); pie(count(mitoseq,2)); dev.off() copia la figura en un fichero de tipo JPG con nombre fig2.jpg. **Respuesta: Código en R y figura insertada en tu fichero RTF (en WordPad usa el Menú "Imagen" y selecciona el fichero JPG que has obtenido).**

**(1.6) Gráficos de sectores (pie) en EPS.** Dibuja un diagrama de tipo **pie()** con el resultado del comando count(mitoseq,3), es decir, con la proporción de codones de nucleótidos. Usando postscript("fig3.eps"); pie(count(mitoseq,3)); dev.off() se copia la figura en un fichero de tipo EPS con nombre fig3.eps. **Respuesta: Código en R. WordPad no permite insertar EPS (aunque Microsoft Word lo permite). ¿Qué aplicación has usado para ver el contenido del fichero fig3.eps? Usa la lupa para ver la calidad del texto. ¿Qué tamaño en disco tienen los ficheros fig1.png, fig2.jpg y fig3.eps? ¿Qué diferencia de calidad hay entre los ficheros, si es que hay alguna? ¿Cuál te parece la más adecuada para un trabajo fin de grado?**

**(2) Búsqueda del genoma de la mitocondria humana y neandertal (Coding sequences).**

**(2.1)** Busca el genoma anotado en formato nucleótidos de la **Mitocondria humana** (Homo sapiens mitochondrion). Descargue todos los genes en la variable **mitogen** de tipo lista de secuencias. ¿Cuántos genes tiene? Usando un bucle while() calcule el contenido GC() de cada gen en un vector llamada **mitoGC**. Dibuje un diagrama de tipo **plot(mitoGC)** con el resultado usando puntos. Use **lines(mitoGC,col="red")** para unir los puntos con una línea roja. Ordene el resultado con sort() y use **points()** de color azul para superponer el resultado en el plot (busque la ayuda de R o consulte las transparencias sobre gráficos para saber cómo funciona este comando). Use otro **lines()** para superponer con línea celeste (cyan) una línea que una dichos puntos. Usando png("fig4.png"), repita los comandos anteriores y acabe con un dev.off() para escribir la figura en un fichero de tipo PNG. **Respuesta: Código en R de todo lo ejecutado y figura insertada en tu fichero RTF.**

**(2.2)** Busca el genoma anotado en formato nucleótidos de la **Mitocondria neandertal** (Homo sapiens neanderthalensis). Descargue todos los genes en la variable **mitonea** de tipo lista de secuencias. ¿Cuántos genes tiene? Usando un bucle while() calcule el contenido GC() de cada gen en un vector llamada **mitoneaGC**. Dibuje un diagrama de tipo **plot(mitoneaGC)** con el resultado usando puntos. Use **lines(mitoneaGC,col="red")** para unir los puntos con una línea roja. Ordene el resultado con sort() y use **points()** de color azul para superponer el resultado en el plot (busque la ayuda de R o consulte las transparencias sobre gráficos para saber cómo funciona este comando). Use otro **lines()** para superponer con línea celeste (cyan) una línea que una dichos puntos. Usando png("fig5.png"), repita los comandos anteriores y acabe con un dev.off() para escribir la figura en un fichero de tipo PNG. **Respuesta: Código en R de todo lo ejecutado y figura insertada en tu fichero RTF.**

**(2.3)** Compare los resultados de los ejercicios (2.1) y (2.2). ¿Qué diferencias encuentra en los genomas mitocondriales del hombre moderno y del hombre de neandertal?

**(3.1) Matrices y comando for ().** Escriba un bucle for() que calcule una matriz **neannn** con cuatro columnas, con nombres a, c, g y t, y tantas filas como genes tiene el genoma de la mitocondria neandertal. En cada fila de la matriz aparecerá el número de nucleótidos a, c, g y t, por columnas, de cada gen. Escriba otro bucle for() que calcule un vector de caracteres **neannnmax** con el nucleótido más abundante para cada gen; para ello recorra las filas de la matriz **neannn** y use el comando max(). Luego escriba un comando apply() que aplicado a **neannn** de el mismo reesultado que **neannnmax**; ¿cómo comprueba que el resultado es idéntico? **Respuesta: Todos los códigos en R que ha ejecutado.**

**(3.2) Matrices y comando while().** Escriba un bucle while() que calcule una matriz **humnnn** con cuatro columnas, con nombres a, c, g y t, y tantas filas como genes tiene el genoma de la mitocondria humana. En cada fila de la matriz aparecerá el número de nucleótidos a, c, g y t, por columnas, de cada gen. Escriba otro bucle while() que calcule un vector de caracteres **humnnnmax** con el nucleótido más abundante para cada gen; para ello recorra las filas de la matriz **humnnn**. **Respuesta: Todos los códigos en R que ha ejecutado. ¿Compare el resultado de los ejercicios (3.1) y (3.2)? ¿Qué conclusiones extrae de dicha comparación?**

# PRÁCTICA 6

**(1) Descarga en el directorio Mis Documentos un fichero FASTA con la secuencia de nucléotidos cuyo código en GenBank está dado por paste0("MG547",181+codigo)), donde codigo es el código del estudiante.** Ejecutael comando library("seqinr"). Si no funciona, instala el paquete seqinr usando el Menú "Paquetes" >> "Instalar paquete(s)..." Lee el fichero FASTA y guarda el contenido en una variable llamada old.dna.

**(1.1) ¿A qué organismo pertenece la secuencia old.dna con dicho código? ¿Cuántos nucleótidos tiene?** Usando la función ModeloMultinomial() vista en clase determine el modelo multinomial de la secuencia old.dna. Dibuje un diagrama con pie() que muestre dicho modelo multinomial. Grabe dicha figura en una fichero tipo PNG. **Respuesta: Modelo multinomial obtenido. Figura PNG obtenida insertada en su fichero de respuesta. ¿Se parece más a una secuencia rica en AT o a una secuencia rica en GC? Razone su respuesta.**

**(1.2) Para generar una secuencia de adn siguiendo un modelo multinomial concreto podemos usar el comando** sample(words(1),size,T,prob=c(pa,pc,pg,pt)) . Genere una secuencia aleatoria de ADN llamada new.dna con la misma longitud que la secuencia old.dna usando el mismo modelo modelo multinomial que la describe. Dibuje una figura con dos diagramas de sectores tipo pie() que muestre el modelo multinomial de las secuencias old.dna y new.dna, uno al lado del otro (grábela en una fichero tipo PNG). **Respuesta: Nuevo modelo multinomial obtenido. Inserte la figura PNG obtenida en su fichero de respuesta. ¿Cuáles son las diferencias entre modelos multinomiales? ¿Por qué se dan dichas diferencias? Razone su respuesta.**

**(1.3) Escriba una función en R llamada secuenciaModeloMultinomial = function (mm=c(0.25,0.25,0.25,0.25), cuantos=100, nn=c("a","c","g","t"))**, que devuelva una secuencia de **cuantos** nucleótidos (con las letras dadas por el parámetro de entrada **nn**) generados por el modelo multinomial dado por el parámetro de entrada **mm** (un vector de cuatro números). Para comprobar que funciona ejecute los siguientes comandos

secuenciaModeloMultinomial(mm=c(0.90,0.1),cuantos=100,nn=c("c","g"))

secuenciaModeloMultinomial(mm=c(0.10,0.80,0.05,0.05),100)

**Respuesta: Código en R para la función que has desarrollado. Escribe un código que usa dicha función para calcular las variables ricaAT2 y ricaGC2 con 271 nucleótidos ricas en AT y en GC, resp. ¿Qué modelo multinomial ha usado?**

**(2.1) Traduzca con translate() la secuencia old.dna en una variable old.aaa .** Usando la función ModeloMultinomial() vista en clase determine el modelo multinomial de la secuencia old.aaa. Dibuje un diagrama con pie() que muestre dicho modelo multinomial. Grabe dicha figura en una fichero tipo PNG. **Respuesta: Modelo multinomial obtenido. Figura PNG obtenida insertada en su fichero de respuesta.**

**(2.2) Usando la función secuenciaModeloMultinomial () genere una secuencia de aminoácidos new.aaa con el modelo multinomial de old.aaa.** Dibuje una figura con dos diagramas de sectores tipo pie() que muestre el modelo multinomial de las secuencias old.aaa y new.aaa, uno al lado del otro (grábela en una fichero tipo PNG). **Respuesta: Nuevo modelo multinomial obtenido. Inserte la figura PNG obtenida en su fichero de respuesta. ¿Cuáles son las diferencias entre modelos multinomiales? ¿Por qué se dan dichas diferencias? Razone su respuesta.**

**(3.1) Busca en la base de datos Nucleotide de GenBankd (NCBI) el genoma completo de los entenovirus D68 ("Enterovirus D68 complete genome"). Saldrán unos 400 resultados, elige el que tenga como ordinal tu código de alumno.** Descarga su secuencia completa en formato nucleótidos y su genoma anotado en formato aminoácidos en sendos ficheros FASTA en Mis Documentos. Lee ambos ficheros FASTA y escribe los resultados en las variables nnn.enteno (un vector de nucleótidos) y

aaa.enteno(una lista de vectores de aminoácidos) ¿Qué cepa (isolate) te ha tocado? ¿Cuántos nucleótidos tiene? ¿Cuántos genes tiene? ¿Cuál es la longitud de su gen más largo?

**(3.2) Algoritmo de composición de bases local (visto en clase) con while().** Vamos a introducir el algoritmo de composición de bases local en secuencias visto en clase esta semana usando un bucle while() para implementar una función llamada

**composicionBasesLocal (dna.sequence=c("a","c","g","t"), window.length=200, offset=100, circular=FALSE )**

que devuelva una matriz de cuatro columnas con las probabilidades estimadas para un modelo multinomial aplicado a cada ventana utilizada. Recuerda que para circular==TRUE se considerará que la secuencia es circular (como se comentó en clase); pero cuando circular==FALSE las ventanas finales tendrán un tamaño más pequeño (como se comentó en clase). **Respuesta: Código de la función desarrollada. Tres figuras para ventanas con longitud 20, 50 y 100 para la secuencia nnn.entero, con offset=20 y circular=TRUE. Inserte en el fichero RTF dichas figuras. Comente lo que observa en dichas figuras. ¿Qué conclusiones extrae?**

**(3.3) Algoritmo de composición de bases local (visto en clase) pero con for().** Repita el ejercicio (3.2) pero sustituyendo todos los bucles while() por bucles for() que realicen lo mismo. Llame a la nueva función **composicionBasesLocalFor(). Respuesta: Código de la función desarrollada. ¿Cómo ha comprobado que ambas funciones dan exactamente el mismo resultado?**

**(4.2) Algoritmo de composición de aminoácidos local con while().** Modifique la función vista en clase **composicionBasesLocal ()** para obtener la nueva función **composicionAminoacidosLocal ()** que devuelva una matriz de 20 columnas con las probabilidades estimadas para un modelo multinomial para aminoácidos aplicado a cada ventana utilizada. Recuerda que con una letra los aminóacidos son a()[-1] y que para circular==TRUE se considerará que la secuencia es circular (como se comentó en clase) y circular==FALSE que las ventanas finales tendrán un tamaño más pequeño, como en (3.2). **Respuesta: Código de la nueva función desarrollada. Tres figuras donde se superpongan los resultados para todos los genes calculados con ventanas de longitud 20, 50 y 100 para la secuencia aaa.entero, con offset=20 y circular=TRUE. Inserte en el fichero RTF dichas figuras. ¿Dibujar todos los genes genera un figura difícil de entender/visualizar? Comente lo que observa en dichas figuras. ¿Son similares en composición los diferentes genes?**

**(4.3) Algoritmo de composición de bases local pero con for().** Repita el ejercicio (4.2) pero sustituyendo todos los bucles while() por bucles for() que realicen lo mismo. Llame a la nueva función **composicionAminoacidosLocalFor(). Respuesta: Código de la función desarrollada. ¿Cómo ha comprobado que ambas funciones dan exactamente el mismo resultado? Tres figuras donde se superpongan los resultados solo para el primer gen calculado con ventanas de longitud 20, 50 y 100 para la secuencia aaa.entero, con offset=20 y circular=TRUE. Inserte en el fichero RTF dichas figuras. ¿Con un solo gen la figura resulta más fácil de entender/visualizar? Comente lo que observa en dichas figuras. ¿Cómo ofecta la longitud de la ventana en el resultado que obtiene? ¿Qué longitud de ventana recomendaría?**

# PRÁCTICA 7

**(1) Descargaremos dos ficheros FASTA: el primero con el genoma anotado en formato aminoácidos de la bacteria *Caulobacter mirabilis* (NZ_CP024201.1), compruebe que tiene 4290 genes, y el segundo con el ADN completo en formato nucleótidos de dicha bacteria, que tiene 4577265 bp.** Lee el genoma anotado y guárdalo en una variable tipo lista llamada genoma. Lee el ADN y guárdalo en una variable tipo vector llamada dna. Toma en la variable midna los 70 000 nucleótidos que están a partir de la posición (inclusive) del nucleótido 1+(codigo-1)*70000 , donde codigo es tu código de alumno. Buscaremos genes en dichos nucléotidos.

**(1.1) Algoritmo de búsqueda de ORFs (candidatos a genes) visto en clase para codones.** Lea entero este enunciado antes de hacer nada. Implementaremos una función buscaORFs(nnn.sec) con el algoritmo que hemos visto en clase esta semana que usa tres bucles while(). La función buscará genes en la secuencia de nucleótidos nnn.sec y delvoverá una matriz llamada result con **dos filas** y tantas columnas como ORFs haya ennnn.sec (las filas tendrán las posiciones del primer nucleótido de los codones de inicio y de parada de cada ORF encontrado). EN CONCRETO, dentro de la función, se traducirá la secuencia de nucleótidos nnn.sec a aminoácidos usando translate() . Luego se buscarán las **metioninas** y las posiciones del primer nucleótido de cada una se guardarán en un vector inicios; se hará lo mismo con los **codones de parada** en un vector paradas. OBSERVE que el algoritmo visto en clase devuelve una matriz de dos columnas y que aquí se le pide que devuelve una matriz con dos filas. APLIQUE LA FUNCIÓN desarrollada a los primeros 3000 nucleótidos de midna usando el marco de lectura por defecto de translate. **Respuesta: Código de la función desarrollada. ¿Cuántos ORFs ha encontrado? Aplique la función a los 70 000 nucleótidos de midna, ¿cuántos ORFs tiene? Dibuje un histograma con la longitud en nucleótidos de todos los ORFs que ha encontrado. Incluya dicha figura en su fichero de respuesta. ¿Cuántos tienen una longitud superior a 100 nucleótidos?**

**(1.2) Algoritmo de búsqueda de ORFs para un marco de lectura dado.** Lea entero este enunciado antes de hacer nada. Implementaremos una función buscaORFs(nnn.sec,frame) con el algoritmo del ejercicio (1.1) pero modificado para que se busque genes en la secuencia de nucleótidos nnn.sec usando el marco de lectura frame, que podrá ser +1,+2,+3,-1,-2,-3 ; se delvoverá una matriz llamada result con dos filas y tantas columnas como ORFs haya ennnn.sec (las filas tendrán las posiciones del primer nucleótido de los codones de inicio y de parada de cada ORF encontrado; CUIDADO, si el marco de lectura es negativo, la primera fila tendrá un número más grande que la segunda fila). EN CONCRETO, dentro de la función, se traducirá la secuencia de nucleótidos nnn.sec a aminoácidos en el marco de lectura frame (si no recuerda cómo hacerlo consulte prácticas anteriores). Luego se buscarán las **metioninas** y las posiciones del primer nucleótido de cada una se guardarán en un vector inicios; se hará lo mismo con los **codones de parada** en un vector paradas. APLIQUE LA FUNCIÓN desarrollada a los primeros 3000 nucleótidos de midna usando el marco de lectura +2 y usando el marco de lectura -2. **Respuesta: Código de la función desarrollada. ¿Cuántos ORFs ha encontrado en cada caso? Aplique la función a los 70 000 nucleótidos de midna en ambos marcos de lectura ¿cuántos ORFs tiene en cada caso? Dibuje dos histogramas (en la misma figura, uno encima del otro) con la longitud de todos los ORFs que ha encontrado en los marcos de lectura +2 y -2. Incluya dicha figura en su fichero de respuesta. ¿Cuántos tienen una longitud superior a 150 nucleótidos?**

**(1.3) Algoritmo de búsqueda de ORFs para nucleótidos (sin importar el marco de lectura).** Lea entero este enunciado antes de hacer nada. Vamos a implementar una función buscaORFs(nnn.sec) que combine en una sola matriz las seis matrices resultado de aplicar buscaORFs(nnn.sec,frame), con frame = +1,+2,+3,-1,-2,-3.Aplique la función a los primeros 3000 nucleótidos de midna. **Respuesta: Código de la función desarrollada. ¿Cuántos ORFs ha encontrado? Aplique la función a midna, ¿cuántos ORFs tiene? Dibuje un histograma con la longitud de todos ORFs que ha encontrado. ¿Cuántos tienen una longitud superior a 200 nucleótidos?**

**(2.1) Algoritmo de búsqueda de genes con contraste de hipótesis.** Lea entero este enunciado antes de hacer nada.Implemente una función buscaGenes(nnn.sec,pval) que use la función buscaORFs(nnn.sec) para determinar los ORFs de nnn.sec y que aplique un contraste de hipótesis basado en la longitud (que hemos visto en las transparencias de clase esta semana; recuerde que se calcula la probabilidad de que un ORF con cierta longitud sea aleatorio y si dicha probabilidad es menor o igual que pval, se acepta como gen, y si no se rechaza por ser un ORF demasiado corto). La función devolverá una matriz con dos filas y tantas columnas como ORFs haya que pasen el contraste de hipótesis con un valor-p de pval. Aplique la función a los primeros 3000 nucleótidos de midna usando un valor p de 0.05. **Respuesta: Código de la función desarrollada. ¿Cuántos genes ha encontrado? Aplique la función a** midna**, ¿cuántos genes tiene? Dibuje un histograma con la longitud de todos genes que ha encontrado.**

**(2.2) Vamos a comparar sus resultados con el genoma anotado de la bacteria (variable** genoma). Lea entero este enunciado antes de hacer nada. Tome el primer gen de midna y busque a mano dicho gen en el genoma anotado en la web de NCBI (use el buscador de Windows Ctrl+F). ¿Encuentra dicho gen? Tiene todo el genoma anotado en la variable genoma. Escriba un código en R que busque el primer gen de midna en la variable genoma (use la función extraeGenLocation(); consulte en prácticas anteriores cómo se hacía esto en R a partir del atributo attr(,"Annot") del gen para entender cómo funciona). ¿Encuentra dicho gen? Busque en midna el primer gen que esté codificado en la cadena complementaria (su nucleótido de inicio es mayor que su nucleótido final) y busque a mano dicho gen en el genoma anotado en la web de NCBI (use el buscador de Windows Ctrl+F). ¿Encuentra dicho gen? Escriba un código en R que busque dicho gen de midna en la variable genoma (use la funciónextraeGenLocation() ). ¿Encuentra dicho gen? **Respuesta: Código R que ha usado. Respuesta a las preguntas (si encuentra el gen escriba qué proteína codifica dicho gen).**

```
extraeGenLocation = function (genAnotado)

{ str = attr(genAnotado,"Annot")

  if (is.null(str)) stop("no es un gen anotado desde fasta")

  pattern="[0-9]+\\.\\.[0-9]+"

  pat.reg = gregexpr(pattern, str)

  pat.res = regmatches(str,pat.reg)[[1]]

  list ( posiciones = as.numeric(strsplit(pat.res,"\\.\\.")[[1]]) ,

        complement = gregexpr("location=complement",str)[[1]][1] > 0 )}
```

**(2.3) Comparación de todos los genes de midna.** Escriba un código en R que busque todos los genes de midna en la variable genoma y determine si todos están en ella o falta alguno. Tiene libertad para hacerlo como quiera (con un bucle for o con un while o usando apply ). Note que en genoma, y lo que extrae la función extraeLocation() , las posiciones de nucleótidos siempre están en orden creciente, pero en su función buscaGenes() a veces están en orden decreciente. **Respuesta: Código R que ha usado. Respuesta ¿cuántos genes de midna se encuentran en genoma? Si alguno no está, ¿qué pasa con los que no se encuentran? ¿Qué conclusiones extrae de esta práctica sobre la anotación en NCBI del genoma de esta bacteria?**

**(3.1) Vamos a obtener los modelos markovianos para algunos genes del genoma anotado.**Lea entero este enunciado antes de hacer nada. Escribe en una variable tipo lista llamada misgenes la sublista de genoma con los 66 genes (recuerde que están en formato aminoácidos) a partir de la posición (inclusive) del gen 1+(codigo-1)*66 (comprueba que esta lista tienes 66 elementos). A partir de la función estimaModeloMarkov(sec) vista en clase el pasado 25 de abril, escriba una nueva función estimaModeloMarkov ( sequence , alphabet ) que estime el modelo markoviano de la

secuencia sequence usando como alfabeto el vector de caracteres alphabet. Compruebe que funciona ejecutando los siguientes comandos:

estimaModeloMarkov ( sample(words(1),666,T) , words(1) )

estimaModeloMarkov ( sample(a()[-1],333,T) , a()[-1] )

Escriba un código en R que obtenga una lista mi**MMlist** con los 66 modelos markovianos asociados a los 66 genes en la variable misgenes. **Respuesta: Código de la función desarrollada. Modelo de Markov obtenido para los genes número 7 y 13 (usando la función anterior). ¿Se parecen ambos modelos markovianos?**

**(3.2) Modelo de Markov promedio a partir de los resultados de la función** estimaModeloMarkov . Escriba un código en R que usando de forma adecuada la media, mean(), obtenga un modelo de Markov promedio miMM con la medias las probabilidades iniciales y las medias de las matrices de transición. VERIFIQUE que el modelo de Markov promedio es realmente un modelo de Markov bien formulado. Escriba un código en R que usando de forma adecuada la desviación típica, sd(), obtenga las desviaciones típicas de las probabilidades iniciales y de las matrices de transición de miMM. **Respuesta: Código en R usado. Media y desviación típica de las probabilidades iniciales. Media y desviación típica de las matrices de transición. ¿Qué conclusiones extrae de los resultados obtenidos? ¿Puede afirmar que todos sus genes se describen por el mismo modelo markoviano?**

**(3.3) Vamos a generar una secuencia con un alfabeto dado a partir de un modelo de Markov.** Escriba una **función**

**secuenciaModeloMarkov ( MM = list ( probInic = rep(c=0.25,g=4), matTrans = matrix (0.25,4,4) ), tam=100 )**

que genere una secuencia de tam caracteres usando como alfabeto los nombres de probInic usando el modelo de Markov dado por la lista MM , que se asume que es compatible con la salida de la **función** estimaModeloMarkov**()** del apartado (3.1). Ejecute el siguiente código:

gen = misgenes[[7]]; MM1 = estimaModeloMarkov **(**gen**)**

MM2 = estimaModeloMarkov **( secuenciaModeloMarkov ( MM1, length(gen) ))**

Compruebe que MM2 es un modelo markoviano correcto. **Respuesta: Código de la función secuenciaModeloMarkov() desarrollada. Compara los modelos markovianos MM1 y MM2, ¿se parecen entre sí? ¿Qué conclusiones extrae de dicha comparación?**

# PRÁCTICA 8

**(1.1) Estimación de una cadena de Markov oculta.** Ejecute los siguientes comandos (vistos en clase); para entenderlos mejor puede consultar las transparencias de clase. Será necesario para lo siguiente. Escriba una función estimaHMM(sec,hid,alph) que devuelva la lista HMM con las probabilidades iniciales, la matriz de transición y la matriz de emisión; la variable sec será una secuencia con el alfabeto alph y hid será un vector de caracteres de la misma longitud que sec. **Respuesta: Código de la función, incluyendo la comprobación de que sec y hid tienen la misma longitud. Ejecute la función y determine las probabilidades iniciales, matriz de transición y matriz de emisión.**

```
sec="actaatgtctgactatcagcggcagcagctgaccgt"; sec=s2c(sec);

hid="AAAAAAAAAAAAAAAAAABBBBBBBBBBBBBBBBB"; hid=s2c(hid);

S = words(1); nS = length(S);

H =  levels(factor(hid)); nH = length(H);

p = count (hid, 1, freq=TRUE, alphabet=H);

names(p)=H;

T = matrix( count(hid,2,freq=TRUE,alphabet=H), nH, nH, byrow=TRUE);

T = T / apply(T,1,sum);

rownames(T)=H; colnames(T)=H;

E = matrix(,nH,nS); rownames(E)=H; colnames(E)=S;

E = t(sapply(H, function(h)

    count( sec[hid==h],1,freq=TRUE,alphabet=S)));

HMM = list ( p=p, T=T, E=E );
```

**(1.2) Reconocimiento de colas LPXTG en proteínas M de membrana en Streptococcos.** Descargue del campus virtual el fichero fasta "LPXTGsequences.txt" y léalo en una variable en R. La primera secuencia del fichero es de 696 aminoácidos y la segunda secuencia es su anotación con 5 estados ocultos. Aplique la función estimaHMM a ambas secuencias para estimar un modelo de Markov oculto para secuencias LPXTG. **Respuesta: Código de llamada a la función. Determine las probabilidades iniciales, matriz de transición y matriz de emisión.**

**(2.1) Generación de una secuencia aleatoria a partir de un modelo de Markov oculto.** Ejecute los siguientes comandos (vistos en clase); para entenderlos mejor puede consultar las transparencias de clase. Será necesario para lo siguiente. Escriba una función generaHMM(HMM,tam) que devuelva una secuencia de tam estados observables generados por el modelo de Markov oculto HMM, cuyas probabilidades iniciales son HMM$p, cuya matriz de transición es HMM$T y cuya matriz de emisión es HMM$E; el alfabeto de estados observables serán los nombres de las columnas de HMM$E y el alfabeto de estados ocultos los nombres de las filas de HMM$E. **Respuesta: Código de la función. Genere 3 secuencias de 50 nucleótidos generadas mediante generaHMM con el modelo HMM del ejercicio (1.1).**

```
tam = 10; SEC = character(tam); HID= character(tam)

HID[1] = sample(H,1,prob=p)

SEC[1] = sample(S,1,prob=E[HID[1],])

for ( i in 2:tam )
```

```
{HID[i] = sample(H,1,prob=T[HID[i-1],])
  SEC[i] = sample(S,1,prob=E[HID[i],])}
```

**(2.2) Secuencias aleatorias de colas LPXTG en proteínas M.** Usando la función generaHMM genere 60 secuencias de 50 aminoácidos usando el modelo de Markov oculto del ejercicio (1.2). Escriba un código en R que determine la posición de las parejas de aminoácidos "LP" y "RT" en dichas 60 secuencias (si alguna no las contiene debe devolver NA). **Respuesta: Códigos en R usados y resultado obtenido con su código.**

**(3.1) Algoritmo de Viterbi.** Ejecute los siguientes comandos (vistos en clase) que ejecutan el algoritmo de Viterbi sobre la secuencia del sec ejercicio (1.1) y su modelo de Markov oculto. Compruebe que el resultado hid.sec coincide con hid. Escriba una función ViterbiHMM(sec,HMM) que aplique el algoritmo de Viterbi a la secuencia sec con el modelo de Markov oculto HMM, cuyas probabilidades iniciales son HMM$p, cuya matriz de transición es HMM$T y cuya matriz de emisión es HMM$E; el alfabeto de estados observables serán los nombres de las columnas de HMM$E y el alfabeto de estados ocultos los nombres de las filas de HMM$E. **Respuesta: Código de la función. Genere 3 secuencias de 50 nucleótidos generadas mediante generaHMM con el modelo HMM del ejercicio (1.1) y aplique el algoritmo de Viterbi a dichas secuencias. ¿Qué resultado obtiene y qué conclusión extrae de dicho resultado?**

```
N = length(sec)
V = matrix(0,ncol=N,nrow=length(H)); rownames(V)=H; colnames(V)=sec;
M = matrix(0,ncol=N,nrow=length(H)); rownames(M)=H; colnames(M)=sec;
 ##
V[ ,1] = p * E[ ,sec[1]]; M[ ,1] = 1:nH
 ## prods = numeric(nH); names(prods) = H
 ## for (iV in 2:N) for (hV in H)
 ## { for (hT in H) prods[hT] = V[hT,iV-1] * T[hT,hV]
 ##   V[hV,iV] = E[hV,sec[iV]] * max ( prods )
 ##   M[hV,iV] = which.max ( prods )
 ## }
for ( iV in 2:N )
{  prodMatr = V[ ,iV-1]*T
   M[ , iV] = apply ( prodMatr, 2, which.max )
   V[ , iV] = E [ ,sec[iV] ] * prodMatr[ cbind(M[,iV],1:nH) ]
}
 ##
hid.sec = character(N)
cual = which.max(V[,N])
hid.sec[N] = H[cual]
for (iM in (N-1):1)
{ cual = M[cual,iM]
```

hid.sec[iM] = H[cual]}

**(3.2) Aplicación del algoritmo de Viterbi a secuencias de colas LPXTG en proteínas M.** Usando el modelo de Markov oculto del apartado (1.2) aplique la función ViterbiHMM(sec,HMM) a la primera secuencia del fichero fasta "LPXTGsequences.txt". Compare el resultado con la segunda secuencia de dicho fichero. ¿En qué se diferencian? Aplique la función a las secuencias tercera y cuarta de dicho fichero. ¿Qué resultado obtiene? **Respuesta: Códigos en R usados, resultados obtenidos y respuesta a las preguntas.**

**(4.1) Algoritmo de Viterbi con logaritmos.** Consulte las transparencias de clase y desarrolle una función ViterbiLogHMM(sec,HMM) que aplique el algoritmo de Viterbi con logaritmos a la secuencia sec con el modelo de Markov oculto HMM, (le recomiendo alterar la función ViterbiHMM(sec,HMM) en los lugares adecuados, según indican las transparencias). Compruebe que ambas funciones dan el mismo resultado al ser aplicados a las secuencias sec y hid del apartado (1.1). **Respuesta: Código de la función. Genere 3 secuencias de 50 nucleótidos generadas mediante generaHMM con el modelo HMM del ejercicio (1.1) y aplique el algoritmo de Viterbi con logaritmos a dichas secuencias. ¿Qué resultado obtiene y qué conclusión extrae de dicho resultado?**

**(4.2) Aplicación del algoritmo de Viterbi con logaritmos a secuencias de colas LPXTG en proteínas M.** Usando el modelo de Markov oculto del apartado (1.2) aplique la función ViterbiLogHMM(sec,HMM) a la primera secuencia del fichero fasta "LPXTGsequences.txt". Compare el resultado con la segunda secuencia de dicho fichero. ¿En qué se diferencian? Aplique la función a las secuencias tercera y cuarta de dicho fichero. ¿Qué resultado obtiene? **Respuesta: Códigos en R usados, resultados obtenidos y respuesta a las preguntas. ¿Funciona en este caso mejor el algoritmo de Viterbi con o sin logaritmo? ¿Por qué?**

# PRÁCTICA 9

**(1) Matrices de Substitución.** Ejecuta library(Biostrings); data(BLOSUM100); data(PAM30); BLOSUM100; PAM30 . Cambie el carácter "*" de los nombres de filas y columnas por "-". Desde el campus virtual descargue el fichero FASTA **samba-virus-comparado-otros-virus.fasta** y léalo con read.fasta() en la variable **virus** (lista de 8 secuencias). Dibuje un diagrama de puntos con dotPlot() del paquete seqinr entre la primera secuencia virus[[1]] y la secuencia cuyo número es 2+( codigoalumno %%7), donde codigoalumno es su código de alumno. **Respuesta: Figura con el último diagrama de puntos que ha obtenido. A la vista de este diagrama qué puede decir del alineamiento de ambas secuencias.**

**(1.1) Algoritmo de alineamiento de secuencias de Needelman-Wunsch.** Escriba una función

```
alineamientoNW = function(s, t, M)
{  ...
   return(list(score,sRes,tRes))
}
```

que a partir de las cadenas de caracteres s y t, y de la matriz de substitución M, aplique el algoritmo de alineamiento global de Needelman-Wunsch visto en clase (y que se presenta en el Campus Virtual) obteniendo la puntuación score y las secuencias alineadas sRes y tRes (ambas de la misma longitud y con carácter "-" como hueco). Compruebe que funciona aplicándola

**alineamientoNW ( "VIVALAVIDA", "VIVALADAVIS", BLOSUM100)**
**alineamientoNW ( "VIVALAVIDA", "VIVALADAVIS", PAM30)**
**Respuesta: Algoritmo de la función que ha desarrollado. Aplique dicho algoritmo a las secuencias** "MRTEIESLWVFALASKFNIYMQQHFASLLVAIAITWFTITI" y "MEDQVGFGFRPNDEELVGH YLRNKIEGNTSRDVEVAISEVNIC" **con matrices de substitución** BLOSUM100 **y** PAM30. **A la vista del resultado, ¿qué puede concluir de ambos alineamientos?**

**(1.2) En el algoritmo visto en clase** se usa Cual[i1,j1]= which.max(vec); cambie dicho comando por Cual[i1,j1]= sample(which(vec==max(vec)),1) . Escriba una nueva función alineamientoNWnew(s,t,M)
con este cambio. **Respuesta: Aplique la nueva función al ejemplo (1.1) con las matrices** BLOSUM100 **y** PAM30 **he indique si observa algún cambio en el resultado. ¿Por qué ocurre?**

**(1.3) Alineamiento de secuencias de nucleótidos.** Aplique los algoritmos **alineamientoNW() y alineamientoNWnew()** a las cadenas de nucleótidos "TTGCACGATAGTTGCATATGCTACAA" y "TTGATTAAGCCCTATGCTTCGTACACAAAAC", con la matriz de substitución

```
subsMatriz = matrix( c(
1, -1, -1.5, -1.5, -2,
-1, 1, -1.5, -1.5, -2,
-1.5, -1.5, 1, -1, -2,
-1.5, -1.5, -1, 1, -2,
-2, -2, -2, -2, NA
), 5,5)
rownames(subsMatriz)=c('A','C','G','T','-')
colnames(subsMatriz)=c('A','C','G','T','-')
```

Aplique dicho algoritmo a las secuencias
"ACTTCACCAGCTCCCTGGCGGTAAGTTGATCAAAGGAAACGCAAAGTTTTCAAG"
"GTTTCACTACTTCCTTTCGGGTAAGTAAATATATAAATATATAAAAATATAATTTTCATC"

**Respuesta: Alineamientos que ha obtenido. ¿Qué conclusión extrae de estos alineamientos? Razone su respuesta.**

**(2.1) Proteína del mimivirus Samba.** Aplique su función alineamientoNW() para alinear la primera secuencia virus[[1]] y la secuencia cuyo número es 2+( codigoalumno %%7), donde codigoalumno es su código de alumno, del fichero FASTA **samba-virus-comparado-otros-virus.fasta. Respuesta: Puntuación (*score*) que ha obtenido al usar las matrices BLOSUM100, BLOSUM45, PAM250 y PAM30. ¿Qué puede comentar sobre los alineamientos que ha obtenido al cambiar la matriz de substitución?**

**(2.2) El fichero FASTA contiene 8 secuencias, vamos a compararlas todas con todas.** Escriba un código en R que rellene una matriz de 8x8 cuyo elemento (i,j) sea el score (puntuación) de comparar las secuencias virus[[i]] y virus[[j]]. Observe que es una matriz simétrica y que sólo tiene que calcular la parte triangular superior (rellenando la inferior aprovechando la simetría de la matriz).
**Respuesta: Matrices de puntuaciones que ha obtenido usando las matrices BLOSUM100, BLOSUM45, PAM250 y PAM30. ¿Cuál es el mejor alineamiento en su opinión? ¿Por qué?**

**(3.1) Algoritmo de alineamiento local de secuencias de Smith-Waterman.** Desarrolle en R el algoritmo visto en las transparencias de clase y escriba una función

```
alineamientoLocal = function(s, t, M)
{ ...
  return(list(score,sRes,tRes))
}
```

que a partir de las cadenas de caracteres s y t, y de la matriz de substitución M, aplique el algoritmo de alineamiento local de Smith-Waterman obteniendo la puntuación score (máxima) y las subsecuencias alineadas sRes y tRes (ambas de la misma longitud y con carácter "-" como hueco). Compruebe que funciona aplicándola a alineamientoLocal ( "VIVALAVIDA", "VIVALADAVIS", BLOSUM100) y **alineamientoLocal ( "VIVALAVIDA", "VIVALADAVIS", PAM30)**. **Respuesta: Código desarrollado.** Aplique dicho algoritmo a las secuencias "MRTEIESLWVFALASKFNIYMQQHFASLLVAIAITWFTITI" y "MEDQVGFGFRPNDEELVGHYLRNKIEGNTSRDVEVAISEVNIC" **las matrices BLOSUM100, BLOSUM45, PAM250 y PAM30. Resultados que ha obtenido. ¿Qué diferencias observa entre usar las diferentes matrices de substitución?**

**(3.2) Uso del algoritmo pairwiseAlignment().** Repita el ejercicio (2.2) usando esta función de Biostrings con la opción type="global" para obtener la puntuación (score). Repita el ejercicio (2.2) usando esta función de Biostrings con la opción type="local" para obtener la puntuación (score). **Respuesta: Matrices de puntuaciones que ha obtenido usando las matrices BLOSUM100, BLOSUM45, PAM250 y PAM30. ¿Cuál es el mejor alineamiento en su opinión? ¿Por qué?**