CURSO: CC471

Practica 09

1. Introducción

PRACTICA : Clustering e Instalacion y uso de R -
Al final Subirá al sitio web en la carpeta tareas un archivo **CC471-LAB09-<Nombre-apellido>.zip con los archivos generados.**

Objetivo general:
1. Análisis de Información utilizando herramientas estadísticas y Data clustering
2.

RECURSOS INFORMÁTICOS
http://www.ncbi.nlm.nih.gov/ (Nacional Center for Biotechnology Information – NCBI)
http://www.ebi.ac.uk/ (European Bioinformatics Institute- EBI- EMBL)
http://pfam.xfam.org (Trust Sanger Institute, Genome Research Limited EMBL-EBI
http://www.ebi.ac.uk/tools/msa/clustalo (Clustal Omega EMBl – EBI)
https://www.digitalocean.com/community/tutorials/how-to-install-r-on-ubuntu-16-04-2

DESARROLLO

I. Instalar R.-
a) Instalar el repositorio mantenido por CRAN

```
sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys
E298A3A825C0D65DFD57CBB651716619E084DAB9

sudo add-apt-repository 'deb [arch=amd64,i386] https://cran.rstudio.com/bin/
linux/ubuntu xenial/'

sudo apt-get update
```

b) instalar R:

```
sudo apt-get install r-base
```

c)Comprobar la instalación:

```
sudo -i R


Output:
R version 3.3.1 (2016-06-21) -- "Bug in Your Hair"
   . . .
   Type 'demo()' for some demos, 'help()' for on-line help, or
   'help.start()' for an HTML browser interface to help.
   Type 'q()' to quit R.
   >
```

2. Instalar paquetes de R desde CRAN
Como ejemplo instalaremos txtplot – Libreria que genera grafos con ASCII y graficos tipo scatterplot, line plot, density plot, acf and barcharts

> install.packages('txtplot')
-Despues de escoger el mirror desde el cual se instalará la libreria, se procederá con la instalación
Luego que la instalacion este completa se puede cargar la libreria:

> library('txtplot')
como ejemplo podemos usar a data que proporciona el paquete (Distancias de frenado y velocidades del año 1920)

>txtplot(cars[,1], cars[,2], xlab = "speed", ylab = "distance")
Para mas ayuda acerca de textplot se puede usar help(txtplot)

Instalacion de la libreria scatterplot
> install.packages('scatterplot3d')
>scatterplot3d(iris$Petal.Width, iris$Sepal.Length, iris$Sepal.Width)

Cualquier paquete precompilado se puede instalar desde CRAN con **install.packages()**
Para saber cuales son los paquetes disponibles se puede consultar **https://cran.r-project.org/**

Ejemplos básicos con R
 # crear el directorio

```
mkdir MiProyectoEnR
# entrar en el directorio
cd MiProyectoEnR
# ejecutar R
R
```

z <- 10 # se define la variable ''z'' y se le asigna el valor 10

```
z = 10 #idem
z # se muestra el contenido
z <- c(10,45,33)  # creamos un vector y lo asignamos a la variable ''z''
z[2] # usamos [] para acceder a un elemento del vector
length(z) # muestra la longitud del vector. () se usa para llamar funciones.

x <- rnorm(100,mean=10,sd=2)  # genero aleatoriamente 100 números con una
distribución normal de media 10 y desvío estandar 2.
x
mean(x) # su media
range(x) # valores mínimo y máximo
sd(x) # el desvío estandar
?hist # ayuda sobre la función hist
hist(x, main="Mi primer histograma") # crea histograma y lo muestra en pantalla
q() # salir!
```

- Se puede escribir varios comandos en un archivo de texto, y ejecutarlo en R asi: source("ruta/MiArchivo.R")
- Usar flechas arriba y abajo para moverse en el historial de comandos
- El historial se puede guardar en cada sesión (?history)
- Pueden ejecutarse múltiples comandos en una línea, separándolos con ";". Ej.: x<-5; y<-15;
- Con TAB, se completan nombres de objetos de R y archivos

3. Ejemplo de Clustering

En este ejemplo vamos a utilizar técnicas básicas para encontrar conglomerados o "clusters" en un conjunto de datos biológicos. La idea es identificar agrupamientos naturales en los datos, que presenten un comportamiento similar entre si, con alguna relevancia biológica. En particular utilizaremos conjuntos de datos provenientes de medidas de expresión génica generadas mediante experimentos con microarrays con muestras tomadas a diferentes tiempos, para identificar grupos o clusters de genes que tengan un perfil de expresión común.

En 1997 DeRisi et. al. estudiaron los cambios transcripcionales de prácticamente todos los genes de Saccharomyces cerevisiae a lo largo del salto metabólico de fermentación a respiración. En el archivo diauxic.txt se encuentra una tabla donde cada fila representa un gen y cada columna el tiempo al cual se tomó la muestra. Así, cada celda contiene una medida de la expresión de un gen particular a un momento determinado.

Trataremos de encontrar respuestas a preguntas como ¿Qué genes presentan un perfil de expresión similar? ¿Cuales son los genes que se sobreexpresan al final (en ausencia de glucosa) y cuales son reprimidos? ¿Cuantos tipos de comportamientos diferentes hay?

2.En el archivo **TablaEjemplo.txt** hay una tabla mínima con datos inventados. La tabla contiene para cuatro genes (A, B, C y D) su nivel de expresión a las 0hs, 1hs y 2 hs luego de algún tratamiento.

Lo primero que vamos a hacer en **R** es cargar los datos de la tabla de ejemplo **TablaEjemplo.txt**. Para esto tipear:

```
>MiTabla=read.csv("/curso/clustering/TablaEjemplo.txt", sep="\t", row.names="gen")
```

Lo que hicimos acá, fue leer el archivo de texto TablaEjemplo.txt utilizando la función **read.csv()** de **R** (csv -- comma separated values -- es la extensión que suele usarse para nombrar archivos de texto conteniendo valores separados por algún delimitador, que comunmente es una coma, o un caracter de tabulación). Además del nombre del archivo, la función **read.csv** nos permite declarar cual es el separador de campos (en este caso son tabulaciones, que se indican como **\t**), y cual es la columna que contiene los identificadores para las filas ("gen" es el nombre de esta columna en nuestro ejemplo). Finalmente, notar que el resultado de ejecutar la función **read.csv** se está almacenando en la variable *MiTabla*.

A continuación vamos a utilizar la función **dist** para calcular una matriz de distancias a partir de los datos de la tabla. Recordar que estos datos ya están cargados en una variable que se llama *MiTabla*.

```
>MisDistancias=dist(MiTabla,method="euclidean")
```

Para ver el resultado, simplemente pedimos el contenido de la variable:

```
>MisDistancias # nos muestra la matriz de distancias
```

Seguidamente vamos a agrupar en forma jerárquica a los genes, de acuerdo a esta matriz de distancias, utilizando la función **hclust()** (**h**ierarchical **clust**ering).

```
>MiClusteringJerarquico=hclust(MisDistancias, method="complete")
```

La función **hclust()** realiza el clustering jerárquico utilizando el criterio de agregación especificado ("complete linkage" o "vecino mas lejano" en este caso). En **R** si quieren obtener ayuda sobre algún comando, por ejemplo para ver qué otras opciones existen para una función, pueden usar la función **help()**. Haciendo:

>help(hclust)
output recortado

```
...
  method: the agglomeration method to be used. This should be (an
          unambiguous abbreviation of) one of '"ward"', '"single"',
          '"complete"', '"average"', '"mcquitty"', '"median"' or
          '"centroid"'.
...
```

En este caso complete = complete linkage, single = "single linkage" (o vecino mas cercano), average = promedio.

Para graficar el resultado de este agrupamiento utilizamos la función plot()

>plot(MiClusteringJerarquico) #Grafica el dendograma que resulta del clustering jerarquico

Los gráficos en R se pueden exportar a PDF y a JPEG facilmente:

```
>pdf("MiImagen.pdf") #alternativamente >jpeg("MiImagen.jpeg")
>plot(MiClusteringJerarquico) # el plot se escribe en el PDF (no se muestra en
pantalla!)
>dev.off() # cierra el archivo, los próximos plots volverán a ser mostrados en
pantalla
```

Ahora vamos a repetir el análisis, estandarizando los datos de expresión disponibles para cada gen, de manera de que todos valores de expresión se encuentren en la misma escala y centrados en cero.

R cuenta con la función **scale()** para estandarizar datos. El problema es que esta función estandariza por columnas, y nosotros necesitamos hacerlo por filas (las columnas contienen datos de expresión para todos los genes). Por lo tanto necesitamos transponer la matriz de datos, de manera de que las columnas pasen a ser filas y viceversa. Para transponer tablas, **R** nos ofrece la función **t()**.

MiTablaTranspuesta=t(MiTabla)
Analizar el contenido de la nueva variable MiTablaTranspuesta para ver como se modificaron los datos. A continuación, podemos utilizar **scale()** sobre esta nueva tabla.

MiTablaEstandarizadaTranspuesta=scale(MiTablaTranspuesta)

Analizar el contenido de la nueva variable MiTablaEstandarizadaTranspuesta para verificar que los datos hayan sido normalizados. Finalmente, necesitamos volver a transponer la tabla estandarizada, para tener nuevamente los datos por gen en filas, y los datos de los distintos tiempos en columnas.

MiTablaEstandarizada=t(MiTablaEstandarizadaTranspuesta)

Esto mismo que hemos hecho en tres pasos (transponer la tabla, normalizarla y volverla a transponer), puede hacerse en un solo paso, así:

>MiTablaSTD=t(scale(t(MiTabla)))

#comparar las medias y las dispersiones....

```
summary(t(MiTabla))
summary(t(MiTablaSTD))
sd(t(MiTablaSTD))
sd(t(MiTabla))
```

Ahora vamos a repetir todos los comandos desde `>MisDistancias=dist(MiTabla,method="euclidean")`inclusive, en adelante, pero utilizando la tabla con los datos estandarizados MiTablaSTD.

Podemos probar con otras medidas de distancia entre datos de expresión, por ejemplo basadas en correlación. Las distancias basadas en correlación permiten comparar "tendencias" en los datos no estandarizados, en forma similar a la distancia euclídea con datos previamente estandarizados. Este tipo de distancias, como la correlación de Pearson, son las mas utilizadas en analisis de expresiòn génica con microarrays.

```
DistanciaCorr=as.dist(1-cor(t(MiTabla)))
```

La funcion cor() calcula la matriz de correlaciones sobre las columnas, por defecto, utiliza la correlación de Pearson. Este coeficiente de correlacion varia entre 1 (genes perfectamente correlacionados) y -1 (correlación negativa perfecta: cuando uno sube, el otro baja), pasando por el 0 (no hay correlación lineal entre ellos). La distancia es 1-cor() de forma tal que si la correlación alta, la distancia sea mínima y viceversa. La función as.dist() convierte la matriz al formato correcto para ser utilizado por las funciones de clustering, como hclust().

Luego de visualizar el nuevo dendograma....

```
>heatmap(MiTablaSTD, Colv=NA)
```

Inspeccionamos el heatmap que es una imagen que muestra niveles de expresion mas altos como mas calientes o blancos/amarillos y los de menos expresion como mas frios o rojos. Colv indica si las columnas (tiempos en este caso) deben ser agrupadas o reordenadas y cómo. Colv= NA(not available), indica que no las reagrupe (respetando el orden natural de la variable tiempo). Las filas (genes) van a ser agrupadas utilizando la funcion hclust() con sus opciones por default y la distancia euclidea, a menos que se indique otra cosa.

Observando el dendograma de los datos estandarizados y el heatmap, podemos ver que hay dos grupos bien distinguibles de genes. Uno que aumenta su expresión a lo largo del tiempo y el otro que disminuye. Podemos cortar el arbol para quedarnos con estos dos grupos a cualquier altura entre ~1 y ~2.5, con la funcion "cutree"...

```
>MiCorte=cutree(MiClusteringJerarquicoSTD,h=1.5)  # cortamos el arbol a la altura
de 1.5.
#o lo que es lo mismo...
>MiCorte=cutree(MiClusteringJerarquicoSTD,k=2) # cortamos el arbol a una altura
tal que el número de clusters obtenido sea de 2.
>MiCorte #visualizamos a que cluster fue asignado cada gen
```

Ahora vamos a agrupar los genes del mismo ejemplo por el método de K-medias, al que se le debe pedir a priori un número K de clusters. En este caso, vamos a pedir K=2.

```
>MiClusteringKMedias=kmeans(MiTablaSTD,2)
>MiClusteringKMedias # vemos el resultado, que como es de esperar para un caso
tan simple, coincide con el método jerárquico.
```

Si el número de clusters a encontrar fuera mayor, es recomendable aumentar el número de inicios al azar; por ejemplo, para inicializar 100 veces en forma aleatoria:

```
>kmeans(MiTablaGrande,5,nstart=100)
```

Ejercicio:

Análisis de Clustering de expresión génica durante crecimeinto diáuxico en levaduras.

“El **crecimiento diaúxico** es un tipo crecimiento microbiano bifásico que tiene lugar cuando hay presentes dos sustratos diferentes que pueden ser utilizados como fuente de carbono. En este tipo de crecimiento microbiano se observa una curva de crecimiento bifásica debido a la utilización secuencial de distintas fuentes de carbono. El metabolismo del organismo es selectivo para uno de los sustratos (se usa la fuente de carbono que permite un crecimiento más rápido) y cuando la agota, comienza a metabolizar el otro”.

Ahora con lo aprendido, importar y analizar el conjunto de datos **diauxic.txt** Identificar grupos de genes que se comporten de manera similar

En Un documento, enumere y describa los pasos que realizo para el análisis. Incluya los resultados gráficos Dendogramas, y tambien los resultados del análisis.

Suba un archivo comprimido con la nomenclatura estandar que incluya sus resultados y sus respuestas al sitio web del repositorio del curso.