CONCEPTOS Y FUNCIONES BÁSICAS DE R Y RStudio

- qué vamos a aprender:
- capturar tablas datos:
  - formato Excel/TSV/CSV (ejemplos: csv, tsv, excel, gtf/gff, sam, vcf)
  - open data:
    - datos.gob.es
    - https://www.europeandataportal.eu/es
    - http://www.fao.org/faostat/en/#data
- analizar datos:
  - obtener información de los datos
  - transformar los datos
- escribir datos en ficheros
- crear gráficas
- ejercicio práctico "real": análisis expresión diferencial RNA-Seq

- cursos online para reforzar lo aprendido y para ampliar (Coursera, edX, Khan Academy, MiradaX,...)

- paneles de R:
- panel script/source
- panel console/terminal/jobs
- panel environment/history/connections
- panel files/plots/packages/help
- barra de menús (primera tarea: Session -> Set Working Directory... -> Choose Directory...)

- flujo de trabajo (interactivo):
- escribir orden R
- evaluar el resultado:
  - si está bien: siguiente orden
  - si está mal: corregir orden (usar historial de comandos con teclas flechas)
- (opcional) crear scripts con las órdenes que hemos comprobado que son correctas

- órdenes R:
- principalmente funciones
- no hay que saberlas todas (hay muchas, nadie las sabe todas):
  - se acaba aprendiendo las más usadas
  - se buscan en Google las que se necesitan y no se conocen:
    - tutorialspoint (sencillo)
    - stackoverflow (todo tipo de programación, preguntas/respuestas)
    - r-bloggers (temas concretos)
    - stat.ethz.ch (específico de R, manual de funciones)
  - ayuda en RStudio: (?funcion, help(función), empezar a escribir la función y F1 cuando salga la ayuda, buscar directamente en el panel de Help)
- el objetivo es aprender la mecánica, no el uso concreto de las funciones

- estructura de las funciones:
- nombre_funcion(parámetro/argumento/opción, ...)
- no hay que saber todas las opciones (ayuda: F1, help(), ?función, args(),...)
- argumentos por defecto
- value (objeto que produce/devuelve):
  - mostrado en la consola
  - capturado en variable

- variables:
- distintos tipos (character, numeric, booleano (TRUE/FALSE), NA (not available), vector, factor, data.frame, etc.; ver con class())
- distintas acciones dependiendo del tipo de variable

- vectores:
- coercion:
  - forzado que R hace implícitamente de TODOS los elementos de un vector a un mismo tipo de variable (character, numeric, booleano, factor)
  - funciones para realizar coercion explícita: as.character, as.numeric, etc.)
- recycling (aritmética de vectores)

- **obtención de uno o más elementos de un vector** (cinco modos básicos):

1- **vector[posicion]** (devuelve el elemento del vector en esa posición):
  - en lugar de una posición, se puede dar una función que devuelva un número (p.ej., which.max())

2- **vector[vector de posiciones]** (devuelve un vector con los elementos del vector en esas posiciones):
  - en lugar de un vector de posiciones, se puede dar una función que devuelva un vector de números (p.ej., grep, order, which)

3- **vector[nombre_del_elemento]** (si el vector tiene nombres):
  - en lugar de un nombre, se puede dar una función que devuelva un nombre (p.ej., grep con value = TRUE, substr)

4- **vector[vector de nombres de los elementos]** (si el vector tiene nombres)
  - en lugar de un vector de nombres, se puede dar una función que devuelva un vector de nombres (p.ej., grep con value = TRUE, substr)

5- **vector[condicion]** (devuelve un vector con los elementos del vector que cumplen la condición):
  - por ejemplo, a[a > 100] devolverá un vector con aquellos elementos del vector a que sean mayores que 100
  - comparaciones (>, <, >=, <=, ==, %in%)
  - se puede combinar dos o más comparaciones con & (que significa 'and') y/o con | (que significa 'or')
  - es muy recomendable usar paréntesis para agrupar correctamente las distintas comparaciones que se usan en una comparación compleja con varios & y/o |

6- **vector[vector de booleanos]** (devuelve un vector con los elementos del vector que hay en las posiciones donde el booleano es TRUE)
  - por ejemplo, suponiendo un vector ‘a’ con tres elementos, a[c(TRUE,FALSE,TRUE)] devolvería un vector con dos elementos: el primer y el tercer elementos del vector ‘a’
  - en realidad, este modo es el mismo que el anterior, pues cuando usamos una condición, lo que hace la condición es evaluarse frente a cada elemento del vector, produciendo en realidad un vector de booleanos, con el mismo número de elementos que el vector sobre el que aplicamos la condición, con TRUE en las posiciones del vector que cumplen la condición y FALSE en las que no la cumplen.
  - por ejemplo, suponiendo el vector a <- c(100,20,500), las siguientes ordenes serían equivalentes:

```
a[a > 40]
a[c(TRUE,FALSE,TRUE)]
```

De hecho, internamente, R convierte la primera a la segunda.

- data frames:
- para capturarlos desde un fichero o URL: read.table/csv/csv2/delim/delim2()
    - opción stringAsFactors para capturar o no como factores las columnas con strings
- para echarles un vistazo: head(), tail(), str(), View, dim()
- obtención de columnas: dataframe$nombre_de_columna (devuelve un vector con los elementos de dicha columna)

- **obtención de elementos del dataframe:**
- los mismos cinco modos básicos que con vectores, salvo que, en lugar de dar una posicion/vector_de_posiciones/nombre/vector_de_nombres/condición, se dan dos posiciones/vectores_de_posiciones/nombres/vectores_de_nombres/condiciones, separados por una coma: el primero será para seleccionar las filas del data frame que se desean, y el segundo para selecionar las columnas del dataset que se desean)
- si no se da uno de esos dos elementos, devuelve todas las filas/columnas

1- **dataframe[posición de la fila, posición de la columna]** (empty for ALL)

2- **dataframe[vector con posiciones de filas, vector posiciones de columnas]**

3- **dataframe[nombre de la fila, nombre de la columna]**

4- **dataframe[vector con nombres de filas, vector con nombres de columnas]**

5- **dataframe[condición que debe cumplir la fila, condición que debe cumplir la columna]**

6- **dataframe[vector de booleanos para las filas, vector de booleanos para las columnas]**

- al igual que con los vectores, en lugar de dar una posicion/vector_de_posiciones/nombre/vector_de_nombres, se puede dar una función que devuelva una posicion/vector_de_posiciones/nombre/vector_de_nombres
- se pueden combinar estos cinco modos, usando un modo para las filas y otro para las columnas. Por ejemplo, dataframe[condicion,c(2,5)] devolvería un data frame con las columnas 2 y 5 de las filas que cumplieran la condición indicada.

- funciones útiles:
- c (creación de un vector)
- seq (obtención de un vector de números desde X hasta Y, cada Z)
- rev (revertir el orden de un vector)
- unique (obtener los elementos de un vector, sin repeticiones)
- substr (obtención de substrings a partir de un string)
- gsub (reemplazo de un substrings dentro de un string)
- grep (obtención de las posiciones donde aparece un determinado elemento dentro de un vector, basándose en un patrón de búsqueda)
- grep con la opción value= (obtención de determinados elementos a partir de un vector, basándose en un patrón de búsqueda)
- paste (unión en un solo string de los elementos de uno o más vectores)
- sum, mean, log
- max (obtención del mayor elemento que hay en un vector)
- which.max (obtención de la posición donde aparece el mayor elemento de un vector)
- min
- which.min
- match (obtención de un vector con las posiciones que ocupan los elementos de un vector en otro vector)
- sort (obtención de los elementos de un vector, ordenados)
- order (obtención de las posiciones en el vector original de los elementos

de un vector ordenados)
- rank (obtención de las posiciones que ocuparía cada uno de los elementos tras ordenarlos)

- Ejemplo de sort/order/rank. Imaginemos el vector a <- c(8,5,1,3):
    - sort(a) daría el vector (1,3,5,8)
    - order(a) daría el vector (3,4,2,1)
    - rank(a) daría el vector (4,3,1,2)

- colnames (obtención o establecimiento de los nombres de las columnas de un dataframe)
- rownames (obtención o establecimiento de los nombres de las filas de un dataframe)
- levels (obtención de los distintos valores que hay en un vector tipo factor)
- cbind (obtención de un dataframe juntando las columnas de dos dataframes)
- rbind (obtención de un dataframe juntando las filas de dos dataframes)
- merge (obtención de un dataframe a partir de dos dataframes, basándose en una columna común a ambos)
- append (añade elementos a un vector)
- read.table/csv/csv2/delim/delim2 (capturar fichero en un dataframe)
- write.table/csv/csv2 (escribir dataframe en un fichero)
- apply (aplica una función a todas y cada una de las filas o las columnas de un dataframe)
- Reduce (aplica una función a una vector de objetos del siguiente modo: aplica la función al primer y el segundo elemento del vector de objetos; luego aplica la función al resultado del paso anterior y al tercer elemento del vector de objetos; luego al resultado del paso anterior y al cuarto elemento del vector de objetos; y así hasta que todos los elementos del vector de objetos han sido usados; https://blog.zhaw.ch/datascience/r-reduce-applys-lesser-known-brother/)

- bucles ‘for’:
- hacer algo para cada elemento de un conjunto de elementos
- estructura formal:
    for (elemento in conjunto_elementos) {órdenes R}
    - la orden anterior ejecutará las órdenes R dadas entre las llaves una vez para cada uno de los elementos del conjunto de elementos dado entre paréntesis

- **condicional ‘if’:**
- hacer algo sólo si se da una condición determinada
- estructura formal:
    if (condición) {órdenes R}
    - la orden anterior ejecutará las órdenes R dadas entre las llaves sólo si se cumple la condición dada entre paréntesis (es decir, si la condición dada entre paréntesis evalúa a TRUE)
- se puede (o no) dar órdenes para que se ejecuten en caso de que no se cumpla la condición:
    if (condición) {órdenes R1} else {órdenes R2}
    - la orden anterior ejecutará las órdenes R1 dadas entre las llaves si se cumple la condición dada entre paréntesis, y las órdenes R2 dadas entre las llaves si no se cumple la condición dada entre paréntesis
- se puede (o no) dar órdenes alternativas para que se ejecuten en caso de que se de una (o más condiciones diferentes):
    if (condición 1) {órdenes R1} else if (condición 2) {órdenes R2}
    - la orden anterior ejecutará las órdenes R1 dadas entre las llaves si se cumple la condición 1 dada entre paréntesis, y, en caso de que no se cumpla la condición 1, ejecutará las órdenes R2 dadas entre las llaves si se cumple la condición 2 dada entre paréntesis
- un ‘if’ puede llevar asociados cero o más ‘else if’ y cero o un ‘else’

- creación de funciones:

- una función consiste en dar nombre a un conjunto de órdenes R para, de este modo, ejecutar dicho conjunto de órdenes R con sólo llamar a la función
- estructura formal:
  nombre_función <- function(variables_input) {órdenes R}
  - la orden anterior creará la función llamada ‘nombre_función’
  - cuando ejecutemos la orden ‘nombre_función(variables)’, se ejecutarán las órdenes R que había entre llaves en el momento de crear la función, pero con las variables que usemos en este momento de ejecutarla
  - si la función produce algún nuevo objeto y queremos capturarlo cuando la ejecutemos, debemos incluir como última orden R la siguiente orden: return(nuevo_objeto). Como con todo en R, si queremos capturar el nuevo objeto en una variable, deberemos hacer: objeto <- nombre_función(variables)