Sesión 2: Manejo de datos.
Ejercicios resueltos.

Recuerda especificar inicialmente el directorio de trabajo en que quieres trabajar.

1. Construir un factor `x` con los niveles 1, 2, 3, 4 y 5, que tengan las siguientes frecuencias: 10, 20, 30, 40 y 50. Comprobar con `table()` que se ha creado bien.

   Utilizar la función `levels()` para asignar unas etiquetas a cada nivel y forzar el vector `x` que sea un factor ordenado.

   Comprobar el número de niveles utilizando la función `nlevels()`.

   Solución:

   ```
   x <- as.factor(sample(rep(1:5, c(10,20,30,40,50))))
   levels(x) <- c("uno", "dos", "tres", "cuatro", "cinco")
   nlevels(x)
   ```

2. Podemos convertir un factor en vector usual mediante `as.vector()` y podemos forzarlo a que tome valores numéricos. Comprobar el funcionamiento con:

   ```
   x <- factor(c("a","b","a","c","b"))
   y <- as.vector(x)
   y <- as.numeric(x)
   ```

3. Utilizar el siguiente código para construir una matriz simétrica de números enteros:

   ```
   Am <- matrix(sample(1:10,100,re=T),10,10); Ad<-Am+t(Am)
   ```

4. Con los siguientes números: 7.3, 6.8, 0.005, 9, 12, 2.4, 18.9, .9

   a) Calcula la media.
   b) Calcula la raíz cuadrada de los números.
   c) Obtén los números que son mayores que su raíz cuadrada.
   d) ¿Cuántos valores son mayores que 1?
   e) Obtén la raíz cuadrada de los números redondeados con dos cifras decimales.
   f) ¿Cuánto difieren los números redondeados de los originales?

   Solución:

   ```
   dat <- c(7.3, 6.8, 0.005, 9, 12, 2.4, 18.9, .9)
   mean(dat); sqrt(dat); sum(dat>1); dat[dat>sqrt(dat)];
   sqrt(datr<-round(dat,2)); dat-datr
   ```

5. Uso de la función `sample()`. Obtén una muestra de tamaño 5 del un vector 1:20, con probabilidades proporcionales al valor del vector.

   Solución:

   ```
   x <- sample(1:20,5,prob=1:20)
   ```

6. Dados dos vectores `x` e `y` que formarán las coordenadas de unos puntos `(x[i],y[i])`, utilizar el producto exterior para calcular una matriz con las distancias euclídeas entre todos los puntos.

   Solución: `MDE <- sqrt((outer(x,x,"-"))^2 + outer(y,y,"-")^2)`

7. Utilizar la función `write()` para guardar, en otro fichero texto, la matriz creada en el ejercicio anterior. Posteriormente eliminar todos los objetos de la memoria de $\mathbb{R}$ y leer con `scan()` la matriz.

   Solución:

   ```
   write(MDE, file = "mat-dist-eucl.txt", ncolumns = ncol(MDE))
   rm(list = ls())
   scan(file = "mat-dist-eucl.txt") ó
   MDE <- matrix(scan(file = "mat-dist-eucl.txt"),ncol=3)
   ```

8. Sea `m<-matrix(1:12,3,4)`, ¿qué produce `diag(diag(m))`?

   Solución:

   ```
   Una matriz diagonal en la que su diagonal es precisamente
   el vector (1,5,9), la diagonal de la matriz m.
   ```

9. Construye una matriz A cuya diagonal esté constituida por la secuencia de enteros del 1 al 6. Construye otra matriz B de idéntico tamaño en la que la primera fila sean todo 1, la segunda todo 2, y así hasta la última, con todos los elementos igual a 6. Construye un vector b de dimensión 6 con la diagonal de la matriz B.

   Solución:

   ```
   A <- diag(1:6)
   B <- matrix(rep(1:6,6),ncol=6)
   b <- diag(B)
   ```

   Realiza las siguientes operaciones:

   a) $b'(A+B)$ **Solución**: `t(b)%*%(A+B)`

   b) $b'(AB)$ **Solución**: `t(b)%*%(A%*%B)`

   c) $A - 1B$ **Solución**: `A - rep(1,6)%*%B` ó `A - 1*B`

   d) Resuelve $(A+B)x = b$ **Solución**: `solve(A+B,b)`

   e) Selecciona la segunda fila de B, llámala $b2$ y calcula $b'b2$ y $b2'A$.
   Solución: `b2 <- B[2,]; t(b)%*%b2; t(b2)%*%A`

10. La funciones que dispone $\mathbb{R}$ para la ordenación son las siguientes:

    - `sort(x, partial=NULL, na.last=NA)`: si `partial=NULL`, obtenemos un vector con los valores de `x` ordenados de forma ascendente. Si partial no es `NULL`, puede contener índices de elementos de `x` que serán ordenados en su posición correcta. Cualquier valor menor que dichos valores tendrá índice menor y cualquier valor mayor quedará parcialmente ordenado con índice mayor. Si `na.last=NA`, los datos faltantes se eliminan del vector resultante ordenado. Si `na.last=TRUE`, los valores `NA` se colocan al final, si `na.last=FALSE`, los valores `NA` se colocan al principio.
    - `rev(x)`: produce una reversión de los argumentos.
    - `order(...)`: proporciona un vector de índices (numérico) permutado de tal forma que sus argumentos quedan ordenados de forma ascendiente. Si disponemos varios vectores, en caso de empate en el primer vector se utilizan los valores del segundo. Si persiste el empate se miraría un tercero, y así sucesivamente. Los valores `NA` se consideran, en este caso, como los valores más grandes.
    - `rank(x)`: proporciona el rango muestral de los valores de `x`. En caso de empates el rango se promedia. Los valores `NA` no se admiten en esta función.
    - `unique(x)`: devuelve un vector como `x`, pero con los elementos duplicados eliminados.
    - `duplicated(x)`: esta función determina que elementos de un vector están duplicados anteriormente y devuelve un vector lógico.

    A la vista de estas funciones, comenta que diferencia crees que habrá entre los comandos `rev(sort(month.name))` y `sort(month.name)`. ¿Cual será el resultado de la expresión `is.numeric(sort(month.name))`?

    Solución: `producirán un vector con los nombres de los meses (en inglés), pero en orden inverso.`
    `Como month.name es un vector de caracteres, no puede ser numérico aunque lo ordenemos.`

    ¿Qué diferencia hay entre `rank(month.name)` y `order(month.name)`?

    Solución: `ahora si son vectores numéricos. Tras ordenar alfabéticamente los meses, el primero nos da para cada mes su correspondiente ordenación (January queda el quinto), mientras que el segundo indica para el vector de meses ordenados que posición ocupaban en el vector sin ordenar (April, el primero ordenado, es el cuarto mes).`

    Si consideramos el vector `x<-c(1,2,3,2,4,1)`, ¿qué resultado producen `duplicated(x)` y `unique(x)`?

    Solución: `nos dará un vector de índices lógico con T en los que se vayan repitiendo (F F F T F T) y un vector con aquellos valores no repetidos (1 2 3 4).`