TRABAJO PRÁCTICO 3 - R Studio

```
#ejercicio 1
N = 10                  #cargo la cantidad de datos en el entorno
1:N-1                   #me muestra del 1 al 10, y a cada componente le resta 1 (como si fuera (1:N)-1)
1:(N-1)                 #me muestra desde 1 hasta N-1 que es 9

#ejercicio 2
#a)
#i
1:10                    #Devuelve una secuencia de elementos (vector) del 1 al 10 pero no se la asigna a ninguna variable

#ii
3:5.5                   #Devuelve un vector de números enteros del 3 al 5 (no tiene en cuenta el 5.5 sino 5 porque va de uno en uno)

#iii
2.8:10.2                #Devuelve un vector del 2.8 al 9.8 porque va de uno en uno (no llega al 10.2)

#iv
seq(1, 30, by=2)        #Devuelve un vector que inicia en 1 y termina en 30, y va de dos en dos: seq(from = , to = , by = , length = )

#v
seq(10, -6, -2)         #Devuelve un vector que inicia en 10 y termina en -6, y va de -dos en -dos

#vi
seq(10, -6, 2)          #Error: el signo del "by" es incorrecto. No se puede ir de 10 a -6 de dos en dos

#vii
rep('a', 5)             #Devuelve un vector que repite 'a' cinco veces: rep(x, times = 1, length.out = NA, each = 1)

#viii
seq(10, 15, length = 6)     #Devuelve una secuencia desde 10 hasta 15, con un largo de 6 elementos (aunque por default se haría igual de 6)
```

```
seq(10,15)

#ix
seq(10, 15, length = 4)        #Devuelve una secuencia desde 10 hasta 15, pero como el largo es 4, los numeros son reales

#x
seq(10, 15, len = 4)           #Mismo resultado ya que admite abreviaciones ("partial matching" diap 10 clase 2)

#xi
rep(c(10, 12), 3)              #Devuelve vector que repite la secuencia 10 12 tres veces.
rep((10:12), 3)                #Si pongo rep((10:12), 3) repite la secuencia 10 11 12 tres veces

#xii
rep(c(5, 8), c(2, 3))          #Devuelve vector que repite el 5 dos veces y el 8 tres veces, porque c(5,8) seria 'x' y c(2,3) seria 'times'

#xiii
rep(1:4, c(2, 1, 5, 2))

#Devuelve vector que repite el 1 dos veces, el 2 una vez, el 3 cinco veces y el 4 dos veces, porque la 'x' es la secuencia 1 2 3 4 y 'times' es el
c()

#xiv
double(20)                     #Devuelve un vector de 20 elementos iguales a cero: double(length = 0)

#b)
V1 = 10:18                             #Vector con secuencia del 10 al 18 - ventana environment
V1 = c(10,11,12,13,14,15,16,17,18)     #Vector con elementos 10 11 13 14 15 16 17 18
V1 = seq(10,18)                        #Vector con secuencia del 10 al 18 con by = 1
V1 = rep(10:18)                        #Vector que repite la secuencia del del 10 al 18 una vez

V2 = c(1,4,7,10,13,16,19,22,25)        #Vector con elementos 1 4 7 10 13 16 19 22 25 - ventana environment
```

```
V2 = seq(1,25,3)                    #Vector con secuencia del 1 al 25 que va de tres en tres
V2 = seq(from = 1, by = 3, length = 9)    #Vector que va desde el 1, de tres en tres, con un largo de 9 elementos

#c)
V1 = c(1,2,4,7,10); length(V1)      #Muestra el tamaño (largo) del vector V1
length(V1) = 20; V1
```

#Define el tamaño (largo) del vector V1 en 20 elementos, pero el vector V1 solo tiene 5 elementos, entonces, como no hay repitencia, completa el vector con NAs

```
length(V1) = 2; V1
```

#Define el tamaño (largo) del vector V1 en 2 elementos, pero como el vector V1 tiene 5 elementos, solo muestra los primeros 2

```
#d)
set.seed(123)
```

#Establezco semilla aleatoria inicial para garantizar que la secuencia de valores aleatorios será la misma cada vez que se ejecute el código

```
x = sample(1:100,20); x
```

#Genero una muestra aleatoria de 20 números enteros entre 1 y 100, sin repetición y se la asigno al vector x - ventana environment

```
x[10]                               #Muestro el elemento que se encuentra en la posición 10 del vector x
x[22]
```

#Me devuelve un NA ya que no hay ningún elemento en la posición 22 porque el vector solo tiene 20 elementos

```
x[c(2,8,4,10:12)]                   #Muestra los elementos de las posiciones 2 8 4 10 y de ahí hasta la 12
x[10:15]                            #Muestra los elementos desde la posición 10 hasta la 15
x[-5]                               #Muestra el vector x sin el elemento que se encuentra en la quinta posición
```

```
x[-(5:15)]                          #Muestra el vector x sin los elementos de la quinta a la quinceava posición
x%%2 == 0                           #Devuelve un vector booleano que marca a los pares con TRUE y a los impares con FALSE
x[x%%2 == 0]                        #Devuelve un vector reducido con los elementos del vector x que son pares (true)
sum(x[x%%2 == 0])                   #Suma los elementos del vector x que son pares
sum(x%%2 == 0)                      #Transforma los TRUE y FALSE en unos y ceros, y cuenta los unos (true)

nombres = c("juan","ana","maria","luis","pedro")    #Al vector "nombres" le asigno los datos en c() - ventana environment
sexo = c("H","M","M","H","H")                       #Al vector "sexo" le asigno los datos en c() - ventana environment
nombres[sexo == "M"]                                #Devuelve los nombres que se encuentran en las posiciones "M" del vector sexo
sum(sexo == "M")                                    #Sumo la cantidad de elementos del vector sexo que son "M"

#ejercicio 3

x = sample(0:10,20,replace = TRUE);x

#Ahora modifique el vector x con una muestra aleatoria de 20 números entre 0 y 10, con repetición - ventana environment

#a)
soloPARES = x[(x%%2 == 0)];soloPARES          #Al vector soloPARES le asigno los valores pares del vector x - ventana environment

#b)
subset(x, x%%4 == 0 & x!=0)         #Esta es una forma de hacerlo, pero es mas larga
b = subset(x, x%%4==0 & x!=0)       #Hago una submuestra con los elementos que cumplen esas condiciones y se la asigno a la variable b
b[1]                                #Busco el elemento en la primera posicion de ese vector porque ya se que va a cumplir la condicion

x[match(TRUE, x%%4 == 0 & x!= 0)]         #Busco en el vector "x" el primer valor (con match) que cumpla la condicion (TRUE)

#c)
match(TRUE, x%%4 == 0 & x!= 0)            #El match solo me da la primera posicion en la que se cumple la condicion en el vector "x"

#d)
```

```
x%%2 == 0                       #Determino elementos pares en el vector x
sum (x%%2 == 0)                 #Sumo cantidad de elementos pares de x

#e)
x[x%%3 == 0]                                #Devuelve vector con multiplos de 3 en x
x[x%%3 == 0 & x%%2 != 0]                    #Devuelve vector con multiplos de 3 y no multiplos de 2 en x
sum (x%%3 == 0 & x%%2 !=0 & x!=0)           #Devuelve la suma de los elementos de x que son multiplos de 3 y no de 2, y que son distintos a 0

#f)
all(x%%2 == 0)                  #NO todos los elementos de x son PARES

#g)
any(x%%10 == 0)                 #Hay AL MENOS un elemento multiplo de 10

#h)
x[x%%2 != 0] = 0                #Reemplazo elementos impares con ceros
x[which(x%%2!=0)] = 0

#ejercicio 4 (matrices)

#matriz dada
set.seed(0)
a = sample(1:100, 12)           #Genero un vector "a"
matriz = matrix(a,3)            #Genero una matriz no simétrica con el vector a llamada "matriz"

#a)
min(matriz)                                 #Devuelve el valor mínimo de toda la matriz
max(matriz)                                 #Devuelve el valor máximo de toda la matriz
mean(matriz)                                #Devuelve el promedio de toda la matriz
c = c(min(matriz),max(matriz),mean(matriz))   #Calculo una nueva variable c que me calcule el mínimo, el máximo y el promedio de la matriz
```

#La solución planteada también funciona para arreglos ya que estos son vectores (arreglos de una dimensión) y matrices (arreglo de dos dimensiones)

#b)
```
apply(matriz, 1, sum)                    #Devuelve la suma de las componentes de cada fila de la matriz
```

#c)
```
min(apply(matriz, 1, sum))               #Devuelve la menor de las sumas de las componentes de cada fila de la matriz
```

#d)
```
nrow(matriz)                             #Devuelve cantidad de filas de la matriz "matriz"
ncol(matriz)                             #Devuelve cantidad de columnas de la matriz "matriz"
nrow(matriz) == ncol(matriz)             #Devuelve FALSO ya que la cantidad de filas no es igual a la cantidad de columnas
matriz == t(matriz)                      #Hace la traspuesta pero en este caso no puede porque no es simétrica
nrow(matriz) == ncol(matriz) & matriz == t(matriz)          #Tienen que cumplirse estas dos condiciones (me da una matriz con T y F)
nrow(matriz) == ncol(matriz) & all(matriz == t(matriz))     #Con el "all" veo que todas las componentes la cumplan
```

#si el nro de filas es igual al nro de columnas hago la segunda parte del &, sino me da FALSE (me da T o F)
#si la primera parte es TRUE, me fijo si la matriz es igual a la traspuesta (me da matriz lógica)
#si es igual a la traspuesta, veo si todos los componentes cumplen la condición (en la diagonal tiene que haber TRUEs y en el resto FALSEs, y me da un valor T o F final)
#con el all ves si tu matriz cumple la condición de la matriz traspuesta booleana

#ejercicio 5

```
set.seed(0)
m = sample(1:20, 9);m          #Genero un vector aleatorio
M = matrix(m,3);               #Lo convierto en una matriz simétrica

sum(M)                         #Devuelve la sumatoria de todos los elementos de la matriz M
```

#ejercicio 6

```
data("ToothGrowth")          #Base de datos que muestra los efectos de la vitamina C en el crecimiento de los dientes de los cobayos
head(ToothGrowth)
View(ToothGrowth)
```

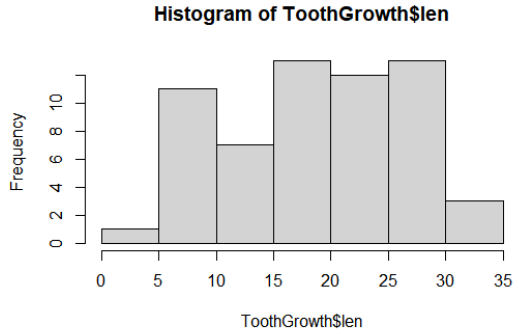
#a)

```
tapply(ToothGrowth$len[ToothGrowth$dose==0.5], ToothGrowth$supp[ToothGrowth$dose==0.5], mean)
```

#Devuelve el tamaño promedio de los dientes del grupo de cobayos tratados con una dosis de 0.5mg agrupados según forma de administración

#b)

```
hist(ToothGrowth)            #Error en hist.default(ToothGrowth): 'x' debe ser numérico
hist(ToothGrowth$len)        #Muestra un histograma del crecimiento de los dientes de los cerdos
```



#ejercicio 7

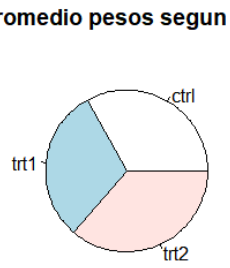
data("PlantGrowth")

#Base de datos que contiene datos de experimentos sobre el crecimiento de ciertas plantas en situaciones de control y de dos tratamientos distintos

head(PlantGrowth)
View(PlantGrowth)

#a)
tapply(PlantGrowth$weight, PlantGrowth$group, mean) #Promedio de peso de cada grupo

#b)
pie(tapply(PlantGrowth$weight, PlantGrowth$group, mean), main = "Promedio pesos segun tipo") #Gráfico de torta



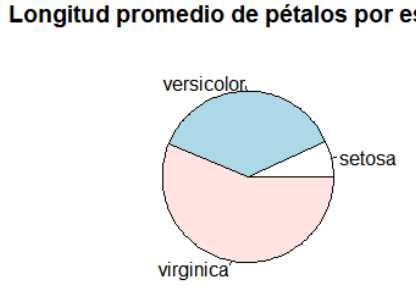
#ejercicio 8

```
data("iris")              #Base de datos que contiene distintas características de las plantas pertenecientes a esa familia
head(iris)
View(iris)

#a)
longPetalos = tapply(iris$Petal.Width, iris$Species, mean); longPetalos

#Genero nueva variable con los promedios de longitud de petalo agrupado por especie

#b)
pie(longPetalos, main = "Longitud promedio de pétalos por especie")              #Gráfico de torta
```



#ejercicio 9

```
data("airquality")    #Base de datos que muestra las mediciones diarias de calidad del aire en Nueva York, de mayo a septiembre de 1973
head(airquality)
View(airquality)
```

#a)
airquality$Temp>75 #Devuelve un vector con TRUEs y FALSEs según si cada elemento cumple o no la condición
sum(airquality$Temp>75) #Devuelve la cantidad de TRUEs, es decir, cuantas veces la temperatura fue mayor a 75 grados

#b)
subset(airquality, airquality$Month == 6) #Subconjunto con datos correspondientes al mes de junio

#c)
subset(airquality, airquality$Day <= 15 & airquality$Month == 8) #Subconjunto con datos correspondientes a los primeros 15 días de agosto

#d)
tapply(airquality$Temp, airquality$Month, mean) #Promedio de temperaturas agrupado por mes

#e)
hist(airquality$Temp, main = "FRECUENCIA DE TEMPERATURAS", xlab = "Temperatura", ylab = "Frecuencia")

#Gráfico de frecuencia de valores de temperaturas