

# Manejo de Vectores numéricos en R

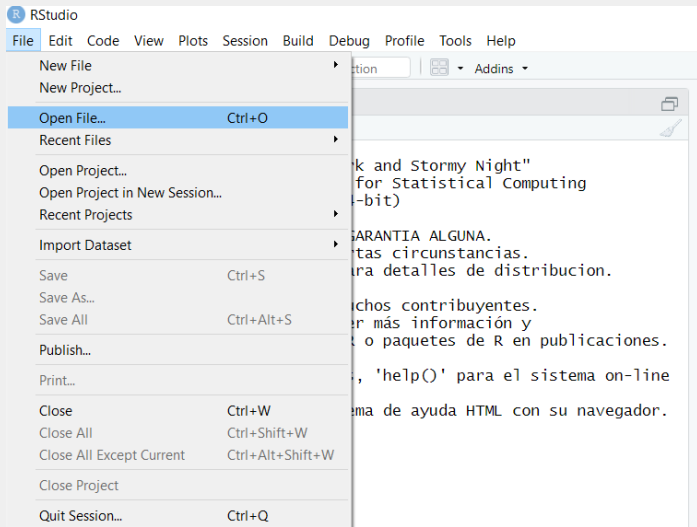
## Sesión 2

Natalie Julian - [www.nataliejulian.com](http://www.nataliejulian.com)

Estadística UC y Data Scientist en Zippedi Inc.

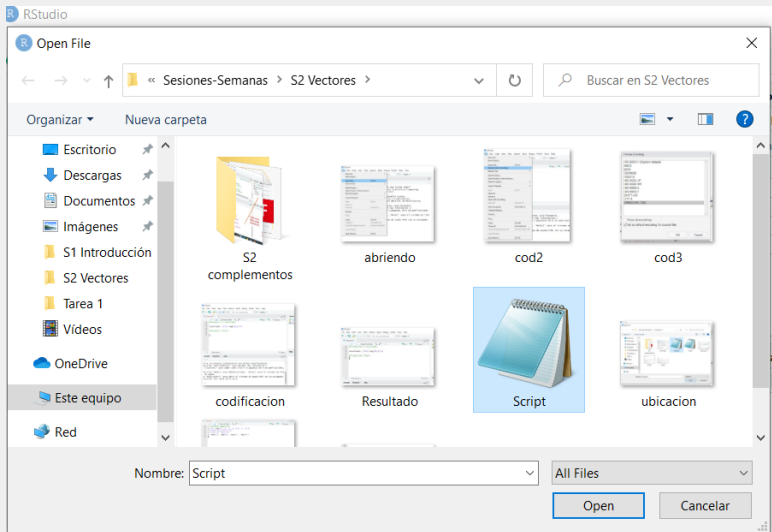
# Abriendo un script

Una vez que seleccionamos la ubicación y guardamos un script, podemos cerrar RStudio y luego volver a retomar nuestro trabajo y abrir dicho script:



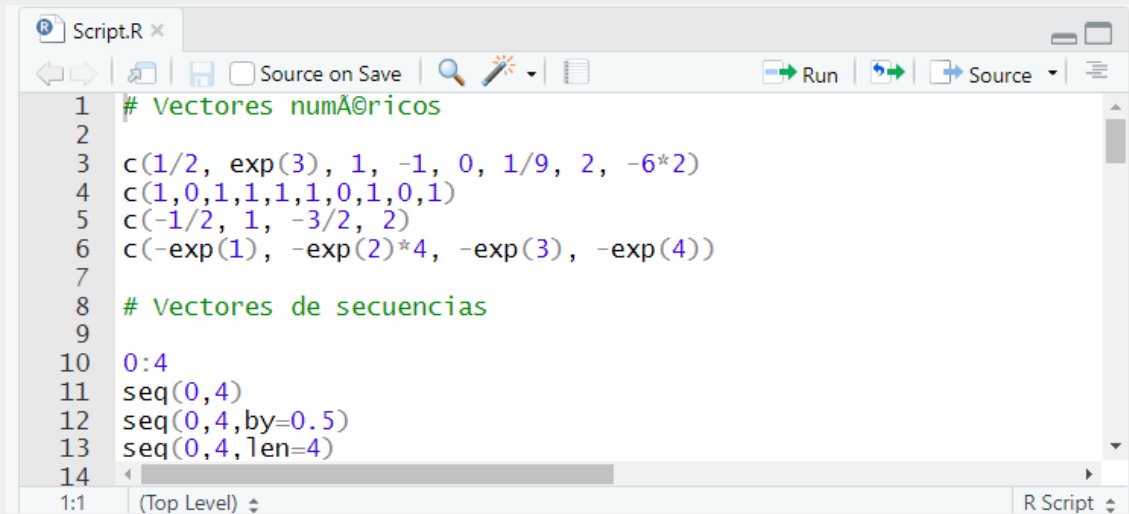
# Seleccionar ubicación del archivo

Al hacer click en `Open file` se abrirá una ventana donde debemos seleccionar nuestro archivo:



# ¡Ups! Problemas de codificación

A veces cuando abrimos un script hay ciertos caracteres que no se leen bien:



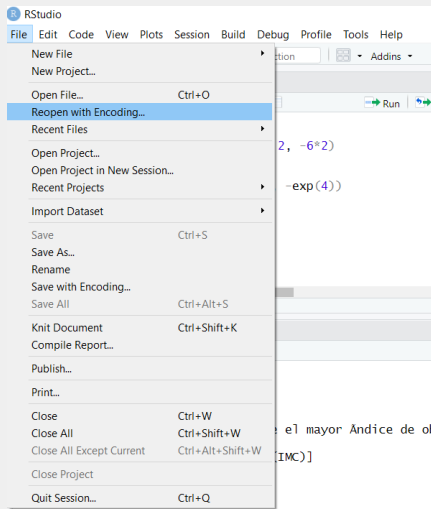
The screenshot shows an R script editor window titled 'Script.R'. The script contains several lines of R code. Lines 1-5 define vectors of numbers, including fractions and mathematical expressions. Lines 8-13 define sequences. The code is as follows:

```
1 # Vectores numéricos
2
3 c(1/2, exp(3), 1, -1, 0, 1/9, 2, -6*2)
4 c(1,0,1,1,1,1,0,1,0,1)
5 c(-1/2, 1, -3/2, 2)
6 c(-exp(1), -exp(2)*4, -exp(3), -exp(4))
7
8 # Vectores de secuencias
9
10 0:4
11 seq(0,4)
12 seq(0,4,by=0.5)
13 seq(0,4,len=4)
14
```

The status bar at the bottom indicates the cursor is at line 1:1, column 1, and the file is an R Script.

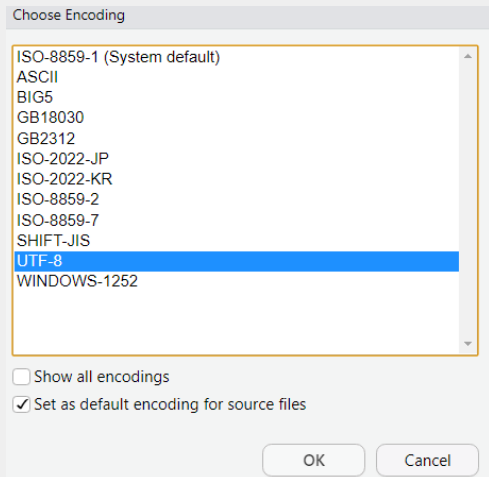
Estos casos se solucionan cambiando la codificación de lectura del script.

# Cómo cambiar la codificación de lectura de un script



# Seleccionar codificación por defecto

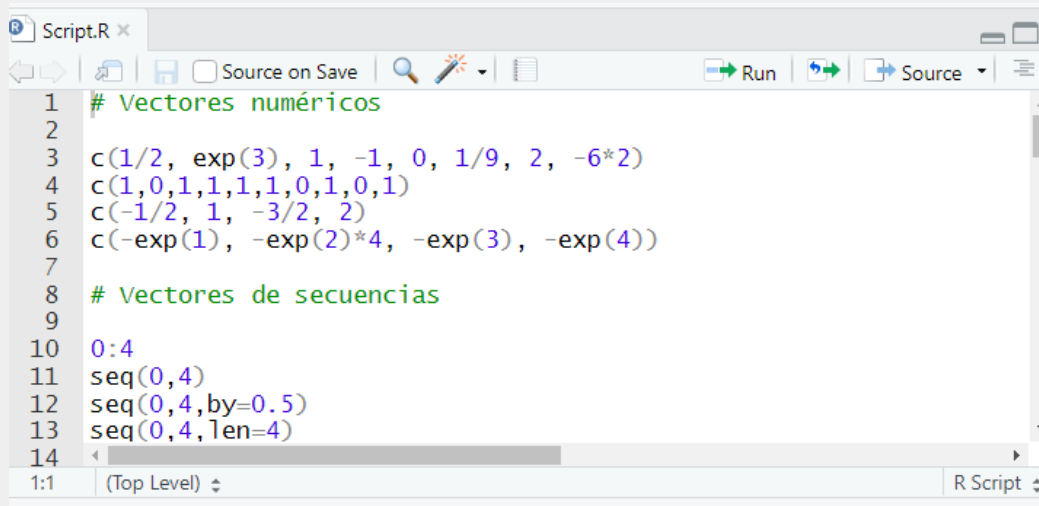
Esta elección dependerá de su computador. Por ejemplo, en este caso, elegiré UTF-8:



Y (de manera opcional) podemos seleccionarla por defecto con Set as default encoding for source files.

# Vista del script

Ahora sí podemos leer correctamente nuestro script, reconociendo las tildes:



```
1 # Vectores numéricos
2
3 c(1/2, exp(3), 1, -1, 0, 1/9, 2, -6*2)
4 c(1,0,1,1,1,1,0,1,0,1)
5 c(-1/2, 1, -3/2, 2)
6 c(-exp(1), -exp(2)*4, -exp(3), -exp(4))
7
8 # Vectores de secuencias
9
10 0:4
11 seq(0,4)
12 seq(0,4,by=0.5)
13 seq(0,4,len=4)
14
```

Un vector es una unidad que contiene de manera secuencial varios elementos del mismo tipo.

En R los vectores se definen con la siguiente estructura:

```
c(elemento1, elemento2, elemento3,...)
```



# VECTORES NUMÉRICOS

# Vectores numéricos

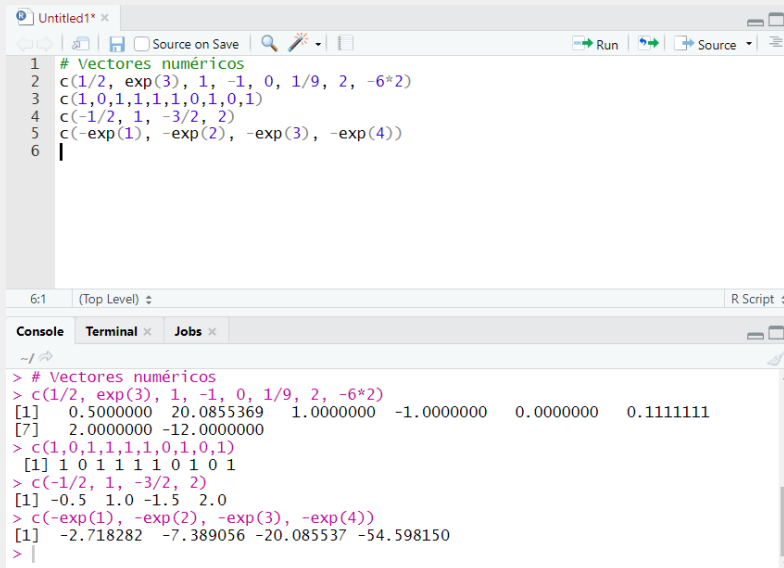
Algunos ejemplos de vectores numéricos:

$c(1/2, \exp(3), 1, -1, 0, 1/9, 2, -6*2)$

$c(1, 0, 1, 1, 1, 1, 0, 1, 0, 1)$

$c(-1/2, 1, -3/2, 2)$

$c(-\exp(1), -\exp(2), -\exp(3), -\exp(4))$



The screenshot shows an RStudio interface with a script editor and a console. The script editor contains R code for creating and displaying vectors. The console shows the execution of this code, including the resulting vectors and their dimensions.

```
1 # Vectores numéricos
2 c(1/2, exp(3), 1, -1, 0, 1/9, 2, -6*2)
3 c(1,0,1,1,1,1,1,0,1,0,1)
4 c(-1/2, 1, -3/2, 2)
5 c(-exp(1), -exp(2), -exp(3), -exp(4))
6 |
```

6:1 (Top Level) R Script

Console Terminal Jobs

```
> # Vectores numéricos
> c(1/2, exp(3), 1, -1, 0, 1/9, 2, -6*2)
[1] 0.5000000 20.0855369 1.0000000 -1.0000000 0.0000000 0.1111111
[7] 2.0000000 -12.0000000
> c(1,0,1,1,1,1,1,0,1,0,1)
[1] 1 0 1 1 1 1 1 0 1 0 1
> c(-1/2, 1, -3/2, 2)
[1] -0.5 1.0 -1.5 2.0
> c(-exp(1), -exp(2), -exp(3), -exp(4))
[1] -2.718282 -7.389056 -20.085537 -54.598150
> |
```

# Vectores de secuencias

Corra las siguientes líneas de código. ¿Qué especifica cada argumento?

```
0:4
```

```
seq(0,4)
```

```
seq(0,4,by=0.5)
```

```
seq(0,4,len=4)
```

# Vectores de secuencias

```
0:4
```

```
[1] 0 1 2 3 4 Una secuencia simple de 0 a 4, de uno en uno.
```

```
seq(0,4)
```

```
[1] 0 1 2 3 4 Una secuencia simple de 0 a 4, de uno en uno.
```

```
seq(0,4,by=0.5)
```

```
[1] 0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 Una secuencia de 0 a 4, de 0,5 en 0,5.
```

```
seq(0,4,len=4)
```

```
[1] 0.0000000 1.333333 2.666667 4.000000 Una secuencia de 0 a 4 de largo 4 o de 4 elementos.
```

# Vectores de repeticiones

Corra las siguientes líneas de código. ¿Qué especifica cada argumento?

```
rep(1, 5)
```

```
rep(1:5, each=2)
```

```
rep(c(1, 5), c(3, 7))
```

```
rep(c(1, 5), len=9)
```

# Vectores de repeticiones

```
rep(1, 5)
```

```
[1] 1 1 1 1 1 Repite 5 veces el valor 1
```

```
rep(1:5, each=2)
```

```
[1] 1 1 2 2 3 3 4 4 5 5 Repite la secuencia de 1 a 5, donde cada elemento lo repite 2 veces
```

```
rep(c(1, 5), c(3, 7))
```

```
[1] 1 1 1 5 5 5 5 5 5 Repite los números 1 y 5, donde el 1 lo repite 3 veces y el 5 lo repite 7 veces
```

```
rep(c(1, 5), len=9)
```

```
[1] 1 5 1 5 1 5 1 5 1 Repite el vector (1,5) hasta que se alcance un largo 9 (9 elementos)
```

# Vector de vectores

Es posible crear un vector a partir de otros vectores, y así alternar el uso de los comandos `seq()` y `rep()`. Vea los siguientes ejemplos:

```
c(rep(c(2,3),3),0,1,-2,seq(15,17,by=0.5))  
c(seq(0,5,by=1),rep(c(6,7,8),len=8),1)
```



# Asignando a un objeto un vector

Tal como le asignamos valor a un objeto con números, podemos asignarle a un objeto un vector.

# Ejemplo

Hace 6 meses Leo comenzó a recibir mesadas y las lleva anotando (500 pesos, 1000 pesos, 350 pesos, 600 pesos, 450 pesos y 760 pesos). Sin embargo, también sabe que todos los meses gasta 150 pesos en stickers. ¿Cuánto dinero restante le quedó en cada mes?

## Ejemplo

Hace 6 meses Leo comenzó a recibir mesadas y las lleva anotando (500 pesos, 1000 pesos, 350 pesos, 600 pesos, 450 pesos y 760 pesos). Sin embargo, también sabe que todos los meses gasta 150 pesos en stickers. ¿Cuánto dinero restante le quedó en cada mes?

```
#Creamos un objeto con las mesadas de Leo:
```

```
mesada<-c(500, 1000, 350, 600, 450, 760)
```

```
#Le restamos los 150 pesos que gasta en stickers:
```

```
mesada-rep(150, 6)
```

```
[1] 350 850 200 450 300 610
```

```
mesada-150 #son equivalentes
```

```
[1] 350 850 200 450 300 610
```

# Funciones aplicables a vectores numéricos

Función	Descripción
<code>length()</code>	Largo del vector
<code>sum()</code>	Suma de los elementos del vector
<code>min()</code>	Mínimo de los elementos del vector
<code>which.min()</code>	En qué elemento del vector se alcanza el mínimo
<code>max()</code>	Máximo de los elementos del vector
<code>which.max()</code>	En qué elemento del vector se alcanza el máximo
<code>mean()</code>	Media de los elementos del vector
<code>median()</code>	Mediana de los elementos del vector
<code>sd()</code>	Desviación estándar de los elementos del vector
<code>var()</code>	Varianza de los elementos del vector
<code>order()</code>	Entrega un vector de índices para ordenar el vector de manera creciente
<code>round()</code>	Entrega el vector redondeado

*También es posible aplicarle operaciones matemáticas a los vectores, como las funciones que ya vimos, `log()`, `exp()`, `**`, `sqrt()`, `sin()`, entre otras.*

# Extraer elementos de un vector

Podemos extraer ciertos elementos de un vector:

```
mesada[1] #Extrae el primer elemento del vector  
[1] 500
```

```
mesada[-1] #Extrae todos los elementos excepto el primero  
[1] 1000 350 600 450 760
```

```
mesada[length(mesada)] #Extrae el último elemento del vector  
[1] 760
```

```
mesada[1:3] #Extrae los primeros tres elementos  
[1] 500 1000 350
```

```
mesada[order(mesada)] #Extrae los elementos ordenados de menor a mayor  
[1] 350 450 500 600 760 1000
```

```
mesada[order(mesada, decreasing=TRUE)] #de mayor a menor  
[1] 1000 760 600 500 450 350
```