GUIA TEÓRICO PRÁCTICA 1:

Primera Parte: Introducción al R

1. Iniciando R

Una vez instalado hay que hacer un doble click en el ícono de R (en Unix/Linux, se escribe R desde el símbolo de comandos (command prompt)). Cuando R se inicia, aparece la ventana del programa "*Gui*" (graphical user interface) con un mensaje de apertura

R Version 2.13.1 (2011-07-8)
Copyright (C) 2011 The R Foundation for Statistical Computing
ISBN 3-900051-07-0

R es un software libre y viene sin GARANTIA ALGUNA.
Usted puede redistribuirlo bajo ciertas circunstancias.
Escriba 'license()' o 'licence()' para detalles de distribucion.

R es un proyecto colaborativo con muchos contribuyentes.
Escriba 'contributors()' para obtener más información y
'citation()' para saber cómo citar R o paquetes de R en publicaciones.

Escriba 'demo()' para demostraciones, 'help()' para el sistema on-line de ayuda,
o 'help.start()' para abrir el sistema de ayuda HTML con su navegador.
Escriba 'q()' para salir de R.

Debajo del mensaje de apertura en la Consola de R se encuentra el "prompt" que es el símbolo **>** ("mayor").

La mayoría de las expresiones en R se escriben directamente a continuación del "prompt" en la Consola de R.

2. Lenguaje.

2.1 Operaciones básicas.

Realice las siguientes evaluaciones

```
> 3 + 7
[1] 10

> 8*39
[1] 312
```

La suma (+) y la multiplicación (*) son las operaciones aritméticas más simples que pueden hacerse en R

```
> 1:5                 #El operador dos puntos : permite obtener sucesiones
[1] 1 2 3 4 5         #El [1] al comienzo es el índice del primer elemento de la fila.

> 1:30
[1]  1  2  3  4  5  6  7  8  9 10 11   # Cuando la salida es un vector largo y
```

```
[12] 12 13 14 15 16 17 18 19 20 21 22  # hay más de una fila cada una está [23]
23 24 25 26 27 28 29 30       #precedida por el índice del primer valor de la fila.
```

2.2 Sintaxis

```
> 3 +                 7     #La mayoría de los espacios son ignorados.
[1] 10
```

No se pueden poner espacios entre el “menor” y el “menos” en el **operador <- de asignación**

2.3 Se distinguen mayúsculas de minúsculas

```
> nuevo.vector <- 1:5
> Nuevo.vector
Problem: Object "Nuevo.vector" not found
Use traceback() to see the call stack
```

3. Vectores

3.1 La función c concatena objetos en un **vector**.

```
> x <- c(1,2,3,4,5,6)          #enteros de 1 a 6
> x <- 1:6                     # lo mismo
```

3.2 Sucesiones

La función **seq** produce secuencias equiespaciadas

```
> x <- seq(1,6)                # lo mismo
> x <- seq(1,6,0.5)            # y ahora que obtiene?
```

Obtenga una sucesión decreciente.

```
> x <- rep(9.1,3)              # resulta (9.1,9.1,9.1)
> x <- rep(c(1,2),4)           # resulta (1,2,1,2,1,2,1,2)
> x <- c(99,rep(1,3))          # resulta (99,1,1,1)
```

3.3 Vector Lógico

Uniendo **valores lógicos** (T o F) se obtiene un vector logico.

```
> x <- c(T,F,F,T)
> x <- rep(c(T,F),6)
```

3.4 Vector de caracteres

Uniendo **cadenas de caracteres** se obtiene un vector de caracteres (character vector).

```
> x <- "Usted cursa análisis de datos."   # vector de caracteres de longitud 1
> x <-  c("muy caliente","tibio","frío")  # vector de caracteres de longitud 3
```

3.5 Generación de números al azar

```
>   y <- runif (10) # qué obtiene? Utilice el help.
```

Repita varias veces

```
>   runif(10)
```

¿obtiene los mismos valores?

Repita varias veces

```
 > set.seed(87) # fija la semilla del generador de números al azar
 >   runif(10)
```

¿obtiene los mismos valores?

4.1 Gráficos simples

```
> x <- runif(100)
> hist(x)
> y <-  2*x
> plot(x,y)

> y <-  2*x + runif(100)
> plot(x,y)

> boxplot(x)
> boxplot(y)
```

4.2 Gráficos paralelos

```
> par(mfrow=c(1,2))
> y <-  2*x
> plot(x,y)

> y <-  2*x + runif(100)
> plot(x,y)

> boxplot(x)
> boxplot(y)

> apply(cbind(x,y),2,boxplot)  #compare con el gráfico anterior
```

5. Acceso a las componentes de un vector.

Los corchetes pueden utilizarse para acceder a cada elemento de un vector. Los elementos pueden designarse por su posición.

5.1 Acceso mediante un vector de valores enteros

```
> x <- runif(100)  # generamos el vector
> x [7]             # muestra el 7mo. elemento de x.
> x [2:10]         # muestra los elementos 2,3,...,10 de x
> x [4] <- 10       # pone al 4to. elemento en 10
```

La expresión dentro del corchete puede ser un vector ó un único número, indicando la posición de los elementos que desea extraer del vector.

```
> x [7:10] <- 0      # asignación en elementos seleccionados
```

En la última expresión ("`x[7:10] <- 0`") ambos lados tienen diferente longitud. En general cuando los dos lados de una asignación tienen distinta longitud R llenará el lado izquierdo repitiendo el lado derecho tantas veces como sea necesario. Si la longitud del lado izquierdo no es un múltiplo de la longitud del derecho R devuelve un **mensaje de advertencia** (warning messages).

```
>  x <- runif(10)
> x [1:10] <- c (1,2)
> x
 [1] 1 2 1 2 1 2 1 2 1 2

> x [1:9] <- c (3,4)
Warning messages:
  Replacement length not a multiple of number of elements
    to replace in: x[1:9] <- c(3, 4)

>x
 [1] 3 4 3 4 3 4 3 4 3 2
```

5.2. Acceso a los elementos de un vector utilizando valores lógicos (T o F).

La sintáxis es "x[a]" donde a es un vector lógico de igual longitud que x. El resultado es un vector que contiene aquellos elementos de x que tienen en x la misma posición que las T's, se omiten los valores donde se encuentran las F´s

```
> x <- 1:4
> x [ c (T,F,F,T) ]
[1] 1 4

 > x <-c("a","b","c","d","e","f")
> x[c(T,F,T,F,T,F)]
[1] "a" "c" "e"

> x[!c(T,F,T,F,T,F)]
[1] "b" "d" "f"
```

Los operadores "==" (igual a), "!=" (distinto a), "<", ">",
"<=" (menor o igual), y ">=" (mayor o igual) darán T's o F's si la condición se satisface ó no se satisface respectivamente. Esto se extiende a los operadores lógicos "&" (y), "!" (no), "|" (o incusivo).

```
> x <- c (1,2,3,4)
> x == 2
[1] FALSE  TRUE FALSE FALSE

> x >= 3
[1] FALSE FALSE  TRUE  TRUE

> (x >= 2)&(x != 3)
[1] FALSE  TRUE FALSE  TRUE

> (x==1)|(x==4)
[1]  TRUE FALSE FALSE  TRUE
```

Los valores lógicos permiten el acceso a los elementos de un vector que cumplen con una condición determinada

```
> x <- c(3,5,2,7)
> x >= 3
[1]  TRUE  TRUE FALSE  TRUE

> x[x >= 3]
[1] 1 2 4
```

Otra forma

```
> which(x>=3)
[1] 1 2 4

> which(x==2)
[1] 3
```

Estos valores lógicos facilitan la recodificación de los datos

```
> x  <-  c (1,2,2,1,3,3,1,2)
> x [ x==3 ]  <- 4
> x
[1] 1 2 2 1 4 4 1 2

> x [ x==2 ] <- NA          # NA indica valor faltante (Not Available)
> x
[1]  1 NA NA  1  4  4  1 NA
```

5.3. Números negativos para omitir elementos de un vector

```
> x <- c("a","b","c","d","e","f")
> x[-1]
[1] "b" "c" "d" "e" "f"

> x[-2]
[1] "a" "c" "d" "e" "f"

> x[-(1:3)]
[1] "d" "e" "f"
```

No debe mezclar subíndices positivos con negativos

```
> x <- c("a", "b", "c", "d", "e", "f")
> x[c(-2, 5)]
Problem in x[c(-2, 5)]: Only 0's may be mixed with negative subscripts

  > x[-1:3]
   Error in x[-1:3]: Only 0's may be mixed with negative subscripts
```

6. Reordenamiento de las componentes de un vector

```
> x <- c("muy caliente", "tibio", "frio")
> x[c(2, 3, 1)]
[1] "tibio"    "frio"     "muy caliente"
```

7. Operaciones con vectores

Cuando los vectores tienen la misma longitud las operaciones básicas "+","-","*","^" se realizan componente a componente:

```
> c(1,2,3)*c(2,3,1)
[1] 2 6 3

> (1:5)^(2:6)
[1]     1     8    81  1024 15625
```

Cuando los vectores no tienen la misma longitud las operaciones básicas "+","-","*","^" también se realizan componente a componente y para ello el vector de menor longitud es repetido hasta obtener la longitud del vector de mayor longitud

Por ejemplo si "x" es un vector de longitud 5 y para realizar la operación "x^2", R expande "2" en "c(2,2,2,2,2)" y luego realiza la operación:

```
> x <-  c(3,4,3,4,3)
> x^2
[1]  9 16  9 16  9

> x <- c(1,2)
> y <- c(1,2,3,4,5,6)
```

Indique como es realizada la siguiente operación: `> y^x.`
¿Que ocurre si `x <- c(1,2,3)`?

8. Atributos de vectores

Cada objeto de datos recibe dos atributos al ser definido: “longitud” y “modo”, son atributos implícitos

```
> length(x)       # da la cantidad de valores en el objeto x
[1] 8
```

`> mode(x)` # da el tipo de objeto (por ej. "numeric", "list", "logical", "NULL", use el help para ver los distintos modos)

```
[1] "numeric"

> mode(7)
 [1] "numeric"

> mode(7==7)
 [1] "logical"
```

8. 1. Longitud

Veamos que ocurre al cambiar la longitud de un vector

```
> x <- c(99,100,20)
> length(x) <- 2
> x
```

```
[1]  99 100

> length(x) <- 5
[1]  99 100  NA  NA  NA
```

8.2 Modo de almacenamiento.

```
> storage.mode(x)  # da el modo de almacenamiento de x.
[1] "double"
```

Los modos de almacenamiento incluyen doble y simple precisión y entero para números ("double" "single" "integer"), caracter ("character"), lógico ("logical"), y complejo ("complex").

Por defecto los datos numéricos se almacenan en doble precisión.

¿Cuales son los siguientes modos de almacenamiento? Ver help.

```
> x <- 2
> x <- as.integer(2)
> x <- 1*x
> x <- c(T,F,T,T,F)
```

Veamos que ocurre al cambiar el modo de almacenamiento.

```
> x <- c(1.7,pi,exp(1))
> x
[1] 1.700000 3.141593 2.718282

> storage.mode(x) <- "integer"
> x
[1] 1 3 2

> storage.mode(x) <- "character"
> x
[1] "1" "3" "2"
```

Algunas veces es necesario crear vectores de una longitud y modo de almacenamiento determinados par ser llenados posteriormente. Podemos hacer esto utilizando las siguientes funciones double(), single(), integer(), character() y logical().

```
> numeric(3)
[1] 0 0 0
> single(3)
[1] 0 0 0
> integer(3)
[1] 0 0 0
> character(3)
[1] "" "" ""
> logical(3)
[1] FALSE FALSE FALSE
```

8.3. Otros atributos

Un vector puede también tener un atributo de nombres (names() attribute). Si existe, el atributo de nombres de un vector es otro vector de cadenas de caracteres de igual longitud que el vector.

```
> x <- c(99,100,20)    # notas de un examen
> names(x)             # sin nombres todavía
NULL
> names(x) <- c("Sebastian","Carolina","Diego")  # asignación de nombres
> names(x)
[1] "Sebastian" "Carolina"  "Diego"
> x
 Sebastian Carolina Diego
        99      100    20

   > x[c(3, 1, 2)]
 Diego Sebastian Carolina
    20        99      100
```

8. 2 Acceso a las componentes de un vector por nombres

```
 > x <- c(99,100,20)
 > names(x) <- c("Sebastian Perez","Carolina Lopez","Diego Aguirre")
 > x["Carolina Lopez"]
 > x["Carolina"]
```

9. Otras funciones para operar sobre vectores.

```
> rev(4:7)          # invierte el orden de los elementos del vector
[1] 7 6 5 4


> sort(c(6,9,2,7,1))        # ordena  los elementos en orden ascendente
      [1] 1 2 6 7 9

> sort(c(6,9,2,7,1),index.return=T)
$x
[1] 1 2 6 7 9               # ordena  los elementos en orden ascendente

$ix
[1] 5 3 1 4 2               # da el lugar donde se encuentra cada elemento

> sort(c("Sebastian", "Carolina", "Diego"))
     [1] "Carolina"  "Diego"     "Sebastian"

   > order(c("Sebastian", "Carolina", "Diego"))
      [1] 2 3 1

   > x <- c("Sebastian", "Carolina", "Diego")

   > x[order(x)]                    # igual que  sort(x)
[1] "Carolina"  "Diego"     "Sebastian"

      > x <- c(6,5,6,6,7,5)

> duplicated(x)    # T si un elemento que ha aparecido previamente
      [1] FALSE FALSE  TRUE  TRUE FALSE  TRUE

      > unique(x)      # devuelve los elementos no duplicados
      [1] 6 5 7
```

Utilice el help para ver las funciones substring(), paste(), and nchar()
que son muy útiles para vectores de caracteres.

Atención: Los vectores admiten valores de un único modo. Si se intenta crear un vector con componentes con distinto modo el R fuerza los elementos a un modo común y **no da error**:

```
 > c(T, 5, 2.8)
[1] 1.0 5.0 2.8
```

10. Ejecución de comandos en una ventana de escritura (Script Window)

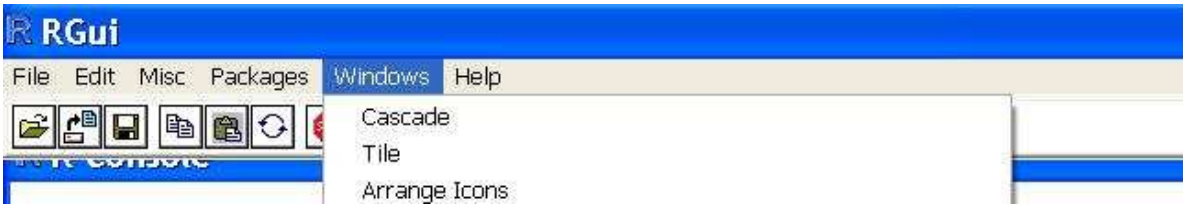
Esta ventana permite la ejecución de varios comandos simultáneamente.

Para abrir una ventana con un script nuevo, simplemente se clickea en el menú principal:

`File -> New script`

Para ejecutar un comando en la ventana del script hay que seleccionar el, o los comandos que interesan y clikear en el ícono de ejecución del menú del editor.
Copie (sin incluir el símbolo > al pintar cada línea en la ventana de comandos) y pegue los últimos comandos ejecutados de la ventana de comandos a la ventana de escritura. Una vez que copíó todos, con la ventana "Untiled – R Editor activa, ejecute todos esos comandos utilizando el botón ejecutar (run) del menú de dicha ventana.

Es útil acomodar las dos ventanas, la de comandos y la del editor, para poder verlas simultáneamente.



Esto puede realizarse cambiando la forma de las ventanas "a mano" o mediante la opción:
`Windows -> Tile` del menú principal (ver figura anterior).

Cuando se ejecuta un comando desde el **`script`** los resultados se muestran en la ventana de la consola inicial del `R.`

Con la ventana **`script`** activa es posible guardarlo desde el correspondiente menú

File -> Save

o

File -> Save as...

11-

- Cree un nuevo Script en R con las instrucciones guardadas en el script de punto 10.5.
- Defina variables cuyos nombres tienen acentos.

12. El espacio de trabajo – Workspace

Todas las variables u "objetos" creados en R están guardados en lo que se llama el espacio de trabajo *workspace*. Para ver que variables están en el espacio de trabajo puede usarse la función **ls()**(esta función no necesita argumentos entre los paréntesis).

```
> ls()   # fíjese cuáles son los objetos que ha creado.
```

13. Salir de R

El programa pregunta si quiere guardar la imagen de su espacio de trabajo (workspace image). Si se clickea en "yes" todos los objetos (los nuevos creados en la sesión actual y en las anteriores) serán guardados y estarán disponibles en la siguiente sesión.
Si clickea en "no", se perderán todos los objetos nuevos y el espacio de trabajo será restaurado al último momento en que la imagen fuera guardada. Es recomendable salvar el trabajo.

Más conveniente aún es *guardar el espacio de trabajo en una carpeta específica*, por ejemplo en C:\ Anàlisis de Datos . Para ello en el **menú principal** de R, seleccionar

File -> Save Workspace -> Save image in

se abre un navegador y se puede elegir la carpeta donde guardar el espacio de trabajo. Al guardarse se genera un ícono de R, que permite abrir el programa desde esa carpeta y con el espacio de trabajo alli guardado.