Leonardo Daniel Rivera Valdez

Básicos de programación en R

Tipos de datos

integer son aquellos con los que no se realizan operaciones, se denominan con una L al final

```
x <- 2L
#typeof() indica el tipo de dato que estamos usando
typeof(x)
```

#**double** es el tipo de dato con el que R realiza las operaciones

```
y <- 2.5
typeof(y)
```

#**complex** son números complejos en R

```
z <- 3 + 2i
typeof(z)
```

#**character** son datos de tipo texto

```
a <- "hola mundo"
typeof(a)
```

#**logical** funciona para indicar elementos de tipo booleano como TRUE o FALSE

```
b <- T
typeof(b)
```

Usos con variables

#Primero se definen unas variables
A <- 5
B <- 10
#Nótese que se almacenan los resultado en diferentes variables como C, C1, etc. Posterior a eso se puede poner sólo la variable y el valor se despliega en la consola (e.j. C sería [1] 15)

#**suma**
C <- A + B

#**resta**
C 1<- A - B

#**división**
C2 <- A / B

#**multiplicación**
C3 <- A * B

#**raíz cuadrada**, aquí se usa la función sqrt para elevar al cuadrado A y se almacena en la variable C4
C4 <- sqrt(A)

Ahora usando datos de tipo character. Se definen unas variables y se genera un mensaje con ellas en la región de ambiente

nombre <- "Pau"
saludo <- "hola"
pregunta <- “cómo estás?
mensaje <- paste(saludo, nombre, pregunta)

Variables y operadores lógicos

#Operadores lógicos, aquí se llaman logical y son:

== igual a, ayuda a ver que dos elementos sean iguales

#TRUE
5 == 5
# FALSE
5 == 4

!= diferente que, nos ayuda a confirmar que dos elementos sean distintos

#FALSE
5 != 5
#TRUE
5 != 4

< menor que

#TRUE
5 < 6
#FALSE
6 < 4

> mayor que

#FALSE
5 > 6
#TRUE
6 > 5

<= menor o igual que

#TRUE
10 <= 10
#FALSE
11 <= 10

>= mayor o igual que

#TRUE
10 >= 10
#FALSE
9 >= 10

! negación, niega el elemento al cual se une

#FALSE
!(10 > 9)
#TRUE
!(10 > 15)

| indica disyunción en sentido booleano, si uno u otra TRUE entonces TRUE

x <- 5 < 6
y <- 5 > 6
z <- 5 < 2

#TRUE
x | y
#FALSE
x | y

& indica conjunción, y sólo es TRUE si ambas lo son

#FALSE
x & y

isTRUE(x) es para saber si algo es verdadero o no

a <- 5 < 4
#FALSE
isTRUE(a)

Ciclo while

#Funciona de tal manera que si la condición en () se cumple, entonces se ejecuta lo que está en {}. Esto lo repite de manera cíclica hasta que () es FALSE

```
#ciclo infinito
while(TRUE){
  print("hola")}

while(FALSE){
  print("hola")}
```

#Este código funciona hasta que contador llega a 12, esto regulado porque cada ciclo contador aumenta +1, por ello sólo puede hacer 11 ciclos

```
contador <- 1
while(contador< 12){
  print(contador)
  contador <- contador +1}

número <- (1 - 2) * 5
while(número < 100){
  print(número)
  número <- número + 10}
```

Ciclo for

#En esencia lo que está haciendo es poner en () un vector o secuencia de iteraciones y en {} la acción a iterar. Su condición es una secuencia en lugar de una operación lógica como en while

```
#Imprime 5 veces “hola R”
for(i in 1:5){
  print("hola R")
}

#Imprime 6 veces “hola R :D”
for(i in 5:10){
  print("hola R :D")
}

#Aquí el código reitera del 5 al 10 e imprime, alternando, los valores de 5:10 con el valor de i en el ciclo +3. Si 1:5 son 5, 6, 7, 8, 9, 10, entonces 5, 8, 7, 10, 8, 11, 9, 12, 10, 13.
for(i in 5:10){
  print(i)
  i <- i + 3
  print(i)
}
```

Condicional if, else, y else/if

#rnorm genera un números aleatorios con una distribución normal, te pide N, media y SD
x <- rnorm(1)

#if es igual a while, donde () es la condición TRUE/FALSE y {} es la acción a ejecutar.La diferencia es que solamente corre una vez
#Para los casos en que queramos que suceda algo cuando () de if no se cumple se usa else

```
x <- rnorm(1)
if (x > 1){
  respuesta <- "Mayor que 1"
}else{
  if(x < -1){
    respuesta <- "Entre -1 y 1"
  }else{
    respuesta <- "Menor a -1"
  }
}
respuesta
```

#El código anterior genera un número al azar cercano a 1 y luego lo clasifica entre mayor que uno, entre -1 y 1, y menor a -1

#Esto es igual, solamente se redujo el espacio al juntar **else if** , así se elimina un corchete de al final. Se llaman **condicionales encadenados y los anteriores anidados**

```
x <- rnorm(1)
if (x > 1){
  respuesta <- "Mayor que 1"
}else if(x < -1){
    respuesta <- "Entre -1 y 1"
  }else{
    respuesta <- "Menor a -1"
  }
```

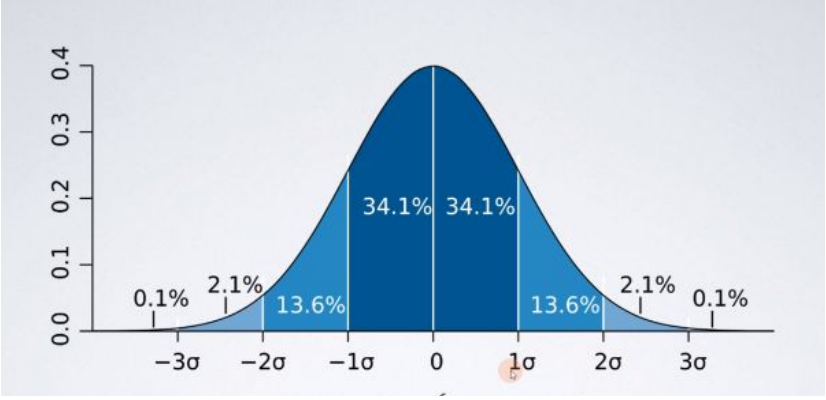
```
respuesta

#Otro ejemplo
x <- rnorm(1, mean= 20, sd= 10)
if(x > 18){
  respuesta <- "Bienvenido al club de los grandes"
}else{
  respuesta <- "Bienvenido al club de los pequeñines"
}
respuesta
```

Ley de los grandes números

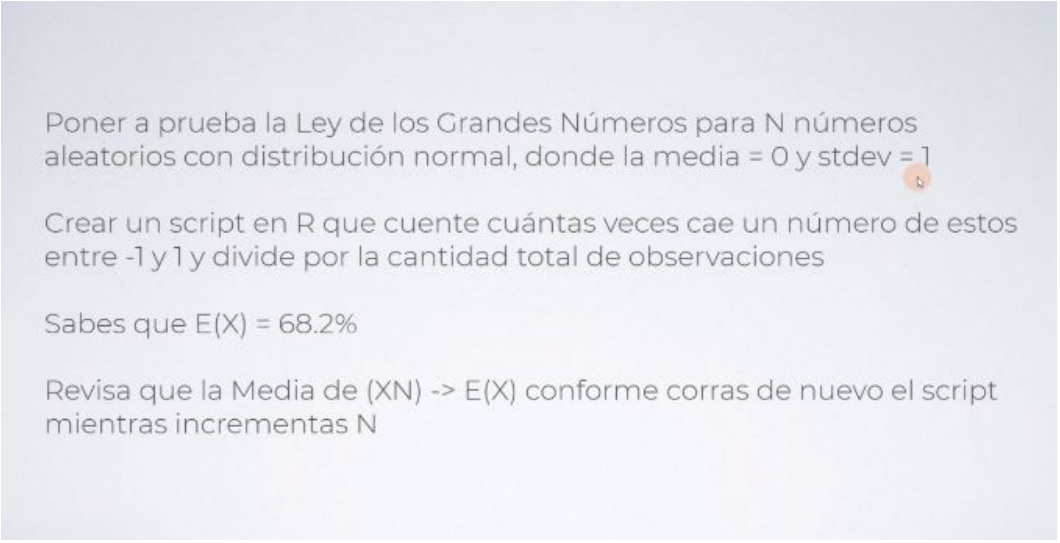
$$\bar{X}_n \longrightarrow E(X) \quad when \quad n \longrightarrow \infty$$

Según esta los valores de la media se aproximan a los esperados conforme el número de casos se acerca al infinito. Si n fuera el tiro de una moneda, entonces a mayor cantidad de tiros de moneda más se acerca la media al valor teórico esperado de P= .50 ó 50%.



Aquí se ve la distribución normal que en esencia todo conjunto de datos que se distribuya de esta manera tiende a que sus valores se acomoden alrededor de su media. Dentro del rango -1 a 1 desviaciones estándar. Si esto es el caso, los %s de arriba son ciertos para esa población.

Instrucciones:



Solución:

```
contador <- 0        # el contador inicial, se empieza sin ningún caso entrelas SD
N <- 100000          # número de casos o vector
for(i in rnorm(N)){
  if(i > -1 & i < 1){        #la condición a evaluar
    contador <- contador + 1        #se van agregando el número de de casos entre -1 y +1 SD
  }
}
contador/ N    #valor se va acercando al 68.2 de la distribución normal a mayor N
```