


# Estadística y modelación de sistemas socioecológicos en R



Laboratorio  
Nacional  
de Ciencias  
de la Sostenibilidad

**Dra. Yosune Miquelajauregui Graf**

# Plan del día

1. Generación de datos
  2. Programación práctica en R : bucles y funciones
  3. Ejercicios
- 

# Generación de datos

```
Años <- seq(1:1000)
```

```
str(Años) int [1:1000] 1 2 3 4 5 6 7 8 9 10 ...
```

[illegible]

# Generación de datos

```
seq(1, 20, by = 3)
```

```
tmp[1] 1 4 7 10 13 16 19
```

```
seq(0, 1, length.out=10 )
```

```
0.0000000 0.1111111 0.2222222 0.3333333
```

```
0.4444444 0.5555556 0.6666667 0.7777778
```

```
0.8888889 1.0000000
```



# Generación de datos

```
Municipio <- c("Tlahuac", "Tlalpan", "Miguel Hidalgo", "Cuauhtemoc", "Venustiano  
Carranza", "Coyoacan", "Benito Juarez", "Cuajimalpa", "Magdalena Contreras",  
"Xochimilco", "Azcapotzalco", "Gustavo A. Madero", "Iztacalco", "Iztapalapa", "Milpa  
Alta", "Alvaro Obregon")
```

```
Mu <- rep(Municipio, each=8)
```

```
str(Mu) chr [1:128] "Tlahuac" "Tlahuac"  
"Tlahuac" "Tlahuac" "Tlahuac" "Tlahuac" ...
```

# Generación de datos

`rep(1:20)`

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

`rep(1:20, each = 2)`

1 1 2 2 3 3 4 4 5 5 6 6 7 7 8 8 9 9 10 10 11 11 12 12 13 13  
14 14 15 15 [31] 16 16 17 17 18 18 19 19 20 20

# Generación de datos

## Generación de números aleatorios

### Funciones en R : rnorm, rpois, rbinom, entre otras

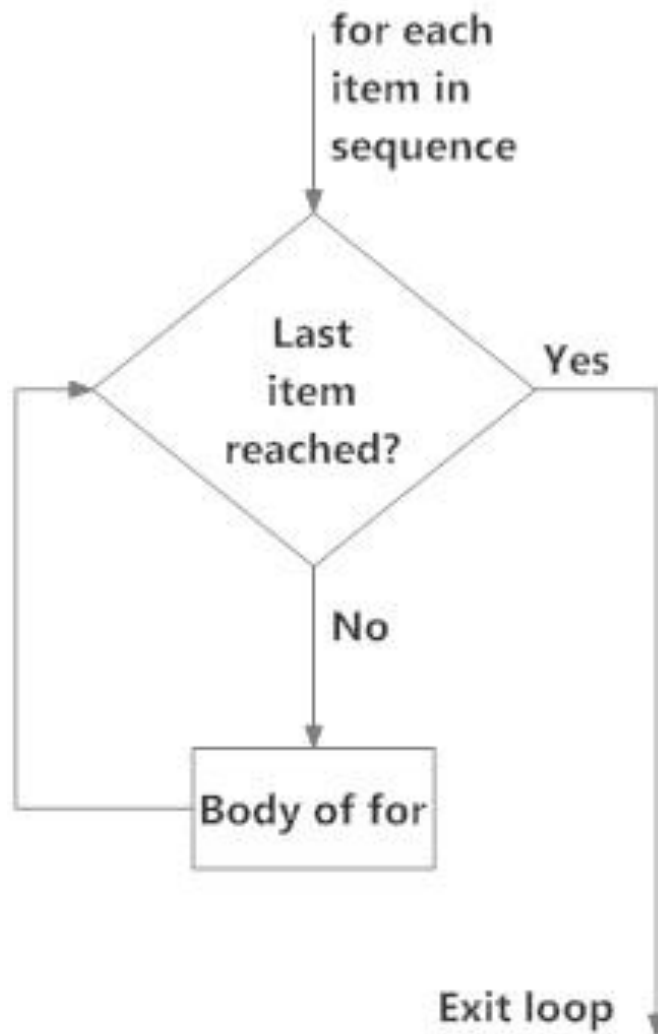
```
Datosaleatoriosnormal <- rnorm(1000,0.5,0.5)  
str(Datosaleatoriosnormal) num [1:1000] -4.904 -7.173 6.063 -0.278 -0.188
```

```
Datosaleatoriosbinom <- rbinom (50, 50,0.5)  
str(Datosaleatoriosbinom) int [1:50] 29 28 31 31 28 23 20 25 24 20
```

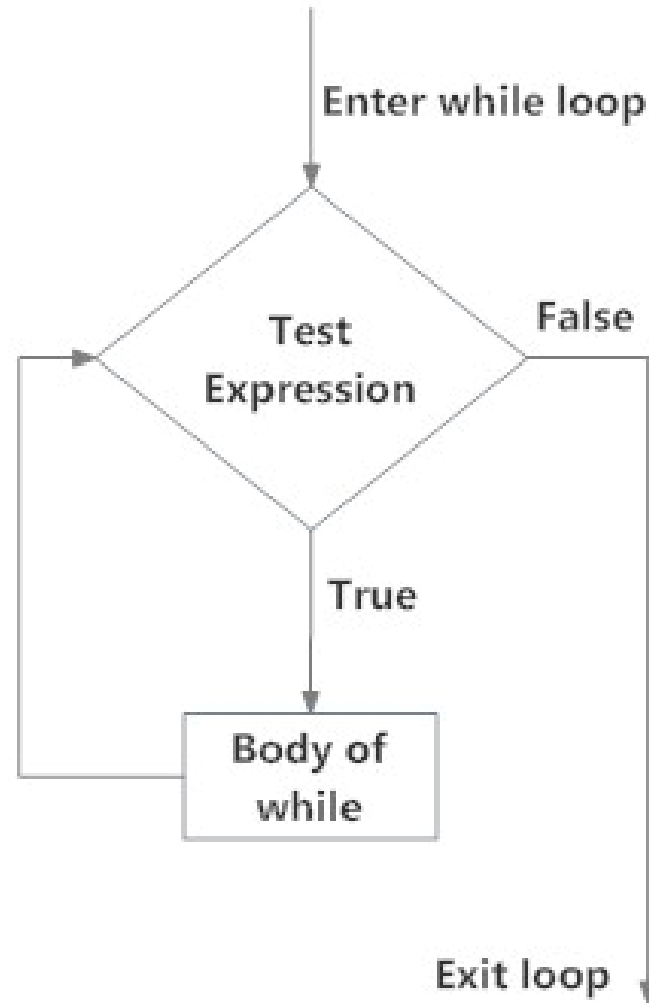
```
Datosaleatoriospoi <- rpois(1000,0.2)  
str(Datosaleatoriospoi) int [1:1000] 0 0 0 0 0 0 0 0 0 0
```

# Bucles

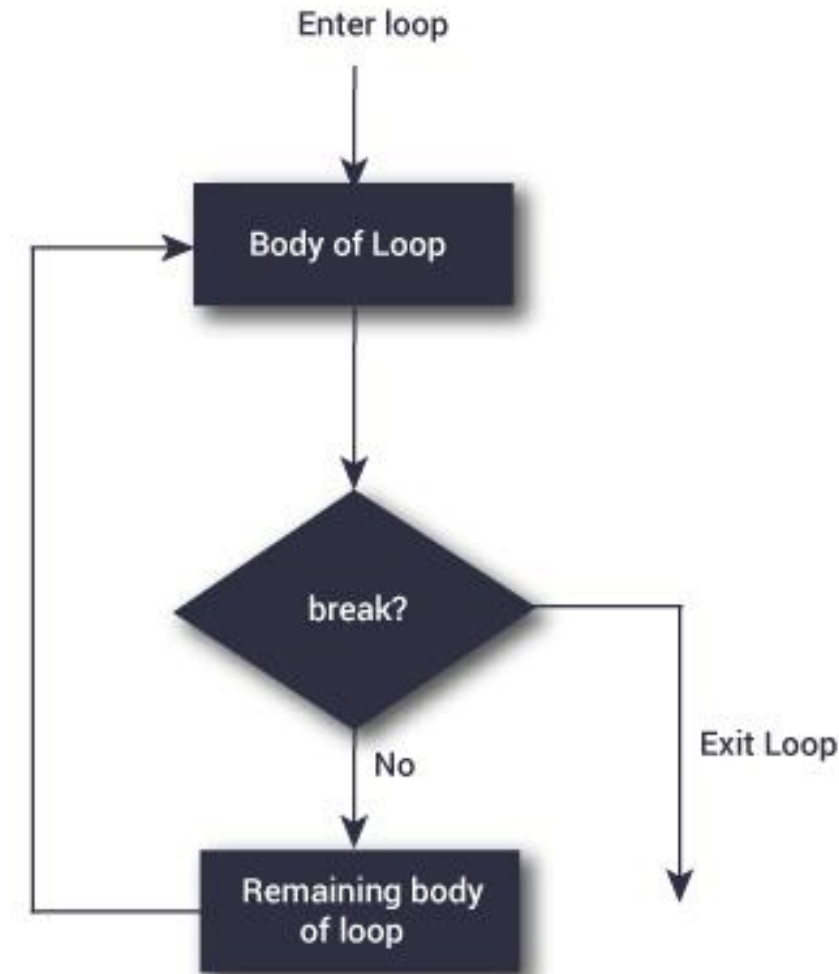
## *FOR*



## *WHILE*



## *REPEAT*





# Bucles

El siguiente bucle *for* calcula la raíz cuadrada de todos los elementos del vector tmp

1. Generar un vector tmp con los datos a evaluar

```
tmp<-seq(30,1000,by=2)  
str(tmp) num [1:486] 30 32 34 36 38 40 42 44 46 48 ...
```

2. Generar un vector vacío para guardar los resultados de la operación. Debe ser de la misma dimensión que tmp

```
rc <- numeric(length(tmp))
```

# Bucles

3. Crear el bucle *for* para evaluar cada elemento  $i$  del vector tmp y guardarlo en la posición  $i$  del vector rc

```
for (i in 1:length(tmp)) {  
  rc[i] <- sqrt(tmp[i])  
  rc  
}
```

# Bucles

4. Llamar al objeto rc que alberga los resultados de la operación

```
> rc
```

```
5.477226 5.656854 5.830952 6.000000 6.164414 6.324555 6.480741  
6.633250 6.782330 6.928203 7.071068 7.211103 7.348469 7.483315  
7.615773 7.745967 7.874008 8.000000 8.124038 8.246211 8.366600  
8.485281 8.602325 8.717798 8.831761 8.944272 9.055385 9.165151  
9.273618 9.380832 9.486833 9.591663 9.695360 9.797959 9.899495  
10.000000 10.099505 10.198039 10.295630 10.392305 10.488088  
10.583005.....
```

# Bucles

```
m <- 20  
n <- 0  
  
for (k in 1:length(m)) {  
  n[k] <- m*2/4  
  n  
}
```

Resultado ??

# Bucles

Inicializar variables

```
PoblacionBallenas <- seq (1200,2000,by=100)
```

```
NuevaPoblacion <- numeric(length(PoblacionBallenas))
```

```
Nacimientos <- 2
```

```
for (b in 1:length(PoblacionBallenas)){  
  NuevaPoblacion[b] <- PoblacionBallenas [b] + Nacimientos  
  NuevaPoblacion  
}
```

```
NuevaPoblacion[1] 1202 1302 1402 1502 1602 1702 1802 1902 2002
```



# Bucles

```
PoblacionBallenas <- seq (1200,2000,by=100)
```

```
CambioPob <- numeric(length(PoblacionBallenas)-1)
```

```
for (b in 1:length(PoblacionBallenas)-1){  
  CambioPob[b] <- PoblacionBallenas [b+1] - PoblacionBallenas [b]  
  CambioPob  
}
```

```
NuevaPoblacion[1] 100 100 100 100 100 100 100 100 100
```



# Bucles

El *while* loop presenta el siguiente formato:

`while (cond) exp`, donde `cond` es la condición a evaluar y `exp` es una expresión

```
i <- 1  
while (i < 6) {  
  print(i)  
  i = i+1  
}
```

# Bucles

El bucle *while* presenta el siguiente formato:

`while (cond) exp`, donde `cond` es la condición a evaluar y `exp` es una expresión

```
[1] 1 2 3 4 5
```



# Bucles

```
numero <- 1
while (numero <5){
  print("este número es menor que cinco")
  numero <- numero+1
}
[1] "este número es menor que cinco"
[1] "este número es menor que cinco"
[1] "este número es menor que cinco"
[1] "este número es menor que cinco"
```

# Bucles

El bucle *repeat* presenta el siguiente formato:

`repeat{exp}(cond) (break)`, donde `exp` es una expresión, `cond` es una condición y `break` es un comando que termina el loop cuando la condición no se cumple más

```
x <- 1
repeat {
  print(x)
  x = x+1
  if (x == 6){
    break }
}
```

1 2 3 4 5

# Funciones

Es otro tipo de objeto. La mayor parte del trabajo en R se realiza a través de funciones con sus respectivos argumentos entre paréntesis. R permite al usuario escribir sus propias funciones con `function ()` para efectuar tareas repetitivas. Éstas tendrán las mismas propiedades que las otras funciones ya implementadas en R.

Expresión o cuerpo de la función → 

```
primerafuncion <- function (x){  
  x <- x*5  
  x  
}
```

← argumentos

# Funciones

```
primerafuncion <- function (x){  
  x <- x*5  
  x  
}
```

```
primerafuncion(5)  
25
```

```
primerafuncion(8)  
40
```

```
primerafuncion(1000)  
5000
```

# Funciones

```
segundafuncion <- function (x, y){  
  tmp <- x*y  
  tmp  
}
```

```
segundafuncion(3,3)  
9
```

```
segundafuncion(1000,3)  
3000
```

```
segundafuncion(8789,8768)  
77061952
```

# Funciones

Una función puede contener más de un argumento. Además, podemos asignar un valor, por defecto, a los argumentos.

```
segundafuncion <- function (x, y=3){  
  tmp <- x*y  
  tmp  
}
```

# Funciones

head(Datos3)	Var1	Var2	Var3
	0.000000000	20.00000	103.0000
	0.01010101	26.86869	105.0404
	0.02020202	33.73737	107.0808
	0.03030303	40.60606	109.1212
	0.04040404	47.47475	111.1616
	0.05050505	54.34343	113.2020

```
> str(Datos3) num [1:100, 1:3] 0 0.0101 0.0202 0.0303 0.0404
... - attr(*, "dimnames")=List of 2 ..$ : NULL ..$ : chr [1:3]
"Var1" "Var2" "Var3">
```

# Funciones

Para extraer los valores utilizamos el operador \$, o los paréntesis cuadrados para acceder a los elementos [].

```
tercerafuncion <- function (x){  
  tmp1 <- mean(x[,1])  
  tmp2 <- mean(x[,2])  
  tmp3 <- mean(x[,3])  
  return (c(tmp1,tmp2,tmp3))  
}
```

```
tercerafuncion(Datos3)  
0.5 360.0 204.0
```



# Funciones

Una función puede arrojar, al evaluarse, distintos elementos. El resultado de una función puede ser una constante, un vector, una hoja de datos, una lista, etc. Es posible además guardar nuestra función como un objeto en R con la función `save()`, y acceder a ella cuando se tenga necesidad con la función `load()`

```
save(segundafuncion,file="segundafuncion.Rdata")  
load(file="segundafuncion.Rdata")
```

# Ejercicio 3

1. Generar una hoja de datos con dos variables: Practica (factor) y Tasa de Secuestro Carbono (numérico).
2. El factor Practica debe tener tres niveles: “Composta”, “Fertilización” y ”Laboreo”. Cada nivel repetido 200 veces.
3. Generar números aleatorios para la tasa de secuestro de carbono en función del factor práctica:
  - a) Tasa de secuestro(Composta): Generar 200 números bajo una distribución normal con media 0.5 y desviación estándar de 0.13.
  - b) Tasa de secuestro (Fertilización) : Generar 200 números bajo una distribución normal con media 0.9 y desviación estándar de 0.13.
  - c) Tasa de secuestro (Laboreo): Generar 200 números bajo una distribución normal con media 0.1 y desviación estándar de 0.13.

# Ejercicio 3

El formato de la hoja de datos debe ser :

Práctica	Tasa_Secuestro
Laboreo	0.06
Laboreo	0.07
Laboreo	.
Fertilización	.
Fertilización	.
Composta	.
Composta	.

# Ejercicio 4

1. Importar los datos AreaBasal.csv. Los datos se obtuvieron del inventario forestal para 10 parcelas de 1.0 ha cada una. Cada parcela tiene información sobre el número de árboles distribuidos en 10 clases diametrales de 2 cm cada una.
2. El objetivo del ejercicio es calcular el área basal total ( $\text{m}^2/\text{ha}$ ) para cada una de las 10 parcelas y guardar la información en un vector llamado Area\_Basal.
3. Sabiendo que el área basal de un árbol ( $\text{m}^2$ )  $= 3.142 * (\text{DAP}^2 / 40000)$  y que el DAP (diámetro a la altura del pecho) para cada clase diametral es:  
2.3, 4.5, 5.5, 6.0, 6.4, 6.8, 7.7, 7.9, 8.1, 9.0
4. Para optimizar la tarea debemos crear un bucle *for* dentro del cual se calcula primero el área basal de cada clase diametral, se multiplica por el número de árboles en cada clase y se suman para así obtener el área basal total.