APUNTES R

print(n) devuelve un breve resumen

rnorm(1) genera un dato al azar muestreado de una distribución normal de media 0 y varianza 1.

```
> 14/5-rnorm(1) [1] 2.108465
```

runif(N) permite obtener N números aleatorios en el intervalo [0, 1] Si volvemos a ejecutar la misma función observamos que el resultado es diferente.

```
> runif(6)

[1] 0.03277419  0.72593542  0.56049218  0.27794481  0.93507531  0.06622219
```

En los objetos se pueden almacenar tanto constantes numéricas como nombres

```
> num1=12; name1="tomate”
```

La instrucción **ls()** permite listar todo lo que hay en el directorio de trabajo

```
> ls() [1] "n" "name1" "name2" "num1" "num2"
```

Podemos listar todos los objetos cuyo nombre contenga un carácter en particular, empleando para ello la instrucción **pat**, por ejemplo si queremos que aparezcan los objetos que contengan el carácter m:

```
> ls(pat="m") [1] "name1" "name2" "num1" "num2"
```

En el caso en que dicho carácter no aparezca

```
> ls(pat="t") character(0)
```

Se pueden borrar elementos de la memoria empleando la función **rm()**

Si se quiere eliminar el objeto pp se emplea **rm(pp)**

Operaciones aritméticas

Suma, resta, multiplicación, división y elevar a una potencia, empleándose respectivamente los símbolos +,−,∗, y /.

```
> a<-5; b<-14; c<-18/7;

> a+b [1] 19

> b-a [1] 9

> b*a [1] 70

> c/b [1] 0.1836735
```

Cuando hay varias operaciones aritméticas en una misma expresión el orden en que el ordenador las ejecuta es: 1. potencias, 2. productos y divisiones, 3. sumas y restas.

Cuando hay varias operaciones del mismo tipo el orden en que se ejecutan es de izquierda a derecha. Sin embargo, el orden en que se ejecutan las operaciones se puede modificar empleando paréntesis.

Constantes numéricas: pi (π), exp(1) (e), 5e4 (5 × 10^4).

Operadores aritméticos < Menor > Mayor <= Menor o igual >= Mayor o igual != Distinto == Igualdad lógica

Algunas funciones matemáticas:

Funciones logarítmicas

log(x) logaritmo neperiano

log10(x) logaritmo en base 10

logb(x,base) logaritmo en cualquier base

exp(x) función exponencial.

Funciones trigonométricas

sin(x) cos(x) tan(x) asin(x) arco seno acos(x) atan(x)

Otras funciones

abs(x) valor absoluto

sqrt(x) raíz cuadrada

factorial(x) factorial

choose(n,x) binomio de Newton n sobre x

Vectores y matrices

La estructura es c().

Un vector es un objeto que almacena un conjunto de valores en una sola fila. Los valores que contiene el vector pueden ser números o cadenas de caracteres

```
> z1<-c(10, -3.5, "pepito")
```

donde el vector z1 tiene 3 componentes. Si queremos que escriba el contenido del vector z1

```
> z1 [1] "10" "-3.5" "pepito"
```

Si ahora deseamos acceder a la 3a componente de dicho vector

```
> z1[3] [1] "pepito"
```

Para sumar componentes

```
> Tutu<-y1[1]+y1[2]
> Tutu [1] 6.5
```

Podemos también realizar operaciones entre dos vectores vector1 y vector2

```
> vector1<-c(17/3,4,exp(pi))
> vector2<-c(sin(4*pi/3),-36,5^3)
> vector1+vector2
[1] 14.80000 92.45000 88.66667 127.39000
> vector1*vector2
```

[1] 23.40000 307.05000 58.66667 348.85200 (el producto de los dos vectores componente a componente).

Si se desea realizar el producto escalar de los dos vectores emplearemos

```
> vector1%*%vector2

[,1]

[1,] 2743.679
```

Matrices

```
> A<-matrix(c(3,4,5,6), nrow = 2, ncol = 2)
```

Si deseamos acceder a una posición

```
> A[2,1]

[1] 4
```

También se puede generar una matriz empleando los comandos cbind y rbind para generar una matriz combinando vectores en forma de columnas o vectores en forma de fila respectivamente.

```
> x1<- c(2, -30, 14)

> x2<- c(sin(2), 6, cos(3))

> x3<- c(exp(-3), 4/5, 0.56)

> A<- cbind(x1,x2,x3)

> A
```

Con matrices se pueden realizar las operaciones aritméticas de suma, resta, producto (%*%) o * para multiplicar matrices elemento a elemento.

Otras operaciones con matrices:

t(A) : transpuesta de la matriz A

det(A) : determinante de la matriz A.

solve(A,b) : solución del sistema de ecuaciones Ax=b.

solve(A) : inversa de la matriz A.

data.frame

La instrucción data.frame sirve para almacenar datos de diferentes tipos.

```
> alumno<- c("rosa","jacinto","edelmiro")

> peso<- c(65, 78, 87)

> estatura<- c(166, 180, 192)

> AA<-data.frame(alumno, peso, estatura)
```

Los elementos del array AA se pueden identificar de manera análoga a como se hace en el caso de matrices:

```
> AA[1,3]

[1] 166
```

Representación gráfica (plot)

```
> meses<- c(1,3,7,9,12)

> produccion<- c(20,40,100,60,18.5)

> plot(meses,produccion)
```

Podemos unir los puntos mediante una recta y colocar títulos a los ejes, mediante >plot(meses,produccion,type="b",xlab="meses del año", ylab="producción de tomates en Tm")

Si, por ejemplo, queremos que una los puntos mediante una línea podemos emplear l de line plot(x,f,type="l") y si deseamos que aparezcan tanto líneas como los puntos, emplearíamos b de both plot(x,f,type="b"), (p=puntos, l=líneas, b=puntos conectados por líneas, h=líneas verticales)

Para funciones:

```
> x 1:50     (entre 1 y 50)

> f=x^2

> plot(x,f)
```

También se puede especificar el color del gráfico mediante la instrucción col (col="red"), además de poner un título al mismo, empleando la instrucción main (main= "Tomates").

Representación gráfica de dos curvas

Se utiliza la estructura:

```
> plot( )

> par(new=TRUE)

> plot( )

> par(new=TRUE)

> plot( )
```

Representación gráfica conjunta de las funciones: sin(x) y cos(x), con $x \in [-\pi, \pi]$.

```
> x=seq(-pi,pi, length = 60)
```

```
>plot(x,sin(x),type='b',lty=5,col='blue',xlim=c(pi,pi),ylim=c(1.1,1.1),xlab='angulo',ylab='sin(x),cos(x)')

>par(new='TRUE')

>plot(x,cos(x),type='b',lty=6,col='red',pch=20,xlim=c(-pi,pi),ylim=c(-1.1,1.1),xlab='',ylab='',
main='Representación seno, coseno')

legend(-3,1, legend=c("sin(x)", "cos(x)"),col=c("blue", "red"), lty=5:6, cex=0.8)
```

Se puede observar que tanto los límites de los ejes coordenados como el texto de la leyenda (legend) y el color de cada gráfica (col) viene expresado en forma de vector, es decir: c(,). Las instrucciones lty y cex se refieren al tipo de línea y tamaño del texto que aparece en el dibujo, respectivamente.

Otra opción es dividir una ventana en varias partes. Para ello podemos emplear la instrucción

```
> par(mfrow=c(nrow,ncol))

> plot()

> plot()
```

> plot() ...

```
par(mfrow=c(2,2))
x=seq(-pi,pi, length = 60)
plot(x,sin(x),type='b',col='blue',lty=5, xlim=c(-pi,pi),ylim=c(-1.1,1.1),xlab='´angulo',ylab='sin(x)')
plot(x,cos(x),type='b',lty=6,col='red',pch=14)
x=seq(0,1,length=60)
plot(x,exp(x),type='b',lty=3,col='green',pch=18)
x=seq(-3,3, length = 60)
plot(x,exp(x),type='b',lty=2,col='purple',pch=18)
```