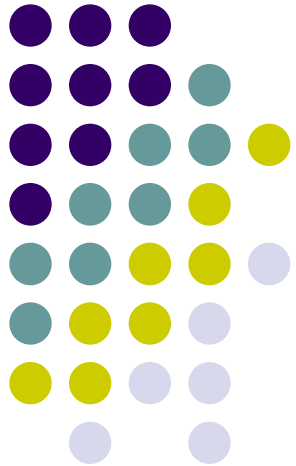
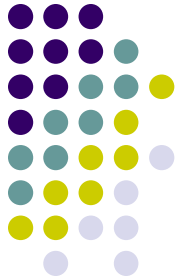
Programación en R

Creación de funciones

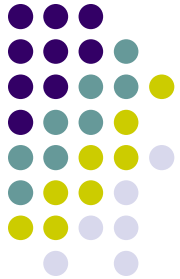


Objetivos



- Aprender a definir y construir funciones nuevas en R (extensiones del lenguaje).
- Construir algunas funciones elementales usando elementos básicos del lenguaje

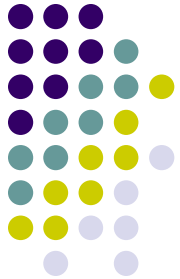
Introducción



- En R existen muchas funciones ya definidas, algunas incluso pueden ser modificadas, pero una de las capacidades más interesantes es la posibilidad de crear nuevas funciones que realicen tareas que no estaban definidas en el momento de instalar el programa.
- En los packages básicos existen funciones elementales de como abs, sqrt, sin, cos, tan, exp, log, min, max, sum, prod, length, mean, range, median, var, cov, summary, sort, rev, order, etc., que están preparadas para ser usadas siguiendo unas reglas de sintaxis básicas.
- Ejemplos:

```
> median(1:10)
> sin(1:10); sin(pi)
> cos(pi)
> sum(1:10)
> prod(1:10)
> summary(1:10)
```

Definición de una función



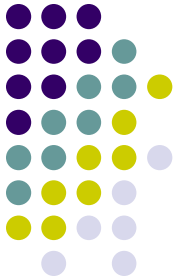
- Una función se define asignando a un objeto la palabra **`function`** seguida de sus argumentos.
- Los argumentos se escriben entre paréntesis y separados por comas. Entre los argumentos de una función se puede usar "tres puntos seguidos", ..., lo que indicaría que el número de argumentos es arbitrario.
- A continuación de los argumentos se escriben las sentencias que definen la función. Si no se pueden escribir en una sola sentencia se agrupan entre llaves **`{…}`**
- Ejemplo. La siguiente función calcula la suma de dos objetos (argumentos "A" y "B") que pueden ser vectores o matrices:

```
> funcionsuma<-function(A=1,B=2) A+B
```

Dado que se han dado valores 1 y 2 por defecto a los argumentos, el uso de la función creada puede ser:

```
> funcionsuma()
> funcionsuma(5,7)
> funcionsuma(1:9,2:4)
```

Definición de una función



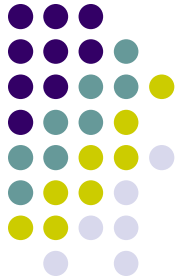
- Si desea poner de manifiesto el valor devuelto, puede utilizar la función **`return`** que termina la función y devuelve su argumento.

```
> f1 <- function(A=1,B=2)
+ {
+ # Devuelve A+B
+ return(A+B)
+ }
```

- Ejemplo. Una variante de la función que devuelve el resultado en un objeto de tipo lista sería:

```
> f4 <- function(A=1,B=2)
+ {
+ # Devuelve en una lista A+B y (A+B)^2
+ list(Suma= A+B,Cuadrado= (A+B)^2)
+ }
```

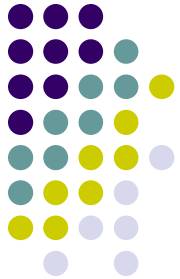
Definición de una función



- Es posible acceder a los argumentos de una función mediante la función **`formals`** y al cuerpo de la misma mediante la función **`body`**.
- Ejemplos:

```
> f1 <- function (A = 1, B = 2)
{
A + B
}
> formals(f1)
> body(f1)
> # Modificamos los argumentos:
> formals(f1)<-alist(X=,Y=-1)
> # La función queda como:
> f1
function (X, Y = -1)
{
A + B
}
```

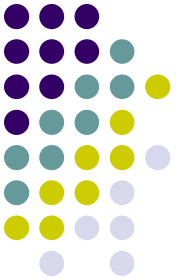
Definición de una función



```
> # Y ahora modificamos el cuerpo como:
> body(f1) <- expression(X*Y)
> # Y la función ahora queda como:
> f1
function (X, Y = -1)
X * Y
```

Algunas funciones elementales

Función media

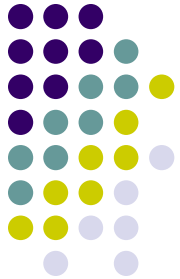


- Vamos a escribir una función alternativa a la función media (existente en R).
- Es una función que tiene un solo argumento, se trata de un vector “x” que por defecto está definido como “x=NA”.
- La función primero elimina del vector “x” los elementos no disponibles (NA). Para ello se utiliza la negación de la función is.na(). Una vez hecho esto se calcula la media del vector resultante.
- La función quedaría definida como sigue:

```
media <- function(x=NA)
{
x<-x[!is.na(x)]
sum(x)/length(x)
}
```

Algunas funciones elementales

Función media



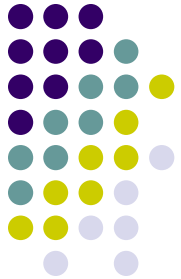
- Algunos ejemplos de uso de la función que acabamos de crear son:

```
> media(c(2,3,7))
[1] 4
> media(c(2,3,7,NA))
[1] 4
```

- **Ejercicio 1.** Escribir una nueva función “media.2” similar a la anterior función “media” que calcule además de la media el número de elementos NA que hay en “x”. De este modo el resultado que devolverá la función será una lista con la media de “x” y el número de elementos NA que hay en “x”.

Algunas funciones elementales

Función varianza



- Vamos a construir ahora una función que realice el cálculo de la varianza de un vector. Para ello podemos utilizar la fórmula derivada de la relación entre los momentos, $\mu_2 = m_2 - m_1^2$, usando así la función “media” que hemos creado antes.

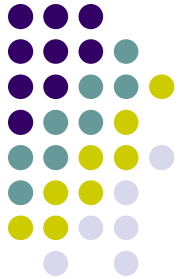
```
> koning<- function(x=NA)
{
media(x^2)-(media(x))^2
}
```

- Ejemplos de uso de la función creada:

```
> koning(c(2,3,7))
[1] 4.666667
> koning(c(2,3,7,NA))
[1] 4.666667
```

Algunas funciones elementales

Función varianza



- No obstante, desde el punto de vista computacional es preferible utilizar directamente la definición de la varianza para obtener la siguiente función:

```
> varianza<- function(x=NA)
{
y<-media(x)
media((x-y)^2)
}
```

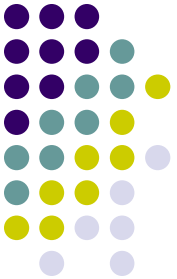
- Ejemplos de uso de la función creada:

```
> varianza(c(2,3,7))
[1] 4.666667
> varianza(c(2,3,7,NA))
[1] 4.666667
```

- Esta variante ofrece (en los ejemplos considerados) los mismos resultados, pero es más estable en otros casos.

Algunas funciones elementales

Función varianza



- La definición de la función se puede simplificar, escribiéndola como sigue:

```
> varianza2<- function(x=NA)
      media((x-media(x))^2)
> varianza2(c(2,3,7))
```

- **Ejercicio 2.** Comparar las funciones anteriores con la definida en R (package base): **`var`**
- **Ejercicio 3.** Escribir funciones que calculen los coeficientes de asimetría y curtosis de un vector. Comparar con funciones disponibles en R para tal finalidad como las funciones **`skewness`** y **`kurtosis`** del package moments.
- **Ejercicio 4.** Escribir funciones que calcule la covarianza y el coeficiente de correlación de Pearson. Usando estas funciones escribir otra función que ofrezca como resultado la recta de regresión de un vector “y” sobre otro vector “x”. Comparar con las funciones disponibles en R para tal finalidad: **`cov,cor,lm`**