Funciones en R

Andres Quiñones

26/2/2020

¿Que es una función?

Matemáticas

Es una forma de relacionar una serie de *inputs* (entradas) y *outputs* (salidas).

Computación

Un subconjunto de instrucciones que realizan una parte de los cómputos de un programa

Uso de las funciones

¿Cuando usar funciones en un programa?

Cuando a lo largo del proyecto existe una misma operación que se usa repetidamente sobre diferentes *objetos* y se esperan diferentes resultados dependiendo del *objeto* al que se le aplique.

¿Por qué se deberían usar funciones?

- Reduce la longitud del código
- Disminuye las oportunidades de tener errores
- Facilita generalizar cambios a lo largo del proyecto
- Facilita el entendimiento del código para otras personas (particularmente para nuestro yo futuro)
- Facilita el use de versiones *vectorizadas* de loops, lo cuál (en ciertos situaciones) facilita paralelizar los cómputos

Funciones en R

¡R está lleno de funciones!

Algunos "*expertos*" argumentan que *R* es un lenguaje basado en functiones (Funtional programming)

En realidad eso depende de si uno como programador usa funciones, o no.

Sintaxis

Declaración de UDF (user defined functions)

```
nombre=function(INPUT){
  cuerpo
  return(OUTPUT)
}
```

Ejecución de funciones

```
nombre(INPUT)
```

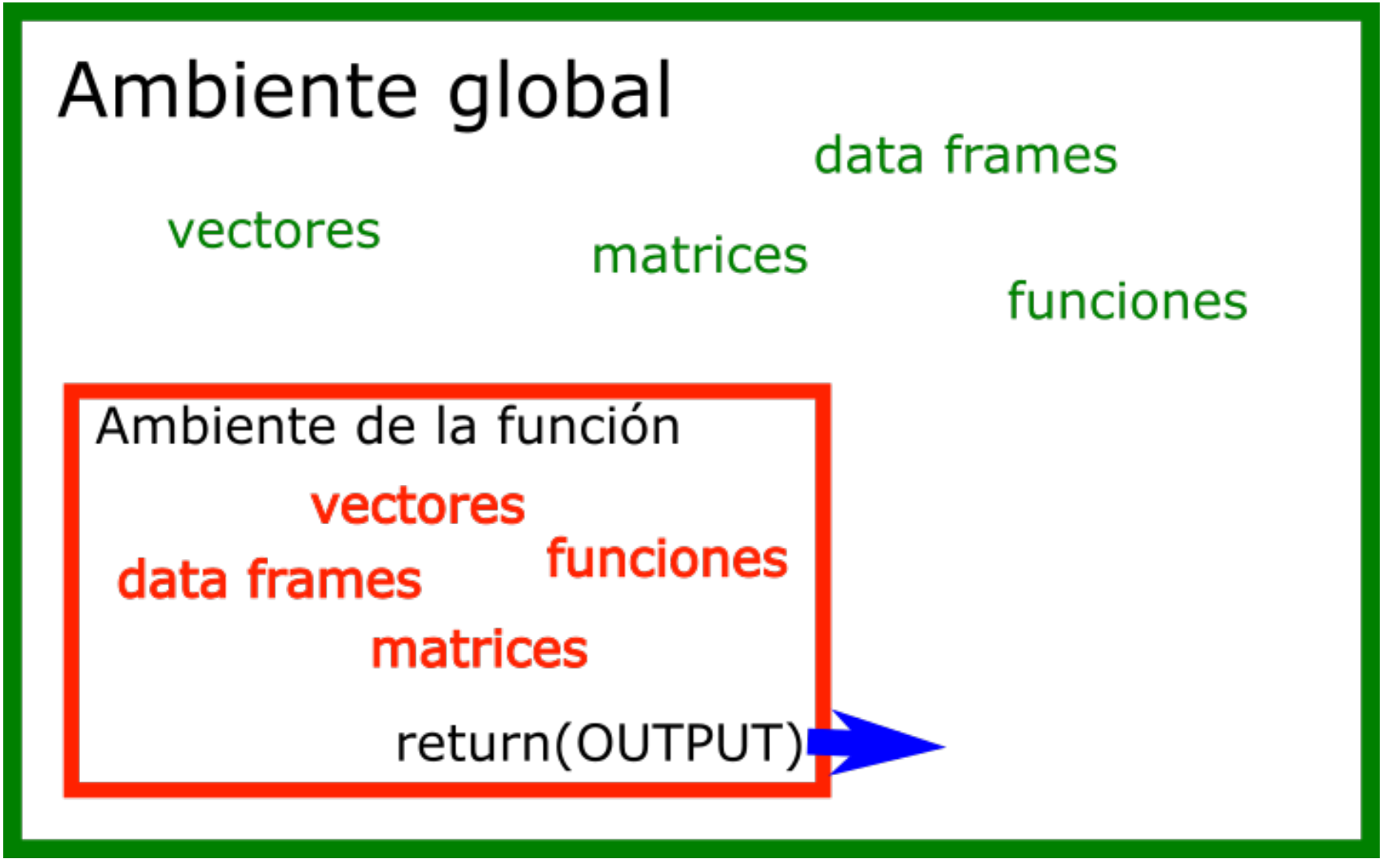
```
## [1] "OUTPUT"
```

Ambientes durante la ejecución

```
knitr::include_graphics(here("globalEnv.png"))
```

Ambientes durante la ejecución

```
knitr::include_graphics(here("localEnv.png"))
```



Usemos funciones!!

Mi primera función

Polinomio

$$y = ax^2 + bx + c$$

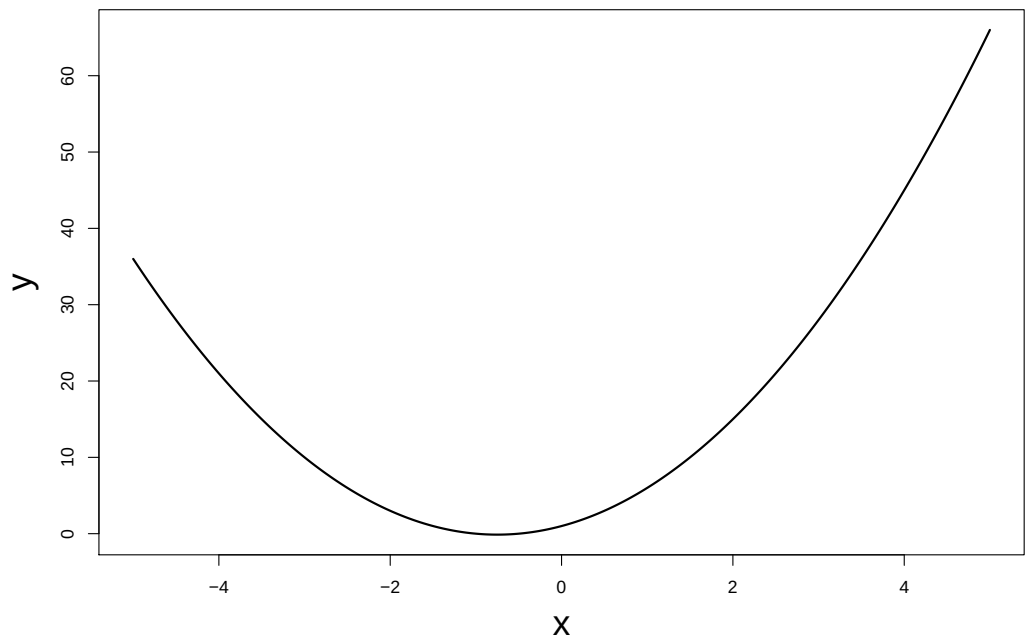
```
poli=function(x){
  a=2
  b=3
  c=1
  y=a*x^2+b*x+c
  return(y)
}
poli(2)
```

```
## [1] 15
```

Polinomio

$$y = ax^2 + bx + c$$

```
rangX=seq(-5,5,length.out = 1000)
yVal=poli(rangX)
plot(rangX,yVal,type="l",ylab="y",xlab="x",cex.lab=2,lwd=2)
```



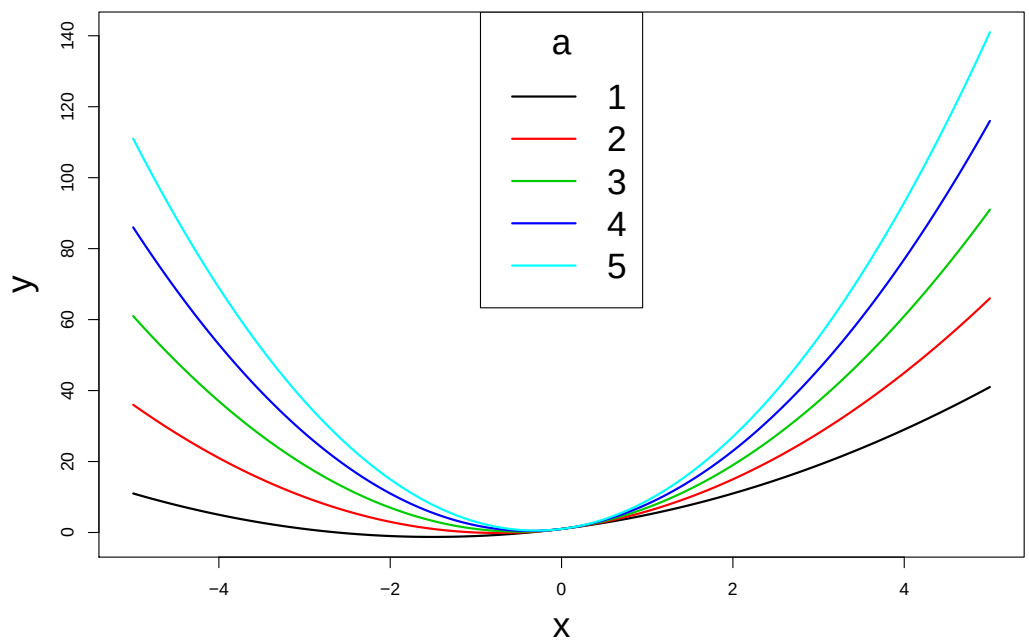
Polinomio

```
poli2=function(a,b,c,x){
  y=a*x^2+b*x+c
  return(y)
}
poli2(a=2,b=3,c=1,2)
```

```
## [1] 15
```

Polinomio

```
valsA=1:5
yVals_A=sapply(valsA,FUN = poli2,x=rangX,b=3,c=1)
matplot(yVals_A,x=rangX,type="l",lty=1,
        col =1:5,ylab="y",xlab="x",cex.lab=2,lwd=2)
legend("top",legend =valsA,col = 1:5,title = "a",lwd=2,cex=
```



Hagamos funciones más útiles

Centrar los datos

$$x = 1, 4, 7, 4....$$

$$x' = x - \bar{x} + a$$

Hagamos funciones más útiles

Centrar los datos

$$x = 1, 4, 7, 4....$$
$$x' = x - \bar{x} + a$$

```
centrar = function(datos, n.centro) {
  n.datos = (datos - mean(datos)) + n.centro
  return(n.datos)
}
x_1=c(4,6,35,45,1,59)
centrar(x_1,0)
```

```
## [1] -21 -19  10  20 -24  34
```

```
mean(x_1)
```

```
## [1] 25
```

```
mean(centrar(x_1,0))
```

```
## [1] 0
```

```
mean(centrar(x_1,5))
```

Centrar los datos

Es común centrar los datos alrededor del 0

Centrar los datos

Es común centrar los datos alrededor del 0

```
centrar = function(datos, n.centro=0) {
  n.datos = (datos - mean(datos)) + n.centro
  return(n.datos)
}
x_1=c(4,6,35,45,1,59)
centrar(x_1)
```

```
## [1] -21 -19  10  20 -24  34
```

```
centrar(x_1,0)
```

```
## [1] -21 -19  10  20 -24  34
```

```
centrar(x_1,5)
```

```
## [1] -16 -14  15  25 -19  39
```

Para todo un juego de datos

```
centrar = function(datos, n.centro=0) {
  n.datos = (datos - mean(datos)) + n.centro
  return(n.datos)
}
means=c(2,56,45,9,-4)
r.data=data.frame(lapply(means, rnorm,n=1000,sd=0.5))
names(r.data)=paste0("R_",1:5)
```

Para todo un juego de datos

```
lapply(r.data, mean) # means=c(2,56,45,9,-4)
```

```
## $R_1
## [1] 2.014847
##
## $R_2
## [1] 56.03096
##
## $R_3
## [1] 45.01292
##
## $R_4
## [1] 8.988989
##
## $R_5
## [1] -4.0022
```

Para todo un juego de datos

```
c.data=centrar(r.data)
```

```
## Warning in mean.default(datos): argument is not numeric
## returning NA
```

Para todo un juego de datos

```
c.data=lapply(r.data, centrar)
lapply(c.data, mean)
```

```
## $R_1
## [1] 1.634389e-16
##
## $R_2
## [1] -2.309238e-15
##
## $R_3
## [1] -7.177357e-16
##
## $R_4
## [1] -1.337603e-17
##
## $R_5
## [1] 1.549836e-17
```

Funciones más complejas

Caractericemos un juego de datos

```
descStats<-function(x){
  m.dat=mean(x,na.rm = T)
  sd.dat=sd(x,na.rm = T)
  cv.dat=sd.dat/m.dat
  mx.dat=max(x,na.rm = T)
  mn.dat=min(x,na.rm = T)
  list(m.dat,sd.dat,cv.dat,mx.dat,mn.dat)
}
```

Funciones más complejas

```
descStats(r.data$R_1)
```

```
## [[1]]
## [1] 2.014847
##
## [[2]]
## [1] 0.5150801
##
## [[3]]
## [1] 0.2556423
##
## [[4]]
## [1] 4.018308
##
## [[5]]
## [1] 0.3170964
```

Funciones más complejas

```
sapply(r.data,descStats)
```

```
##          R_1         R_2          R_3         R_4         R_5
## [1,] 2.014847   56.03096     45.01292   8.988989   -4.0022
## [2,] 0.5150801 0.5013479   0.5039943  0.5067372 0.525662
## [3,] 0.2556423 0.008947693 0.01119666 0.0563731 -0.13134
## [4,] 4.018308   57.72323     46.5948    10.4156   -2.35554
## [5,] 0.3170964 54.3742      43.53946   7.657378  -5.85103
```

Gráficas

```
pdf(here("graphs.pdf"))
randN<-runif(1000)
lapply(r.data, function(x){
plot(randN,x,pch=20,col="red")
l.mod<-lm(x~randN)
abline(l.mod)
})
```

```
## $R_1
## NULL
##
## $R_2
## NULL
##
## $R_3
## NULL
##
## $R_4
## NULL
```