Introducción al R y RStudio

Tomado de R. P. Dobrow.*Probability: With Applications and R*.
Wiley, 2013.

Probabilidad 2020 - Facultad de Ciencias

Contenidos

- Introducción a R
- R como calculadora
- Funciones integradas del R
- Navegar por el teclado
- La ayuda (help) del R
- Declarando vectores
- Simulación de vectores aleatorios
- Operaciones con vectores
- Gráficos de vectores (Plots)
- Programas (scripts)
- Funciones

Introducción al R[1]

Utilizamos el software estadístico R a través del entorno R Studio.

- `https://www.r-project.org/` (Download)
- `https://www.rstudio.com/` (Download)

[1] Adaptado de: R. P. Dobrow. *Probability: With Applications and R*. Wiley, 2013.

Referencias:

- Robert P. Dobrow.*Probability: With Applications and R*. Wiley, 2013.
- Robert P. Dobrow.*Introduction to stochastic processes with R*. Wiley, 2016.
- Stefano. M. Iacus. *Option Pricing and Estimation of Financial Models with R*. Wiley, 2011.

R como calculadora

Cuando abres el R, la primera ventana que ves es la consola del R. (Es una de las 4 consolas del R studio) Se escribe directamente en esa consola para utilizarla como calculadora. Por ejemplo:

```
> 1+1
[1] 2
> 2-2
[1] 0
> 3*3
[1] 9
> 4/4
[1] 1
> 5^5
[1] 3125
> 4^(4^4)
[1] 1.340781e+154
```

Funciones integradas del R

Los nombres de las funciones integradas son en general intuitivos.

```
> pi
[1] 3.141593
> cos(pi)
[1] -1
> exp(1)
[1] 2.718282
> abs(-6)
[1] 6
> factorial(4)
[1] 24
> sqrt(9)
[1] 3
```

Navegar por el teclado

Quizá las teclas mas importantes del teclado son las flechas $\uparrow$ y $\downarrow$. Cada comando que se utiliza se guarda en la memoria. Las flechas $\uparrow$ y $\downarrow$ recorren la historia de los comandos utilizados para atrás y para adelante.
Supongamos que queremos calcular $\log 8$ y escribimos

```
> Log(8)
Error:  could not find function "Log"
```

Obtenemos un error porque el R distingue mayúsculas de minúsculas. Todos los comandos comienzan con minúscula.
En vez de re-escribir el comando, con la flecha $\uparrow$ traemos el último comando y lo corregimos.

```
> log(8)
[1] 2.079442
```

La ayuda (help) del R

Cualquier comando de R precedido por el signo de pregunta ?:

```
> ?log
```

abre una ventana separada de ayuda:

log {base} R Documentation

Logarithms and Exponentials

Description

`log` computes logarithms, by default natural logarithms, `log10` computes common (i.e., base 10) logarithms, and `log2` computes binary (i.e., base 2) logarithms. The general form `log(x, base)` computes logarithms with base `base`.

`log1p(x)` computes *log(1+x)* accurately also for *|x| << 1*.

`exp` computes the exponential function.

`expm1(x)` computes *exp(x) - 1* accurately also for *|x| << 1*.

Usage

```
log(x, base = exp(1))
logb(x, base = exp(1))
log10(x)
log2(x)

log1p(x)

exp(x)
expm1(x)
```

Arguments

`x`	a numeric or complex vector.
`base`	a positive or complex number: the base with respect to which logarithms are computed. Defaults to *e*=`exp(1)`.

Si el signo ? no funciona, intenta ??, por ejemplo: ??cholesky

Declarando vectores

El comando a:b crea un vector de enteros de a hasta b.

```
> 1:10
[1] 1 2 3 4 5 6 7 8 9 10
> -3:5
[1] -3 -2 -1 0 1 2 3 4 5
```

Para generar sucesiones mas complicadas se utiliza el comando seq.

```
> seq(1,10)
[1] 1 2 3 4 5 6 7 8 9 10
> seq(1,20,2)
[1] 1 3 5 7 911 13 15 17 19
```

Para crear un vector y asignarlo a una variable, escribir

```
> x <- c(2,3,5,7,11)
> x
[1] 2 3 5 7 11
```

Si queremos agregar elementos a un vector, escribimos

```
> x <- c(x,13,17)
> x
[1]  2  3  5  7 11 13 17
```

La letra `c` significa *concatenar*

A los elementos de un vector accedemos mediante [ ].

```
> x[4]
[1] 7
> x[1:3]
[1] 2 3 5
```

Simulación de vectores aleatorios

Podemos simular un vector de variables aleatorias independientes [2] de largo n para las distribuciones mas usuales, mediante la declaración de sus parámetros en el comando correspondiente, que comienza siempre por una `r` (de “random”). Por ejemplo

```
> runif(3,0,1)
[1] 0.2758678 0.2295485 0.6165712
```

genera tres números aleatorios con distribución uniforme en [0,1].

[2] Asumimos que son independientes. En realidad se trata de números pseudo-aleatorios

Para guardar en un vector x 100 números aleatorios con distribución normal con media 0 y desvío estándar [3]:

```
> x<-rnorm(100,0,0.1)
```

Si no declaramos los parámetros, R asume los valores por default de cada distribución: `runif(3)` genera 3 números uniformes en [0,1], y `rnorm(100)` genera 100 números normales estándar (con media cero y varianza uno)

[3] Cuidado: 0.1 es el desvío, la varianza es 0.01.

Operaciones con vectores

Podemos averiguar el largo, la suma, la media, el desvío estándar, y la varianza de un vector x:

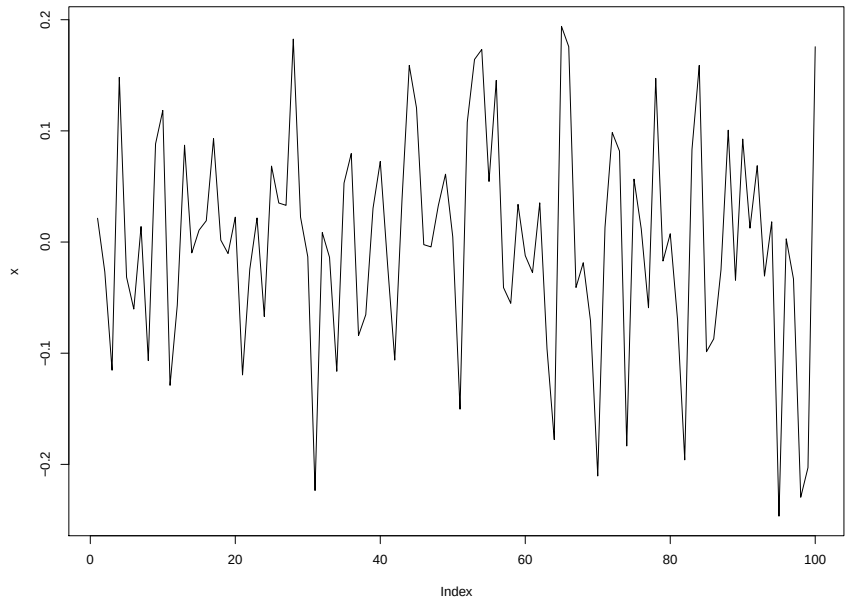
```
> length(x)
[1] 100

> sum(x)
[1] 0.3076999

> mean(x)
[1] 0.003076999

> sd(x)
[1] 0.09927355

> var(x)
[1] 0.009855239
```

Gráficos de vectores (Plots)

Podemos graficar el vector x:

```
> plot(x,type="l")
```

produce un gráfico en una ventana separada:



`type="l"` produce línea continua en el gráfico.
Más opciones con `?plot`

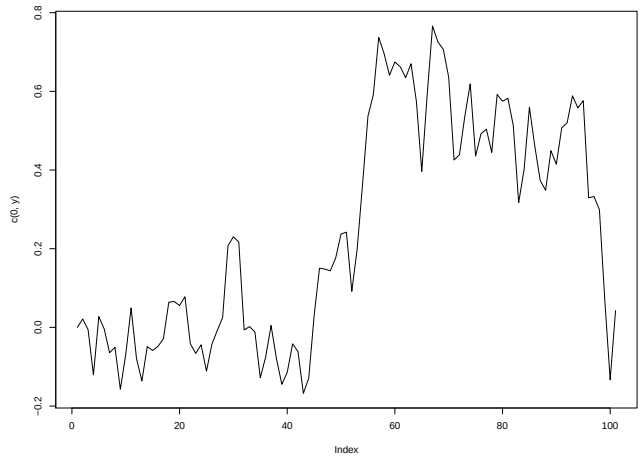
Gráficos de vectores (cont.)

Más interesante es graficar el vector $y = (y_1, y_2, \dots)$ de las *sumas parciales* (*cumulative sums*) de $x = (x_1, x_2, \dots)$ definido mediante

$$y_0 = 0, \quad y_n = x_1 + \cdots + x_n,$$

que se calcula con el comando `cumsum` (no incluye y_0):

```
> y<-cumsum(x)
> plot(c(0,y),type="l")
```



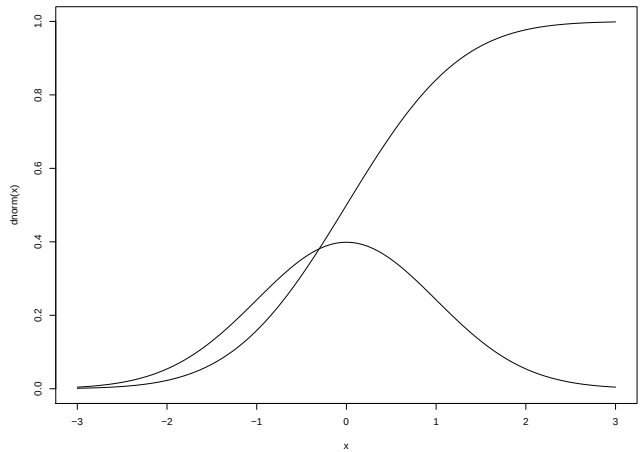
El comando `c(0,y)` agrega el valor incial 0 al vector y.

Gráficos de funciones

También se pueden graficar funciones definidas mediante fórmulas. Por ejemplo

```
> curve(dnorm,xlim=c(-3,3),ylim=c(0,1))
> curve(pnorm,add=TRUE)
```

produce la densidad y la distribución normal en una misma ventana:



En este caso los comandos `xlim` y `ylim` se aplican a ambos gráficos, y `add=TRUE` agrega el segundo gráfico al primero.

Programas (scripts)

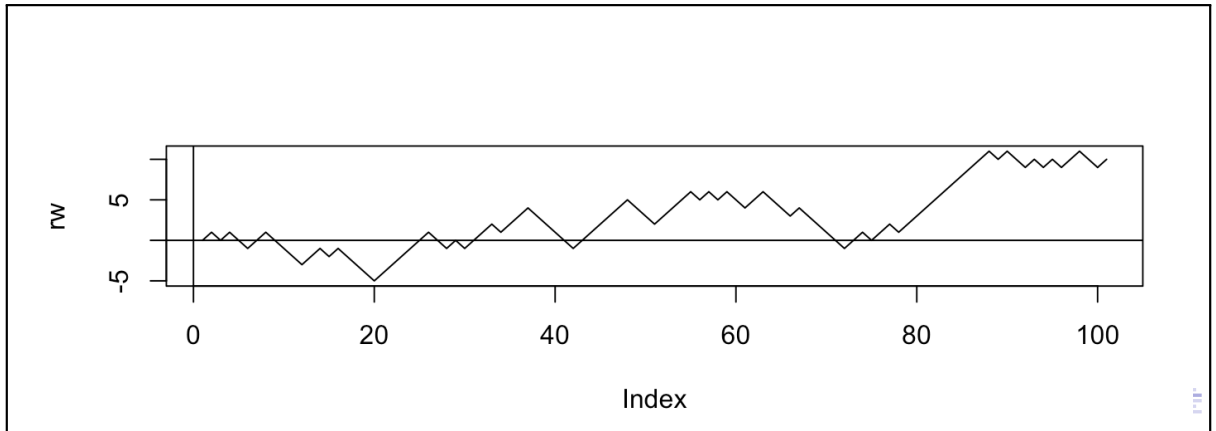
Un script (un programa breve) es un archivo de texto con la extensión .R que contiene una sucesión de comandos de R, y se puede correr en la consola:

```r
# Simulating Random Walk
n <- 100
# sample chooses -1 or 1 with equal probab. n times
increments<-sample(c(-1,1),n,replace = TRUE)
rw <- c(0, cumsum(increments))
plot(rw,type="l")
# abline adds horizontal and vertical lines at 0,0
abline(h=0,v=0)
```

Para correr el script:

```
> source("~/Dropbox/.../random-walk.R")
```

produce un gráfico en una ventana separada:



Funciones

La sintaxis de una función es

```
> name <- function(arg1,arg2,...) expresión
```

Ejemplos:

```
> f<- function(x) 1/(1+x)
> f(1)
[1] 0.5
```

Para simular un vector de dos variables estándar independientes a partir de dos uniformes escribimos la función `boxmuller`:

```
> boxmuller<-function(u1,u2) c(sqrt(-2*log(u1))*cos(2*pi*u2),sqrt(-2*log(u1))*sin(2*pi*u2))
> boxmuller(runif(1),runif(1))
[1] 0.3089062 0.8901110
>
```