R, el llenguatge de computació estadística per excel·lència

Adrià Cabeza Sant'Anna
Departament de Computació

January 5, 2019

Índex

1 Introducció

El llenguatge de programació R és un llenguatge de programació usat per a l'obtenció de càlculs i gràfics estadístics. Fou creat originalment per en Ross Ihaka i Robert Gentleman a la Universitat d'Auckland (Nova Zelanda) a l'any 1993.

És una implementació de S, un llenguatge de programació creat als Bell Labs per en Rick Becker, John Chambers i l'Allan Wilks i l'origen del seu nom es deu a les inicials dels seus dos creadors a més d'un joc de paraules envers al S.

2 Tipus de Data

Per a assignar a una variable ho podem fer de diverses maneres:

```
> x <- c(10.4, 5.6, 3.1, 6.4, 21.7)
> x = c(10.4, 5.6, 3.1, 6.4, 21.7)
> assign("x", c(10.4, 5.6, 3.1, 6.4, 21.7))
> c(10.4, 5.6, 3.1, 6.4, 21.7) -> x
```

2.1 Vectors

Els vectors són el tipus de dades més bàsic. Els vectors poden ser pensats com moltes cel·les contigües amb dades. Hi ha sis tipus atòmics de vectors: lògic, enter, real, complex, string(o char) i raw. També tenim arrays i matrius que són vectors que guarden informació en més d'una dimensió fent servir el paràmetre *dim*. Números o strings *i.e. 4.2, "hola", 32* també són vectors de tamany 1. Els vectors es declaren de la següent manera:

```
> v<-c(1,2,3,4,5)
> v
[1] 1 2 3 4 5
> n<-c("a","d","r","i","a")
> n
[1] "a" "d" "r" "i" "a"
```

2.2 Llistes

Les llistes o vectors genèrics són un altre tipus de dades molt important. En aquest cas però cada element d'una llista pot ser un objecte d'R. Els elements d'una llista poden ser accedits de tres maneres diferents. Per exemple, considerant la següent llista:

```
> movie <- list(title='Monty Python\'s The Meaning
of Life', year=1983, cast=c('Graham Chapman','John
Cleese','Terry Gilliam','Eric Idle','Terry
Jones','Michael Palin'))
```

Es poden accedir o bé com si fos un vector fent servir "[ ]" i la posició de l'element que es vol accedir, o bé escrivint l'element que es vol (semblant a com fariem amb C i els seus structs) fent servir [[ ]] o $:

```
> movie$title
[1] "Monty Python's The Meaning of Life"

> movie[2]
$year
[1] 1983

> movie[["cast"]]
[1] "Graham Chapman" "John\nCleese"   "Terry Gilliam"
"Eric Idle"
[5] "Terry\nJones"   "Michael Palin"
```

2.3 Data frames

Un data frame és una taula on cada columna conté el valor d'una variable i cada fila conté el conjunt de valors de cada columna. En un data frame cada columna ha de tenir el mateix nombre d'elements i el nom de les files hauria de ser únic.

```
emp.data <- data.frame(
   emp_id = c (1:5),
   emp_name = c("Rick","Dan","Michelle","Ryan","Gary"),
```

```
   salary = c(623.3,515.2,611.0,729.0,843.25),
   start_date = as.Date(c("2012-01-01", "2013-09-23",
    "2014-11-15", "2014-05-11","2015-03-27")),
   stringsAsFactors = FALSE
   #per indicar si tractar els strings com a text
)
print(emp.data)

 emp_id    emp_name     salary     start_date
1      1     Rick        623.30     2012-01-01
2      2     Dan         515.20     2013-09-23
3      3     Michelle    611.00     2014-11-15
4      4     Ryan        729.00     2014-05-11
5      5     Gary        843.25     2015-03-27
```

2.4 Funcions

Les funcions són objectes de primera classe en R. Poden ser passats com a argument a altres funcions i retornats com a valor d'altres funcions. La sintàxis per a escriure una funció és la següent:

```
function (arglist) body
```

Normalment les funcions s'assignen a símbols però no és necessari, a les funcions que no se'ls hi posa nom les anomenem funcions anònimes. Veiem exemples simples de funcions en R:

```
> echo <- function(x) print(x)
> echo(c(1,2,3,4))
[1] 1 2 3 4

> (function(x) x*x) (c(1,2,3,4,5))
[1]  1  4  9 16 25
```

3 Paradigma de Programació

- **Tipat**: Dèbilment tipat i d'assignació dinàmica

- **Imperatiu**: És fan servir declaracions on es descriu una seqüència de passos que canvien l'estat del programa.
- **Orientat a objectes**: Tot en R són objectes, per això es pot efectuar programació orientada a objectes.
- **Funcional**: Disposa de diverses eines per a crear i manipular funcions.
- **Interpretat**: R és un llenguatge interpretat i això fa que el temps que se li dedica per a esciure codi sigui molt menor. La desavantatge que té és que el temps de computació és major.
- **Reflexiu**: Es pot modificar la estructura i comportament en temps d'execució.
- **Orientat a arrays**: En R és un llenguatge orientat a arrays ja que les operacions s'apliquen un cop a cada element de tot l'array.

3.1 Funcional

Tal i com s'ha mencionat a l'apartat 2.4, les funcions son objectes de primera classe i qualsevol cosa que es pugui fer amb un vector també es pot fer amb una funció: assignar-les a variables, guardar-les en llistes, passar-les com a paràmetres, crear-les dintre d'altres funcions o retornar-les com a resultat d'una funció. A continuació veurem uns quants exemples per a veure diferents característiques funcional de R:

3.1.1 Closures

Com podem veure a continuació, en R podem fer les denominades *closures* on una funció està escrita per una altra funció. Això pot ser útil en el cas que volguem tenir dos nivells de paràmetres: el nivell pare que controla la operació i la del fill que és el que fa la feina. En el següent exemple s'aprofita la idea per a crear una funció pare *power()* que crea la funció fill *square()*.

```
power <- function(exponent) {
function(x) {
x ^ exponent
}
```

```
}
square <- power(2)
square(2)
[1] 4
```

3.1.2 Lapply

En aquest exemple veiem una funció molt important en la part funcional de R, la funció lapply(); lapply() agafa tres inputs: una llista, una funció i els diferents paràmetres que se li passen a la funció. Llavors s'aplica la funció a cada element de la llista i retorna una llista nova de la mateixa mida.

A més de lapply() també existeixen altres variants com sapply(), vapply() que produeixen vectors, arrays o matrius com a output en comptes de de llistes.

```
> lapply(c(3,4), function(x,y) x-y, 4)
[[1]]
[1] -1

[[2]]
[1] 0

> fun2 <- list(mean, sum, median)
> x<- c(1,2,3,4,5)
> lapply(fun2, function(f) f(x))
[[1]]
[1] 3

[[2]]
[1] 15

[[3]]
[1] 3
```

3.1.3 Map, Reduce i Filter

Tot llenguatge funcional té tres eines bàsiques per a manipular llistes: Map(), Reduce() i Filter().

- **Map()**, té una funcionalitat molt semblant a la funció lapply explicada anteriorment. En aquest cas Map() itera sobre múltiples estructures de dades en paral·lel. A lapply() només un argument de la funció varia, els altres estan fixos, en canvi amb Map() no és així.
- **Reduce()**, redueix un vector a un sol valor cridant recursivament una funció. Agafa els dos primers valors i els combina, del resultat en combina el tercer i va fent. Necessita dos paràmetres: la funció i el vector.
- **Filter()**, aquesta funció redueix un vector a un altre seleccionant els elements que compleixen un predicat definit per el resultat d'una funció. Necessita dos paràmetres: la funció i el vector.

```
> Reduce(function(a, b) a + b, 1:10)
[1] 55

> Reduce(‘-‘, 1:3)
[1] -4

> Filter(function(i) i %%3 == 0, c(1,2,3,4,5,6,7,8,9))
[1] 3 6 9

>> Map(function(x,y) x^y, c(9,2), c(3,5))
[[1]]
[1] 729

[[2]]
[1] 32
```

3.1.4 Lazy evaluation

En R disposem de lazy evaluation, és a dir els paràmetres no es calculen fins que es necessiten. Per a mostrar-ho s'ha realitzat el següent exemple. Com podem veure a la funció *sum.of.squares* necessitem dos paràmetres: x, i la seva mitjana. Posteriorment es crida a la funció amb un vector com a x. Aquest vector conté NA, és a dir li falten dades. No obstant, degut a que no es necessita calcular el paràmetre about en un principi, podem aprofitar la laziness per a extreure aquest missing value i calcular-ho després.

```
> sum.of.squares = function(x, about = mean(x)) {
x = x[!is.na(x)]
sum(square(x - about))
}

> sum.of.squares(c(-1, 1, NA))
[1] 2
```

3.2 Orientat a objectes

Un objecte és una estructura de dades amb atributs i mètodes que actuen en els atributs. En el cas de R tenim tres tipus de sistemes de classes. Els anomenats S3, S4 i classe de referència.

• S3, la majoria de classes built-in en R són d'aquest tipus. És el tipus de classe més primitiu i es pot crear afegint un atribut de classe a un objecte.

```
> s <- list(name = "John", age = 21, GPA = 3.5)
> class(s) <- "studentS3"
> s
$name
[1] "John"

$age
[1] 21
```

```
$GPA
[1] 3.5

attr(,"class")
[1] "studentS3"
```

• S4, són una millora de S3. Tenen una estructura definida que ajuda a que els objectes d'una mateixa classe tinguin un aspecte similar. Els components es defineixen fent servir la funció *setClass* i els objectes són creats fent servir la funció *new()*.

```
> setClass("studentS4", slots=list(name="character",
 age="numeric", GPA="numeric"))
> s <- new("studentS4",name="John", age=21, GPA=3.5)
> s
An object of class "studentS4"
Slot "name":
[1] "John"

Slot "age":
[1] 21

Slot "GPA":
[1] 3.5
```

• Classe de Referència, es van introduir posteriorment i són el més semblant a la majoria de llenguatges OOP. Són bàsicament S4 amb un entorn afegit.

```
> student <- setRefClass("student",
  fields = list(name = "character",
age = "numeric", GPA = "numeric"))
> s <- student(name="Adri", age = 20, GPA= 3.0)
> s
Reference class object of class "student"
Field "name":
[1] "Adri"
Field "age":
[1] 20
Field "GPA":
[1] 3
```

3.3 Reflexiu

R té l'habilitat d'examinar i modificar la seva estructura i comportament en temps d'execució. Per a treballar amb les característiques reflexives de R tenim funcions com *get*, *getAnywhere*, *showMethods*, *assign* o *eval*.

```
> f <- function (x) x+1
> get("f")(3)
[1] 4

> eval(parse(text="mean(1:5)"))
[1] 3
```

4 Funcions i operadors aritmètics

Els operadors aritmètics elementals són +, -, *, / i ^. A més també hi ha varies funcions aritmetiques disponibles com log, exp, sin, cos, tan, sqrt; max i min, que selecciona l'element més gros i més petit d'un vector; range que et retorna el c(min(x),max(c)); length(x), que retorna el nombre d'elements; sum(x) o prod(x).

```
> c <- c(1,2,3,4,5)
> sqrt(c)
[1] 1.000000 1.414214 1.732051 2.000000 2.236068
> range(c)
[1] 1 5
> sin(c)
[1]  0.8414710  0.9092974  0.1411200 -0.7568025 -0.9589243
> prod(c)
[1] 120
```

5 Usos

R s'usa principalment per a data science: visualització i anàlisis de dades, previsions, estadística, machine learning... Això és degut a que R té diverses característiques que el fan òptim per a aquesta tasca:

- Llenguatge interactiu. L'anàlisis de data és inherentment un procés interactiu.
- Gràfics
- Estructures de dades
- Funcions com a objectes de primera classe
- Comunitat i múltiples paquets
- Té *missing values*

A més, té incloses moltes funcions per a efectuar operacions estadístiques no trivials, des de regressions linears o logístiques, arbres de decisió fins a xarxes neuronals.

Algunes companyies que fan servir R i els seus usos:

- **ANZ**, el quart banc més gran d'Austràlia. Per a analitzar el risc de crèdit.
- **Ford**. Per a millorar el disseny dels vehicles.
- **Twitter**. Per a monitoritzar l'experiència d'usuari.
- **The New York Times**, per a organitzar i preparar gràfics per al seu diari o pàgina web.
- **The Human Rights Data Analysis Group**, per a quantificar l'impacte de la guerra.

6 Llenguatges similars

Dintre el camp dels llenguatges similars podem tenir diversos aspectes en compte i depenent de cada aspecte ens trobem amb uns llenguatges o altres:

En el grup dels llenguatges usats per a estadística i càlcul hi entrarien **Python**, el seu gran contrincant en el camp del Data Science, **Octave**, **Matlab** i **Julia**. En quan a sintaxi trobem l'**S**, llenguatge del qual deriva.

En aquest aspecte les diferències són mínimes. Per exemple, l'R permet "=" per a assignacions a més del " ←". A més també té similituds superficials amb **C**.

En quan a semàntica tindríem una varietat de llenguatge de programació funcional i podríem trobar llenguatges com **Lisp** i **APL**.

Bibliografia

[1] H. Wickham, *Advanced R*
Souce: http://adv-r.had.co.nz/

[2] *Statistics with R*. Source: http://zoonek2.free.fr/UNIX/48_R/all.html

[3] L. Lamport, *LATEX: a document preparation system: user's guide and reference manual* (Addison-Wesley Pub. Co., cop. 1994)

[4] *An introduction to R* (W. N. Venables, D. M. Smith and the R Core Team) Source:https://cran.r-project.org/doc/manuals/R-intro.pdf

[5] *What is R programming used for?*
Source: https://www.quora.com/What-is-R-programming-used-for

[6] *R, llenguatge de programació*. Source: https://ca.wikipedia.org/wiki/R_llenguatgeprogramacio

[7] *R Docs*. Source: https://cran.r-project.org/doc/manuals/r-release/R-lang.html

[8] *Apply, lapply, sapply functions in R*, Source:https://www.r-bloggers.com/apply-lapply-rapply-sapply-functions-in-r/

[9] *R Classes and Objects*, Source: https://www.datamentor.io/r-programming/object-class-introduction/