# Modelos y paradigmas de programación

## Análisis del lenguaje de programación R

Harold Armando Achicanoy Estrella


### Resumen

El objetivo de este artículo es presentar una corta revisión a partir de los modelos y paradigmas de programación vistos durante el curso para el lenguaje de programación R. R es un lenguaje de programación y ambiente de desarrollo usado ampliamente para el análisis estadístico y la visualización de datos. El lenguaje de programación soporta los paradigmas: Orientado a objetos y funcional. Por otro lado, no soporta el paradigma declarativo, ni concurrencia alguna.


### Introducción

R es un lenguaje de programación y entorno de desarrollo gratuito de código abierto usado ampliamente para el análisis estadístico y la visualización de datos. Su surgimiento ocurre a inicios de los años 90 por dos profesores de la Universidad de Auckland en Nueva Zelanda: Ross Ihaka y Robert Gentleman. La primera versión del lenguaje fue lanzada en agosto de 1993, no obstante, la versión estable de R estuvo disponible desde febrero del 2000. Los orígenes de R provienen del lenguaje de programación S el cual fue desarrollado en 1970 por John Chambers y su equipo de trabajo en los laboratorios Bell cuyo fin también estuvo destinado al análisis estadístico de datos.

R fue altamente influenciado por S y Lisp-Stat, estos ambientes alentaron a los estadísticos a convertirse en programadores de tal manera que ellos puedan tener un control más efectivo sobre el proceso de análisis de datos (Gentleman & Ihaka, n.d.). Adicionalmente, R permite la integración con otros lenguajes de programación, como C/C++, Java, Python, entre otros. Puede ser usado prácticamente en cualquier plataforma y sistema operativo tales como Unix, Linux, Windows o Mac OS y es hoy en día el lenguaje de programación para análisis estadístico más popular que está siendo usado por gigantes de la industria como Facebook y Google.

Desde el punto de vista de paradigmas de programación, R es un lenguaje de programación imperativo lo que indica que tiene estado explicito, soporta una componente de evaluación perezosa de las funciones creadas y el enfoque orientado a objetos, convirtiéndolo en un lenguaje de tipo multi-paradigma.

A continuación se describe el contenido del documento en función de las secciones a desarrollar: en primera medida se presenta el entorno, la sintaxis y los tipos básicos de datos usados en R, posteriormente se muestra una revisión de los principales paradigmas de programación implementados en R así como sus respectivas pruebas. Finalmente, se presentan las conclusiones obtenidas del análisis del lenguaje.

### Entorno, sintaxis y tipos básicos de objetos usados en R

R presenta un desarrollo vertiginoso produciendo al año un promedio de 2 a 3 actualizaciones en las cuales se van corrigiendo detalles o agregando nuevas características en función de obtener un buen desempeño del lenguaje y sus correspondientes retos. La interfaz de R nos presenta una consola de análisis donde se ingresan las líneas de código correspondientes a lo que se desee ejecutar. En la Figura 1 se muestra la consola de R.

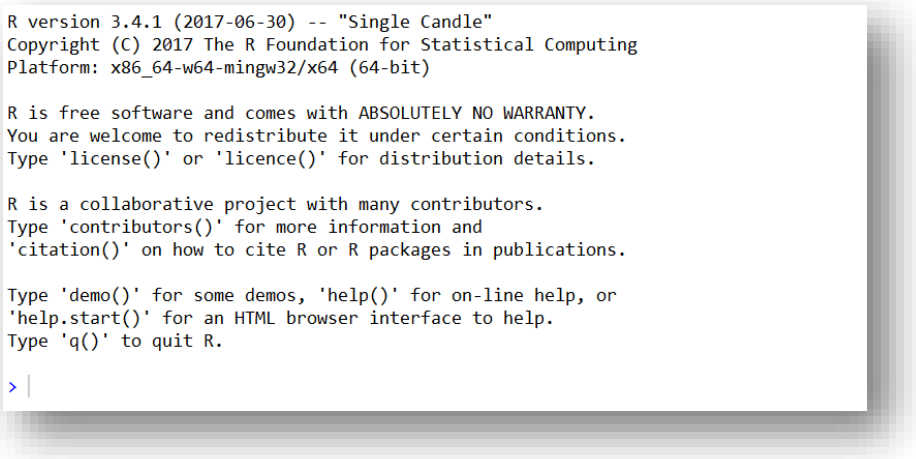


*Figura 1. Consola de R*

El tipo básico de datos usado en R es el vector, que corresponde a una colección ordenada de valores del mismo tipo. Los tipos de valores pueden ser numéricos, caracteres o lógicos. Otros tipos de datos incluyen las listas que se denominan vectores heterogéneos debido a que pueden tener valores de diferentes tipos y las funciones (Morandat, Hill, Osvald, & Vitek, 2012).

El operador básico de asignación para definir el contenido de una variable es el símbolo:

```
<-
```

Por tanto para declarar como contenido de la variable X el valor 5, se emplearía la siguiente expresión:

```
X <- 5
```

Donde X constituye un vector de longitud 1 el cual es declarado en el entorno global de R y donde el contenido de la variable se puede asignar y reasignar tantas veces se desee a diferencia de un lenguaje de programación de tipo declarativo. Las operaciones aritméticas se pueden desarrollar sobre los vectores elemento a elemento. Otros tipos de datos pueden ser creados a partir de los vectores, como las matrices, arreglos, y marcos de datos. La evaluación de las funciones se realiza de forma perezosa y el alcance del lenguaje es dinámico.

***Sintaxis del lenguaje núcleo de R***

La sintaxis del lenguaje núcleo de R, consiste de expresiones denotadas por la letra **e**, y abarca:

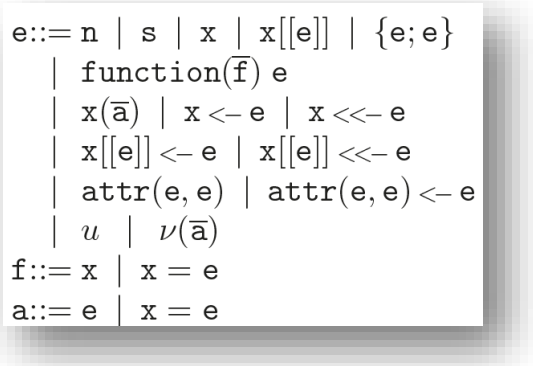


*Figura 2. Sintaxis de R*

**n**: literales numéricos
**s**: literales de texto (string)
**x**: símbolos
**x[[e]]**: acceso a arreglos
**{e; e}**: bloques
**function(f) e**: declaración de funciones
**x(a)**: llamada de funciones
**x <- e**: asignación de variables
**x <<- e**: súper-asignación de variables
**x[[e]] <- e**: asignación de arreglos
**x[[e]] <<- e**: súper-asignación de arreglos
**attr(e, e)**: extracción de atributos
**attr(e, e) <- e**: asignación de atributos

Adicionalmente, las expresiones pueden incluir también valores, **u**, y llamadas a funciones parcialmente reducidas, **v(a)**. Los parámetros en la declaración de una función, denotados por f pueden ser variables o variables con un valor por defecto o una expresión. Argumentos simétricos de las invocaciones a funciones, denotadas por la letra **a** son expresiones que pueden ser nombradas a través de símbolos. Con estos elementos se compone el lenguaje núcleo de R.

## Paradigmas de programación implementados en R

***Programación orientada a objetos***

Existen dos tipos de valores en programación orientada a objetos implementados en R, los objetos de tipo S3 implementados en 1990 que soportan polimorfismo ad-hoc y consisten en listados de cadenas de texto usadas para los métodos de envío y los objetos de tipo S4 implementados en 1998 que agregaron clases apropiadas y múltiples métodos.

En el modelo de objetos S3 no hay entidades de clase ni prototipos, aquí los objetos corresponden a tipos de valores base de R etiquetados por un atributo llamado clase cuyo valor es un vector de strings. Como todos los atributos, un objeto clase se puede actualizar en cualquier momento. Dado que no se requiere ninguna estructura en los objetos, dos instancias que tienen la misma etiqueta de clase pueden ser completamente diferentes.

```
> who <- function(x) UseMethod("who")
> who.man <- function(x) print("Ceasar!")
> who.default <- function(x) print("??")
> me <- 42; who(me)
[1] "??"
> class(me) <- "man"; who(me)
[1] "Ceasar!"
```

*Figura 3. Creación de un objeto tipo S3*

En el modelo tipo S4, se admite la herencia múltiple, y se permite la herencia repetida, pero solo afecta el algoritmo de envío del método. Cuando una clase hereda un espacio con el mismo nombre desde dos caminos diferentes, la etiqueta de clase que proviene de la primera superclase se conserva y las etiquetas de los otros padres deben ser subclases de esa etiqueta. Las clases pueden ser redeclaradas en cualquier momento.

```
> setClass("Point", representation(x="numeric", y="numeric"))
> setClass("Color", representation(color="character"))
> setClass("CP", contains=c("Point","Color"))
> l <- new("Point", x = 1, y = 1)
> r <- new("CP", x = 0, y = 0, color = "red")
> r@color
[1] "red"
>
> setGeneric("add", function(a, b) standardGeneric("add"))
[1] "add"
> setMethod("add", signature("Point", "Point"),
+           function(a, b) new("Point", x= a@x+b@x, y=a@y+b@y))
[1] "add"
> setMethod("add", signature("CP", "CP"),
+           function(a, b) new("CP",x=a@x+b@x,y=a@y+b@y,color=a@color))
[1] "add"
> add(r, l)
An object of class "Point"
Slot "x":
[1] 1

Slot "y":
[1] 1
```

*Figura 4. Creación de clases y métodos con objetos tipo S4*

## Conclusiones

El lenguaje de programación R de amplio uso actualmente, aunque toma ciertos conceptos de otros paradigmas de programación está centrado en la programación funcional y orientada a objetos. Pudiéndose considerar un lenguaje multi-paradigma.

## Referencias


Gentleman, R., & Ihaka, R. (n.d.). Lexical scope and statistical computing.

Morandat, F., Hill, B., Osvald, L., & Vitek, J. (2012). Evaluating the Design of the R Language. *ECOOP 2012 – Object-Oriented Programming*, *7313*, 104–131. https://doi.org/10.1007/978-3-642-31057-7_6