Ingeniería y Competitividad
ISSN: 0123-3033
inycompe@gmail.com
Universidad del Valle
Colombia

Velásquez, Juan D.; Montoya, Olga L.; Castaño, Natalia
¿Es el proyecto R para la computación estadística apropiado para la inteligencia computacional?
Ingeniería y Competitividad, vol. 12, núm. 2, 2010, pp. 81-94
Universidad del Valle
Cali, Colombia

Ingeniería y Competitividad, Volumen 12, No. 2, p. 81 - 94 (2010)

¿Es el proyecto R para la computación estadística apropiado para la inteligencia computacional?

Juan D. Velásquez[*§], Olga L. Montoya[*], Natalia Castaño[*]

Facultad de Minas, Universidad Nacional de Colombia, Sede Medellín
§ e-mail: jdvelasq@unal.edu.co

(Recibido: Enero 27 de 2010 – Aceptado: Diciembre 7 de 2010)

Resumen

En este artículo, se revisa el proyecto R para el cálculo estadístico y gráficos. Se presenta una revisión de las principales características del ambiente y del lenguaje de programación subyacente. Se muestra la utilidad de esta herramienta para programar paradigmas de las ciencias de la computación por medio de varios ejemplos. Finalmente, se argumenta por qué el lenguaje R es una herramienta interesante para desarrollar software en el campo de la inteligencia computacional.

Palabras Claves: Projecto-R, Lenguajes de programación, Redes neuronales, Sistemas difusos, Neurocomputación.

¿Is the R project for statistical computing appropriate for computational intelligence?

Abstract

In this paper it is reviewed the R project for statistical computing and graphics. It is presented a review of the main features of the environment and the underlying programming language. It is argumented the utility of this tool for programming computer sciences paradigms by mean of several examples. Finally, it is argued why R language is an interesting tool for developing software in the computational intelligence field.

Keywords: R-project, Programming languages, Neural networks, Fuzzy systems, Neurocomputation.

1. Introducción

Resulta indudable que los avances que se han dado en el desarrollo del software y del hardware han impactado enormemente el desarrollo de las ciencias y la ingeniería, tal que estas últimas se han convertido en clientes naturales de los desarrollos que se den en las ciencias de la computación. Más aun, los recursos que tienen tanto el investigador y el profesional para realizar cálculos y gráficos complejos juegan un papel fundamental dentro del desempeño profesional y científico. Un resultado directo es que el término "computacional" se ha venido acuñando en diferentes disciplinas con el ánimo de enfatizar su orientación hacia el uso de métodos basados en el uso intensivo de complejos cálculos numéricos y en los lenguajes más apropiados para el desarrollo de procesos algorítmicos; algunos ejemplos son:

• La estadística computacional (Sawitzki, 2009) cobija tanto avances metodológicos en el área de la estadística que influyen en desarrollos computacionales como en sentido inverso. El área se diferencia de la estadística tradicional, en que hay una fuerte influencia de los aspectos computacionales sobre los desarrollos tanto matemáticos y teóricos como en los algorítmicos y su implementación.

• La inteligencia computacional (Sumathi & Paneerselvam, 2010) que es una rama de la inteligencia artificial enfocada a simular el comportamiento inteligente usando sistemas que requieren computación intensiva, tal como las redes neuronales artificiales, algoritmos bio-inspirados (computación evolutiva, enjambres, etc.), sistemas difusos y sistemas inteligentes híbridos.

• Las finanzas computacionales (Los, 2001; Seydel, 2009) que se enfocan en resolver problemas financieros que incluyen la valoración de opciones y derivados, estructuración de portafolios, riesgo financiero, valoración de activos financieros y el pronostico de precios a partir de metodologías basadas en el uso intensivo del computador, tales como simulación, inteligencia computacional o estadística computacional.

• La econometría computacional (Belsley & Kontoghiorghes, 2009; Kleiber & Zeileis, 2008; Pfaff, 2008; Vinod, 2008) en que priman los aspectos relacionados con las herramientas computacionales disponibles, estimación numérica de modelos, algoritmos de optimización, modelado de series de tiempo no lineales, estimación de propiedades estadísticas a partir de métodos numéricos, etc.

Consecuentemente, se ha tratado de dotar a los lenguajes clásicos de programación con librerías numéricas que faciliten el desarrollo de algoritmos numéricos complejos y la construcción de gráficos, tal como GSL (http://www.gnu.org/software/gsl/), IMSL o Scinet Math; pero la principal dirección de desarrollo es la construcción entornos computacionales que usan lenguajes diseñados específicamente para estas tareas; entre los más conocidos y empleados se encuentran:

• Basados en la manipulación de matrices: MATLAB, IDL (y sus clones como GNU Octave, FreeMat, Fawlty, Rlab, GNU data language y Jasymca), GAUSS, SciLab y OxMatrix.

• Basados en notaciones matemáticas o algebraicas como APL, MAPLE o Mathematica.

• Basados en la extensión de lenguajes generales de programación como PDL (Perl Data Language) o PythonXY.

Finalmente, también se ha dado el desarrollo de ambientes especializados en tareas específicas que incluyen su propio lenguaje, tal como S-Plus o R. Particularmente, el lenguaje de programación R para la computación estadística es un ambiente para realizar cálculos estadísticos que es ampliamente aceptado y usado por dicha comunidad científica. No obstante, ha venido tomando fuerza en otras comunidades científicas aunque es casi completamente desconocido en la comunidad dedicada a las ciencias de la computación.

El objetivo de este artículo es argumentar y presentar ejemplos de por qué el lenguaje R es de interés para profesionales e investigadores pertenecientes al área de las ciencias de la

computación.

Este artículo está organizado como sigue: en la Sección 2 se presenta una revisión del entorno de programación y su lenguaje; seguidamente, se listan las fuentes de información disponibles en la Sección 3. Varios ejemplos ilustrativos se desarrollan en la Sección 4. Una discusión sobre la utilidad de R en inteligencia computacional es presentada en la Sección 5. Finalmente, se concluye en la Sección 6.

2. El entorno de programación R y su lenguaje

El entorno, desarrollado originalmente por Ihaka & Gentleman (1996), implementa un lenguaje de programación que es un clon de los lenguajes S diseñado originalmente en AT&T Laboratories (Becker, et al, 1988; Chambers, 1998; Chambers & Hastie, 1992) y S-Plus (que es la versión comercial de S); S es un lenguaje específicamente diseñado para la visualización de datos y la exploración, modelado estadístico y programación con datos (Insightful, 2007). De ahí, que muchos programas escritos en S puedan ejecutarse en S sin modificaciones. La interacción con el usuario se basa en una interfaz de línea de comandos que puede resultar intimidante inicialmente para el usuario, pero que resulta apropiada para la manipulación interactiva de datos; en la Figura 1 se ilustra la interfaz bajo el sistema operativo Windows. De ahí, que se hayan diseñado interfaces alternativas de usuario con el ánimo de facilitar el uso del entorno; ellas incluyen: R-Commander (Fox, 2005), el Integrated Computing Environment for R (Sriplung, 2006) y Tinn-R (http://www.sciviews.org/Tinn-R/).

El entorno R es un software libre en código fuente bajo la definición dada en la licencia GNU (General Public Licence) de la FSF (Free Software Fundation), el cual puede descargarse de la Internet ya sea como código fuente o como distribuciones que corren en Linux (Debian, Redhat, SUSE o Ubuntu), Windows o MacOS. A la fecha se encuentra disponible la versión 2.10.1.

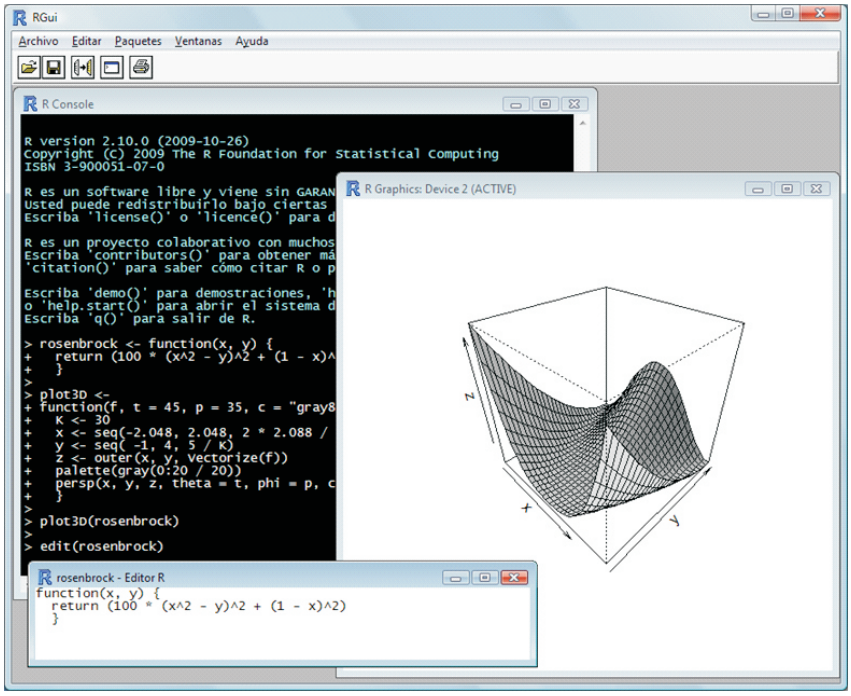


Figura 1. Entorno de R para el sistema operativo Windows.

Tanto el entorno como todo el material complementario pueden obtenerse en http://www.r-project.org/ o en cualquiera de los servidores web o ftp pertenecientes a CRAN.

La sintaxis del lenguaje R es similar, al menos superficialmente, a la de C y C++ (Grunsky, 2002), pero su semántica sigue los paradigmas de la programación funcional y la programación orientada a objetos, tal como lo hacen lenguajes como LISP y Scheme; esto último implica que el lenguaje tiene la capacidad de manipular directamente los objetos del lenguaje, aplicar reglas de sustitución y evaluar expresiones.

R es un lenguaje orientado a objetos, tal que, inclusive los tipos de datos más básicos, tales como: booleanos, enteros, reales, caracteres, vectores, matrices, listas y hojas de datos son objetos mismos. Esta característica permite que el usuario interactúe de forma transparente, ya que las llamadas se realizan a funciones genéricas, como print, summary o plot, las cuales determinan internamente que método debe ser llamado dependiendo de la clase de objetos a las que pertenecen sus argumentos. El sistema también permite que el usuario defina sus propias clases específicas y los métodos correspondientes para cada clase (Grunsky, 2002).

El ambiente de programación (sistema) cuanta con:

• Mecanismos para la manipulación y almacenamiento de grandes cantidades de datos de manera eficiente y rápida; en este tópico se incluyen mecanismos para escritura y lectura de datos, acceso a bases de datos, manejo de fechas, tablas indizadas, manipulación de caracteres y agregación de datos (Spector, 2008).

• Una amplia colección de herramientas estadísticas para el análisis de datos. El sistema base permite diferentes cálculos relacionados con el uso de distribuciones de probabilidad; para una distribución paramétrica, el usuario puede: generar números aleatorios, calcular sus parámetros a partir de una muestra de datos o calcular su función de densidad de probabilidad; igualmente, el sistema brinda otras herramientas , para el análisis de la distribución de los datos, tales como: histogramas, gráficos Q-Q, contrastes de normalidad. Igualmente, el sistema base da facilidades para ajustar de modelos aditivos generalizados y realizar análisis de varianza y comparación de modelos. Sin embargo, el sistema no está limitado a los modelos mencionados y brinda facilidades para trabajar con árboles de regresión y clasificación, modelos generalizados, regresión robusta y modelos mixtos. Igualmente, el sistema cuenta con funciones especializadas para diferentes áreas como inferencia bayesiana (Hoff, 2009; Albert, 2009), econometría computacional, finanzas empíricas, estadística multivariada, análisis de series de tiempo (Pfaff, 2008; Cowpertwait & Metcalfe, 2009), aprendizaje estadístico y de máquinas, diseño de experimentos, econometría (Kleiber & Zeileis, 2008), modelos dinámicos lineales (Petris, et al, 2009), métodos Monte Carlo (Robert & Casella, 2010), entre otros.

• Herramientas de alto nivel para la construcción de gráficos y su posterior análisis; véase a (Correa y González 2002). El entorno cuenta con una gran cantidad de funciones, tanto primitivas como de alto nivel, para construir diversos tipos de gráficos en 2D y 3D. Entre ellos se encuentran: histogramas, árboles de tallo y hoja, boxplots, gráficos de barra, gráficos en coordenadas polares, gráficos de dispersión, series de tiempo, superficies, contornos, entre otros.

• Un mecanismo de extensión de la funcionalidad del entorno a través de paquetes (Gentleman, 2008a, 2008b). El lenguaje puede extenderse mediante el uso de paquetes, los cuales pueden incluir rutinas compiladas usando Fortran 77 o lenguaje C. En el momento, hay 8 paquetes suministrados con la distribución base de R, pero a la fecha, hay una colección de 2162 paquetes disponibles en CRAN, y la cual se expande continuamente; dichos paquetes traen una serie de funciones básicas para realizar cálculos numéricos, pero también presenta funciones matemáticas y estadísticas avanzadas tales como la estimación de modelos de clasificación, cópulas, análisis multivariante. En su estructura, cada paquete está compuesto por un conjunto de variables que contienen datos, y un grupo de funciones que permiten ejecutar cálculos, realizar

gráficos e imprimir resultados. Para obtener cualquier paquete de interés basta visitar el sitio www.r-project.org y dar clic en el link CRAN, en el cual se encuentra una lista de contribuciones realizadas para R listas para descargarse.

• Un lenguaje de programación, simple y efectivo, que incluye condicionales, saltos, definición de funciones recursivas y fácil manejo de los datos de entrada y salida. Operadores para ejecutar cálculos sobre vectores y matrices.

• Un sistema para la depuración de código y manejo de excepciones.

3. Información disponible sobre R

Existe publicada una cantidad muy importante de información sobre el entorno y sus aplicaciones. La información disponible incluye:

• Los manuales de referencia

• FAQ (frequently answer questions)

• Tutoriales escritos por los usuarios

• Listas de correo

• El periódico R-News

• Un wiki (http://wiki.r-project.org)

• Publicaciones arbitradas y seriadas en las que se describen nuevos paquetes y sus funcionalidades: "The R Journal" y "Journal of Statistical Software".

• Más de 96 libros publicados desde 1998 hasta la fecha, de los cuales algunos se encuentran disponibles libremente en formato pdf.

• Memorias de conferencia: "International R User Conference" y "Directions in Statistical Computing"

Existen varios sitios de Internet donde se actualiza constantemente la información sobre la herramienta. Entre los principales se encuentran:

• www.r-project.org

• www.inside-r.org

• Los sitios donde se publican las memorias de la "International R User Conference"; su última versión se encuentra disponible en http://user2010.org/

• www.rseek.org

• http://finzi.psych.upenn.edu/search.html

• R graphical manual (http://bm2.genes.nig.ac.jp/RGM2/index.php?clear≤all)

• The R wikibook (http://en.wikibooks.org/wiki/R_Programming)

• R bloggers (http://www.r-bloggers.com/)

4. Algunos ejemplos de R

El objetivo de esta sección es ilustrar algunas de las capacidades del lenguaje R a través de ejemplos típicos de las Ciencias de la Computación y de la Inteligencia Computacional.

4.1 Abstracciones de alto nivel

R es un lenguaje de programación que le permite al usuario expresar procedimientos complejos utilizando principios fundamentales que incluyen recursión, iteración y formulación de abstracciones de alto nivel.

Un ejemplo típico de la programación funcional es expresar el concepto factorial de un número como una función recursiva que genera un proceso recursivo:

```
> fact <- function(n)  {
+   if (n == 0 || n == 1)
+     return(1)
+   else
+     return (n * fact(n - 1))
+   }
>
> fact(4)
[1] 24
```

O como una función recursiva que genera un proceso iterativo a partir del concepto de

sustitución:

```
> fact <- function(n)  {
+          fiter   <-   function(counter,
accumulator) {
+     if (counter == n )
+        accumulator * n
+     else
+           fiter( counter + 1, counter *
accumulator )
+     }
+   fiter(1, 1)
+   }
>
> fact(4)
[1] 24
```

Es posible construir abstracciones complejas de alto nivel a partir de las siguientes características del lenguaje:

• Las funciones pueden pasarse como parámetros.

• Es posible crear funciones anónimas.

• Las funciones pueden escribirse como funciones generales.

• Es posible retornar una función como resultado de una función.

Para ejemplificar lo anterior, se presenta a continuación la codificación en R del concepto sumatoria de una función $f(x)$, y su uso utilizando una función anónima que representa la función identidad $f(y) \leq y$; se ejemplifica su uso para calcular la sumatoria de la serie 1, 2, 3, 4, 5.

```
> summatory <- function(f, x) {
+   accumulator <- 0
+       for  (e  in  x)  accumulator  <-
accumulator + f(e)
+   return (accumulator)
+   }
>
> summatory( function(y) y, 1:5 )
[1] 15
```

Finalmente, la función anterior podría usarse para calcular 1!+2!+3!+4!:

```
> summatory( fact, 1:4)
[1] 33
```

4.2 Construcción de gráficos

En el código presentado a continuación se ilustra la construcción de un gráfico tridimensional de la función de Rosenbrock $f(x,y) \leq 100\ (x^2 - y)^2 + (1 - x)^2$, la cual es comúnmente usada para probar algoritmos de optimización; la función es graficada para x en el intervalo [-2.048, 2.048] y y en [-1, 4]. Los parámetros de la función plot3D controlan los ángulos de graficación, el color y el sombreado.

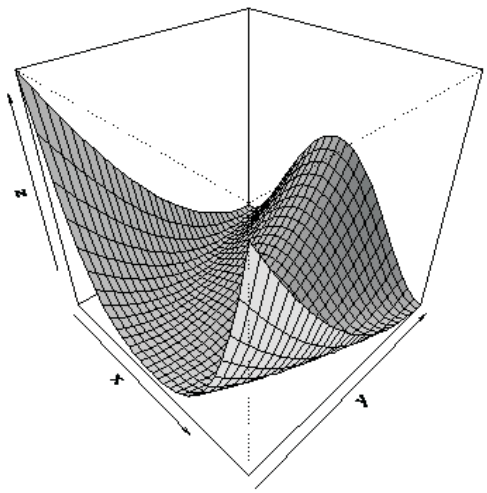


Figura 2. Gráfico 3D de la función de Rosenbrock generada usando R.

```
> rosenbrock <- function(x, y) {
+   return (100 * (x^2 - y)^2 + (1 - x)^2)
+   }
> plot3D <-
+ function(f, t = 45, p = 35, c = "gray85",
s = 0.25) {
+   K <- 30
+   x <- seq(-2.048, 2.048, 2 * 2.088 / K)
+   y <- seq( -1, 4, 5 / K)
+   z <- outer(x, y, Vectorize(f))
+   palette(gray(0:20 / 20))
+   persp(x, y, z, theta = t, phi = p, col =
c, shade = s)
+   }
>
> plot3D(rosenbrock)
```

La gráfica obtenida se presenta en la Figura 2.

4.3 Optimización numérica

El óptimo de una función *f* (x) con una región factible definida por restricciones de la forma L ? x ? U puede ser obtenido numéricamente muestreando la región factible por medio de números aleatorios uniformes; esta técnica es conocida como optimización de Monte Carlo. En el siguiente código se presenta una función genérica que implementa dicha metodología, la cual se aplica a una función que retorna la sumatoria del cuadrado de sus argumentos de entrada.

```
> squares <- function( x ) {
+   return( sum( x ^ 2 ))
+   }
>
> mc <- function(f, x0, L, U, M = 10000)  {
+   f.min <- f( x0 )
+   x.min <- x0
+   for (iter in 1:M) {
+     x  <- L + runif( length(x0) ) * (U -
L)
+     fx <- f( x )
+     if (fx < f.min) {
+       f.min <- fx
+       x.min <- x
+       }
+     }
+   return (x.min)
+   }
>
> mc( f=squares, x0=c(10,10), L=c(-10,-
10), U=c(10, 10))
[1] 0.006802971 0.145513765
```

A continuación se presenta un ejemplo optimizando la misma función, pero usando Temple Simulado.

```
>   optim(par=c(10,10),   fn=squares,
method="SANN",
+ lower=c(-10,-10), upper=c(10, 10))
$par
[1] 3.087949e-20 3.087949e-20

$value
[1] 1.907086e-39
```

4.4 Memoria asociativa lineal

Los tres ejemplos anteriores permiten ejemplificar, de una forma general, el potencial del lenguaje. Velásquez (2010) desarrolla un ejemplo

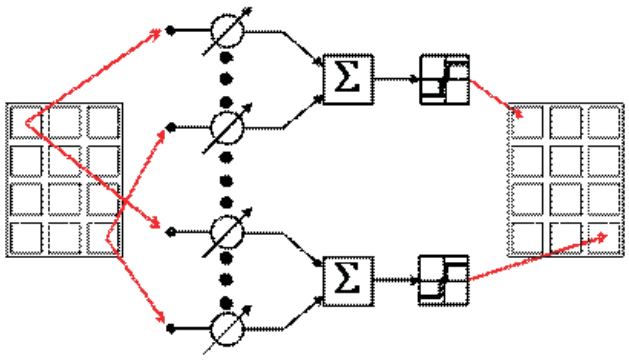
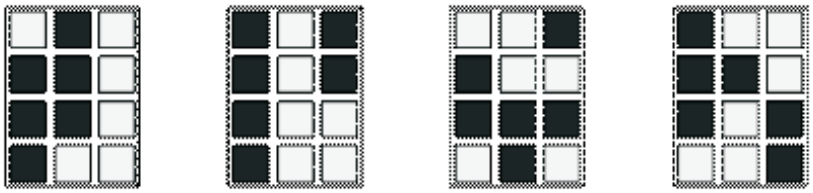
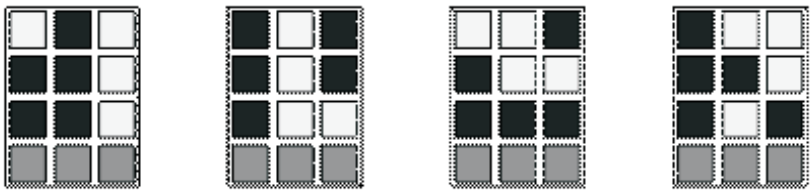


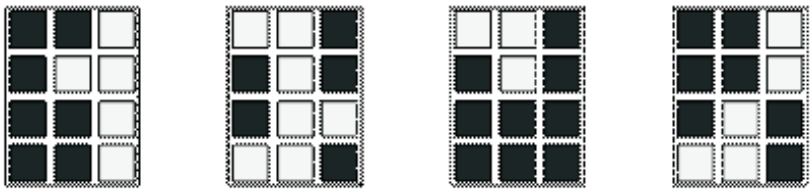
Figura 3. Esquema de una red neuronal autoasociativa que permite recuperar el patrón binario original de salida ante un patrón binario de entrada que está incompleto (se desconoce el valor de algunos bits) o contaminado (algunos de los bits originales se han invertid). (Adaptado de Velásquez (2010)).



a) Patrones de ejemplo para construir una memoria lineal auto-asociativa



(b) Patrones incompletos obtenidos al hacer ceros los elementos correspondientes a la última fila de los patrones originales



(c) Patrones contaminados al invertir aleatoriamente algunos elementos de los patrones originales.

Figura 4. Patrones binarios para implementar y probar una memoria lineal auto-asociativa (adaptado de Velásquez (2010)).

completo mucho más avanzado, en el que se ilustra la implementación de una memoria lineal auto-asociativa usando el sistema de clases S3 del lenguaje R. Esta memoria es un tipo de red neuronal artificial con entradas y salidas binarias donde cada patrón de entrada se asocia consigo mismo. En este caso, se tiene una neurona de entrada y una neurona de salida por bit del patrón de entrada. En la Figura 3 se presenta un diagrama esquemático de dicha red neuronal donde el patrón binario está organizado como una matriz de cuatro filas por tres columnas. Una vez la red neuronal artificial ha memorizado un conjunto de patrones (Figura 4a), ella es capaz de recuperar el patrón original aunque el patrón de entrada este incompleto (se desconoce el valor de algunos bits, como en el ejemplo de la Figura 4b) o contaminado con ruido (Figura 4c).

El ejemplo desarrollado por Velásquez (2010) consta de las siguientes funciones:

• `arrayplot`: es una función para graficar una matriz de valores binarios, como las presentadas en la Figura 4.

• `amemory`: crea una memoria lineal asociativa a partir de los conjuntos de patrones de entrada y de salida.
• `predict`: calcula la salida de una memoria lineal asociativa creada usando `amemory`, dado un patrón de entrada.

Siguiendo el trabajo de Velásquez (2010), los cuatro patrones binarios bipolares de la Figura 4a pueden ser representados usando vectores:

```
>   P1   =   c(+1,-1,-1,-1,-1,-1,-1,+1,+1,+1,+1,+1)
>  P2  =  c(-1,-1,-1,-1,+1,+1,+1,+1,-1,-1,+1,+1)
> P3 = c(+1,-1,-1,+1,+1,+1,-1,-1,-1,+1,-1,+1)
> P4 = c(-1,-1,-1,+1,+1,-1,+1,+1,+1,+1,-1,-1)
```

La Figura 4a es creada usando las variables anteriores y la función `arrayplot`:

```
> par(mfrow = c(1,4), mar = c(1, 1, 1, 1))
> arrayplot(matrix(P1,4,3))
> arrayplot(matrix(P2,4,3))
> arrayplot(matrix(P3,4,3))
> arrayplot(matrix(P4,4,3))
```

La memoria lineal auto-asociativa es creada y almacenada en la variable `M` usando la función `assmem`:

```
> K = cbind(P1,P2,P3,P4)
> M = assmem(A = K, B = K, type = 'ALM')
```

y puede usarse para reconstruir un patrón incompleto usando la función `predict`, por ejemplo:

```
>   O1   =   c(+1,-1,-1,-0,-1,-1,-1,+0,+1,+1,+1,+0)
> predict(M, A = O1)
[1] 1 -1 -1 -1 -1 -1 -1 1 1 1 1 1
```

4.5 Simulación de un controlador de un vehículo

Una aplicación típica de las redes neuronales y los sistemas difusos, es su uso como controladores para el parqueo de vehículos. En este caso, las capacidades gráficas de R pueden utilizarse para la construcción de un simulador en 2D que muestra la ruta seguida por un automóvil que inicia su trayectoria en un punto cualquiera del plano

El controlador es una función que calcula el ángulo del volante para girar el vehículo usando

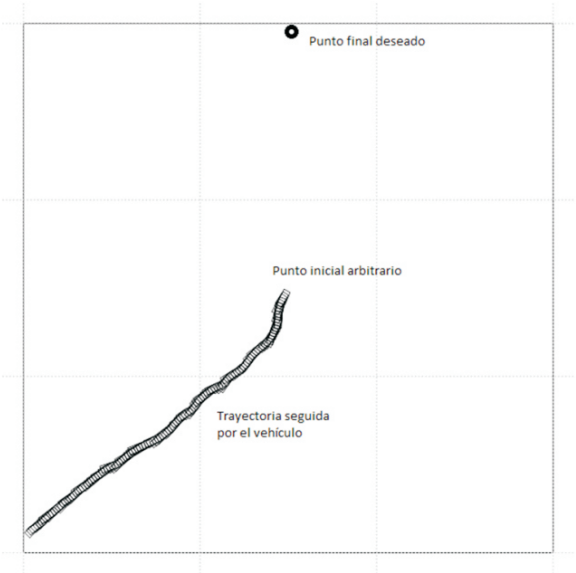


Figura 5. Simulador de un vehículo cuya dirección es determinada usando un controlador.

como información el tiempo, la posición y orientación actuales del vehículo. Por ejemplo, el siguiente controlador retorna simplemente un ángulo aleatorio:

```
zz   <- function (it, x, y, z) {return
(runif(1, min=-0.7, max = 0.7)) }
```

La gráfica presentada en la Figura 5 se generó invocando el siguiente comando:

```
truck.sim(Jn=zz, x0 = 75, y0 = 75, z0 = -2,
dt = 1, maxit = 200, velocity = 0.8)
```

Este controlador se usó en clases prácticas, donde el estudiante debe diseñar y entrenar una red neuronal artificial supervisada que permite parquear el automóvil en una posición fija y predeterminada. Al finalizar la fase de entrenamiento, la red neuronal artificial obtenida es codificada como una función de R y se realiza la simulación para distintos puntos y orientaciones iniciales para verificar su adecuado funcionamiento.

El simulador también es aplicable cuando el controlador es diseñado usando lógica difusa.

4.6 Optimización numérica usando algoritmos evolutivos

En R, existen varios paquetes para la optimización numérica de funciones usando diferentes metaheurísticas, entre los que se incluyen los algoritmos genéticos y la evolución diferencial. No obstante, el lenguaje puede usarse para la ejemplificación (con fines de docencia) del funcionamiento de los algoritmos de optimización. Por ejemplo, en la Figura 6 se presenta la evolución del mejor individuo cuando la optimización se realiza usando la metodología de Estrategias Evolutivas (1+1), esto es, un padre y un hijo.

El código en R para obtener la Figura 6 es el siguiente:

```
example.EvoEst.search <-
function()  {
  x0     = -0.78
  y0     = 2.00
  M      = 10000
   Sxc     = 0.5; Sxc.min = 0.1   # sigma
mutacion
   Syc     = 0.5; Syc.min = 0.1   # sigma
mutacion
  x      = matrix(0, nrow=M, ncol=1)
  y      = matrix(0, nrow=M, ncol=1)
  f      = matrix(0, nrow=M, ncol=1)
  t      = 2 # variable control aciertos
  x[1]   = x0
  y[1]   = y0
  f[1]   = rosenbrock( x0, y0 )
  xc     = x[1]
  yc     = y[1]
  fc     = f[1]
  for( iter in 2:M ) {
    Nt = rnorm(1,0,1)
       Sx  =  Sxc*exp(  rnorm(1,0,1)/2+Nt  /
sqrt(  2 * sqrt( 2 )))
    Sy = Syc * exp( rnorm(1,0,1) / 2 + Nt /
sqrt( 2 * sqrt( 2 )))
    if( Sx < Sxc.min ) Sx = Sxc.min
    if( Sy < Syc.min ) Sy = Syc.min
    Dx = Sx * rnorm(1,0,1)
    Dy = Sy * rnorm(1,0,1)
    fiter = rosenbrock( xc + Dx, yc + Dy )
    if( fiter < fc )  {
      xc   = xc + Dx
      yc   = yc + Dy
      fc   = fiter
      Sxc  = Sx
      Syc  = Sy
      x[t] = xc
      y[t] = yc
      f[t] = fc
      t    = t + 1
      }
    }
  }
```

4.7 Predicción de series de tiempo usando redes neuronales artificiales

Otro uso característico de las redes neuronales artificiales es la predicción de series de tiempo no lineales. Existen varios paquetes de R que implementan algunos tipos de redes neuronales artificiales (nnet, kernlan, klaR y rdetools, entre otros) pero para su uso en la solución de problemas de regresión y clasificación. No obstante, ello es suficiente para realizar algunos experimentos sobre el pronóstico de series de tiempo; particularmente, Vega, et al (2010) han usado algunas de estas implementaciones para evaluar la capacidad de pronóstico de las máquinas de vectores de soporte en el caso de la demanda mensual de electricidad en Colombia. En la Figura 7, se presenta la predicción un mes adelante para la

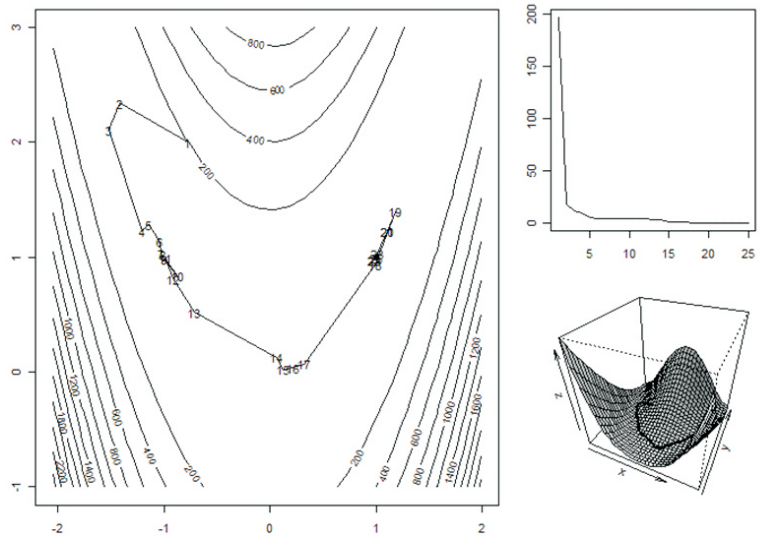


Figura 6. Ejemplo de una corrida usando la metodología de Estrategias de Evolución (1+1), para optimizar la función de Rosenbrok.

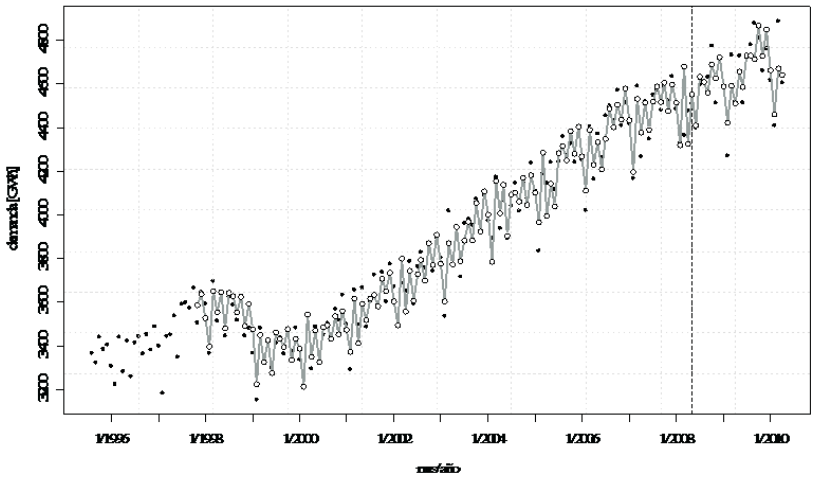


Figura 7. Predicción de una serie de tiempo usando una máquina de vectores de soporte.

serie en estudio obtenida usando una máquina de vectores de soporte implementada en el lenguaje R.

5. Discusión

Tal como ya se indicó, la inteligencia computacional (Sumathi & Paneerselvam, 2010) se basa en la solución de problemas a partir de metodologías que requieren cómputo numérico intensivo. Varios paradigmas bien conocidos y difundidos son el tema central de esta área: las redes neuronales artificiales, los sistemas de inferencia borrosa y los algoritmos bio-inspirados. Sus líneas de investigación son:

• El desarrollo de nuevas técnicas computacionalmente intensivas que impacten de forma explícita los métodos específicos de cada uno de los paradigmas mencionados.

• El desarrollo, evaluación y validación de software y algoritmos.

• El desarrollo de métodos computacionales originales y novedosos en que se apliquen los métodos específicos propios de la inteligencia computacional con aplicación a la solución de problemas específicos en exploración y clasificación de datos, econometría computacional, reconocimiento de patrones, economía computacional, etc.

• Desarrollo de aplicaciones, en que se incluye el desarrollo de herramientas, la comparación entre metodologías y la solución de problemas reales no triviales.

Las líneas presentadas están relacionadas con tres aspectos fundamentales:

•La docencia: el docente-investigador del área requiere herramientas computacionales que le permitan ejemplificar y demostrar las falencias y virtudes de las metodologías expuestas en la solución de problemas. El lenguaje R tiene dos ventajas fundamentales: primero, su uso es libre y gratuito, de tal forma que los estudiantes pueden descargarlo, instalarlo y usarlo en sus máquinas personales facilitando el aprendizaje. Segundo, es un lenguaje relativamente sencillo y bien documentado haciendo que su tiempo de aprendizaje sea corto; el único inconveniente es la gran cantidad de paquetes existentes, ya que puede ser difícil saber si hay implementaciones que permitan realizar una tarea específica. Así, el estudiante puede experimentar con herramientas ya programadas, o en proyectos más ambiciosos, implementar paquetes específicos.

• La aplicación en el mundo real. El desarrollo de paquetes permite que el código sea depurado, validado y documentado, de tal manera que las herramientas desarrolladas puedan ser usadas por un tercero para la solución de problemas reales.

• La investigación. Los diferentes paradigmas de la inteligencia computacional se encuentran en continuo desarrollo y cada momento surgen nuevas direcciones de investigación. Las características del lenguaje R permiten que se puedan realizar desde prototipos rápidos de las distintas metodologías hasta paquetes completamente documentados. Esto facilita que se haga un desarrollo por etapas, maximizando la productividad y evitando la recodificación de código.

En relación a los paradigmas propios de la inteligencia computacional, se puede decir que:

• No se encontraron evidencias de paquetes orientados al modelado de sistemas utilizando sistemas de inferencia borrosa. Esto es claramente explicado en el hecho de que R está orientado al área de estadística. En este sentido, existe una necesidad clara de desarrollar herramientas basadas en este paradigma.

• Redes neuronales artificiales: Según la investigación realizada, existen paquetes orientados a la aplicación de perceptrones multicapa (paquetes nnet y amore) y máquinas de vectores de soporte (paquetes kernlan, klaR y rdetools) en problemas de regresión y clasificación. No existen paquetes específicos para tareas como la predicción de series de tiempo. Igualmente, existen muchos otros tipos de arquitecturas mejores que los perceptrones multicapa como DAN2, ARNN o cascada-correlación para las que no existen implementaciones disponibles. Así, el lenguaje R

presenta una clara oportunidad en este tópico.

• Otra rama de singular importancia es el desarrollo y aplicación de algoritmos metaheuristicos para la optimización de funciones complejas. En R, existen implementaciones basadas en algoritmos genéticos (gafit, rgenoud, genalg), temple simulado (función optimo del paquete stats) y evolución diferencial (DEoptim). Sin embargo, no existen implementaciones para otros algoritmos que han ganado mucha popularidad en la última década tales como enjambres de partículas o métodos híbridos. El desarrollo y análisis de metaheurísticas para optimización es una clara oportunidad en sí misma, pero el uso de metodologías robustas es un requisito fundamental en la aplicación de paradigmas como las redes neuronales artificiales y los modelos híbridos.

Finalmente, la disponibilidad de metodologías robustas para el análisis y la manipulación de datos que ya se encuentran implementadas en el lenguaje R, hacen que muchas tareas relacionadas con la preparación de la información sean fácilmente ejecutadas; igualmente, es posible contrastar los resultados contra otros modelos de origen puramente estadístico, y así verificar las ganancias derivadas del uso de las técnicas propias de la inteligencia computacional.

6. Conclusiones

En este artículo se presenta una revisión del Proyecto-R para la computación estadística y la graficación con énfasis en su potencial para aplicaciones en inteligencia computacional y ciencias de la computación. Aunque algunos de los ejemplos presentados no corresponde propiamente al área de la inteligencia artificial, si demuestran claramente el potencial del lenguaje al incorporar elementos conceptuales propios de la programación funcional. Siguiendo los elementos esbozados en dichos ejemplos, fácilmente puede vislumbrarse el potencial del lenguaje para implementar paradigmas complejos como modelos de redes neuronales artificiales, sistemas neurodifusos o sistemas híbridos. El principal objetivo de este trabajo ha sido atraer la atención de los profesionales e investigadores del área de la inteligencia computacional sobre este lenguaje de programación, con el fin de fomentar su uso, y la implementación de herramientas y paquetes específicos que permitan ampliar la gama de herramientas disponibles.

7. Referencias bibliográficas

Albert, J. (2009). *Bayesian Computation with R*. Springer Series in Statistics. Springer, 2nd edition.

Amman, H.A, Kendrick, D.A. & Rust, J. (1996). *Handbook of Computational Economics, Volume I*. Elsevier/North-Holland: Amsterdam.

Becker, R., Chambers, J. M., & Wilks, A. (1988). *The (new) S language: A programming environment for data analysis and graphics*. Pacific Grove: Wadsworth & Brooks/Cole.

Belsley, D.A. & Kontoghiorghes, E. (2009). *Handbook of Computational Econometrics*. Wiley Interscience.

Chambers, J. M. (1998). *Programming with data: A guide to the S language*. New York: Springer-Verlag.

Chambers, J.M. & Hastie, T. J. (1992). *Statistical Models in S*. Chapman & Hall, London.

Correa, J.C. & González, N. (2002). *Gráficos estadísticos con R*. Universidad Nacional de Colombia, Sede Medellín.

Cowpertwait, P.S.P. & Metcalfe, A. (2009). *Introductory Time Series with R*. Springer Series in Statistics. Springer.

Fox, J. (2005). The R commander: A basic statistics graphical user interface to R. *Journal of Statistical Software* 14(9).

Gentleman, R. (2008a). *Bioinformatics with R*. Chapman & Hall/CRC, Boca Raton, FL.

Gentleman, R. (2008b). *R Programming for Bioinformatics*. Computer Science & Data Analysis. Chapman & Hall/CRC, Boca Raton, FL.

Grunsky, E. C. (2002). R: a data analysis and statistical programming environment -- an emerging tool for the geosciences. *Computers Geosciences* 28 (10), 1219-1222.

Hoff, P.D. (2009). *A First Course in Bayesian Statistical Methods. Springer Series in Statistics for Social and Behavioral Sciences*. Springer.

Ihaka, R. & Gentleman, R. (1996). R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics* 5, 299–314.

Insightful. (2007). *S-Plus 8 for Windows. User's Guide*. Insightful Corporation, Seattle, WA.

Kendrick, D.A, Mercado, R.P. & Amman, H.M. (2005). *Computational Economics*. Princenton University Press.

Kleiber, C. & Zeileis, A. (2008). *Applied Econometrics with R*. Springer

Los, C.A. (2001). *Computational Finance: A scientific perspective*. World Scientific Publishing Co. Pte. Ltd.

Petris, G., Petrone, S. & Campagnoli, P. (2009). *Dynamic Linear Models with R*. Use R. Springer.

Pfaff, P. (2008). *Analysis of Integrated and Cointegrated Time Series with R*. 2nd Edition. Springer, New York.

Robert, C. & Casella, G. (2010). *Introducing Monte Carlo Methods with R*. Use R. Springer.

Sawitzki, G. (2009). *Computational Statistics:An Introduction to R*. Chapman & Hall/CRC Press, Boca Raton (FL).

Seydel, R. (2009). *Tools for Computational Finance*. 4th edition. Springer.

Spector, P. (2008). *Data Manipulation with R*. Springer.

Sriplung, H. (2006). *Integrated computing environment for R*. R package Version 1.0–1. URL: http://www.r-ice.org.

Sumathi, S. & Paneerselvam, S. (2010). *Computational Intelligence Paradigms: Theory & Applications using MATLAB*. CRC Press.

Tesfatsion, L. & Judd, K.L. (2006). *Handbook of Computational Economics, Volume II: Agent-Based Computational Economics*. Elsevier/North-Holland: Amsterdam, 2006, 904pp

Vega, W., Velásquez, J.D. & Franco, C.J. (2010). *Pronóstico de la demanda mensual de electricidad en el mercado energético Colombiano usando máquinas de vectores de soporte*. Reporte Técnico, Universidad Nacional de Colombia.

Velásquez, J.D. (2010). Implementación de una memoria asociativa lineal usando el lenguaje R. *Avances en Sistemas e Informática*, 7 (2), 97-103.

Vinod, H.D. (2008). *Hands-on Intermediate Econometrics Using R: Templates for Extending Dozens of Practical Examples*. World Scientific, Hackensack, NJ.