Introducción a la Inteligencia Predictiva en R con aplicaciones:

Introducción a R.

Diego J. Pedregal

Universidad de Castilla-La Mancha

Diego.Pedregal@uclm.es

Universidad de Sevilla

15-16 de Noviembre de 2018

- Objetivo principal del curso es demostrar la utilidad y potencia de R en el análisis de datos y ayudar a saltar la barrera de entrada que supone este tipo de aplicaciones.
- Práctico.
- Es un curso muy abierto.
- A programar se aprende …
- Se admiten sugerencias.
- 8 horas divididas en 4 módulos de 2 horas.

Horario

- Día 1 (15 Noviembre):
  - Sesión 1 (2 horas): Introducción a R.
  - Sesión 2 (2 horas): Técnicas de Regresión.
- Día 2 (16 Noviembre):
  - Sesión 3 (2 horas): Introducción a las Series Temporales.
  - Sesión 4 (2 horas): Predicción.

- Otros modelos:
  - Otros modelos lineales: Ad-hoc, Componentes No Observables, Alisado Exponencial, etc.
  - Modelos no lineales: GAS, GARCH, TAR, etc.
- Predicción de grandes bases de datos:
  - Predicción de demanda intermitente, discreta, granulosa, etc.
  - Predicción jerárquica.
  - Redes Neuronales Artificiales.
  - Deep learning (CNN, RNN, LSTM, etc.).

¿Qué es R?

- Lenguaje de alto nivel, especialmente apto para investigación en estadística. Barrera de entrada.
- Existen otros: MATLAB, Python, Julia, Go, etc.
- Flexibilidad.
  - Permite desarrollar subrutinas propias de forma acumulada.
  - Permite mezclar librerías (packages).
  - Existen muchas librerías en la WEB.
  - Más flexible que paquetes cerrados.

Entorno de trabajo RStudio

RStudio

File Edit Code View Project Workspace Plots Tools Help

Go to file/function | Project: (None)

diamondPricing.R* | formatPlot.R | diamonds

Source on Save | Source

```
1  library(ggplot2)
2  source("plots/formatPlot.R")
3
4  view(diamonds)
5  summary(diamonds)
6
7  summary(diamonds$price)
8  aveSize <- round(mean(diamonds$carat), 4)
9  clarity <- levels(diamonds$clarity)
10
11 p <- qplot(carat, price,
12             data=diamonds, color=clarity,
13             xlab="Carat", ylab="Price",
14             main="Diamond Pricing")
15
```

15:1 (Top Level) R Script

Console ~/

```
       x                y                z
 Min.   : 0.000   Min.   : 0.000   Min.   : 0.000
 1st Qu.: 4.710   1st Qu.: 4.720   1st Qu.: 2.910
 Median : 5.700   Median : 5.710   Median : 3.530
 Mean   : 5.731   Mean   : 5.735   Mean   : 3.539
 3rd Qu.: 6.540   3rd Qu.: 6.540   3rd Qu.: 4.040
 Max.   :10.740   Max.   :58.900   Max.   :31.800
> summary(diamonds$price)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    326     950    2401    3933    5324   18820
> aveSize <- round(mean(diamonds$carat), 4)
> clarity <- levels(diamonds$clarity)
> p <- qplot(carat, price,
+            data=diamonds, color=clarity,
+            xlab="Carat", ylab="Price",
+            main="Diamond Pricing")
>
> format.plot(p, size=24)
>
```

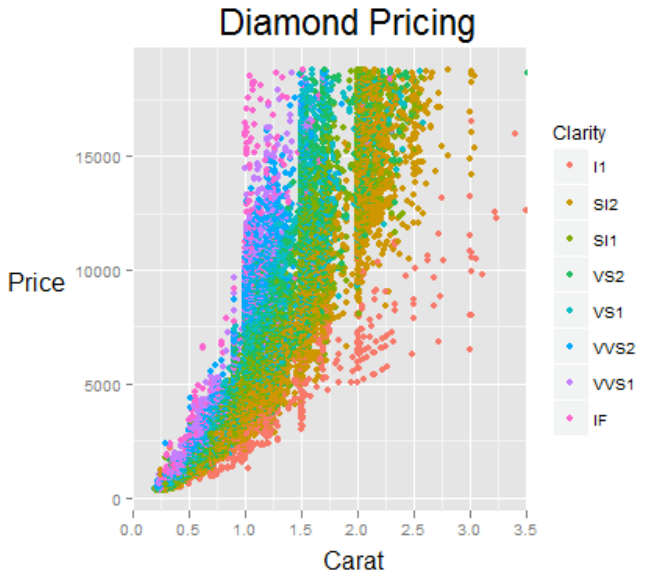
Workspace | History

Load | Save | Import Dataset | Clear All

Data	
diamonds	53940 obs. of 10 variables
Values	
aveSize	0.7979
clarity	character[8]
p	ggplot[8]
Functions	
format.plot(plot, size)	

Files | Plots | Packages | Help

Zoom | Export | Clear All



Instalación

- Instalación: https://cran.r-project.org/bin/windows/base/
- Instalación RStudio: https://www.rstudio.com/products/rstudio/download/
- En la consola se pueden instalar packages con el comando install.packages()
- Instalar package forecast: install.packages("forecast")
- Recuperar la carpeta de trabajo: getwd()
- Cambiar la carpeta de trabajo: setwd()
- Ayuda: ?comando
- Ayuda: Buscar en ventana de ayuda
- Ayuda: Buscar en WEB
- Crear proyecto para el curso desde Rstudio

Lenguaje de programación

- Asignación. R distingue entre mayúsculas y minúsculas:
  - variable <- expresión (<- es el operador de asignación)
  - variable = expresión (= es otro operador de asignación)
  - Ejemplos: a<- 1; A= 2; B= a %% 3
  - Ejemplos: a<- TRUE; b= FALSE;
- Tipos de Variables:
  - Números reales de doble precisión
    - 3/1.6; 1/0; 0/0
  - Vectores
    - y= c(1, 2, 3, 4, 5, 6)
    - length(y)
    - y[2:5]
    - y/2; y*2

Lenguaje de programación

- Matrices
  - m<- matrix(c(1, 2, 3, 4, 5, 6), 2, 3); m
  - A= dim(m); A
  - A[1]; A[2]
- Hipermatrices
  - m<- array(1:30, c(2, 5, 3)); m
  - A= dim(m); A
  - A[1]; A[2]; A[3]
- Cadenas
  - a= "esto es una cadena de caracteres"
- Vectores de cadenas
  - nombre= c("Yo", "Tú", "Él", "Ella")

Lenguaje de programación

- Frames
  - edad<- c(10, 20, 30, 40)
  - DATA= data.frame(nombre, edad)
  - DATA$nombre[3]; DATA$edad[3]
- Listas

  C<- list(c(1,2,3), c(TRUE, FALSE, TRUE), c("a", "b"))

  C[1]; C[2]; C[3]

  C[[1]]; C[[2]]; C[[3]]

  typeof(C[1])

  typeof(C[[1]])

Matrices y vectores

- 1.3. Operaciones con matrices y vectores:
  - Transposición:
    A= matrix(c(1, 2, 3, 4, 5, 6), 2, 3); t(A)
  - Producto:
    B= matrix(runif(9), 3,3)
    A%*%B; A*2
    A*A
    A[1, 2]; A[2, 1]; A[, 1]
  - Inversión:
    C= solve(B); C%*%B
  - Diagonal de una matrix:
    diag(B)

PRÁCTICA #1.1

Matrices y vectores

– Algunos comandos que crean matrices:

Matriz identidad:

diag(3)

Matriz de ceros o unos:

matrix(0, 3, 4); matrix(1, 1, 2)

Dimensiones de matrices:

n<- dim(A); n[1]; n[2]; n

Concatenación vertical:

rbind(A, B)

Concatenación horizontal:

cbind(A, C)

Operadores

- Operadores relacionales:
  - `<` : 1<2; 2<1
    c= c(1, 2, 0); d= c(2, 2, 0); c< d
  - `>` : 1>2 ; 2>1 ; c> d
  - `<=` : 1<=2 ; 2<=1 ; 1<=1 ; c<= d
  - `>=` : 1>=2 ; 2>=1 ; 1>=1 ; c>= d
  - `==` : 1==1 ; 2==1 ; c== d
  - `!=` : 1!=1 ; 2 !=1 ; c !=d
- Operadores lógicos:
  - `&` (AND): 1<2 & 2==1; 1<2 & 2<3 ; c< d & d>=0
  - `|` (OR): 1<2 | 2<1 ; c< d | d>=0
  - `!` (NOT): !(1<2) ; !(1<2 & 2==1) ; !(c< d & d>=0)

Scripts y funciones

- Ficheros de comandos y funciones
  - R puede utilizarse sin necesidad de introducir los comandos uno a uno en la ventana de comandos, para ello se pueden utilizar dos tipos de ficheros:
    - Ficheros de comandos (script). Son sencillamente un conjunto de comandos R aglutinados en un fichero con extensión .R
    - Funciones, que son en realidad comandos tipo R escritos en fichero con extensión .R. En realidad una función es mucho más, puesto que las variables que en ella aparecen son locales y permite la entrada y salida de argumentos.

Funciones

- Sintaxis general de funciones en R:

```
Nombre__función <- function(arg1, arg2, ... )
{
    # Comentarios
    Comandos
     return(objeto)
}
```

- Todas las variables definidas dentro de la función son locales.
- Las funciones pueden llamar a otras funciones y pueden llamarse a sí mismas (peligro de llamadas infinitas!!)
- En la declaración se pueden incluir valores por defecto de los argumentos de entrada:

```
Nombre <- function(data, alfa= 1, beta= 0, name= "Fran")
```

PRÁCTICA #1.2

Bifurcaciones

- **Bifurcaciones**: A menudo interesa que dependiendo de determinadas condiciones se ejecute una parte de código o bien otra. Para ello se utilizan las bifurcaciones.

```
if (condición){
    Comandos cuando la condición es cierta
} else {
    Comandos cuando la condición es falsa
}
```

- Ejemplo:

```
i= 1
if (i==1) print("i vale 1")
```

- Las bifurcaciones se pueden anidar.

Bucles

- Bucles: Se utilizan para repetir una tarea un número de veces.

1. Bucles “for”:

```
for (variable in conjunto){
    Comandos que se repiten
}

for (year in 2009:2017) {
    print(paste("El año es: “, year))
 }
```

Bucles

2. Bucles “while”:

```
while (condición){
    Comandos cuando la condición es cierta
}
```

- Los bucles se pueden anidar.
- Peligro de bucles infinitos.

```
i<- 1
while (i== 1)
    print("Esta condición es cierta")
```

- ¿Qué diferencia hay con…?

```
if (i== 1)
    print("Esta condición es cierta")
```

PRÁCTICA #1.3