Prácticas de Aprendizaje Automático

Parte 3: Programando en R

Curso 2017-18

Índice

Cuando hablamos de datos

- **Variable** : Característica de interés *una dataframe, una matriz, un vector*.
- **Muestra Observada** : Conjunto de valores de la variable obtenidos de manera homogénea *una fila del dataframe, fila de la matriz, componente del vector*
- **Tamaño muestral** : Número de datos observados *longitud del dataframe, de la matriz, del vector*.
- La manera de describir la muestra (nuestros datos) depende del tipo de atributo:
  Cualitativo : Intrínsecamente no tiene carácter numérico (categórica) *ej. sexo, nivel de estudios*
  Cuantitativo : Intrínsecamente numérico
  - Discreto (cantidad finita o numerable de valores): *número de hijos*
  - Contínuo (valores reales): *Altura*

Índice

Simulando datos

- Para simular datos contínuos

R tiene las distribuciones de probabilidad más comunes implementadas en la librería BASE. En otras librerías disponemos de otras tantas. Para cada una de ellas *(distrib)*, disponemos de 4 versiones:

datos contínuos

función densidad/probabilidad `d` distrib
función distribución `p` distrib, calcula $P(X <= x)$
función inversa distribución (cuantiles) `q` distrib, calcula x t.q. $P(X <= x) = p$
generador de numeros aleatorios `r` distrib

así para la normal, las distribución es **norm**

```
dnorm(x,mean = 0,sd = 1, log = FALSE)
qnorm(p,mean = 0,sd = 1, lower.tail = TRUE, log.p = FA
pnorm(q,mean = 0,sd = 1, lower.tail = TRUE, log.p = FA
rnorm(n,mean = 0,sd = 1, log = FALSE)
```

Simulando datos

```
> x <- rnorm(10)
> x
[1] 1.38380206 0.48772671 0.53403109 0.66721944
[5] 0.01585029 0.37945986 1.31096736 0.55330472
[9] 1.22090852 0.45236742

> x <- rnorm(10, 20, 2) #si no tomamos valores xdefect
> x
[1] 23.38812 20.16846 21.87999 20.73813 19.59020
[6] 18.73439 18.31721 22.51748 20.36966 21.04371
```

Para que sea reproducible, es conveniente fijar la semilla

```
> set.seed(1)
> rnorm(5)
[1] -0.6264538  0.1836433 -0.8356286  1.5952808  0.329
> rnorm(5)
[1] -0.8204684  0.4874291  0.7383247  0.5757814 -0.305
> set.seed(1)
> rnorm(5)
```

Distribuciones de probabilidad

Distribuciones de probabilidad en la librería BASE.

Función	Comando
Normal	rnorm(n, mean=0, sd=1)
exponencial	rexp(n, rate=1)
gamma	rgamma(n, shape, scale=1)
Poisson	rpois(n, lambda)
Weibull	rweibull(n, shape, scale=1)
Cauchy	rcauchy(n, location=0, scale=1)
beta	rbeta(n, shape1, shape2)
t de Student	rt(n, df)
F (Snedecor)	rf(n, df1, df2)
Pearson X^2	rchisq(n, df)
binomial	rbinom(n, size, prob)
...	

Simulando datos

Supongamos que queremos generar datos apartir de un modelo lineal de la siguiente forma:
$y = \beta_0 + \beta_1 x + \xi$ donde,
ξ sigue una $N(0, 2^2)$, x sigue $N(0, 1^2)$ y $\beta_0 = 0.5$ y $\beta_1 = 2$.

```
set.seed(3)
x <- rnorm(100)
e <- rnorm(100, 0, 2)
y <- 0.5 + 2 * x + e
plot(y)
```

Medidas de localización y dispersión

Función	Utilidad
sum(..., na.rm=FALSE)	Suma
max(..., na.rm=FALSE)	Máximo
min(..., na.rm=FALSE)	Mínimo
which.min(x)	Posición del máximo
which.max(x)	Posición del mínimo
pmax(...,na.rm=FALSE)	Máximo en paralelo
pmin(...,na.rm=FALSE)	Mínimo en paralelo
mean(x, trim=0, na.rm=FALSE)	Media
weighted.mean(x,w,na.rm=FALSE)	Media ponderada
median(x,na.rm=FALSE)	Mediana
quantile(x,prob=(0,0.25,0.5,0.75,1),na.rm=F)	Cuantiles
summary(x, na.rm=FALSE)	min,1c,mediana,media,3c,max
range(...,na.rm=FALSE, finite=FALSE)	Rango
var(x, y=x, na.rm=FALSE, use)	Varianza
sd(x, na.rm=FALSE)	Desviación Típica

Ejemplos de uso

```
> summary(y)
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-4.67800 -1.31300  0.07138  0.55990  2.93000  7.24500


mean(x);            median(x);            quantile(x);
quantile(x,c(0.35,0.9)); sd(x);             var(x);
range(x);           min(x);             which.min(x);
x[which.min(x)];              pmin(x[1:5],x[6:10]);
max(x);             which.max(x);             x[which.max(x)];
pmax(x[4:8],x[2:6]
```

Simulando otros datos

- Para simular datos numéricos discretos

La función **sample()** genera un conjunto de valores apartir de un conjunto especificado de entrada. Con o sin sustitución (reemplazo).

así para la normal, las distribución es **norm**

```
> set.seed(1)
> sample(1:10, 4)
[1] 3 4 5 7
> sample(1:10, 4)
[1] 3 9 8 5
> sample(letters, 5)
[1] "q" "b" "e" "x" "p"
> sample(1:10)
[1] 4 710 6 9 2 8 3 1 5
> sample(1:10)
[1] 2 3 4 1 9 5 10 8 6 7
> sample(1:10, replace = TRUE)
[1] 2 9 7 8 2 8 5 9 7 8
```

Índice

Factores/Variables categóricas

- Un factor es un vector utilizado para especificar una clasificación discreta de los elementos de otro vector de igual longitud.
  - Pueden ser no ordenados (nominales):
    No existe jerarquía entre ellos (p.e., colores)
  - Ordenados (ordinales): Existe jerarquía entre ellos (p.e., cursos, grupos de edad)
- Se pueden crear a partir de un vector numérico con las funciones **as.factor(), o bien gl()**.
- También a partir de un vector de caracteres utilizando **factor()**.
- Las etiquetas se asignan con **levels()**.

Cuando trabajamos con factores, a veces nos interesa conocer cuantos hay de cada clase o categoría. La frecuencia de cada una de las clases se puede obtener mediante la función **table()**

Factores/Variables categóricas

```
>f <- as.factor(c(1,2,3,1,2,1,1,3,2))     #Factor 3 cat
>f
[1] 1 2 3 1 2 1 1 3 2
Levels: 1 2 3
levels(f)<-c("Bajo", "Medio", "Alto")
>f
[1] Bajo  Medio Alto  Bajo  Medio Bajo  Bajo  Alto  Me
Levels: Bajo Medio Alto
> ford<-as.ordered(f)              # Factor ordenado
> ford
[1] Bajo  Medio Alto  Bajo  Medio Bajo  Bajo  Alto  Me
Levels: Bajo < Medio < Alto
```

```
dolor <- sample(rep(0:3, each=4)
fdolor <- factor(dolor,levels=0:3)
levels(fdolor) <- c("nodolor","leve","medio","alto")
table(fdolor)
```

Índice

Representar gráficos

Las facilidades gráficas de R constituyen una de las componentes más importantes de este lenguaje.

R incluye muchas y muy variadas funciones para hacer gráficas estadísticas estándar: desde gráficos muy simples a figuras de gran calidad para incluir en artículos y libro

R permite exportar y salvar estos gráficos en múltiples formatos `.pdf`, `.jpeg`

Los comandos de dibujo que vamos a ver se dividen en:

- Funciones de alto nivel
- Funciones de bajo nivel
  que añaden información a gráficos ya existentes

La función plot

Cuando creamos un gráfico, asociado a éste se encuentra el dispositivo gráfico, que puede ser una ventana donde se represente dicho gráfico. funciones como: `x11()` , `window()`, `dev.new()`, `dev.off()`, ... van trabajar con estos dispositivos cuando usemos funciones de bajo nivel.

El procedimiento gráfico de alto nivel más habitual para dibujar datos es **plot()**.

```
x<-(0:65)/10
y<-sin(x)
plot(x)
plot(x, y) #abre automaticamente
plot(x, y, main= "Funcion Seno")
z <-cos(x)
windows()  #Crea ventana si windows (X11() para linux)
plot(x, z, main="Funcion Coseno"
```

Tenemos varios dispositivos activos

Parámetros de la función plot

main : Cambia el título principal del gráfico

sub : Cambia el subtítulo del gráfico

type : Tip o de gráfico (puntos, líneas, etc.)

xlab,ylab : Cambia las etiquetas de los ejes

xlim, ylim : Cambia el rango de valores de los ejes

lty : Cambia el tipo de línea;

lwd : Cambia el grosor de línea

col : Color con el que dibuja

```
plot(x, y, main="Seno", type="l")
plot(x, z, main="Coseno", lty=2, col="red", type="l")
plot(x, z, main="Coseno", lty=3, col="blue", type="l",
xlim=c(0,2), ylab="cos(x)")
```

Parámetros de la función plot

Hay 9 formas básicas de dibujar un conjunto de puntos, esto se define con los valores de type:

- **p** : points (es el default)
- **l** : una line
- **b** : "both", puntos conectados con segmentos de línea
- **c** : solo con segmentos
- **o** : "overplotted", puntos y lineas sobreescritas
- **h** : como un histograma
- **s** : stair en forma de escalera
- **S** : una escalera alternative
- **n** : none, no pinta los puntos

```
plot(x, y, type = "p");
plot(x, y, type = "l");
plot(x, y, type = "b");
plot(x, y, type = "c");
```

Parámetros de la función plot

• Hay 25 formas de puntos diferentes que se definen con el argumento **pch** También se puede modificar el punto y en su lugar utilizar caracteres para pintar.

```
plot(x, y, type = "p", pch=2);
plot(1:10,pch = c("@","a","#","&","%"))
```

• Hay 6 tipos de líneas que se pueden definir con **lty** de 1 a 6

Para controlar el tamaño (grosor) de los puntos, se usa el argumento **cex** que indica un múltiplo de veces sobre el tamaño normal.

```
plot(x, y, lty=6, cex=0.5);
```

R tiene colores por defecto pueden ser número $1:8$ o texto `"red"` ...

Parámetros de la función plot

R tiene 657 nombres de colores que se pueden pasar como argumento **col** .
Para ver los nombres, consulte la función **colors()** alguna muestra de azúl
`"aquamarine1"` `"aquamarine3"` `"aquamarine4"` `"azure"` `"azure1"`
Hay determinadas paletas de colores que se pueden seleccionar en en el parámetro **col** por ejemplo `heat.colors()` ...

```
plot(x,y, type= "o", pch=3, col = rainbow(3))
```

Índice

Funciones de bajo nivel

Hay una serie de funciones que permiten dibujar sobre una gráfica ya creada.
Los más habituales:

- **points(x, y, ...)** : Dibuja una nube de puntos
- **abline()** : Dibuja una recta con intercept y pendiente
- **lines(x, y, ...)** : Dibuja una linea que une todos los puntos
- **polygons(x, y, ...)** : Dibuja un polígono cerrado
- **text(x, y, labels, ...)** : Escribe texto en unas coordenada
- **mtext** texto en los márgenes

```
plot(x, y,  main="Fun seno coseno",type="l")
lines(x,  z, col="blue",  lty=2)
text(x=c(0.5,0.5),y=c(0, 1),
labels=c("sin(x)", "cos(x)"),
col=c("black","blue"))
```

Leyendas

Para poner una leyenda sobre como interpretar las líneas se usa la función **legend(x, y, legend, ...)**

- **x,y** esquina superior izquierda de la leyenda
- legend : Texto de la leyenda
- bty: tipo de borde, (n omitir)

```
plot(x,y, main="Fun seno coseno", type="l")
lines(x, z, col="blue",lty=2)
legend(x=3, y=1,legend=c("sin(x)","cos(x)"), lty=c(1,2
col=c("black", "blue"))
```

Como se distribuye el área para el dibujo

En toda figura hay una región principal donde se sitúa el plot, con o sin ejes, medido en unidades del propio gráfico. Esta región está delimitado por 4 márgenes.

```
plot(x,y,xlab=" texto eje X",ylab="texto eje Y", main=
                # se vuelve a abrir la ventana
text(-1,48, "texto en -2,48")    # anade texto
mtext(paste("lado",1:4), side= 1:4,line=-1,font=2)    #
```

Como se distribuye el área para el dibujo

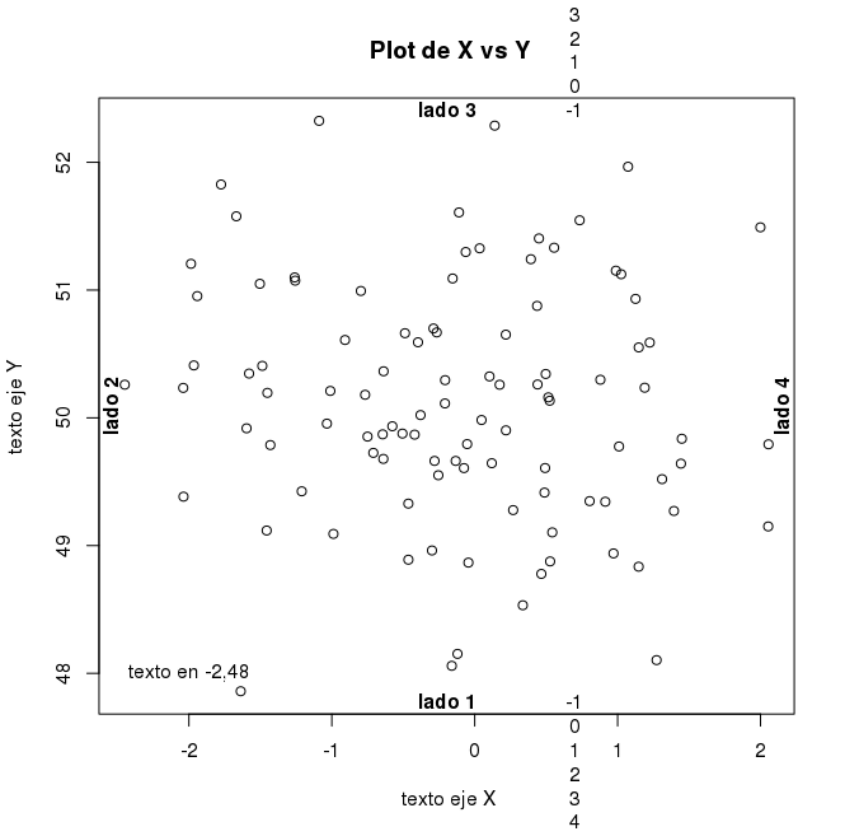


Figure 1: Disposición de regiones en un gráfico

Estableciendo el fondo

La forma más simple de pensar es ver como definimos el papel sobre el que vamos a pintar y luego se añaden elementos sobre ese área.

La función **par()** establece un como *parámetro gráfico* y define el papel sobre el que se va a dibujar después, y se hereda en los sucesivos plot de la página. Así se puede poner varios gráficos por página, consulte

```
n.col <- 2
n.row <- 2
par(mfrow = c(n.row,n.col))
plot(x, y, main="Seno", type="l")
plot(x, z, main="Coseno", lty=2, col="red", type="l")
plot(x, z, main="Coseno", lty=3, col="blue", type="s")
plot(x, z, main="Coseno", lty=6, col=5)
```

O bien se quiere aprovechar más el papel

```
> par(mar=c(4,4,2,2)) # se gana espacio si no hay titu
> plot(x,y,col=2, xlab=" texto eje X",ylab="texto eje
```

Vamos a representar datos

Podemos estudiar la distribución de 2 atributos

```
X <- matrix(rnorm(100), ncol = 2);
colnames(X) <-c("aAtributo", "bAtributo")
plot(X)
```

Podemos estudiar la distribución **por pares** de atributos con **pairs()**

```
X <-matrix(rnorm(100), ncol=5)
colnames(X) <- c("a", "id", "edad", "loc", "peso")
pairs(X)
```

```
especie <-unclass(iris$Species) # data(iris)
plot(iris[1:2],pch=21,bg = c("red", "green3", "blue")[especie])
pairs(~ .,data = iris, main = "Iris Dataset",
pch = 21, bg = c("red", "green3", "blue")[unclass(iris$Species)]
```

Hay otros tipos de gráficos

Pero hay más tipos de gráficos comunmente utilizados:

- Boxplots : gráficos de bigotes
- Barplots
- Histogramas
- Pies

```
x <- rnorm(100,5,3)
hist(x) # genera un histograma
hist(x, main="Histograma", breaks=10)
```

boxplot(rnorm(50),rnorm(100),rlnorm(50))

Índice

Para guardar los gráficos

Se puede guardar el dispositivo activo
bien através del menú export ...
bien usando comando `pdf()` abre un dispositivo gráfico que produce un fichero `.pdf` Conviene cerrar dispositivo gráfico. Es probable que tengamos varios dispositivos gráficos abiertos... la ventana gráfica y varios `pdf()`

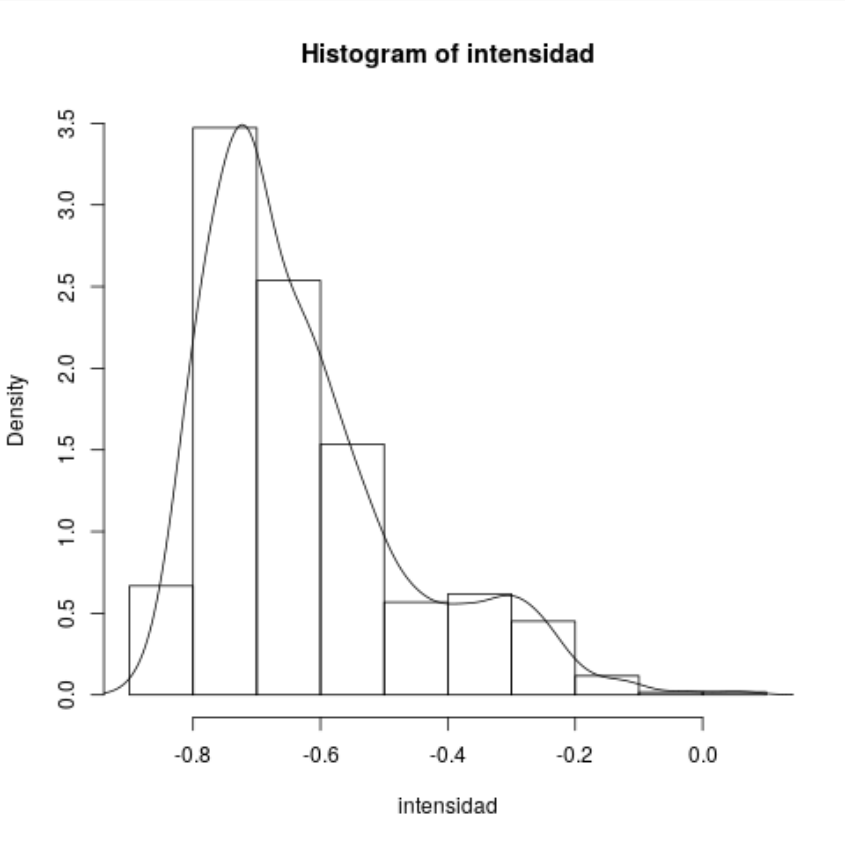
```
getwd()                    # comprobar el directorio actu
pdf("plot03.pdf")    # cada vez se abre un dispositivo
hist(intensidad,freq = F)
lines(density(intensidad))
```

Através del entorno gráfico se han ejecutado varias veces instrucciones del tipo abrir y cerrar dispositivo, que podemos controlar de la forma:

```
>  dev.list()        # se consulta la lista
>  dev.off(dev.cur())    # cierra el actual o se espefi
```

Para otros formatos, se puede usar `savePlot` . Con el argumento `type` seleccionamos el formato en el que vamos a guardar la figura, por defecto, se guarda el gráfico del dispositivo actual, dev.cur()

Gráficas de cajas boxplot



Esta es la salida de las gráficas superpuestas