Universitat de les Illes Balears

#SOM UIB

AprendeR: Introducción al tratamiento de datos con R y RStudio

Módulo 3

Lección Extra
Rmarkdown

Campus Extens
UIB Virtual

http://moocs.uib.cat

Edició: abril 2016
Edita: Campus Extens – UIB Virtual
Disseny portada: Direcció de l'Estratègia de Comunicació i Promoció Institucional (dircom.uib.cat)

Aquesta obra està subjecta a una llicència CC
Reconeixement-NoComercial-SenseObraDerivada 4.0 Internacional

Extras de *R Markdown*

Para la mayor parte de las necesidades de este curso en lo que se refiere a composición de ficheros *R markdown*, el documento *Markdown Quick Reference* que lleva *RStudio* y que mencionamos en la Lección 0 es suficiente. No obstante, a lo largo del curso ampliaremos sus contenidos en algunos temas cuando lo creamos necesario. En esta lección, vamos a tratar diversos puntos: cómo escribir fórmulas matemáticas bien formateadas; cómo controlar el comportamiento de los bloques de código (*chunks*) al compilar el fichero *R markdown*, y su aspecto en el documento final; cómo controlar el aspecto de los gráficos producidos en los *chunks*; y como incluir de manera sencilla tablas bien formateadas.

1. Fórmulas matemáticas

La manera de incluir fórmulas matemáticas en *R Markdown* se basa en la sintaxis del sistema de composición de textos científicos LaTeX, que es el que hemos usado para escribir las lecciones de este curso. Esta misma sintaxis, con pequeñas modificaciones, se usa para escribir fórmulas matemáticas bien formateadas en otros contextos: en los foros de *Moodle*, en las entradas y comentarios de blogs en *Blogger* o *Wordpress*, en la *Wikipedia*, etc.

Incluir fórmulas en un texto *R Markdown* no tiene ningún misterio. Solo hay que introducir el código que representa la fórmula de una de las dos formas siguientes:

- Para las fórmulas o ecuaciones dentro del mismo párrafo, se escribe el código entre dos dólares: `$código$`.
- Para las fórmulas o ecuaciones que queramos que aparezcan centradas en una línea aparte, se escribe el código entre dos dobles dólares: `$$código$$`.

Al componer una fórmula a partir del código, *RStudio* ignora los espacios en blanco que hayamos escrito en ella, y añade los espacios en blanco a partir del significado lógico de sus elementos. Por ejemplo (y dejamos algunos espacios en blanco innecesarios para que veáis que no tienen ningún efecto en el resultado), el código siguiente:

```
Las raíces de la ecuación $x^2= 2$ son $x=\sqrt{ 2}$ y $x=-\sqrt{2} $;
en general, las raíces de $ax^2+b  x+c=0$, con $a\neq 0$, vienen dadas
por la fórmula $$x=\frac{-b\pm\sqrt{b^2-4 a c}}{2a}.$$
```

produce el texto siguiente:

> Las raíces de la ecuación $x^2 = 2$ son $x = \sqrt{2}$ y $x = -\sqrt{2}$; en general, las raíces de $ax^2 + bx + c = 0$, con $a \neq 0$, vienen dadas por la fórmula
>
> $$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

Observad el código:

- Las potencias, y en general los superíndices, se indican con `^`.

- La raíz cuadrada de *algo* se indica con `\sqrt{`*algo*`}` (de *square root*).
- Una fracción se indica con `\frac{`*numerador*`}{`*denominador*`}` (de *fraction*).
- Los signos $\pm$ y $\neq$ se indican con las marcas `\pm` (de *plus-minus*) y `\neq` (de *not equal*), respectivamente.

Como podéis ver, las marcas de LaTeX que definen los diferentes elementos de las fórmulas matemáticas tienen nombres intuitivamente claros.

A continuación damos algunas tablas con las marcas correspondientes a algunos de los signos matemáticos más usuales:

Algunos operadores binarios

$+$	`+`	$-$	`-`	$\pm$	`\pm`	$\times$	`\times`	$\div$	`\div`	$\cdot$	`\cdot`
$/$	`/`	$\circ$	`\circ`	$\cap$	`\cap`	$\cup$	`\cup`	$\vee$	`\vee`	$\wedge$	`\wedge`

Algunos signos para relaciones

$=$	`=`	$\neq$	`\neq`	$<$	`<`	$>$	`>`	$\leqslant$	`\leq`
$\geqslant$	`\geq`	$\subseteq$	`\subseteq`	$\supseteq$	`\supseteq`	$\subsetneq$	`\subsetneq`	$\in$	`\in`
$\equiv$	`\equiv`	$\sim$	`\sim`	$\mid$	`\mid`	$\approx$	`\approx`	$\cong$	`\cong`

Algunos operadores

$\sum$	`\sum`	$\prod$	`\prod`	$\coprod$	`\coprod`	$\bigoplus$	`\bigoplus`
$\bigcap$	`\bigcap`	$\bigcup$	`\bigcup`	$\bigsqcup$	`\bigsqcup`	$\int$	`\int`
$\bigvee$	`\bigvee`	$\bigwedge$	`\bigwedge`				

Algunos delimitadores

$($	`(`	$)$	`)`	$[$	`[`	$]$	`]`
$\{$	`\{`	$\}$	`\}`	$\langle$	`\langle`	$\rangle$	`\rangle`
$\lfloor$	`\lfloor`	$\rfloor$	`\rfloor`	$\lceil$	`\lceil`	$\rceil$	`\rceil`

Algunas letras griegas

α	`\alpha`	β	`\beta`	γ	`\gamma`	δ	`\delta`
ϵ	`\epsilon`	ε	`\varepsilon`	ζ	`\zeta`	η	`\eta`
θ	`\theta`	γ	`\gamma`	κ	`\kappa`	λ	`\lambda`
μ	`\mu`	ν	`\nu`	ξ	`\xi`	π	`\pi`
ρ	`\rho`	σ	`\sigma`	τ	`\tau`	υ	`\upsilon`
ϕ	`\phi`	φ	`\varphi`	χ	`\chi`	ψ	`\psi`
ω	`\omega`	Γ	`\Gamma`	Δ	`\Delta`	Θ	`\Theta`
Λ	`\Lambda`	Ξ	`\Xi`	Π	`\Pi`	Σ	`\Sigma`
Υ	`\Upsilon`	Φ	`\Phi`	Ψ	`\Psi`	Ω	`\Omega`

Algunos acentos en matemáticas

$\hat{x}$	`\hat{x}`	$\bar{x}$	`\bar{x}`	$\dot{x}$	`\dot{x}`	$\tilde{x}$	`\tilde{x}`	$\vec{x}$	`\vec{x}`

Algunos acentos «expansibles»

$\widetilde{xyz}$	`\widetilde{xyz}`	$\widehat{xyz}$	`\widehat{xyz}`
$\overline{xyz}$	`\overline{xyz}`	$\underline{xyz}$	`\underline{xyz}`
$\overleftarrow{xyz}$	`\overleftarrow{xyz}`	$\overrightarrow{xyz}$	`\overrightarrow{xyz}`
$\overbrace{xyz}$	`\overbrace{xyz}`	$\underbrace{xyz}$	`\underbrace{xyz}`

Algunas flechas

$\leftarrow$	`\leftarrow`	$\Leftarrow$	`\Leftarrow`	$\rightarrow$	`\rightarrow`
$\Rightarrow$	`\Rightarrow`	$\longleftarrow$	`\longleftarrow`	$\Longleftarrow$	`\Longleftarrow`
$\longrightarrow$	`\longrightarrow`	$\Longrightarrow$	`\Longrightarrow`	$\leftrightarrow$	`\leftrightarrow`
$\Leftrightarrow$	`\Leftrightarrow`	$\mapsto$	`\mapsto`	$\longleftrightarrow$	`\longleftrightarrow`
$\Longleftrightarrow$	`\Longleftrightarrow`	$\uparrow$	`\uparrow`	$\downarrow$	`\downarrow`

Algunas funciones

sin	`\sin`	cos	`\cos`	arcsin	`\arcsin`	arc cos	`\arccos`
tan	`\tan`	arctan	`\arctan`	exp	`\exp`	log	`\log`
ln	`\ln`	máx	`\max`	mín	`\min`	lím	`\lim`
sup	`\sup`	ínf	`\inf`	det	`\det`	arg	`\arg`

Otros signos

$\ldots$	`\ldots`	$\cdots$	`\cdots`	$\vdots$	`\vdots`	$\ddots$	`\ddots`
$\aleph$	`\aleph`	$\exists$	`\exists`	$\forall$	`\forall`	∞	`\infty`
$\emptyset$	`\emptyset`	$\neg$	`\neg`	$\top$	`\top`	$\bot$	`\bot`
$\backslash$	`\backslash`	∂	`\partial`	\$	`\$`	%	`\%`

Algunos puntos que hay que tener en cuenta en la composición de fórmulas:

- Los subíndices y superíndices se indican con los signos `_` y `^`, respectivamente. Si el subíndice o superíndice está formado por dos o más caracteres, hay que entrarlo entre llaves. Por ejemplo, `x_i` produce x_i y `x^{25}` produce x^{25}, pero, cuidado, `x^25` produce x^25.

- Disponéis de diversos tipos de letra para usar en fórmulas matemáticas; las más útiles son:

  - Negrita, que se indica con `\mathbf{...}`; por ejemplo, `\mathbf{X}` y `\mathbf{a}` producen $\mathbf{X}$ y $\mathbf{a}$, respectivamente.
  - La llamada «Negrita de pizarra», que se usa en las notaciones de algunos conjuntos de números y se indica con `\mathbb{...}`; por ejemplo, `\mathbb{N}` produce $\mathbb{N}$ (el conjunto de los números naturales), `\mathbb{Z}` produce $\mathbb{Z}$ (los números enteros), `\mathbb{Q}` produce $\mathbb{Q}$ (los números racionales), `\mathbb{R}` produce $\mathbb{R}$ (los números reales), y `\mathbb{C}` produce $\mathbb{C}$ (los números complejos).

  - Caligráfica, que se indica con `\mathcal{...}`; así, por ejemplo, `\mathcal{A}` y `\mathcal{S}` producen $\mathcal{A}$ y $\mathcal{S}$, respectivamente. Este tipo sólo se puede usar en mayúsculas.

- `\sqrt` produce raíces cuadradas o de orden superior: `\sqrt{xyz}` produce $\sqrt{xyz}$ mientras que `\sqrt[n]{xyz}` produce $\sqrt[n]{xyz}$.

- `\frac` produce fracciones; su tamaño y composición depende de si la fórmula ha de aparecer en el interior de un párrafo o ha de aparecer en línea aparte: `$\frac{abc}{xyz}$` y `$$\frac{abc}{xyz}$$` producen, respectivamente, $\frac{abc}{xyz}$ y

$$\frac{abc}{xyz}$$

- Los operadores $\sum$, $\prod$ etc. también tienen dos versiones, según se escriban entre dólares o entre dobles dólares; por ejemplo `$\sum_{i=1}^n x_i$` y `$$\sum_{i=1}^n x_i$$` producen, respectivamente, $\sum_{i=1}^{n} x_i$ y

$$\sum_{i=1}^{n} x_i$$

- Podemos especificar que los delimitadores se adapten a la altura de la expresión que envuelven, combinándolos con `\left` y `\right`. Comparad `$$(\frac{abc}{xyz})$$`, que produce

$$(\frac{abc}{xyz})$$

  con `$$\left(\frac{abc}{xyz}\right)$$`, que produce

$$\left(\frac{abc}{xyz}\right)$$

- Podemos incluir matrices, o, más en general, tablas, en las fórmulas matemáticas. Una tabla se define empezando con `\begin{array}{`*formato*`}` y acabando con `\end{array}`. El formato es una secuencia de letras `l` (de izquierda, *left*), `r` (de derecha, *right*) o `c` (de centrada): el número de letras indica el número de columnas, y cada letra indica el tipo de alineamiento de la columna correspondiente. Así, por ejemplo, `\begin{array}{rccl}` define una tabla de cuatro columnas: la primera alineada a la derecha, la segunda y la tercera centradas, y la cuarta alineada a la izquierda.

  Entre el `\begin{array}{...}` y el `\end{array}` se introducen por filas los valores de la tabla: los elementos de cada fila se separan con el signo `&` y el cambio de fila se indica con `\\`.

  Para definir una matriz, hemos de envolver la tabla con los delimitadores `\left(` y `\right)`. Por ejemplo,

```
$$
\left(\begin{array}{ccc}
a_{1,1} & a_{1,2} & a_{1,3}\\
a_{2,1} & a_{2,2} & a_{2,3}
\end{array}\right)
$$
```

produce la matriz (en línea aparte)

$$\left(\begin{array}{ccc} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \end{array} \right)$$

De manera similar, para indicar el determinante de una matriz, hemos de usar los delimitadores `\left|` y `\right|`.

Dos cuestiones a tener en cuenta:

- Si sólo queremos usar un delimitador a un lado de la tabla, tenemos que incluir al otro lado `\left.` o `\right.`, según corresponda; las marcas `\left` y `\right` siempre han de ir en parejas. Por ejemplo,

  ```
  $$
  \left.\begin{array}{l}
  2x+3y=5\\
  6x-2y=8
  \end{array}\right\}
  $$
  ```

  produce el sistema de ecuaciones

  $$\left. \begin{array}{l} 2x + 3y = 5 \\ 6x - 2y = 8 \end{array} \right\}$$

- Si en una entrada de una tabla en modo matemático queremos introducir *texto*, lo tenemos que incluir en una caja de texto definida con la instrucción `\mbox{`*texto*`}` (`mbox` es la abreviatura de *make a box*). Por ejemplo,

  ```
  $$
  f(x)=\left\{
  \begin{array}{ll}
  2x & \mbox{si } x\leq 0\\
  3x & \mbox{si } x\geq 0
  \end{array}
  \right.
  $$
  ```

  produce

  $$f(x) = \left\{ \begin{array}{ll} 2x & \text{si } x \leqslant 0 \\ 3x & \text{si } x \geqslant 0 \end{array} \right.$$

  Observad que hemos dejado un espacio en blanco dentro de los `\mbox{si }` para separar los «si» de las fórmulas que los siguen: el contenido de un `\mbox{...}` se transforma en texto, y por lo tanto se tienen en cuenta los espacios en blanco igual que en el texto normal.

Si necesitáis escribir expresiones matemáticas de manera regular en vuestros documentos *R Markdown*, os recomendamos que consultéis la sección de Matemáticas del *Wikibook* de LaTeX[1] y el manual *Ecuaciones en LaTeX* de Sebastián Santisi, que encontraréis en el *url* `http://web.fi.uba.ar/~ssantisi/works/ecuaciones_en_latex/`.

[1] `http://en.wikibooks.org/wiki/LaTeX/Mathematics`

2. Parámetros de los *chunks* de R

Los bloques de código de R, o *chunks*, se indican dentro de un documento *R Markdown* de la manera siguiente:

```
```{r}
x=1+1
x
```
```

Para crear un bloque de código, podemos usar la opción «*Insert Chunk*» del menú desplegable «*Chunks*» en la esquina superior derecha de la ventana de ficheros. Si no queréis usar ese menú, tened en cuenta que cada signo ` se produce con un acento grave seguido de un espacio en blanco.

La parte entre llaves que empieza por `r` puede contener diversos parámetros. En primer lugar, se puede incluir una etiqueta, distinta para cada bloque. Estas etiquetas han de ser cadenas de caracteres sin espacios en blanco ni puntos, y se han de separar de la `r` por medio de un espacio en blanco, y son útiles para navegar entre bloques con la opción «*Jump To*» del menú «*Chunks*».

A continuación, y separadas por comas entre ellas y de la etiqueta (o de la `r`, si no hay etiqueta) se pueden especificar algunas opciones que sirven para determinar el comportamiento del bloque al compilar el documento pulsando el botón «*Knit*». Las opciones disponibles, y sus posibles valores, se os mostrarán en un menú desplegable cuando escribáis la coma; las más útiles son:

- `echo`, que permite indicar si se ha de mostrar el código fuente de R en el documento final: igualada a `TRUE`, que es el valor por defecto, lo mostrará, e igualada a `FALSE`, no.
- `eval`, que permite indicar si queremos que se evalúe el código: sus valores son `TRUE`, que es el valor por defecto, y `FALSE`, que hace que no se evalúe.
- `results`, que permite indicar cómo queremos ver el resultado de la ejecución del código en el documento final: sus posibles valores son palabras, y por lo tanto se han de entrar entre comillas. Los más útiles son:
  - `"hide"`, hace que no se muestre el resultado en el documento final.
  - `"markup"`, que es su valor por defecto, muestra los resultados en el documento final línea a línea, encabezados con dos marcas de comentario `##`, y literales, sin que el programa que abre el documento final los interprete como código.
  - `"asis"`, devuelve los resultados en el documento final línea a línea de manera literal, y el programa con el que se abre el documento final los interpreta como texto y formatea adecuadamente.
  - `"hold"`, muestra todos los resultados de golpe al final del bloque de código.

  Observad que no es lo mismo especificar que no se evalúe el código (`eval=FALSE`) que hacer que se evalúe, pero no mostrar el resultado (`results="hide"`): en el segundo caso, el resultado del cálculo no aparecerá en el documento final, pero se podrá usar en bloques de código posteriores.

- `message`, que permite indicar si se han de mostrar en el documento final los mensajes que `R` produce al ejecutar el código: sus valores son `TRUE`, que es el valor por defecto, y `FALSE`.
- `warning`, que permite indicar si se han de mostrar en el documento final los mensajes de advertencia que a veces producen algunas funciones al ejecutarse: sus valores son de nuevo `TRUE`, su valor por defecto, y `FALSE`.

Veamos algunos ejemplos:

- El bloque

```
```{r,echo=TRUE,results="markup"}
x=1:10
x
sqrt(x)
```
```

  produce, en el documento final:

```
x=1:10
x
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
sqrt(x)
```

```
##  [1] 1.000000 1.414214 1.732051 2.000000 2.236068 2.449490
##  [7] 2.645751 2.828427 3.000000 3.162278
```

- El bloque

```
```{r,echo=TRUE,results="asis"}
x=1:10
x
sqrt(x)
```
```

  produce, en el documento final:

```
x=1:10
x
```

[1] 1 2 3 4 5 6 7 8 9 10

```
sqrt(x)
```

[1] 1.000000 1.414214 1.732051 2.000000 2.236068 2.449490 2.645751 2.828427 3.000000 3.162278

- El bloque

```
```{r,echo=TRUE,results="hold"}
x=1:10
x
sqrt(x)
```
```

produce, en el documento final:

```
x=1:10
x
sqrt(x)
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
##  [1] 1.000000 1.414214 1.732051 2.000000 2.236068 2.449490
##  [7] 2.645751 2.828427 3.000000 3.162278
```

- El bloque

```
```{r,echo=FALSE}
x=1:10
x
sqrt(x)
```
```

no aparece en el documento final, y sólo se muestra el resultado:

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
##  [1] 1.000000 1.414214 1.732051 2.000000 2.236068 2.449490
##  [7] 2.645751 2.828427 3.000000 3.162278
```

- El bloque

```
```{r,echo=TRUE,message=TRUE}
library(magic)
magic(5)
```
```

produce, en el documento final:

```
library(magic)
```

```
## Loading required package: abind
```

```
magic(5)
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    9    2   25   18   11
## [2,]    3   21   19   12   10
## [3,]   22   20   13    6    4
## [4,]   16   14    7    5   23
## [5,]   15    8    1   24   17
```

- El bloque

```
```{r,echo=TRUE,message=FALSE}
library(magic)
magic(5)
```
```

produce, en el documento final:

```
library(magic)
magic(5)
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    9    2   25   18   11
## [2,]    3   21   19   12   10
## [3,]   22   20   13    6    4
## [4,]   16   14    7    5   23
## [5,]   15    8    1   24   17
```

3. Los *chunks* en modo línea

Los bloques de código que hemos explicado en la sección anterior sirven para generar resultados en línea aparte. Si queremos introducir dentro de un párrafo un trozo de código de `R` que se ejecute al compilar el documento y muestre el resultado en el documento final, hay que hacerlo con `` `r ...` ``.

Por ejemplo, si en el documento *R Markdown* escribimos

```
El cubo de dos es `r 2^3`, o lo que es lo mismo, $2^3=$ `r 2^3`
```

produce, al pulsar «Knit», la salida

El cubo de dos es 8, o lo que es lo mismo, $2^3 = 8$

Veamos otro ejemplo más práctico. Supongamos que nos dan una muestra y queremos calcular su media, su varianza, su desviación típica y su tamaño muestral. Entonces, podemos cargar los datos y efectuar los cálculos en un bloque de código (que, si queremos, podemos ocultar completamente en el documento final) y a continuación en un párrafo ir llamando los resultados mediante pequeños bloques. Por ejemplo, podríamos usar el bloque

```
```{r,echo=FALSE,result="hide"}
muestra=c(1,2,3,NA,2.8,3.1,4.9)
media=mean(muestra,na.rm=TRUE)
n=length(na.omit(muestra))
varianza=round(var(muestra,na.rm=TRUE)*(n-1)/n,3)
desv.tipica=round(sqrt(varianza),3)
```
```

y a continuación escribir el párrafo

```
La muestra es de tamaño $n=$ `r n`, su media es $\overline{x}=$
`r media`, su varianza es $s^2=$ `r varianza` y su desviación típica
es $s=$ `r desv.tipica`.
```

Entonces, en el documento final, el bloque con los cálculos permanecería oculto, y este párrafo produciría

> La muestra es de tamaño $n = 6$, su media es $\overline{x} = 2.8$, su varianza es $s^2 = 1.403$ y su desviación típica es $s = 1.184$.

4. Figuras

En el curso se estudian varias funciones que producen gráficos, empezando por la función básica `plot`. En un fichero *R markdown*, podemos controlar el tamaño, la posición etc. de los gráficos en el documento final (html, pdf o *Word*) por medio de opciones en el encabezamiento de sus *chunks*. Veamos algunas de las opciones más útiles para el nivel básico de este curso:

- `fig.height` y `fig.width` sirven para especificar la altura y la anchura, respectivamente, del gráfico «real». Sus posibles valores son números enteros y se sobreentiende que están en pulgadas (no pueden usarse otras unidades); por defecto, ambos valen 7.
- `out.height` y `out.width` sirven para especificar la altura y la anchura, respectivamente, del gráfico *en el documento final*. Estas opciones son útiles si se quieren indicar las dimensiones del gráfico en tamaños relativos respecto de las medidas de la caja de texto. Así, por ejemplo, al compilar con `Knit PDF` o `Knit HTML`, si se especifican `fig.height` y `fig.width` (o se dejan con sus valores por defecto) y se incluye la opción `fig.width="60%"` (entrecomillado), el gráfico aparecerá en el documento escalado de manera que su ancho sea un 60 % del ancho del texto. Pero cuidado, cuando se compila con `Knit Word`, `fig.width="60%"` no tiene ningún efecto.

  Como norma general, os recomendamos que uséis `fig.height` y `fig.width` para especificar las dimensiones del gráfico y lo incluyáis tal cual en el documento. Solo os recomendamos usar `out.height` o `out.width` cuando de trate de gráficos que se producen de manera diferente según sus dimensiones, como pueden ser las nubes de palabras de la Lección 12. Para incorporar una nube de palabras a tamaño reducido en un documento *R markdown*, lo mejor es dejar que `R` la produzca con su tamaño por defecto y luego reducirla con `out.width`: si reducís sus `fig.height` y `fig.width`, puede que algunas palabras no quepan en el gráfico o que queden muy apelmazadas.
- `fig.asp` permite fijar la proporción altura/anchura de un gráfico si solo se especifica su `fig.width`.
- `fig.show` sirve para controlar cómo se incluyen en el documento final los gráficos producidos en el *chunk* cuando hay más de uno. Su valor por defecto es `"as.is"`, que los va incluyendo a medida que se generan. Con `fig.show="hold"` se dibujan todos los gráficos de golpe al final del *chunk*.
- `fig.align` permite alinear los gráficos respecto del texto. Sus posibles valores son `left`, `right` y `center` (a la izquierda, a la derecha y centrados, respectivamente). Si no se

especifica, no se efectúa ningún alineamiento específico y se adapta al alineamiento del texto que rodea el *chunk*.

- `dev` sirve para especificar el tipo de fichero gráfico que se genera. Solo hay un caso en que nos parece útil usar esta opción: si pensáis publicar vuestro documento html como una página web, os conviene usar `dev="svg"` y el gráfico se creará en formato «*Scalable Vector Graphics*», con lo que se escalará de manera correcta y sin perder calidad al aumentar o disminuir el ancho de la página web en el navegador.

Para más opciones, podéis consultar la Guía de Referencia de *R markdown*, que encontraréis en el *url* `http://www.rstudio.com/wp-content/uploads/2015/03/rmarkdown-reference.pdf` y en el repositorio de ficheros del curso, o la página web `http://yihui.name/knitr/options/`.

Por otro lado, dentro de un *chunk* podemos usar las funciones `par` y `layout`, que también aparecen en algunas lecciones del curso. Estas funciones sirven para modificar el aspecto de los gráficos generados con `R`, y en particular pueden usarse para especificar el modo como se agrupan los gráficos producidos en un mismo *chunk*. Su efecto es el mismo al usarlas en la consola, salvo que allí los cambios introducidos con estas funciones son permanentes hasta que volvamos a cambiarlos, mientras que en un documento *R markdown* solo afectan a su *chunk*.

Con la función

`par(mfrow=c(`x`,` y`))`

organizaremos los gráficos producidos en el *chunk* formando una matriz de x filas e y columnas, llenando sus entradas por filas. A modo de ejemplo, si un *chunk* produce 3 gráficos y queremos mostrarlos juntos, con la opción `fig.show="hold"` aparecerán al final del chunk, uno debajo del otro. En cambio, si no especificamos la opción `fig.show` y al principio del *chunk* entramos `par(mfrow=c(1,3))`, se dibujarán uno al lado del otro formando un único gráfico (y entonces conviene ajustar con `fig.height`, `fig.width` o `fig.asp` sus dimensiones: por ejemplo, seguramente no querremos que este gráfico compuesto sea cuadrado).

Esta misma construcción se puede usar en la consola, pero entonces hay que tener en cuenta que a partir de la instrucción `par(mfrow=c(`x`,` y`))` todos los gráficos se agruparán de esta manera. Para evitarlo, entrad `par(mfrow=c(1,1))` cuando queráis volver al modo por defecto de los gráficos uno a uno. Veamos un ejemplo:

```
> par(mfrow=c(2,2))
> plot(iris[iris$Species=="setosa",1:2], main="Sépalos de setosa",
  xlab="Longitud", ylab="Anchura", pch=20)
> plot(iris[iris$Species=="versicolor",1:2], main="Sépalos de
  versicolor", xlab="Longitud", ylab="Anchura", pch=20, col="red")
> plot(iris[iris$Species=="virginica",1:2],main="Sépalos de virginica",
  xlab="Longitud", ylab="Anchura", pch=20, col="blue")
> plot(iris[ 1:2],main="Sépalos de iris",xlab="Longitud",
  ylab="Anchura", pch=20, col="green")
> par(mfrow=c(1,1))
```

produce la Figura 1 y restaura el formato original. En general, con la función `par` se pueden modificar «hasta nueva orden» muchos aspectos de los gráficos. Si lo hacéis, es conveniente que antes de modificarlos guardéis su valor actual con, por ejemplo, `par.anterior=par()`. A continuación, ya podéis modificar los parámetros de `par` que queráis; si en algún momento

queréis volver al estilo original, bastará que entréis `par(par.anterior)`.

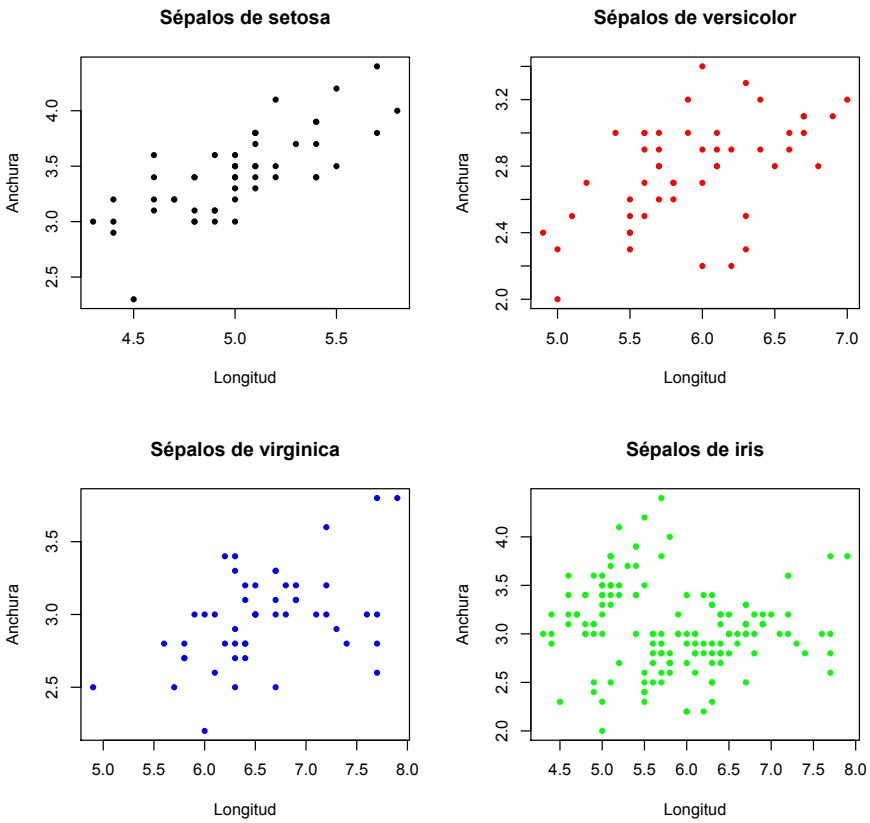

Figura 1. Cuatro gráficos en uno organizados con la instrucción `par(mfrow=c(2,2))`.

Pasemos a la función `layout`. Su sintaxis básica es

`layout(`M`, widths=..., heights=...)`

donde

- M es una matriz que indica la composición en forma de cuadrícula de los gráficos que queremos agrupar en una misma figura. Sus entradas han de ser $0, 1, 2, \ldots, n$ donde n el número de gráficos que queremos organizar (el 0 puede faltar, pero no podemos saltarnos ninguna entre 1 y n). Entonces, si la entrada (i, j) de la matriz M es el número $k \geqslant 1$, el k-ésimo gráfico producido en el *chunk* se situará en la posición (i, j) de la cuadrícula; si la entrada (i, j) de la matriz es 0, la posición (i, j) de la cuadrícula quedará vacía. Si un valor se repite en diferentes entradas de M, el gráfico correspondiente se reparte entre estas entradas.
- El parámetro `widths` sirve para especificar las anchuras relativas de las diferentes columnas (por defecto, todas iguales): para ello, se iguala al vector de estas anchuras.
- El parámetro `heights` sirve para especificar, de manera similar, las alturas relativas de las diferentes filas; por defecto, todas son iguales.

Así, el código

```
> M=matrix(c(1,1,1,2,3,4),nrow=2,byrow=TRUE)
> M
     [,1] [,2] [,3]
[1,]    1    1    1
[2,]    2    3    4
> layout(M)
> plot(iris[ 1:2], main="Sépalos de iris", xlab="Longitud",
  ylab="Anchura", pch=20, col="green")
> plot(iris[iris$Species=="setosa",1:2], main="Sépalos de setosa",
  xlab="Longitud", ylab="Anchura", pch=20)
> plot(iris[iris$Species=="versicolor",1:2],main="Sépalos de
  versicolor", xlab="Longitud", ylab="Anchura", pch=20, col="red")
> plot(iris[iris$Species=="virginica",1:2],main="Sépalos de virginica",
  xlab="Longitud", ylab="Anchura", pch=20, col="blue")
> layout(1)
```

produce la Figura 2, que no es ninguna maravilla, pero sirve para observar cómo la matriz M organiza los gráficos: el primer `plot` ocupa toda la primera fila, y los otros tres, las tres posiciones de la segunda fila ordenados de izquierda a derecha. Como antes, el efecto de `layout` es permanente, por lo que si queréis volver al modo «gráficos de uno en uno» lo más práctico es que entréis al terminar vuestro gráfico compuesto la instrucción `layout(1)`, como hemos hecho en el código anterior. Encontraréis otro ejemplo del uso de `layout` al final de la Lección 11.

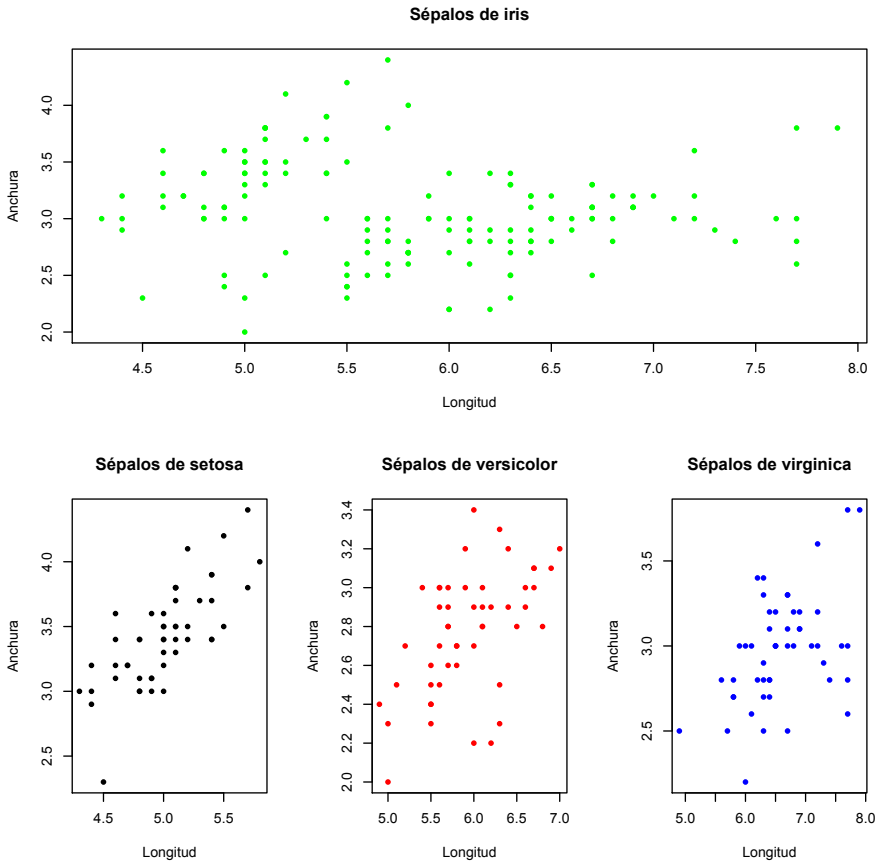

Figura 2. Tres gráficos en uno organizados con la instrucción `layout(matrix(c(1,1,1,2,3,4),nrow=2,byrow=TRUE))`.

5. Tablas

La manera más sencilla de incluir tablas producidas con `R` en un fichero *R markdown* es usando el paquete `printr`. A día de hoy (mayo 2016) aún no ha sido incorporado al repositorio de la *CRAN*, y para instalar su última versión hay que ejecutar en la consola

```
> install.packages("printr", type="source",
    repos=c("http://yihui.name/xran"))
```

Una vez instalado, para usarlo en un fichero *R markdown* hay que cargarlo con `library` en un *chunk* al principio; recordad que para usar un paquete en un fichero *R markdown* no basta que esté cargado en la sesión de la consola, se tiene que cargar específicamente en el fichero.

Este paquete hace que las matrices, tablas de contingencia y *data frames* se presenten como tablas de manera adecuada, que además dependerá del formato de salida elegido. Veamos algunos ejemplos de *chunks* y cómo se ven sus resultados. Para empezar, veamos como muestra los *data frames* en el documento final. El *chunk*

````
```{r}
library(printr)
head(iris)
```
````

produce en un documento html la tabla (a) de la Figura 3, en un documento pdf, la tabla (b), y en un documento *Word*, la tabla (c). A partir de ahora, y en lo que resta de sección, sólo mostraremos el resultado en un documento pdf. Además, en los *chunks* que siguen ya suponemos que hemos cargado el paquete `printr` en el *chunk* anterior.

Veamos ahora como se generan las tablas de contingencia en el documento final. El *chunk*

````
```{r}
Sexo=c("M","H","H","H","H","M","M","H","M","H","H","M","M","H",
  "H","M","H","M","H","H","H","M","H","M","H","H","H","H","H","M")
Respuesta=c("No","Si","Si","Si","No","No","Si","No","No","No",
  "No","No","Si","No","No","Si","Si","Si","Si","Si","Si","No","Si",
  "No","Si","Si","Si","No","Si","Si")
Edad=c("40-60","40-60",">60",">60","40-60","20-40","20-40",
"40-60","20-40","<20",">60","<20","40-60","40-60","<20",
"20-40","40-60","<20",">60","40-60","40-60","<20","20-40",
"20-40",">60","20-40","40-60","20-40","40-60","<20")
table(Edad)
table(Respuesta,Edad)
table(Respuesta,Sexo,Edad)
ftable(Respuesta,Sexo,Edad)
```
````

produce las tablas de contingencia unidimensional, bidimensional y tridimensional de la Figura 4.

Veamos finalmente como se ven las matrices en el documento final. El *chunk*

````
```{r}
````
````

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa

(a)

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa

(b)

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa

(c)

Figura 3. Tabla de las seis primeras filas de `iris` producidas con el paquete `printr`: (a) en formato html; (b) en formato pdf; (c) en formato *Word*.

```
M=matrix(c(2,8,5,5,9,7,3,6,6,5,2,7,3,7,7,8,9,6,9,3,2,
   3,3,10,8,10,2,10,8,7,3,3,2,6,3,2,1,9,8,5),nrow=8)
M
dimnames(M)=list(NULL, paste("Col",1:5,sep="."))
M
```

produce, con la primera M, la tabla (a) de la Figura 5 y con la segunda, con nombres en las columnas, la (b).

Si en algún momento queréis volver a la presentación usual de matrices, *data frames* y tablas de contingencia, basta que descarguéis el paquete `printr` entrando en un *chunk*

```
detach("package:printr", unload=TRUE)
```

y luego ya lo volveréis a cargar si lo necesitáis de nuevo. Así, con

```
```{r}
detach("package:printr", unload=TRUE)
M
```
```

<20	>60	20-40	40-60
6	5	8	11

(a)

Respuesta/Edad	<20	>60	20-40	40-60
No	4	1	4	4
Si	2	4	4	7

(b)

Respuesta	Sexo	Edad	Freq
No	H	<20	2
		>60	1
		20-40	1
		40-60	3
	M	<20	2
		>60	0
		20-40	3
		40-60	1
Si	H	<20	0
		>60	4
		20-40	2
		40-60	6
	M	<20	2
		>60	0
		20-40	2
		40-60	1

(c)

Figura 4. Tablas de contingencia producidas en formato pdf con el paquete `printr`: (a) unidimensional; (b) bidimensional; (c) tridimensional.

obtenemos

```
##       Col.1 Col.2 Col.3 Col.4 Col.5
## [1,]     2     6     9     8     2
## [2,]     8     5     6    10     6
## [3,]     5     2     9     2     3
## [4,]     5     7     3    10     2
## [5,]     9     3     2     8     1
## [6,]     7     7     3     7     9
## [7,]     3     7     3     3     8
## [8,]     6     8    10     3     5
```

2	6	9	8	2
8	5	6	10	6
5	2	9	2	3
5	7	3	10	2
9	3	2	8	1
7	7	3	7	9
3	7	3	3	8
6	8	10	3	5

(a)

Col.1	Col.2	Col.3	Col.4	Col.5
2	6	9	8	2
8	5	6	10	6
5	2	9	2	3
5	7	3	10	2
9	3	2	8	1
7	7	3	7	9
3	7	3	3	8
6	8	10	3	5

(b)

Figura 5. Matrices producidas en formato pdf con el paquete `printr`: (a) sin nombres de columnas; (b) con nombres de columnas.

Universitat de les Illes Balears

Campus Extens
UIB Virtual

http://**campusextens**.uib.cat


@campusextensUIB


http://www.scoop.it/t/recursos-i-eines-per-al-professorat


http://campusextensrecursos.uib.es/