Extras de *R Markdown* 1: Fórmulas y *chunks*

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* debería ser suficiente. Este documento lo podéis encontrar pulsando el botón "?" situado en la barra superior de la ventana de ficheros y escogiendo "Markdown Quick Reference". También os puede ser útil la chuleta de R Markdown rm-cheatsheet.pdf que encontraréis en:

`http://shiny.rstudio.com/images/rm-cheatsheet.pdf.zip`.

No obstante, a lo largo del curso ampliaremos sus contenidos en algunos temas cuando lo creamos necesario. Para empezar, vamos a tratar dos puntos: cómo escribir fórmulas matemáticas bien formateadas y cómo controlar el comportamiento de los bloques de código (*chunks*) al compilar el fichero *R markdown*, y su aspecto en el documento final.

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 símbolos $\pm$ 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 símbolos matemáticos más usuales:

Algunos operadores binarios

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

Algunos símbolos 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 símbolos

$\ldots$	`\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 símbolos `_` 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

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

  Si queremos componer en el interior de un párrafo una fracción del estilo de las fracciones compuestas en línea aparte, hay que usar `\dfrac` en lugar de `\frac`.

- 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 `$(\dfrac{abc}{xyz})$`, que produce $(\dfrac{abc}{xyz})$, con `$\left(\dfrac{abc}{xyz}\right)$`, que produce $\left(\dfrac{abc}{xyz}\right)$.

- 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 símbolo `&` 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 por cualquier motivo 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 queréis usar ese menú, tened presente que cada símbolo ` 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 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 $s = 1.184$.