

# Aprendiendo a escribir en QMD

(Web del curso [aquí](#))

26 de junio de 2023

# Lo que ya sabemos

---

- Trabajamos con Qprojects
- Documento fuente escrito en Qmd genera diferentes outputs

# ¿Qué es Quarto?

---

Quarto is a multi-language, next generation version of R Markdown, with many new features and capabilities.

Puedes ver [este video](#) de 100 segundos

# ¿Qué es Quarto?

- Un nuevo sistema de publicación científica y técnica de código abierto basado en Pandoc
- Quarto no es un paquete, es un programa independiente, un CLI
- Permite incorporar **texto y código** para producir documentos (reproducibles) en múltiples formatos
- Es ... la “segunda generación de Rmarkdown”
- Muy parecido a Rmarkdown, pero ... **no requiere R**. Soporta lenguajes como Python, Julia y Observable.
- Quarto utiliza Knitr para ejecutar el código R; así que es **capaz de procesar también los ficheros .Rmd** sin modificarlos

## Extensión: Ventajas de Quarto sobre RMD

- Proyecto en [desarrollo activo](#) ... mientras que Rmarkdown [it's not going away](#) pero ...
- Quarto **unifica funcionalidades** de varios paquetes del entorno Rmd como xaringan, bookdown, blogdown , ...
- **Por ejemplo:** Cross references, Call-outs, Advanced Layout (tb para imágenes), Extensiones, Interactividad, YAML intelligence, Quarto Pub, Divs, Spans
- Para ver si estas ventajas merecen la pena para ti puedes leer a [Occasional Divergences](#), [Nick Tierney](#), [Alison Hill](#), [Danielle Navarro](#), [Jumping Rivers](#) o [esta pregunta](#) de Stack Overflow.

## Extensión: ¿Qué es Rmarkdown? ¿Para qué sirve?

- El predecesor de Quarto
- Un “[entorno](#)” para hacer informes/publicaciones/transparencias **REPRODUCIBLES** con R.

Is an authoring framework for data science, combining your code, its results, and your prose. R Markdown documents are fully reproducible and support dozens of output formats, like PDFs, Word files, slideshows, and more.

- Con Rmd se pueden generar **multitud de outputs**. Por ejemplo, visita [está galería](#) o [este listado](#)

### Una oda a Rmarkdown

- [How Rmarkdown changed my life](#): charla de Rob Hyndman sobre su proceso hasta llegar a usar Rmarkdown para hacer sus documentos científicos y webs.

# Para poder practicar lo que vayamos aprendiendo ...

## Tarea 3.0: Vamos a crear un nuevo Qproject

- Cierra RStudio
- Crea un nuevo Qproject llamado “my\_qproject”
- Inspecciona los documentos que hay dentro de “my\_qproject”
- Borra el archivo `my_qproject.qmd`
- Crea un archivo `.qmd`
- Procésalo y llámalo `prueba_01.qmd`

# Qmarkdown: guía rápida ( `.qmd` )



# Los documentos **.qmd** tienen 3-4 partes

1. Encabezamiento (**yml** header)
2. Trozos de **código** R (R chunks)
3. **Texto** (escrito en Markdown) ...  
... y **todo lo demás**: imágenes, links, ecuaciones, etc ...

## Un ejemplo

# Mi primer documento con Quarto

AUTHOR

Pedro J. Pérez

## Introducción a Quarto

Este es un documento Quarto (formato .qmd).

El bloque de abajo es un “chunk” con código R.

```
2 + 2
```

```
[1] 4
```

## ¿Cómo se procesa un .qmd?

Al clicar en el botón “Render” se generará el documento final en formato

`.html`

El documento final contendrá el texto (debidamente formateado), el código R y **el resultado de la evaluación del código**.

Esto es todo por ahora!!

```
1 ---
2 title: "Mi primer documento con Quarto"
3 author: "Pedro J. Pérez"
4 format: html
5 ---
6
7 ## Introducción a Quarto
8
9 Este es un documento Quarto (formato .qmd).
10
11 El bloque de abajo es un "chunk" con código R.
12
13 ```{r}
14 2 + 2
15 ```
16
17 ### ¿Cómo se procesa un .qmd?
18
19 Al clicar en el botón "Render" se generará el documento final en formato `.html`
20
21 El documento final contendrá el texto (debidamente formateado), el código R y
22 el resultado de la evaluación del código.
23
24 Esto es todo por ahora!!
25
```

# (I): YAML

---

- Se utiliza para especificar metadatos (opciones) del documento final

# (I) El encabezamiento o “yaml header”

- Se (suele) poner al principio del documento, entre estas marcas: `---`
- En el yaml son MUY importantes los **espacios y la indentación**

## Ejemplos de yaml

ejemplo 1

ejemplo 2

ejemplo 3

```
1 ---  
2 title: "Mi primer archivo qmd"  
3 date: "2023-08-08"  
4 format: html  
5 ---
```

- Veremos más opciones de YAML cuando veamos la creación de páginas web

### Tarea 3.1: Vamos a jugar con el `yaml`

- Utiliza el archivo `prueba_01.qmd` para probar como quedaría `prueba_01.html` al usar los 3 anteriores `yaml`'s
- Una vez hayas probado a usar el tercer YAML, prueba algún otro `theme` de los 25 disponibles. [Aquí](#) tienes el listado, y [aquí](#) hay ejemplos para ver cómo quedan y decidir cual te gusta más.

# (II): CHUNKS

---

- En documentos `.qmd` podemos incorporar código
- Código que puede ser **ejecutado**
- Así se podrá **mostrar** en el documento final tanto el código como **los resultados** (de la evaluación del código)
- Esto es lo que permite que los documentos sean **reproducibles**
- La documentación oficial está [aquí](#)

## (II) Code Blocks o chunks (código R)

- Para que Quarto sepa qué partes del `.qmd` es **código R**, debe ir dentro de estas marcas:

```
```${r}```  
```\n`
```

- Por ejemplo:

```
```${r}```  
# se pueden poner comentarios  
# de la forma habitual  
aa <- c(1, 4, 9, 16, 25)  
sqrt(aa)  
```\n`
```

- Cuando Quarto/**knitr** procesen el chunk, se interpretará como código R y **ejecutará las instrucciones**, de forma que, en el documento final, se **mostrará el output** generado por el chunk.

```
# se pueden poner comentarios  
# de la forma habitual  
aa <- c(1, 4, 9, 16, 25)  
sqrt(aa)
```

```
## [1] 1 2 3 4 5
```



## (II) Chunks: los chunks pueden tener opciones.

- Por ejemplo:

```
```${r}
#| echo: true
#| eval: false
2 + 2
```
```

- Las principales opciones son:

| Opción               | (valor por defecto) | ¿Que hace?                               |
|----------------------|---------------------|--|
| <code>echo</code>    | true                | Incluye el código en el documento final  |
| <code>eval</code>    | true                | Evalúa el código                         |
| <code>warning</code> | true                | Incluye los avisos en el documento final |
| <code>message</code> | true                | Incluye los mensajes                     |

## (II) Chunks: los chunks pueden tener opciones.

Extensión: hay más opciones para los chunks

| Opción               | (valor por defecto) | ¿Que hace?   |
|----------------------|---------------------|--|
| <code>output</code>  | true                | Incluye el resultado de la evaluación del código             |
| <code>error</code>   | false               | Si pones <code>true</code> , se mostrarán los errores (!!!!) |
| <code>include</code> | true                |  |
| <code>label</code>   |                     | Si quieres poner label al chunk                              |

- Puedes ver todas las opciones [aquí](#) 🐱

## Tarea 3.2: Vamos a incorporar un chunk a nuestro `.qmd`

- Utiliza el archivo `prueba_01.qmd` para insertar un chunk de código R con las siguientes instrucciones:

```
1 library(tidyverse)
2 ggplot(iris, aes(Sepal.Length, Petal.Length)) + geom_point()
```

Si no tuvieses el paquete `tidyverse` instalado, puedes usar este otro chunk:

```
1 plot(iris$Petal.Length, iris$Petal.Width)
```

### Ayudita con las marcas de los chunks

- Para incorporar las marcas que delimitan los chunks de código, puedes usar este atajo del teclado: Alt-Ctrl-I

### Tarea 3.2b: Vamos a jugar con la opciones de los chunks

- Incorpora al chunk las siguientes opciones: `echo` y `eval`
- ¿Cuales son las opciones por defecto para `echo` y `eval`?
- Incorpora también al chunk las siguientes opciones: `message` y `label`

### Tarea 3.2b: Una solución

```
1 #| eval: true
2 #| echo: true
3 #| message: false
4 #| label: my-primer-chunk
5
6 library(tidyverse)
7 ggplot(iris, aes(Sepal.Length, Petal.Length)) + geom_point()
```

## Extensión: más detalles sobre las opciones de los chunks

- `include`: si en un chunk pones `#| include: false`, el código de ese chunk **se ejecutará pero no se mostrará nada**, ni el código, ni el resultado, ni mensajes, ni errores. Esta opción es útil, por ejemplo, para cargar los paquetes de R.
- `#| label: setup`: si un chunk se llama `setup`, **este chunk se ejecutará siempre que ejecutes otro chunk** del mismo documento. Esta opción suele usarse para cargar los paquetes necesarios para un análisis, de formas que estos siempre estén disponibles cuando ejecutes otro chunk. Esto puede ser una buena práctica.

## Extensión: dos detalles más sobre opciones de los chunks

- `#| output: asis` : permite generar raw-markdown. Un ejemplo [aquí](#)
- `echo`: además de los típicos `true` y `false`, ahora **incorpora un nuevo valor** `#| echo: fenced` que facilita mostrar las marcas de los chunks en el documento final. Documentación [aquí](#).

## (II) Chunks: opciones de los chunks en el YAML

- En el YAML, podemos fijar los valores por defecto de las opciones de los chunks
- Se han de poner anidadas dentro de `execute:`

Por ejemplo:

## .Qmd (chunk options en el chunk)

```
1 ---
2 title: "Documentos con Quarto (.qmd)"
3 author: "Pedro J. Pérez"
4 date: "`r Sys.Date()`"
5 format: html
6 ---
7
8 ```{r}
9 #| label : sumando
10 #| echo: true
11 #| eval: false
12 2 + 2
13 ```
14
```

## .Qmd (chunk options en el yaml)

```
1 ---
2 title: "Documentos con Quarto (.qmd)"
3 author: "Pedro J. Pérez"
4 date: "`r Sys.Date()`"
5 format: html
6 execute: |
7   echo: true
8   eval: false
9 ---
10
11 ```{r}
12 #| label: sumando
13 2 + 2
14 ```
15
```



### Tarea 3.3: opciones de los chunks en el YAML

- Utiliza el archivo `prueba_01.qmd` para insertar las opciones de los chunks en el YAML del documento
- Por ejemplo, indica en el YAML que `echo: true` pero `eval: false`

### Tarea 3.3: Solución

```
1 execute:  
2   echo: true  
3   eval: false
```

### Extensión: uso de la opción `include`

- ¿Se puede poner la opción `include:` en el YAML?

## (II) Chunks: Más opciones de `knitr`

- Si usamos `knitr` para ejecutar los chunks, entonces podemos usar todas las [opciones nativas de knitr](#), como: `collapse`, `fig.width`, `comment`, etc ... Más información [aquí](#). Un ejemplo:

```
1 ---
2 title: "Mi segundo post"
3 author: "Pedro J. Pérez"
4 execute:
5   echo: fenced
6 knitr:
7   opts_chunk:
8     collapse: false
9     comment: "#>"
10 ---
11
12 Este es mi **segundo** post
13
```

- Si quieres ver todas las opciones disponibles para los chunks en `knitr` tendrás que ir a la [página web de knitr](#), a la [cheat sheet sobre Rmarkdown](#), o a la [Reference Guide](#)

## Extensión: Mi chunk de setup en RMD

```
1  ```{r chunk-setup, include = FALSE}
2  knitr::opts_chunk$set(echo = TRUE, eval = TRUE, message = FALSE, warning = FALSE,
3                        #results = "hold",
4                        cache = FALSE, cache.path = "/caches/", comment = "#>",
5                        #fig.width = 7, #fig.height= 7,
6                        #out.width = 7, out.height = 7,
7                        collapse = TRUE, fig.show = "hold",
8                        fig.asp = 7/9, out.width = "60%", fig.align = "center")
9
10 options(scipen = 999) #- para quitar la notación científica
11  ```
```

- Las opciones del chunk de setup, en el YAML quedarían como:

```
1  ---
2  title: "My Document"
3  execute:
4    echo: true
5  knitr:
6    opts_chunk:
7      results: "hold"
8      collapse: true
9      comment: "#>"
10     fig.asp: 7/9,
11     out.width: "60%"
12     fig.align: "center"
13     fig.show: "hold"
14     R.options:
15       scipen: 999  #- para quitar la notación científica
16  ---
```

## (II) Chunks: otras opciones de Quarto para los chunks

Ahora, en Quarto, hay más opciones para los chunks. Documentación oficial [aquí](#). Por ejemplo:

- Hacer **folding code** con `#| code-fold: true`
  - Puedes combinarlo con `#| code-summary: "Para ver el código, pincha"`
- Si el código es muy largo, puedes usar `#| code-overflow: wrap` (o `scroll`)
- Puedes hacer que se muestren los **números de línea** con `#| code-line-numbers: true`

Todas estas opciones **también** pueden ir en el YAML.

### Tarea 3.4: Nuevas opciones de los chunks

- Utiliza el archivo `prueba_01.qmd` para probar algunas de las nuevas opciones de los chunks en el YAML
- Por ejemplo:

```
1 format:
2   html:
3     code-fold: true
4     code-summary: "Para ver el código, pincha"
5     code-line-numbers: true
```

- Puedes también probar a usar la opción `code-tools: true` y ver que ocurre.

La documentación oficial la tienes [aquí](#) y [aquí](#)

## (II) Chunks: Code annotation

- Desde Quarto 1.3 podemos incluir en los chunks **line-based annotations**. Documentación [aquí](#). Por ejemplo:

- El código:

```
1  ```r
2  library(tidyverse)
3  library(palmerpenguins)
4  penguins |>                                # <1>
5    mutate(                                   # <2>
6      bill_ratio = bill_depth_mm / bill_length_mm, # <2>
7      bill_area  = bill_depth_mm * bill_length_mm # <2>
8    )                                           # <2>
9  ```
10 1. Take `penguins`, and then,
11 2. add new columns for the bill ratio and bill area.
```

- El resultado:

```
1  library(tidyverse)
2  library(palmerpenguins)
3  penguins |>                                ①
4    mutate(                                   ②
5      bill_ratio = bill_depth_mm / bill_length_mm,
6      bill_area  = bill_depth_mm * bill_length_mm
7    )
```

- ① Take **penguins**, and then,
- ② add new columns for the bill ratio and bill area.

(\*) (faltan las {} en el chunk)

## (II) Código: inline code

- La mayoría del código suele ir en los chunks, pero a veces necesitamos **inline code**; es decir, código R dentro de nuestro texto.

### Por ejemplo

- Si quiero describir un conjunto de datos puedo hacerlo escribiendo: “El data.frame iris tiene 150 filas y 5 variables”
- Pero es **mejor hacerlo con inline code**. Para ello tienes que poner el código R dentro de estas marcas: ``r``
- Habría que escribirlo así:
  - El data.frame iris tiene ``r nrow(iris)`` filas y ``r ncol(iris)`` variables.

## Tarea 3.5: Código inline

- Utiliza el archivo `prueba_01.qmd` para incorporar un párrafo de texto que contenga código inline.
- Por ejemplo, puedes incorporar este párrafo:

Este archivo que estoy escribiendo lo hice el 26 de junio de 2023.

pero lo que queremos es que la fecha se modifique cada vez que se procese el documento `prueba_01.qmd`

**Pista:** puedes probar a usar parte del siguiente código R

```
1 Sys.Date()  
2 format(Sys.Date(), "%d %B, %Y")
```



## (II) Chunks: bloques de código **no ejecutables**

- Hemos visto que podemos incluir código R en nuestro `.qmd`.
- El código R normalmente **es ejecutado** por el paquete `knitr`. Para ello hay que usar estas marcas:

```
` `` {r}  
2 + 2  
` ``
```

- Pero, además, como estamos usando Markdown, podemos incorporar en nuestro `.qmd` bloques de código **no ejecutables**.
- Se usan las mismas marcas `` ```, pero sin las llaves:

```
1 ` `` r  
2 2 + 2  
3 ` ``
```

- Pandoc soporta syntax highlighting para cerca de [140 different languages](#). si no estuviese el lenguaje que necesitas, usa el `default` language.
- Documentación oficial [aquí](#).

### Tarea 3.6: bloques de código no ejecutable

- Utiliza el archivo `prueba_01.qmd` para incorporar:
  - un bloque de código R **no ejecutable**
  - un bloque de código Phyton **no ejecutable**

## Extensión: diferencias entre Quarto y Rmd

- **No hace falta chunk inicial:** con Quarto se pueden poner las opciones de los chunks en el YAML
- En ficheros `.qmd`, las opciones de los chunks se pueden especificar globalmente en el YAML y a nivel individual en cada uno de los chunks.
- En los **chunks individuales** ahora se utiliza la **sintaxis YAML** (`key: value`) en líneas dentro del chunk que empiezan con `#|`. Por ejemplo:

```
1 ---
2 title: "Mi segundo post"
3 execute:
4   echo: false
5   eval: true
6 ---
7
8 Este es mi **segundo** post con Quarto
9
10 ```{r}
11 #| eval: false
12 2 + 2
13 ```
14
15 ```{r}
16 summary(mtcars)
17 ```
```

- Un ejemplo (diferencias `.qmd` vs. `.Rmd`):

.qmd

```
```\r\n#| label: sumando\r\n#| echo: true\r\n#| eval: false\r\n2 + 2\r\n```\r\n
```

.Rmd

```
```\r sumando, echo=TRUE, eval=FALSE\r\n2 + 2\r\n```\r\n
```

## Extensión: convertir de Rmarkdown a Quarto

Esta perfectamente bien explicado por Thomas Mock [aquí](#) y [aquí](#) pero, por si alguna vez desaparecen sus slides, lo replicaré aquí.

La mayoría de las veces, Quarto puede procesar los ficheros `.Rmd`, pero si quieres convertirlos a `.qmd`

- Cambia la **extensión** del fichero de `.Rmd` a `.qmd`
- Cambia la **sintaxis** del `yaml`: de `output: html_document` a `format: html`
- Puedes convertir los **chunks de código**: `knitr::convert_chunk_header("doc.qmd", output = identity)`

# (III) Texto

---

- El texto (o narrativa) **se escribe en Markdown** (concretamente en [Pandoc's Markdown](#))
- Markdown es un **lenguaje de marcado ligero** con sintaxis sencilla que permite dar formato y estructura a un texto y convertirlo a `.html`
- Documentación oficial de Quarto [aquí](#) y su [documento fuente](#)
- Una fantástica guía sobre Markdown [aquí](#)

# (III) Texto: ¿qué narices es esto de Markdown?

- Es un **lenguaje de marcas** diseñado para escribir fácilmente **para la web** y que también sea fácilmente legible.
- Markdown es un lenguaje **estoico** para escribir (para la web)
- Markdown fue creado por [John Grueber](#) y [Aaron Swartz](#) en 2004. Para saber más sobre Aaron Swartz puedes ver [este documental](#).
- Se creó con el objetivo de crear un formato de texto fácil de escribir y leer y que se convirtiera fácilmente en **.html**

Markdown is a text-to-HTML conversion tool for web writers.

- La ventaja de escribir en Markdown es que es un lenguaje muy fácil de aprender y que, como está basado en un formato de texto plano, es y será compatible con la mayoría de plataformas.

- En este otro [tutorial de Markdown](#), se puede leer lo siguiente:

Markdown is a way to write for the web. It's written in what nerds like to call "plaintext". Plaintext is just the regular alphabet, with a few familiar symbols. Unlike cumbersome word processing applications, text written in Markdown can be easily shared between computers, mobile phones, and people. It's quickly becoming the writing standard for academics, scientists, writers, and many more. Websites like GitHub and reddit use Markdown to style their comments.



# (III) Texto: ideas básicas (otra vez)

- En los documentos `.qmd`, “todo” lo que no es el yaml o un chunk, es texto.
- En Quarto el texto se escribe en [Markdown](#)
- Como Quarto se basa en **pandoc**, se usa “[Pandoc`s markdown](#)”, una versión revisada y extendida de la propuesta original de Grueber
- Tienes una fantástica guía de Markdown [aquí](#), y [aquí](#) una cheatsheet.

# (III) Texto: Dando formato al texto

- Sintaxis básica de `markdown`

## Si escribes esto:

texto normal

texto en **\*\*negrita\*\***

texto en *\*cursiva\**

un superíndice<sup>2</sup>

un subíndice<sub>2</sub>

palabras ~~tachadas~~

``verbatim code``

> un blockquote

## Se verá esto:

texto normal

texto en **negrita**

texto en *cursiva*

un superíndice<sup>2</sup>

un subíndice<sub>2</sub>

palabras ~~tachadas~~

`verbatim code`

un blockquote



## Tarea 3.7: texto en markdown

Para practicar a escribir con Markdown podemos:

- Hacerlo en RStudio, por ejemplo podemos utilizar el archivo [prueba\\_01.qmd](#) para incorporar un párrafo que contenga los elementos que hemos visto en la tabla anterior
- Alternativamente, podemos usar editores online como [este](#) o [este](#)
- Aquí tienes un ejemplo:

```
1 Cuando quise darme cuenta la clase había terminado ....
```

## Tarea 3.7: una “solución”

```
1 texto en negrita
2
3 texto en cursiva
4
5 un superíndice^2^
6
7 un subíndice~2~
8
9 palabras ~~tachadas~~
10
11 `verbatim code`
12
13 > un blockquote
```

<https://go.uv.es/pjperez/intro.to.quarto>



# (III) Texto: Dando formato al texto

- Más posibilidades:

Si escribes esto:

```
[texto subrayado]{.underline}
```

Se verá esto:

texto subrayado

```
<mark>texto destacado</mark>
```

texto destacado

```
[En versalita]{.smallcaps}
```

EN VERSALITA

```
el emoji :joy:
```

el emoji 😂

```
> :warning: **Cuidado!!** no
```

⚠ Cuidado!! no

(\*\*) Los emojis mejor desde el editor visual o copiándolos, por ejemplo, desde [aquí](#)

# (III) Texto: Notas al pie

Se usa la marca <sup>^</sup>, y hay 2 sintaxis:

## 1. Con sintaxis inline:

```
1 Sintáxis (inline) para insertar una nota al pie^[Se llama inline
2 porque el contenido de la nota al pie se entremezcla con el texto principal]. Suele ser
3 la sintáxis que yo uso.
```

## 2. No-inline

```
1 Aquí voy a poner otra nota al pie [1] , y otra más.[b]
2
3 [1]: Esta otra sintaxis me gusta menos. Pero igual es más clara
4
5 [b]: No sufras por la numeración. lo hará Pandoc!!
```

- Documentación oficial [aquí](https://go.uv.es/pjperez/intro.to.quarto)

# (III) Texto: Títulos

Markdown:

Se verá esto:

---

```
# Header 1
```

# Header 1

---

```
## Header 2
```

## Header 2

---

```
### Header 3
```

### Header 3

---

```
#### Header 4
```

#### Header 4

---

```
##### Header 5
```

##### Header 5

---

```
##### Header 6
```

###### Header 6



# (III) Texto: Creando listas

- En las listas es **crucial la indentación** (normalmente 2 espacios). Documentación [aquí](#)

| Markdown Syntax  | Output  |
|--|---|
| <pre>* unordered list   + sub-item 1   + sub-item 2     - sub-sub-item 1</pre> | <ul style="list-style-type: none"><li>• unordered list<ul style="list-style-type: none"><li>◦ sub-item 1</li><li>◦ sub-item 2<ul style="list-style-type: none"><li>▪ sub-sub-item 1</li></ul></li></ul></li></ul> |
| <pre>*   item 2      Continued (indent 4 spaces)</pre>                         | <ul style="list-style-type: none"><li>• item 2<ul style="list-style-type: none"><li>Continued (indent 4 spaces)</li></ul></li></ul>   |
| <pre>1. ordered list 2. item 2    i) sub-item 1       A.  sub-sub-item 1</pre> | <ol style="list-style-type: none"><li>1. ordered list</li><li>2. item 2<ol style="list-style-type: none"><li>i. sub-item 1<ol style="list-style-type: none"><li>A. sub-sub-item 1</li></ol></li></ol></li></ol>   |

- Mas posibilidades [aquí](#) y [aquí](#)

# (III) Texto: Creando listas

- Lista ordenada

```
1 1. First item
2 2. Second item
3 3. Third item
4   3.1. Indented item
5   3.2. Indented item
6 4. Fourth item
```

- Lista no ordenada

```
1 - First item
2 - Second item
3 - Third item
4   - Indented item
5   - Indented item
6 - Fourth item
```

# (III) Texto: Párrafos

- Para crear un **nuevo párrafo** has de dejar una (o más) **lineas en blanco**
- Para **crear más espacio entre párrafos**; es decir, añadir una linea en blanco extra, podemos usar uno o varios: `<br>` o `<br><br>`

## Dentro de un párrafo

- Si **dentro de un párrafo** necesitas que **una frase empiece en otra linea** entonces, **has de dejar 2 (o más) espacios al final**; en inglés esto se conoce como un “hard line break”

### Extensión: más formas de crear “hard line breaks”

- El “hard line break” también se puede hacer con `<br>`
- El “hard line break” también se puede hacer con `"\"` seguido de nueva linea.
- En una tabla, `\` es la única forma de crear un salto de línea, ya que se ignoran los espacios finales en las celdas.

# (III) Texto: Lineas, espacios, ...

- Si en una linea de texto hay 2 espacios o más, estos se reducen a un espacio.

## Extensión: ¿cómo dejar más espacio entre palabras?

- Por lo tanto, si quieres separar palabras con más de un espacio, tendrás que usar lo que en html se conoce como “entities”: `&nbsp;` (1 espacio) `&ensp;` (2 espacios) y `&emsp;` (4 espacios).
- Otra forma sería con la etiqueta `<pre>`.

```
1 <pre>
2 Hola      amigos    !!!!
3 </pre>
```

## Extensión: un detalle que a veces es útil

- Los espacios al principio de las lineas que empiezan con `|` se mantienen (🧐)

# (IV) Más elementos

---

- Además de YAML, chunks y texto, **Markdown tiene más elementos para escribir**
- Por ejemplo: links, imágenes, vídeos, ecuaciones, tablas, bibliografía, etc ....

# (IV) Más elementos: hyper-links

Si escribes esto:

```
<https://quarto.org>
```

Al procesarse se verá esto:

<https://quarto.org>

```
[Web de Quarto] (https://quarto.org)
```

[Web de Quarto](https://quarto.org)

## Extensión: Cómo hacer que los links se abran en nueva página

Si quieres que la página enlazada **se abra en otra pestaña** del navegador puedes:

1. Especificarlo en el YAML con `link-external-newwindow: true`
2. Añadir `{target="_blank"}`, con lo que nuestros enlaces quedarían como:

```
1 [Web de Quarto] (https://quarto.org) {target="_blank"}
```

3. La versión 1.3 de Quarto permitirá esto en los enlaces en el fichero `_quarto.yml`. Puedes leerlo [aquí](https://go.uv.es/pjperez/intro.to.quarto)

# (IV) Más elementos: Imágenes

- Documentación oficial [aquí](#)

```
1  { fig-align="left" width="12%" }
```



```
1  { fig-align="right" width="20%" }
```



- Puedes ver como hacer composiciones de imágenes en el blog, concretamente [aquí](#), además de cuestiones como el tamaño y aspecto de las imágenes.



### Tarea 3.8: Markdown, incorporar una imagen de internet

- Incorpora [esta imagen](#) al archivo `prueba_01.qmd`
- La ruta a la imagen es:

```
1 https://upload.wikimedia.org/wikipedia/commons/thumb/f/f1/Nacho_Vegas_-_Rock_%26_Clown_-_A_Estrada.jpg/800px-Nacho_Vegas_-_Rock_%26_Clown_-_A_Estrada.jpg
```

### Tarea 3.8: solución

```
1 ![] (https://upload.wikimedia.org/wikipedia/commons/thumb/f/f1/Nacho_Vegas_-_Rock_%26_Clown_-_A_Estrada.jpg/800px-Nacho_Vegas_-_Rock_%26_Clown_-_A_Estrada.jpg)
```

### Tarea 3.8b: Markdown, incorporar una imagen local

- Guarda [esta otra imagen](#) dentro de `my_qproject`, concretamente dentro de la carpeta `imagenes`
- Incorpora esa imagen (guardada en tu ordenador) al archivo `prueba_01.qmd`

### Tarea 3.8b: Solución

```
1 ![] (./imagenes/my_imagen.jpg)
```

# (IV) Más elementos: Imágenes generadas con código

- Documentación para fijar el tamaño [aquí](#)

```
1  ```{r}
2  #| label: fig-penguins-01
3  #| fig-cap: Gráfico con datos de pingüinos
4  #| fig-width: 7
5  #| fig-asp: 0.618
6  #| output-location: column-fragment
7  #| code-line-numbers: "2|3|4-5|6|7"
8  library(tidyverse)
9  library(palmerpenguins)
10 pp <- ggplot(penguins, aes(
11     x = bill_length_mm, y = bill_depth_mm,
12     color = species, shape = species)) +
13     geom_point() +
14     theme_minimal() +
15     labs(x = "Bill length (mm)", y = "Bill depth (mm)")
16
17 pp
18 ```
```

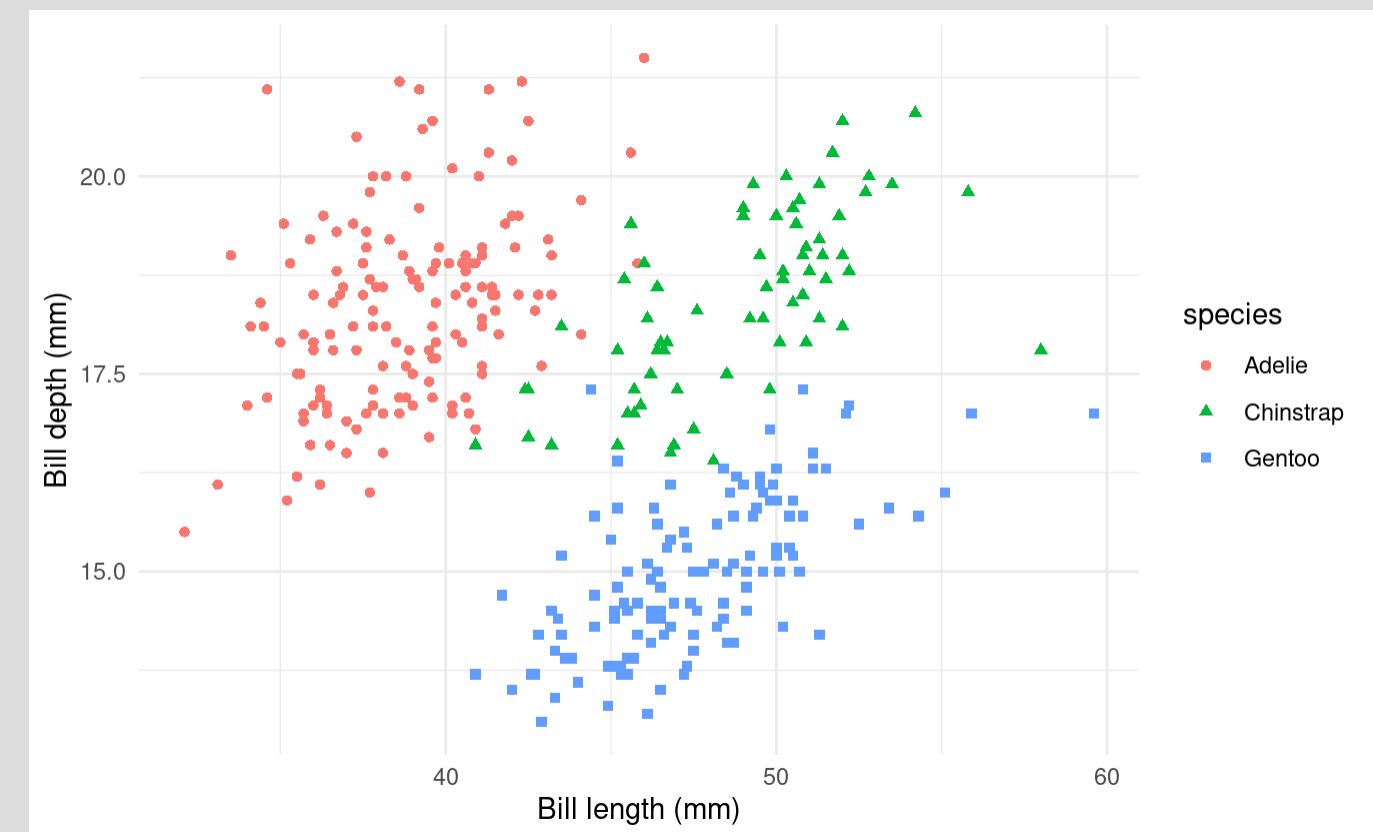


Figura 1: Gráfico con datos de pingüinos

# (IV) Más elementos: Tablas con Markdown

- Documentación oficial para tablas [aquí](#)

## Sintaxis Markdown para tablas

```
Right	Left	Default	Center
12	12	12	12
123	123	123	123
1	1	1	1

: Título de la tabla
```

Genera lo siguiente:

| Título de la tabla |      |       |        |
|--------------------|------|-------|--------|
| Default            | Left | Right | Center |
| 12                 | 12   | 12    | 12     |
| 123                | 123  | 123   | 123    |
| 1                  | 1    | 1     | 1      |

- Como ves no es fácil-fácil generar tablas con Mardown pero, **puedes usar un generador de tablas**, por ejemplo [este](#) o [este](#). También se puede usar el [editor visual de Rstudio](#)

## (IV) Más elementos: Tablas con R

- R tiene muchos paquetes para hacer tablas.
- Por ejemplo: `knitr::kable(df)`, o `gt::gt(df)`, o `DT::datatable(df)` ...
- Un ejemplo sencillo:

```
1 head(iris) |> gt::gt()
```

| 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  |

- Si te interesa el tema de las tablas puedes visitar [esto](#) o [esto](#) o [esto](#)

## (IV) Más elementos: Tablas **con R** desde ficheros Excel

- Podemos importar los datos para las tablas de un fichero Excel:

```
1 df <- rio::import("./datos/matriculados.xlsx")  
2 DT::datatable(df)
```

# (III) Más elementos: Ecuaciones

- Markdown admite ecuaciones escritas en Latex:
  - si van entre un **\$** serán ecuaciones inline
  - **\$\$** para ecuaciones independientes.

## Markdown Syntax

ecuación inline: `$E = mc^{2}$`

ecuación independiente:

`$$E = mc^{2}$$`

## Output

ecuación inline:  $E = mc^2$

ecuación independiente:

$$E = mc^2$$

- Tienes editores online de ecuaciones en Latex [aquí](#) y [aquí](#). Puedes ver algunos ejemplos de ecuaciones escritas en Latex [aquí](#)
- Además, podemos reusar el código Latex de ecuaciones en la web (si se ha usado MathJax). Por ejemplo [aquí](#)

# RStudio Visual editor

---

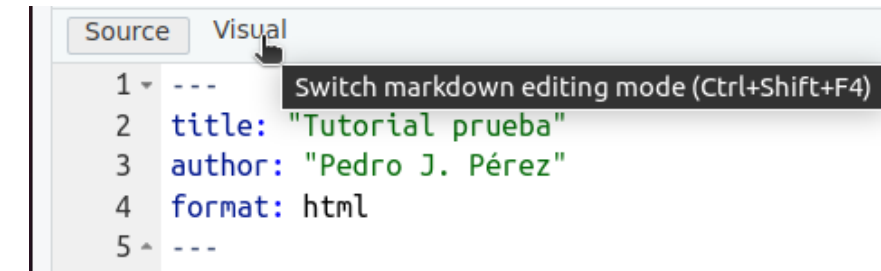
- ¿Me acordaré de todas la marcas que hemos visto?

, ya te digo yo que No



# Editor Visual en RStudio

- Desde 2020, concretamente desde la [versión v.1.4](#), RStudio dispone de un editor visual
- El editor visual **facilita la inserción** de los **elementos** que hemos ido viendo durante el curso. Documentación oficial [aquí](#) y [aquí](#)
- Hasta ahora hemos trabajado en nuestros QMD en formato “Source”, pero **podemos cambiar el modo de edición a “Visual”**, solo hay que seleccionar la pestaña adecuada:



# Editor Visual: Modo de edición

Edición en modo Visual:

En modo Source:

---

SourceVisualB I </> Header 2 Format Insert Table

```
---
title: "Tutorial prueba"
author: "Pedro J. Pérez"
format: html
---
```

## Quarto

Quarto enables you to weave together content and executable code into a finished document. To learn more about Quarto see <https://quarto.org>.

## Running Code

When you click the **Render** button a document will be generated that includes both content and the output of embedded code. You can embed code like this:

```
{r}
1 + 1
```

The `echo: false` option disables the printing of code (only output is displayed).

FilesPlotsPackagesHelpTutorialViewer

← → ✖ Edit Sync Editor

# Tutorial prueba

AUTHOR  
Pedro J. Pérez

## Quarto

Quarto enables you to weave together content and executable code into a finished document. To learn more about Quarto see <https://quarto.org>.

## Running Code

When you click the **Render** button a document will be generated that includes both content and the output of embedded code.

1 + 1

[1] 2

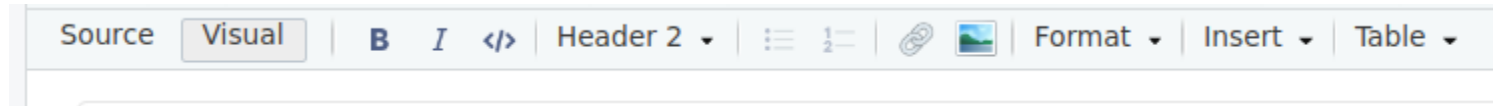
The `echo: false` option disables the printing of code (only output is displayed).

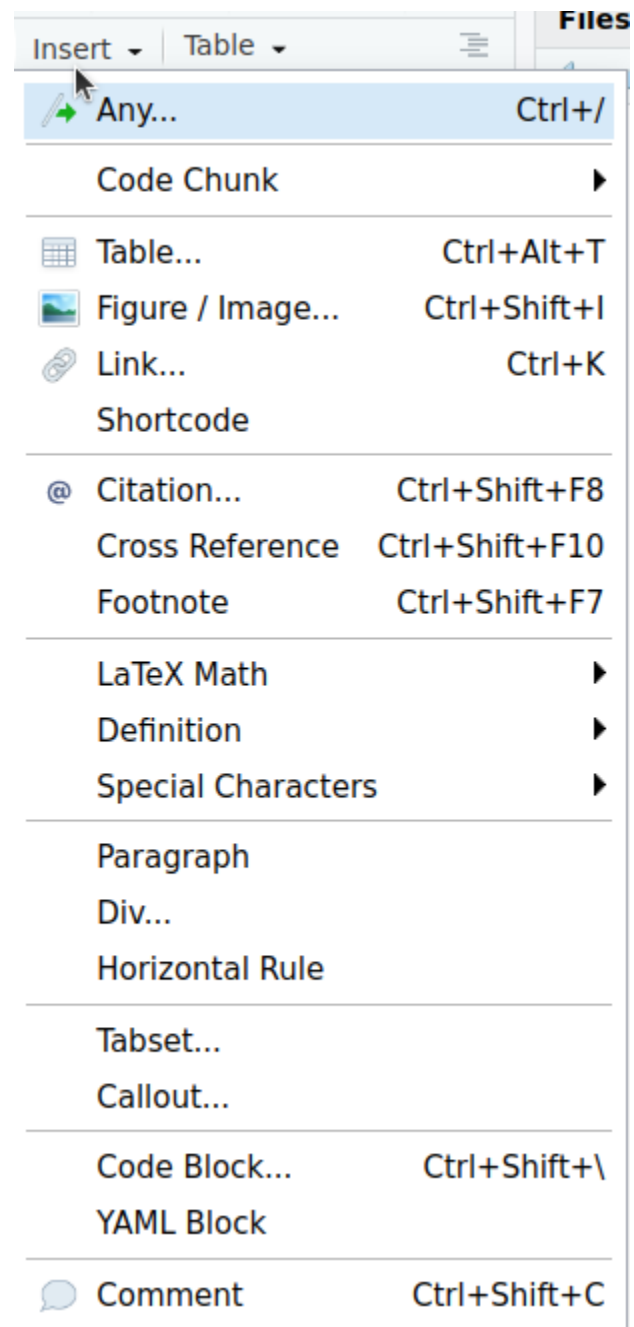
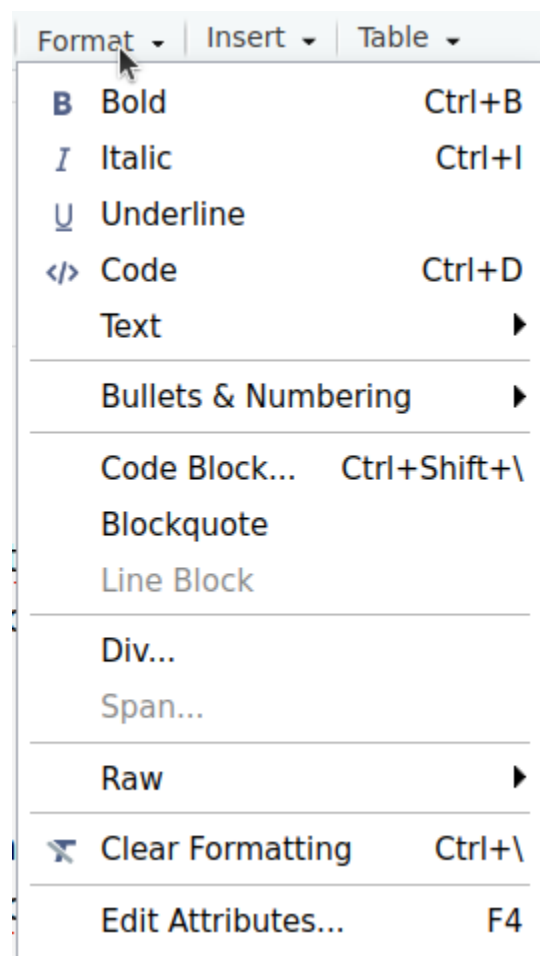
<https://go.uv.es/pjpperez/intro.to.quarto>

64

# Editor Visual: Barra de herramientas

La barra de herramientas del editor visual facilita el acceso a la mayoría de elementos que podemos insertar en nuestros documentos QMD:





### Tarea 3.9: Practicando con el editor visual

- Incorpora una imagen a nuestro `.qmd`
- Incorpora una tabla

# Aún quedan más elementos!!

---

- Pandoc fenced divs, Pandoc bracketed spans
- Layout, CSS, ....
- ... pero lo dejamos para más adelante

