

# Más cosas con Quarto

(Web del curso [aquí](#))

26 de junio de 2023

# Con Quarto se pueden hacer muu uuchas más cosas!!!

---

Veamos algunas de ellas

# Subir la web a la UV

- Una vez hemos generado nuestra web en local con Quarto nos queda **subirla a la web**
- Hay muchos servicios de alojamiento, [por ejemplo](#), pero nosotros tenemos un **espacio en la web de la UV**.
- Solo **tenemos que subir los ficheros** de la carpeta [docs](#) o [\\_site](#) a la web de la UV
- Pero antes, **hay que tener activado vuestro espacio web en UV**. Veamos cómo hacerlo con [este post](#)

# Gestión de bibliografía

- Imagina que en uno de nuestros post queremos **incorporar referencias bibliográficas** y además **referenciarlas** en el cuerpo del artículo ¿te suena, no?

Además,

- **no queremos tener que copiar a mano las referencias y**
- **tampoco queremos tener que cambiar el formato 🤖🤖 de las referencias si es que decidimos cambiar de revista**

Entonces, ¿cómo lo hacemos? os lo cuento en [este post](#)

# Diagramas

- Quarto permite hacer **diagramas** con [Mermaid](#) y [Graphviz](#). Lo que, según la [documentación oficial](#), posibilita la creación de flowcharts, sequence diagrams, state diagrams, Gantt charts, and more **usando una sintaxis** (de texto) **similar a Markdown**.
- Un ejemplo:

## El código

```
1  ```{mermaid}
2  %%| eval: false
3  flowchart LR
4      qmd --> J([Jupyter])
5      qmd --> K([knitr])
6      J --> md
7      K --> md
8      md --> P([pandoc])
9      P --> pdf
10     P --> html
11     P --> docx
12  ```
```

## El resultado:



## • Otro ejemplo

```
1 ```{mermaid}
2 gantt
3 dateFormat YYYY-MM-DD
4 title Adding GANTT diagram to mermaid
5 excludes weekdays 2014-01-10
6
7 section A section
8 Completed task      :done,      des1, 2014-01-06,2014-01-08
9 Active task         :active,     des2, 2014-01-09, 3d
10 Future task        :            des3, after des2, 5d
11 Future task2       :            des4, after des3, 5d
12 ```
```

```
gantt
dateFormat YYYY-MM-DD
title Adding GANTT diagram to mermaid
excludes weekdays 2014-01-10
```

```
section A section
Completed task      :done,      des1, 2014-01-06,2014-01-08
Active task         :active,     des2, 2014-01-09, 3d
Future task        :            des3, after des2, 5d
Future task2       :            des4, after des3, 5d
```

# Documentos con múltiples formatos

- Los documentos `.html` que generemos , ya sean estos documentos individuales o dentro de una web, pueden incluir links al mismo documento en **otros formatos**. Documentación [aquí](#).
- Para ello hay que incluirlo en el YAML, por ejemplo:

```
1 title: Mi documento
2 author: Pedro J. Pérez
3 toc: true
4 format:
5   html: default
6   epub: default
```

- **Un detalle:** El render del documento se ha de hacer desde la Terminal:

```
1 quarto render mi-multi-format-document.qmd
```

# Ejercicios interactivos

- Hay varios paquetes como [learnr](#) que permiten incorporar **ejercicios interactivos** a documentos [.html](#) generados con Quarto, pero ...
  - ... tienen un **inconveniente**: los tutoriales deben ser ejecutados localmente o alojados en un servidor shiny
- El paquete [webexercises](#) **soluciona ese problema** ya que genera “standalone HTML files that require only a JavaScript-enabled browser”.
- El objetivo de **webexercises** es:

To enable instructors to easily **create interactive web pages** that students can use in **self-guided learning**



# Tablas estadísticas

- Ya vimos [aquí](#) que se podían hacer tablas muy chulas con R, pero muchas veces hay que **presentar los resultados** de contrastes o de la **estimación de modelos estadísticos**.
- Vamos a profundizar en ello con [este post](#) del blog

# Journal articles

- El curso se ha centrado en la realización de documentos `.html` pero desde Quarto se pueden generar documentos en [múltiples formatos](#), incluido `pdf``s.
- Además, disponemos de **plantillas** para generar, directamente desde Quarto, documentos en el formato adecuado para un conjunto de Journals. Puedes ver el **listado de Journals** [aquí](#) y [aquí](#)
- En **Rmarkdown** tenemos el paquete [rticles](#) con plantillas para generar documentos de un conjunto amplio de [Journals](#)

## Extensión: más sobre publicación en Journals desde Quarto

- Un [hilo de Mastodon](#) con las vicisitudes de M. Mahoney para publicar en Elsevier desde Quarto:
- [Journals](#) para publicar código.
- [Esta charla](#) de Mine Çetinkaya-Rundel habla de este tema en el minuto 14 aprox.

# Incorporar tablas desde Google Sheets (o desde Excel)

- A veces veces tenemos unos datos, o un listado de algo: tareas, estudiantes, calendario etc... que queremos incorporar a nuestros documentos, ¿Cómo lo hacemos?
- Si tenemos los datos en el fichero `datos/matriculados.xlsx` solo habría que:

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

- Si los datos los tenemos en Google Sheets

```
1 # googlesheets4::gs4_deauth() #- si tuvieses q autorizar
2 my_url <- "https://docs.google.com/spreadsheets/d/16XpV9I0_hoyPqV6o7Pkir4c_mmwgHM4UuGXuM7t7vUc/edit?usp=sharing"
3 df <- googlesheets4::read_sheet(my_url)
4 gt::gt(df)
```

# Con Quarto AÚN se pueden hacer muu uuchas más cosas!!

---

Veamos algunas de ellas

# Countdown

- Incluir un **reloj para hacer cuentas atrás** cuando pones un ejercicio a resolver en clase.  
Documentación [aquí](#)

- Un ejemplo

```
1 ```{r}
2 #| eval: false
3 #| # devtools::install_github("gadenbuie/countdown")
4 countdown::countdown(minutes = 0, seconds = 42)
5 ```
```

# Crear un glosario de términos

- El paquete `glossary` permite crear un glosario de términos al final de un documento `.html` y referenciar términos del glosario en el texto.
- ¿Cómo? No he llegado a probarlo pero básicamente tienes que
  - crear el glosario de términos en un documento `.yaml`,
  - proporcionar su ruta a la función `glossary_path()`
- Para añadir un término al glosario:

```
1 glossary_add(term = "Ishi",  
2              def = "Ishi fue el nombre dado al último miembro de la tribu de los yahi de California, Estados Unidos.")
```

- Para referenciar palabras del glosario en el texto, se hace algo como:

Se considera a `glossary::glossary("Ishi")` el último nativo de la California septentrional que vivió fuera de la influencia de la cultura occidental. Su historiase popularizó a raíz de un libro de Theodora Kroeber, madre de Ursula K. Le Guin.

- Además, puedes definir el estilo del glosario con `glossary_style()`. Por ejemplo:

```
1 glossary::glossary_path("psyteachr.yml")  
2 glossary::glossary_style("purple", "underline")
```

# Crear botones de descarga

- Se trata de crear unos botoncitos (chulos!!) en nuestra web que sirvan para descargarse los documentos (o recursos) que queramos
- Vamos a probar a hacer estos botones con la extensión [downloadthis](#).
- Para ello vamos a usar la documentación de la extensión y [este post](#) del blog del curso.



# Extensiones

- En el curso se habla de las extensiones. En las slides nº5, concretamente [aquí](#) y en [este post](#) y en [este otro](#)
- [Aquí](#) un post que cuenta como hacer extensiones
- Por ejemplo podemos probar a usar la extensión [fontawesome](#).

Algunas extensiones chulas:

- [code-fullscreen](#): permite que al pinchar en un **bloque de código**, este pase a estar a **pantalla completa**
- [nustshell](#): permite crear “**expandable explanations**”. Lo mejor es pinchar y ver los ejemplos
- [collapse-social-embeds](#): permite crear 6 tipos de **nuevos callouts** para social contents (Github gist, Twitter tweets, Mastodon toots, y vídeos de Loom, Vimeo y Youtube). Puedes ver un ejemplo [aquí](#). Seguramente acabaré usándolo en el curso!!
- [downloadthis](#): permite crear botones de descargar muy chulos. Ya la hemos usado
- [pointer](#) y [quarto-spotlight](#): 2 extensiones para **iluminar/agrandar el puntero del ratón** en revealjs slides
- [line-highlight](#): similar a [code-line-numbers](#) pero ahora es para documentos html, no para RevealJs slides
- [webR](#): permite to run R code in the browser without the need for an R server to execute the code!!
- [Sverto](#): permite incorporar Svelte components that can seamlessly react to your ObservableJS code!!!
- [quarto-animate](#): permite crear unas animaciones muy “aparatosas”. Puedes ver un ejemplo <https://go.wes/pjperex/intro.to.quarto>

# Curriculums con Quarto

Desde Quarto se pueden hacer curriculums. Por ejemplo:

- Plantilla para crear CV con Quarto: [quarto-cv](#)
- Otra plantilla para hacer CV's, de [@BeaMilz](#): el [código](#) y el [resultado](#)
- Tengo el CV en un [.docx](#) ¿puedo pasarlo a Quarto? Sí, [aquí](#) la explicación. Aconsejan pasarlo primero a Google doc's

## CV's con [.Rmd](#)

- Un currículum con [pagedown::html\\_resume](#).
- Curriculum con [datadrivencv](#) package

# Posters con Quarto

- Yo aún no he visto que se pueda, pero [la gente esta esperando](#) poder hacer posters con Quarto
- Con `.Rmd` habían varios paquetes para hacer posters, por ejemplo con [posterdown](#) o con [pagedown::poster\\_relaxed](#).

# Más formatos con **. Rmd**

- Business cards con [pagedown::business\\_card](#)
- “Cuadros de mando” con [flexdashboard](#): tienes algunos ejemplos [aquí](#). Permite [distintos layouts](#), incluidos los [storyboards](#), por ejemplo [este](#)
- Shiny apps con el paquete [shiny](#): algunos [ejemplos](#)
- Formatos posibles gracias al paquete [rmdformats](#)

# Trucos CSS y SASS

- En el improbable caso de que tengamos tiempo, veremos **algunos trucos para tunear** algunos aspectos de nuestros documentos y web/blog
- Lo haríamos con [este post](#) del blog

# AÚN más cosas!!!

---

Ejemplos que se me han quedado en el tintero (quizás para otra edición)

# Informes parametrizados

- un ejemplo con notas y quizás mandarlo por mail
- o con datos de inflación de varios países: <https://datageeek.com/2023/03/16/food-inflation-interactive-chart-with-ggiraph/>



# Cosas de “Rendering”

When you render a Quarto document, first **knitr** executes all of the code chunks and creates a new markdown (.md) document which includes the code and its output. The markdown file generated is then processed by **pandoc**, which creates the finished format.

- Rendering con opciones [aquí](#)

```
1 quarto render document.qmd --to pdf
2 quarto render document.qmd --to html -M code-fold:true
3 quarto render document.qmd --to html -M code-fold:true -P alpha:0.2 -P ratio:0.3
```

- Para crear **epub** en Quarto bastaría con hacer en la Terminal: **quarto render --to epub**; pero mejor leer los consejos de [Bruno Rodrigues](#) y de [Sam Parmar](#)

- Para hacer `quarto::quarto_render()` y que el archivo de salida se cree en un **directorio** que no sea el directorio raíz del proyecto, **parece que no se puede**; así que de momento hay que hacer un workaround con `file.copy(full_path_source, full_path_destination)` para copiar los archivos de salida a otra carpeta.
- Para convertir un **.docx** a `.md`, hay que ejecutar en la terminal:

```
1 pandoc -f docx -t markdown foo.docx -o foo.markdown
```

# Runing Stata in Quarto documents

Tampoco lo he probado, entre otras cosas porque no tengo Stata instalado, pero :

- Podemos correr código de Stata en Quarto document: [aquí](#) se explica cómo usando un paquete de Phyton, [pystata](#).
- [Stata Facade](#): una extensión de Quarto that hides the evidence of faking Stata dynamic content with Python code blocks and Stata cell magic.
- También se puede con R: [Statamarkdown](#) y [RStata](#)

# Runing SPSS in Quarto documents

No he visto que sea posible correr SPSS code dentro de R, pero para que tenemos [R GUI's](#). Otra [comparación de GUI's](#). Un post sobre [Jamovi](#)

- [r2spss](#) y su [vignette](#)
- [Rmimic](#)
- [expss](#)

# Shiny en Quarto

- <https://appsilon.com/interactive-quarto-report-translation-tutorial/#final>

# Queries to bibliographic databases

- **openalexR**: “helps you interface with the **OpenAlex** API to retrieve bibliographic information about publications, authors, venues, institutions and **concepts**”
- **rscielo**: “offers functions to easily scrape bibliometric information from scientific journals and articles hosted on the **Scileo Platform**”
- **JCRImpactFactor**: “JCRImpactFactor: Journal Citation Reports (‘JCR’) Impact Factor by Clarivate Analytics”
- **Scrapping WOS**
- **Bibliometrix**: “bibliometrix package provides a set of tools for quantitative research in bibliometrics and scientometrics”
- **RISmed**: A set of tools to extract bibliographic content from the National Center for Biotechnology Information (NCBI) databases, including PubMed.

# Trabajando con documentos WORD

- [officedown](#): facilita el formateo de informe de Word desde R. [Aquí](#) un libro y dos posts: [aquí](#), otro [aquí](#)
- [officer](#): The officer package lets R users manipulate Word (.docx) and PowerPoint (\*.pptx) documents. In short, one can add images, tables and text into documents from R.
- [docxtractr](#): Extract Data Tables and Comments from Microsoft Word Documents. [Aquí](#) un post sobre su uso.

# Plantillas/cosas de pdf's

- Plantillas para hacer pretty pdfs: [aquí](#) y [aquí](#)
- Plantilla para [compact pdf](#)
- Nicola Rennie te dice cómo hacer pdfs parametrizados: [aquí](#)
- Quarto YouTube [Playlist](#)
- Title Pages [templates](#) for adding a cover page to your Quarto pdf books. Un [ejemplo](#)
- Great looking pdf with pagedown: vídeo [aquí](#)
  
- Muchísimas más en [Quarto awesome](#)

## Otras plantillas



# Ejemplitos (de copiar y hacer)

- [Quarto sketchy html](#)
- [Slides chulas](#) pero sencillas de Mine. El repo [aquí](#)
- Un ejemplo de slides chulas: [Intro to Quarto](#), el repo [aquí](#).
- Un póster con [posterdown](#) o con [pagedown::poster\\_relaxed](#).
- Business cards con [pagedown::business\\_card](#)
- “Cuadros de mando” con [flexdashboard](#): tienes algunos ejemplos [aquí](#). Permite [distintos layouts](#), incluidos los [storyboards](#), por ejemplo [este](#)
- Shiny apps con el paquete [shiny](#): algunos [ejemplos](#)
- Formatos posibles gracias al paquete [rmdformats](#)

```
1 pak::pak('rstudio/pagedown')
```

