Investigación reproducible con Quarto y R

Felipe Ortega María Jesús Algar Emilio López Cano

2024-10-24

Tabla de contenidos

Preface

Este taller explica cómo utilizar Quarto, un software de creación de documentación científica, para crear publicaciones de calidad que integren contenido de texto formateado, gráficos, tablas, así como resultados de ejecución de código software en varios lenguajes, todo ello integrado en el propio documento.

Quarto se ha convertido en una herramienta muy versátil y potente en el conjunto de herramientas de los programadores científicos, en especial por proporcionar soporte para implementar buenas prácticas de **investigación reproducible**, incluyendo los principios FAIR. El intenso movimiento iniciado desde hace años por la comunidad científica para garantizar acceso en abierto no solo al producto final (e.g. una publicación) sino también a materiales adicionales (códig fuente, conjuntos de datos, figuras, procesos de trabajo, archivos de configuración, etc.) se ha convertido en un objetivo insoslayable para académicos y especialistas en muchos campos diferentes. Muchas publicaciones científicas de prestigio exigen ahora a los autores enviar estos materiales auxiliares junto con los borradores de sus manuscritos, para permitir que otros colegas reproduzcan y validen los resultados, repliquen sus estudios sobre nuevas cohortes de individuos o elementos o para contribuir a la interpretación de los resultados obtenidos.

Con la herramienta Quarto se puede combinar texto formateado (escrito en Markdown) junto con secciones de código ejecutable, todo integrado en un mismo documento. Estas secciones o *chunks* de código ejecutable pueden estar escritas en varios lenguajes: R, Python, Julia u Observable. Es incluso posible combinar en un mismo documento o colección de documentos secciones de código escritas en diferentes lenguajes de programación.

Este es un **taller práctico** que presenta ejemplos reales y comandos para crear paso a paso tus propios documentos con Quarto en poco tiempo. Además, junto a la explicación de los conceptos clave para entender este proceso también se ofrecen recomendaciones sobre buenas prácticas de trabajo, para guiar a los aprendices de Quarto en la dirección correcta.

Puedes aprender muchos más detalles sobre cuarto en la guía en línea https://quarto.org/docs/guide/. En particular, en https://quarto.org/docs/books se documenta en detalle cómo crear libros como este utilizando Quarto.

Parte I

Quarto

1 Documentos científicos

En su actividad diaria, los estudiantes, académicos y especialistas científicos producen gran cantidad de documentación de todo tipo: notas de laboratorio, apuntes, memorandos, informes técnicos y, sobre todo, artículos científicos para publicar sus descubrimientos y avances en un área de conocimiento. Normalmente, la creación de este tipo de documentos científicos conlleva una gran cantidad de tareas que involucran diferentes herramientas y posibles puntos de fallos.

La Figura 1.1 muestra una descripción general esquemática de un proceso de trabajo clásico para la creación de documentos científicos. Con frecuencia, el elemento principal es un archivo maestro de procesador de textos (Word, OpenOffice/LibreOffice, etc.), una página web o un fichero LaTeX (si vamos a crear un documento PDF) que recoge todo el contenido.

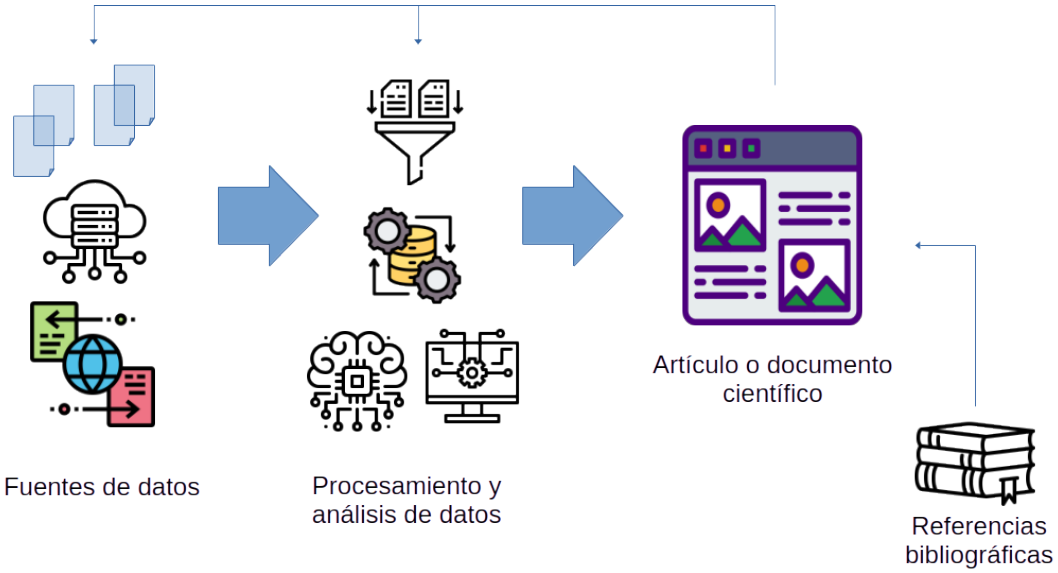


Figura 1.1: Proceso de creación de documentación y artículos científicos

Este archivo maestro se va llenando de contenido que procede de diversas fuentes, como por ejemplo:

- figuras y esquemas generados manualmente o mediante código software (como gráficos de visualización de datos);

- tablas y resúmenes que describen conjuntos de datos y resultados;
- resultados y evaluación del rendimiento de modelos o algoritmos; estadísticos o de aprendizaje automático;
- fórmulas y ecuaciones matemáticas;
- tablas de datos y otra información de utilidad;
- referencias bibliográficas (normalmente generadas con ayuda de algún programa de gestión de información bibliográfica).

Muchos de estos elementos fuerzan a los usuarios a ejecutar una y otra vez herramientas y programas externos, procedimientos y otras tareas para incorporar luego los nuevos resultados al fichero maestro. Debemos admitir que este proceso, en su mayor parte manual, además de tedioso puede ser muy propenso a que cometamos errores o descuidos. "¡Espera! He olvidado actualizar la Figura 1". "¿Seguro que estos son los últimos resultados de evaluación del modelo M?" "¿Has comprobado que hemos cargado la última versión del fichero de datos D?" Estas son preguntas comunes que surgen en el día a día de los equipos científicos.

Sin embargo, sería genial si no fuese necesario realizar todo ese proceso manual y, en ocasiones, muy frustrante de forma manual. ¿Tenemos alguna alternativa para evitarlo? Sí, la tenemos. La respuesta a nuestras necesidades nos la brinda un concepto muy poderoso: la **programación literaria**.

1.1 Programación literaria

El concepto de programación literaria fue acuñado por el profesor Donald E. Knuth (1984). Sí, no has leído mal, hace más de 40 años. Este concepto establece que debería ser posible integrar, en un solo documento científico, texto formateado y resultados de la ejecución de código software para componer dicho documento de forma dinámica. Entonces, ¿por qué hemos tardado tanto en poner en práctica esta idea? La visión de Knuth, aunque muy adelantada a su tiempo, era correcta, pero la tecnología de la época no permitía ponerla en práctica.

Sin embargo, hoy día contamos con todos los elementos indispensables para llevar esta idea a la práctica. Es más, contamos con una herramienta, Quarto, que nos va a permitir automatizar y gestionar todo el proceso de creación de documentos de programación literaria de forma rápida y fiable.

1.2 Investigación reproducible

Durante muchas décadas, el método científico se ha basado en la publicación de investigaciones que describen el resultado de análisis de datos y experimentos. En todos los casos, resulta fundamental poder confiar en las condiciones, los datos recabados, el método de análisis y

de ejecución de los experimentos así como en las herramientas de diversas clase, incluído el software, que los autores de la publicación han empleado para llevarla a cabo.

Sin embargo, los numerosos avances experimentados en los últimos años en las herramientas y métodos de análisis permiten que ahora sea mucho más sencillo comprobar el resultado de estos análisis. Podríamos suponer que esto facilita mucho el trabajo de científicos y científicas, pero en realidad sucede todo lo contrario. Veamos algunos ejemplos:

- **Oncología** (Begley & Ellis, 2012): El Departamento de Biotecnología de la firma Amgen (Thousand Oaks, CA, EE.UU.) sólo pudo confirmar 6 de un total de 53 artículos de investigación emblemáticos publicados en este área. Por su parte, Bayer HealthCare (Alemania) tan sólo pudo validar un 25% de los estudios analizados.
- **Psicología** (Wicherts et al., 2006): El 73% de los autores de un total de 249 artículos publicados por la APA no respondieron en un periodo de 6 meses a las preguntas y requerimientos formulados acerca de los datos que emplearon en sus investigaciones.
- **Economía y finanzas** (Burman et al., 2010): La comparación de diferentes paquetes software aplicados en la ejecución de varios análisis de modelos financieros y estadísticos refleja que cada uno de dichos paquetes produce resultados *muy distintos* empleando *las mismas técnicas* estadísticas directamente aplicadas sobre *datos idénticos* a los empleados en la publicación original.

De hecho, han llegado a aparecer artículos que sugieren que buena parte de los resultados publicados en áreas como Medicina podrían no ser del todo fiables (Ioannidis, 2005). Como resultado de todos estos hallazgos recientes, se ha generado en toda la comunidad científica e investigadora una gran polémica, acompañada de una profunda crisis de confianza.

A pesar de todo, como muy bien recoge una conocida tira cómica sobre el mundo académico y la investigación (ver Figura 1.2), el proceso de desarrollo de las publicaciones científicas se basa primordialmente en la revisión continua de los métodos y resultados (empezando por los propios estudiantes y sus supervisores).

La Figura 1.3 muestra un gráfico publicado en la prestigiosa revista Science Magazine [CITATION], que representa los datos sobre evolución del número de artículos de investigación retractados o retirados por diveresas causas, entre 1997 y 2014. En este gráfico, podemos constatar cómo la mejora de las herramientas y la mayor disponibilidad de recursos permite analizar y revisar un mayor volumen de publicaciones y análisis, lo que permite detectar un mayor número de casos problemáticos.

1.2.1 Reproducibilidad y replicabilidad

Se habla con frecuencia de *reproducir* y *replicar* un análisis de datos o un experimento científico (Leek & Peng, 2015). Sin embargo, se pueden citar numerosas evidencias que demuestran que existen definiciones incompatibles sobre estos dos términos y otros relacionados con ellos

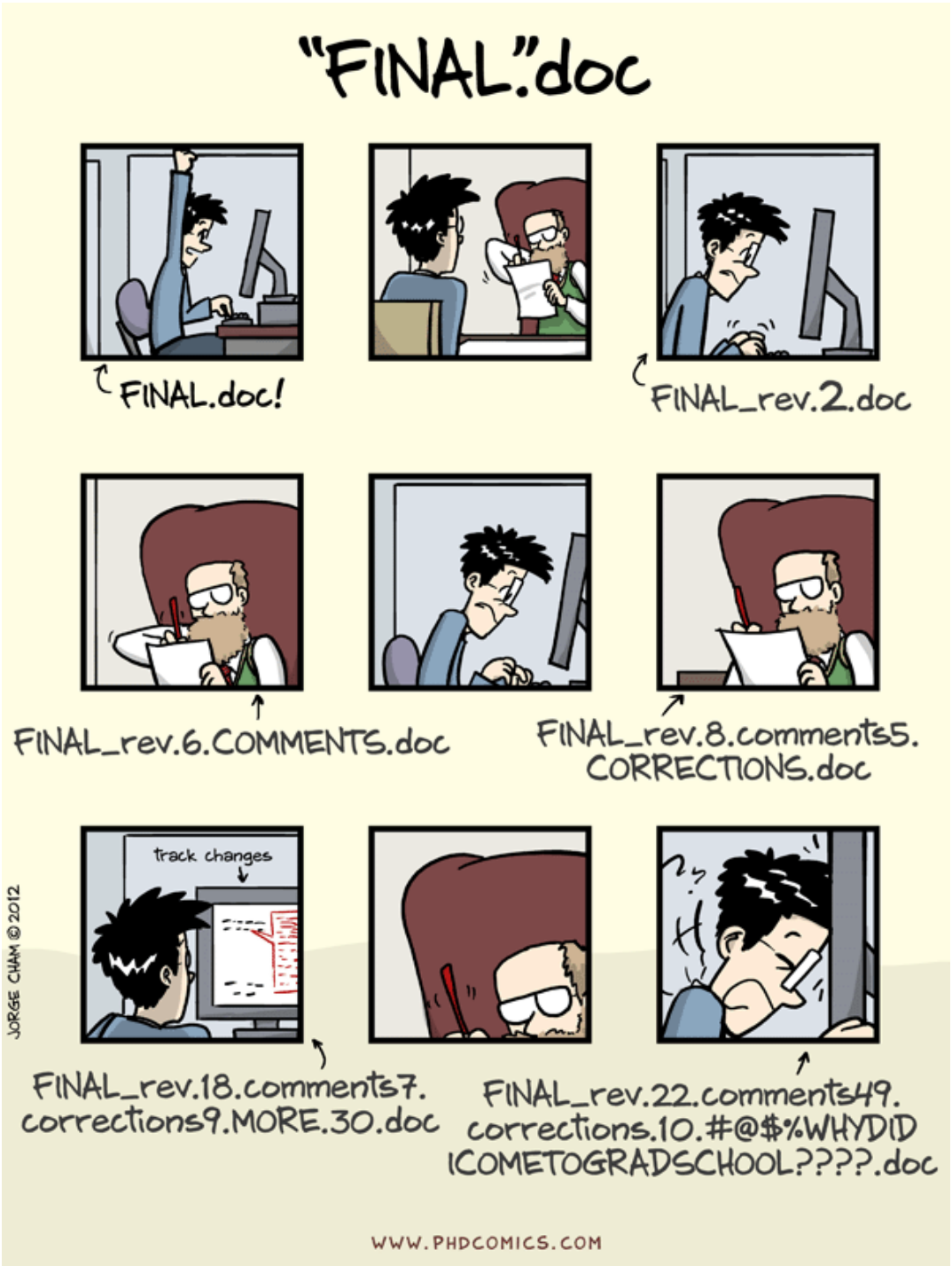


Figura 1.2: Tira cómica que representa el modelo de revisión de publicaciones científicas. Fuente: PhD comics.

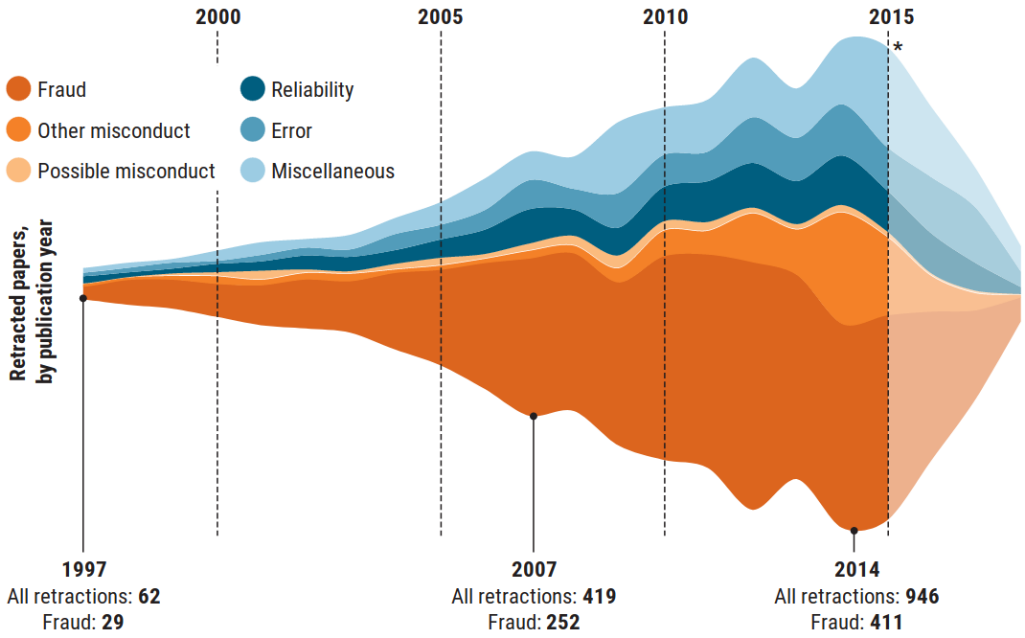


Figura 1.3: Evolución del número de publicaciones científicas retractadas o retiradas por diversas causas, entre 1997 y 2014. Fuente: Science Magazine (Brainard et al., 2018).

(Barba, 2018). Mucho cuidado, por tanto, porque dependiendo de la comunidad científica o el campo de conocimineto en que nos encontremos, el significado de estos dos terminos puede llegar a ser incluso *enteramente opuesto* a su definición aceptada en otras áreas [1]. Aquí vamos a ceñirnos a la definición aceptada en un gran número de áreas, incluyendo estadística o computación científica (véase Barba, 2018, p. 33):

- **Reproducibilidad**: Se define como la capacidad para recomputar los resultados de un análisis, con los mismos datos que se emplearon en el análisis original, y conociendo los detalles de la secuencia (*workflow* o *pipeline*) de operaciones de componen dicho análisis. Se debe poder garantizar ciertas premisas:
  - Si usamos las mismas herramientas (e.g. R, un cierto listado de paquetes, las mismas versiones de todos los paquetes y dependencias), así como el mismo código (*scripts* de R) sobre los mismos datos, los resultados y conclusiones han de ser consistentes con los del análisis original.
  - Los autores del análisis original deben proporcionar todos los elementos (datos, código y procedimiento empleado) para permitir que el análisis sea reproducible (Barba, 2018).
- **Replicabilidad**: Se define como la capacidad para realizar un experimento o análisis independiente del original, que aborde el mismo objetivo pero sobre un conjunto de datos diferente del empleado en el estudio inicial. En caso de que los resultados no sean consistentes, será necesario realizar más réplicas y armonizar los resultados y conclusiones por medio de técnicas adecuadas, como por ejemplo el **meta-análisis**.

1.2.2 Niveles de replicación

En función de los elementos publicados por los autores del estudio original, así como del grado de detalle con el que se describe el proceso para llevar a cabo el estudio, los pasos que se han seguido y las herramientas empleadas, tenemos diferentes niveles de replicabilidad o reproducibilidad, representados en la Figura 1.4.

- *No reproducible*: No se proporcionan datos, código ni ninguna descripción concreta sobre la implementación del estudio o análisis. Muchas publicaciones científicas ya no aceptan publicar artículos en estas condiciones.
- *Código* o *Datos*: Un buen número de editoriales solicitan que los conjuntos de datos empleados en el análisis o estudio de la publicación sean accesibles mediante una URL, bien porque estén disponibles en un repositorio público o bien porque los autores del

[1] Entre los ejemplos más importanes de definiciones que contradicen las que damos en este taller, están las adoptadas por la Federation of American Societies for Experimental Biology (FASEB), en inmunología y microbiología, así como las adoptadas por la Association for Computer Machinery (ACM) en ciencias de la computación.

artículo lo han publicado. Así mismo, muchas publicaciones exigen que el código software para llevar a cabo el análisis también esté accesible públicamente, en un repositorio de código abierto o bien en un proyecto de un servicio de control de versiones con acceso libre.

- *Código y datos*: Lo ideal es que tanto el código como los datos estén públicamente accesibles a disposición de quien los quiera examinar o bien utilizar para reproducir los resultados (validación) o replicar el análisis con otros datos u otros casos.
- *Entorno de ejecución y datos enlazados*: Un paso más para facilitar la reproducibilidad de los estudios consiste en publicar archivos de código y metadatos con información más precisa sobre el lenguaje de programación, los paquetes software empleados y cualquier otra dependencia necesaria para llevar a cabo el mímo estudio o análisis. Otra variante para facilitar la reproducibilidad es la de encapsular el código y las dependencias en un contenedor virtual ya preconfigurado, que se pueda descargar y ejecutar directamente.
- *Gold standard*: El nivel más avanzado consistiría en documentar todos los procedimientos realizados durante el estudio o análisis, incluyendo la codificación de las tareas de obtención, limpieza y preparación de los datos, así como la generación de gráficas de visualización de resultados o cualquier otro resultado derivado del estudio.

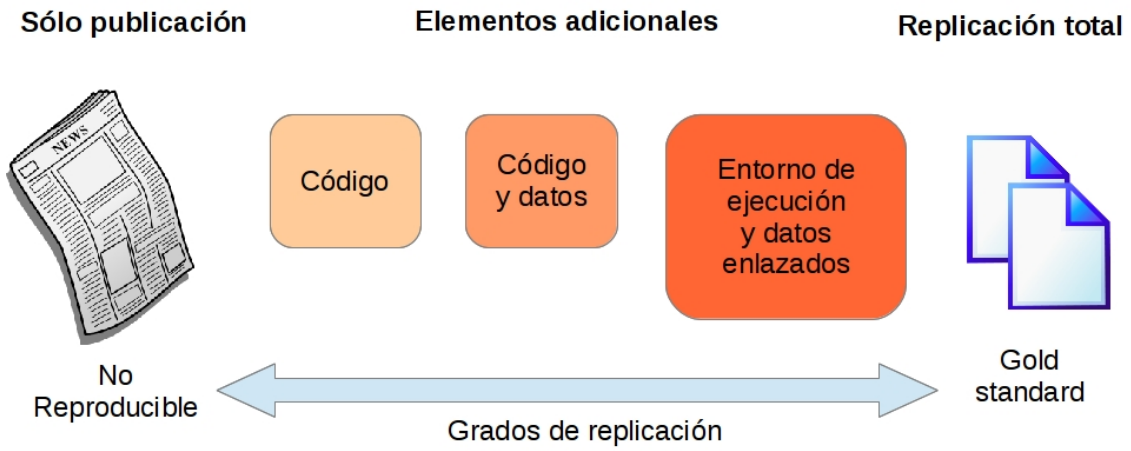


Figura 1.4: Espectro de los diferentes niveles de replicación en publicaciones científicas. Fuente: Peng (2011).

1.2.3 Herramientas para la replicabilidad

Ciertas tecnologías y herramientas que se han refinado y sofisticado durante los últimos años están permitiendo facilitar la replicabilidad del procesamiento y análisis de datos.

- Sistemas de Control de Versiones para código software (SCV): herramientas como Git, Mercurial y servicios web como GitHub o GitLab han popularizado la creación y publicación de proyectos que permiten gestionar el código software que se ha creado, controlando los cambios y las versiones liberadas. Los servicios web integran, además, un buen número de herramientas para dar soporte a diferentes facetas del proceso de desarrollo de software, tales como generación de documentación, manuales y ejemplos, informes de error y peticiones de mejoras, integración continua y despliegue continuo (CI/CD), testeo sistemático del código generado, etc. Si todavía no te has planteado en qué puede beneficiarte utilizar una herramienta de control de versiones de código fuente, echa un vistazo a la Figura 1.5 en la que revivirás una situación lamentablemente muy habitual entre los investigadores y científicos que desarrollan soluciones software.

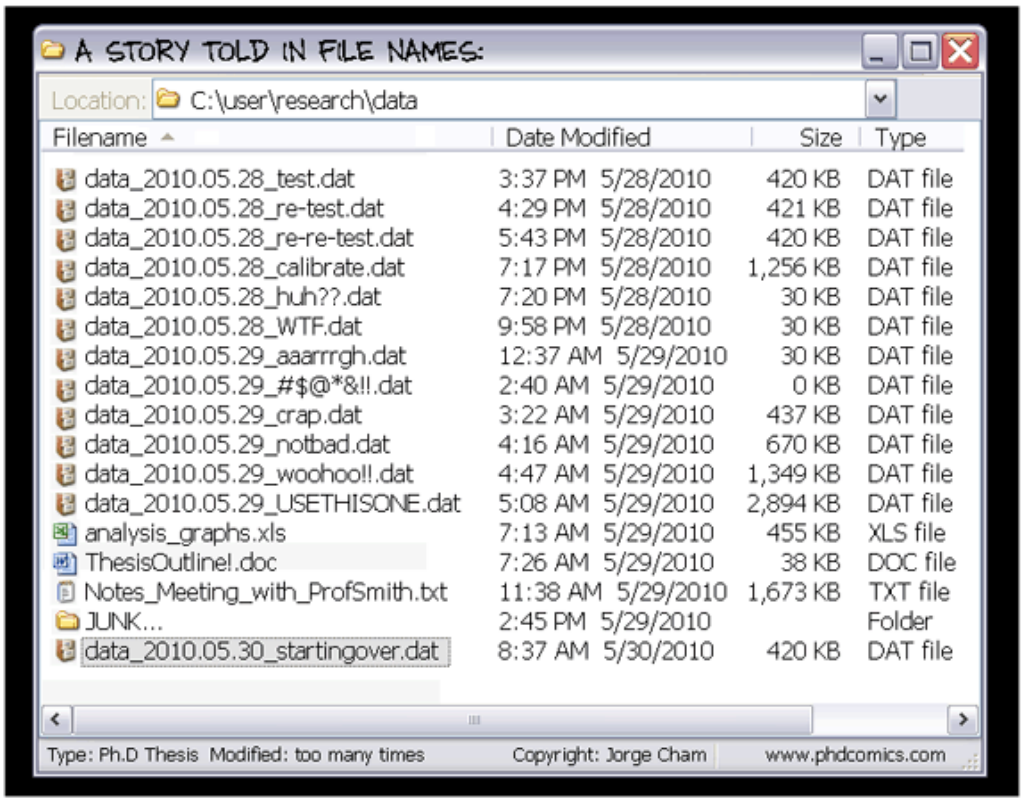


Figura 1.5: Control de versiones de software. Fuente: PhD Comics

- Virtualización de software y contenedores: en un entorno tecnológico dominado por la contratación y despliegue de infraestructura de computación y servicios en arquitecturas en la nube (*cloud computing*), las herramientas de empaquetado y virtualización de aplicaciones y servicios software que pueden instalarse y desplegarse en poco tiempo han

revolucionado la forma en la que se publican y gestionan los productos software, incluidos los de procesamiento y análisis de datos.

- Control de versiones de datos: De forma análoga a los SCV para código fuente, está apareciendo software para aplicar los mismos principios a los ficheros de datos. De esta forma, podemos controlar diferentes versiones de cada archivo de datos, modificaciones efectuadas en los mismos, etc. Una de estas herramientas es Data Version Control (DVC), que permite versionado de datos y modelos. Como resultado, podemos saber en todo momento qué versión de los datos y qué listado de *features* se han incluido en cada modelo considerado durante el análisis, manteniendo integrada la información descriptiva sobre estos tres componentes esenciales que siempre deben ir cohesionados.

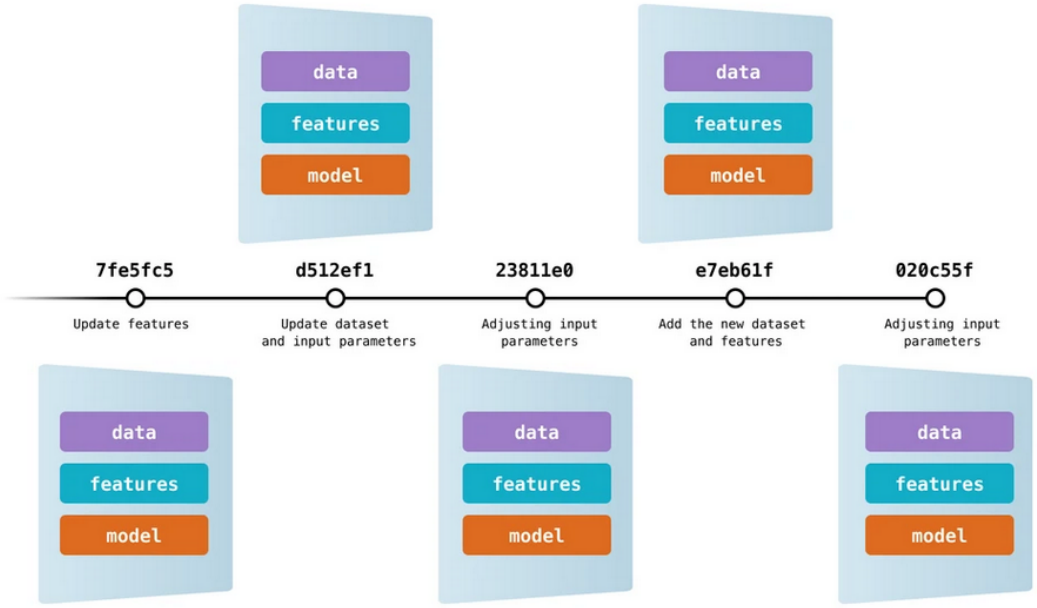


Figura 1.6: Ejemplo del versionado de datos, código y modelos que DVC es capaz de mantener. Fuente: Documentación DVC.

- Gestión de modelos y experimentos: Otro tipo de herramienta de gestión de proyectos de aprendizaje automático es la que permite la organización, seguimiento, comparativa y selección de los experimentos y modelos que hemos realizado. Uno de los ejemplos destacados más recientes es ML Flow, que proporciona soporte para ajuste, evaluación y optimización de modelos, despliegue de los mismos en entornos en producción, creación de un registro de modelos pre-entrenados, etc. Por supuesto, es posible combinar este tipo de herramientas con otras como DVC, creando como resultado un entorno integral de gestión de nuestros proyectos.
- Creación y gestión de *pipelines* de procesamiento de datos: el último elemento indispensa-

ble en todo proyecto de procesamiento y análisis de datos que deba cuidar la escalabilidad es una herramienta para creación y gestión de flujos de procesamiento y análisis o *pipelines* de datos. El conjunto de todos los *pipelines* de nuestro proyecto componen el *workflow* general del mismo. A estas herramientas se las conoce como *orquestadores* de datos o de flujos de trabajo. En esta categoría, contamos tanto con herramientas muy potentes y llenas de características como Apache Airflow o Prefect como con otras más sencillas y directas como Luigi.

Por supuesto, la comunidad de R no se ha mantenido ajena a estas nuevas tendencias, muy en particular la iniciativa R OpenSci, dentro de la cual encontramos muchos paquetes (publicados en el repositorio oficial CRAN) que cubren diversos aspectos del trabajo científico, incluyendo la gestión de *pipelines* y *workflows* mediante el paquete `targets`.

- Manual de uso del paquete R `targets`.

1.3 Quarto para publicaciones científicas

Ahora que ya conocemos el concepto fundamental sobre el que se asienta el funcionamiento de Quarto y su aplicación para conseguir un mayor nivel de reproducibilidad y transparencia en nuestro proceso científico, vamos a explicar con más detalle el proceso que sigue Quarto para componer un documento. La Figura 1.7 presenta un esquema con el proceso de creación del documento y los elementos y herramientas que entran en juego para conseguirlo.

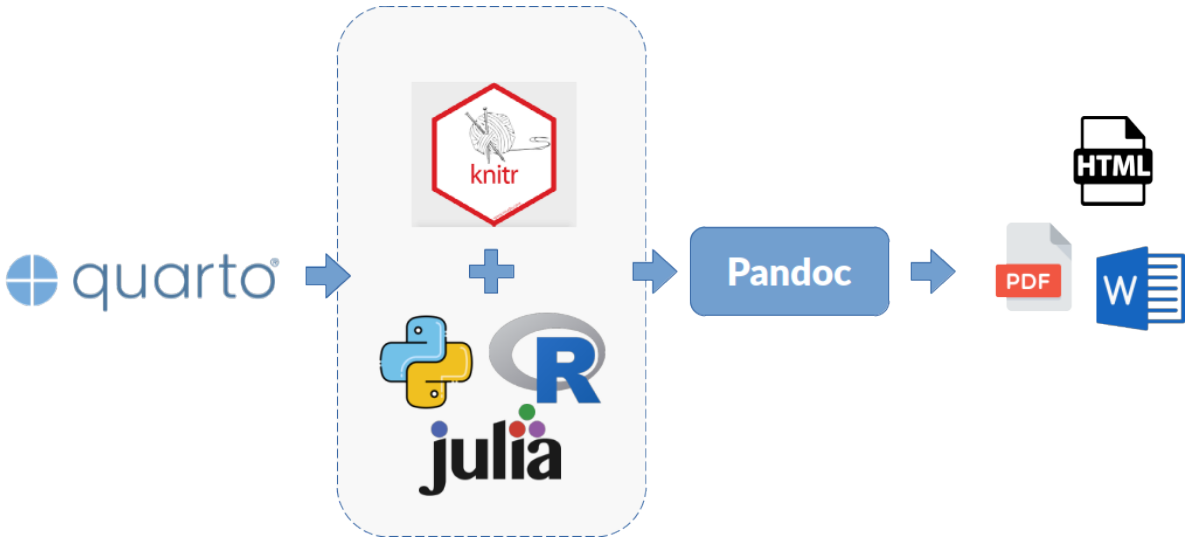


Figura 1.7: Proceso de creación de contenidos con Quarto.

- **Quarto**: un software que permite crear documentación científica siguiendo los principios de la programación literaria.

- **Kintr y lenguaje de programación**: el paquete `knitr` se encarga de la conexión con un intérprete de un lenguaje de programación (R, Python, Julia) que se pueda ejecutar en un entorno REPL, para poder ejecutar fragmentos de código software integrados en el documento y generar como resultado contenido en formato Markdown.
- **Markdown** (contenido formateado): lenguaje de marcado de contenido textual que permite formatear de forma sencilla la información de nuestros documentos creados con Quarto.
- **Pandoc** (traductor universal de formatos documentales): este software recibe el contenido ya formateado usando el estándar Markdown, para convertirlo en el tipo de salida seleccionado. Existen varias opciones disponibles: HTML, PDF o Word, así como diapositivas, websites o paneles interactivos (*dashboards*).

1.4 Instalación de Quarto

Para instalar la última versión del software Quarto en nuestro sistema, dirigimos nuestro navegador web a la página https://quarto.org/docs/get-started/. Aquí, descargamos e instalamos el archivo correspondiente a nuestro sistema operativo.

En este momento, la última versión de Quarto disponible es la `1.5.57`.

> **Requisitos para generar documentos en PDF**
>
> Por defecto, el formato de salida de los documentos generados con Quarto es HTML. Si queremos generar documentos en PDF, necesitamos tener instalada una distribución LaTeX. Para más información, consulta la Sección 3.3.

2 Tipos de documentos

En este capítulo, presentamos los principales tipos de documentos y colecciones de contenidos científicos que podemos generar con Quarto.

2.1 Documentos individuales

La forma más sencilla de trabajar con Quarto es crear un documento individual. Dicho documento podrá utilizar las secciones o *chunks* de código para leer datos de entrada o descargarlos de alguna fuente, procesarlos, analizarlos y mostrar los resultados. Se pueden añadir gráficos, tablas, ecuaciones, referencias bibliográficas y muchos otros elementos.

Los documentos tienen siempre una estructura estándar:

- *Preámbulo*: en el que se especifican opciones de configuración para la creación del documento con Quarto y sus herramientas asociadas.
- *Cuerpo*: la sección que alberga el contenido principal del documento, incluyendo secciones de texto formateado en Markdown y secciones de código ejecutable. El código software se podrá mostrar, si resulta de utilidad, o quedar oculto en el resultado final.
- *Referencias*: Al final del documento se incluyen las referencias bibliográficas, como es habitual en los textos científicos.

2.2 Libros

La evolución natural del caso anterior es reunir una colección de documentos individuales en un solo libro. *Quarto books* permite crear este tipo de documentos, estructurados en partes, capítulos y secciones. Las opciones de configuración permitirán confeccionar una portada de introducción para el sito web que contiene los capítulos (un documento por capítulo) o bien los elementos necesarios para crear un libro en PDF, semejante a los publicados por una editorial.

2.3 Artículos y publicaciones

Uno de los resultados clave en todo proceso científico es la producción de artículos y publicaciones (informes técnicos, etc.) que recojan los resultados y avances científicos conseguidos. En este caso, Quarto también nos podrá ayudar, con la colaboración de otros elementos indispensables como el paquete R `rticles`, que proporciona plantillas para generar artículos según las especificaciones de las principales publicaciones y editoriales científicas en multitud de campos de conocimiento.

2.4 Presentaciones

También es posible generar presentaciones (normalmente, en formato HTML) con diapositivas mediante Quarto. En este caso, tendríamos el soporte de varios paquetes y entornos de creación de presentaciones web a nuestra disposición, como `reveal.js` (HTML), Beamer (para LaTeX/PDF) o formato PPTX de MS Office.

Este caso no lo trataremos en este taller, pero se puede obtener más información en la guía online, disponible en https://quarto.org/docs/presentations/.

2.5 Sitios web

Otra opción que puede resultar interesante es crear sitios web personales (por ejemplo, para mostrar nuestro CV y una selección de trabajos destacados, publicaciones, etc.), blogs e incluso sitios web corporativos (organización, grupo de investigación) de forma rápida mediante Quarto. Existen numerosas plantillas gratuitas y de pago ya disponibles para crear sitios web con un aspecto armonizado, aunque necesitaremos aprender un poco de HTML y CSS para poder personalizar aún más nuestra web.

Aquí tenemos un ejemplo de sitio web de un investigador en tecnología medioambiental creado con Quarto: https://www.mm218.dev/. Más ejemplos de diferentes tipos de sitios web generados con Quarto: https://drganghe.github.io/quarto-academic-site-examples.html.

Se puede conseguir más información y tutoriales para crear sitios web con Quarto en https://quarto.org/docs/websites/.

2.6 *Dashboards*

Por último, es posible crear cuadros de mandos o *dashboards* personalizados para monitorización de datos, análisis de modelos y resultados o bien para ejemplos y aplicaciones docentes utilizando Quarto, tal y como se describen en la guía https://quarto.org/docs/dashboards/.

En este caso podemos incluir entre las herramientas Shiny, un paquete software para R (también disponible para Python) con el que crear aplicaciones interactivas basadas en datos de forma rápida y sencilla.

3 Proceso de trabajo

En este apartado vamos a explicar algunos detalles más sobre el proceso de creación de los documentos en Quarto, para comprender mejor los componentes que intervienen en este proceso y las opciones de configuración que tenemos disponibles. La Figura 3.1 resume a alto nivel las fases de creación de un documento con Quarto.

Figura 3.1: Proceso de creación de contenidos con Quarto. Fuente: RStudio.

3.1 Cadena de montaje de un documento

Podemos considerar el proceso de creación de un documento en Quarto como una cadena de montaje en la que varias herramientas software se aplican secuencialmente para producir el documento final en diferentes formatos de salida.

- Quarto: El primer agente que interviene en la interpretación del fichero con extensión `.qmd` es Quarto. El programa debe estar instalado previamente en nuestro equipo para que el entorno de desarrollo que decidamos emplear (RStudio, Microsoft VS Code, etc.) pueda encontrarlo. Quarto se encarga de interpretar el contenido del archivo y considerar las diferentes opciones de configuració que hemos insertado en la cabecera del documento, así como en los fragmentos de código ejecutable, para delegar las tareas de construcción de las distintas partes del documento a otras herramientas.

  Quarto también se encarga de insertar de forma automática algunos elementos de autoría de documentos (como las llamadas de atención, explicadas en la Sección 4.5.4), mientras que la generación de otros elementos (numeración de figuras, tablas, citas bibliográficas, etc.) se delega a otras herramientas como Pandoc.

- Motor de ejecución de código (`engine`): Una de las grandes ventajas de la producción de documentos en Quarto es la posibilidad de insertar fragmentos de código ejecutable en nuestros documentos. Quarto puede trabajar con diferentes motores (*engines* en la terminología de Quarto) que interpretan el código y devuelven el resultado de su ejecución para integrarlo en el documento final.
  - Todos los bloques de código en lenguaje R utilizan el paquete `knitr` como *engine* de ejecución.
  - Los bloques de código ejecutable en otros lenguajes soportados en Quarto (Python, Julia, Observable) utilizan los *kernel* disponibles en la herramienta Jupyter para su ejecución, excepto en el caso explícito de que se combinen bloques de código en R y Python en el mismo documento.
  - En el caso particular de que un mismo documento combine bloques de código en R y Python, entonces se utiliza la herramienta Knitr para el código en R junto con el paquete `reticulate` de R para ejecutar los bloques escritos en Python. Esta combinación tiene la ventaja adicional de que es posible "trasvasar" variables y resultados entre los entornos de ejecución de R y Python, de forma que podamos usar variables y estructuras de datos creados en la parte de R dentro de nuestro código Python y viceversa.
- Markdown: El producto intermedio de todas las fases anteriores es un archivo en formato Markdown (extensión `.md`), que ya integra el contenido textual formateado y muchos de los resultados y elementos adicionales generados por las distintas herramientas que han entrado en juego hasta este punto. En caso de que fuese útil, este archivo intermedio se puede almacenar para ser revisado o para utilizarlo para otros fines. A continuación, este archivo se envía a Pandoc para la última etapa de creación del documento.
- Pandoc: El proyecto Pandoc ofrece una herramienta sofwtware para traducir documentos entre diferentes formatos de representación de información. Como se puede ver rápidamente en la página principal del proyecto, la lista de formatos soportados es realmente extensa. A efectos prácticos, en Quarto se utiliza la capacidad de Pandoc para recibir como entrada un documento en formato Markdown y generar una salida en tres posibles formatos: HTML, DOC/DOCX o PDF. El archivo Markdown debe incluir una sintaxis específica para codificar determinados elementos (referencias cruzadas, citas bibliográficas, figuras, tablas, ecuaciones, etc.) que iremos presentando en los siguientes capítulos, para que Pandoc pueda interpretar estos elementos y representarlos de forma apropiada en cada formato de salida.

Por último, cabe destacar que se necesitan programas de visualización de documentos para cargar los documentos de salida, según el formato: navegador web (HTML), MS Word (archivos DOC/DOCX), visor PDF (archivos PDF).

3.2 Producir HTML

La opción por defecto para el formato de salida de los documentos Quarto es generar un documento HTML, que se puede visualizar con la mayoría de navegadores web modernos. Este formato de salida tiene varias ventajas:

- Es bastante probable que el receptor del documento ya disponga de uno o varios navegadores instalados en su sistema para ver el documento, si se lo enviamos o compartimos directamente.
- Es más sencillo publicar este tipo de documentos en la Web, utilizando alguna de las distintas plataformas disponibles para este fin:
  - RPubs, un servicio de publicación de documentos de la empresa Posit PBC, integrado con RStudio.
  - Los documentos individuales (véase Capítulo 4) y los libros o colecciones de documentos (véase Capítulo 5) se pueden publicar de forma sencilla y rápida en sitios de hospedaje de proyectos software como GitHub o GitLab, que además proporcionan servicios de control de versiones, gestión de informes de error/mejoras, documentación, testing, etc.

3.3 Producir PDF

Al contrario que en HTML, cuando generamos documentos en PDF se añade un paso adicional de compilación del documento a final de toda la cadena de montaje, utilizando LaTeX y el motor de compilación XeLaTeX para generar la salida PDF. Por tanto, si seleccionamos esta opción de salida es **imprescidible** tener **instalada una distribución TeX/LaTeX** previamente en nuestro sistema, para compilar y generar los documentos. Si no disponemos de ninguna todavía, se puede instalar TinyTeX, una distribución ligera de TeX Live que tiene mucho menor tamaño (~100 MB frente a los más de 4 GB de TeX Live completa).

3.3.1 Personalizar documentos PDF

Se pueden utilizar plantillas de documentos LaTeX predefinidas. Por defecto, Quarto utiliza varias plantillas de la colección de paquetes LaTeX koma-script.

Algunas de estas plantillas pueden funcionar de forma relativamente sencilla en Quarto, mientras que otras requieren cierta adaptación, para lo cual se necesitarán algunos conocimientos sobre LaTeX. Este es probablemente un tema más avanzado para muchos usuarios, por lo que por ahora no lo vamos a tratar en este taller introductorio.

No obstante, a modo de ejemplo, ofrecemos a continuación un listado de algunos ejemplos que ilustran las enormes posibilidades de este tipo de plantillas:

- El profesor R.J. Hyndman ha publicado plantillas de documentos Quarto de la Monash University, que pueden utilizarse como punto de partida para personalizarlas en nuestros propios proyectos.
- El repositorio Awesome Quarto Thesis recoge un listado de plantillas Quarto para generar memorias de TFG/TFM y tesis doctorales para algunas universidades. También se enlaza una plantilla genérica de extensión para Quarto, concebida para facilitar que otros usuarios puedan personalizarla según los criterios marcados por su propia institución para generar estos trabajos.

4 Documentos individuales

La manera más sencilla de comenzar a utilizar Quarto es crear documentos individuales. Se trata de documentos autocontenidos, que incorporan texto formateado y código ejecutable en un único archivo.

Para crear un documento nuevo con Quarto, simplemente podemos usar las opciones del menu de RStudio o MS Visual Code, o bien crear un archivo con extensión `.qmd`.

4.1 Creación del documento con RStudio

Antes de empezar, comprueba que has instalado el software Quarto en tu máquina. Es un programa software independiente, que tiene que estar instalado para que el resto del proceso funcione (consulta la sección Sección 1.4).

Si ya tenemos instalada una versión reciente de RStudio, necesitaremos instalar los siguientes paquetes para el ejemplo:

```
install.packages("tidyverse")
install.packages("palmerpenguins")
install.packages("quarto")
```

Ahora, en RStudio creamos un nuevo proyecto eligiendo la opción *Quarto project*, tal y como aparece en la Figura 4.1.

Podemos nombrar el directorio de nuestro proyecto como `primer-ejemplo` y pulsamos `Create Project`.

Como resultado, nos debe aparecer un nuevo proyecto abierto en pantalla, con el aspecto que se muestra en la Figura 4.2.

En concreto, en el panel superior izquierdo podemos comprobar que, por defecto se ha abierto el editor *Visual*, que permite crear documentos Quarto de forma más intuitiva. Sin embargo, para empezar a familiarizarnos desde el principio con la estructura de un documento en quarto vamos a cambiar al editor *Source* para ver el código fuente, pulsando en el botón que se muestra en la figura Figura 4.3.

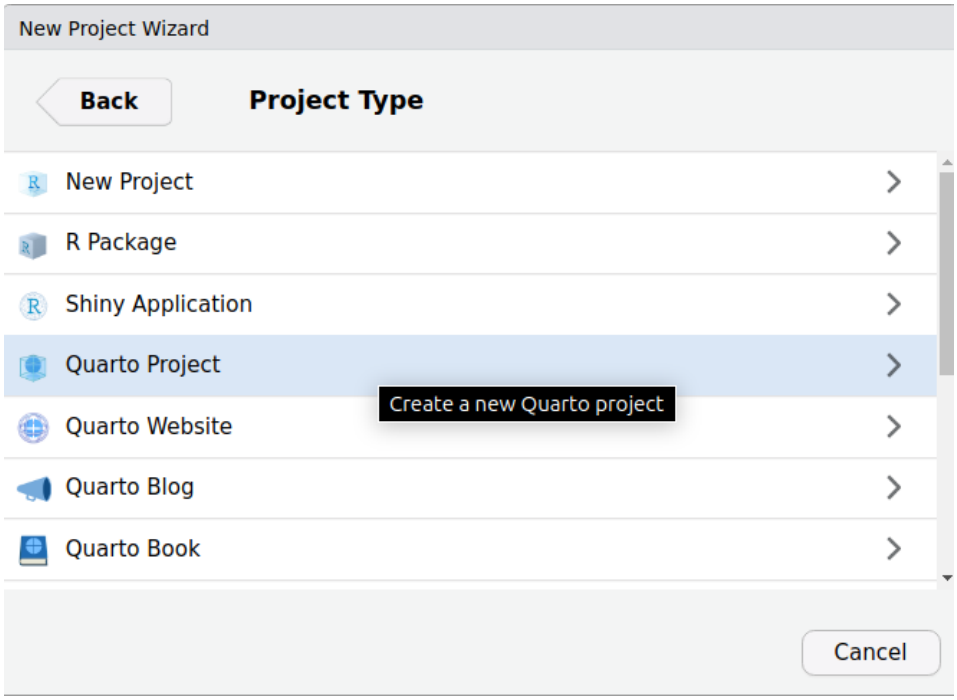


Figura 4.1: New Quarto project

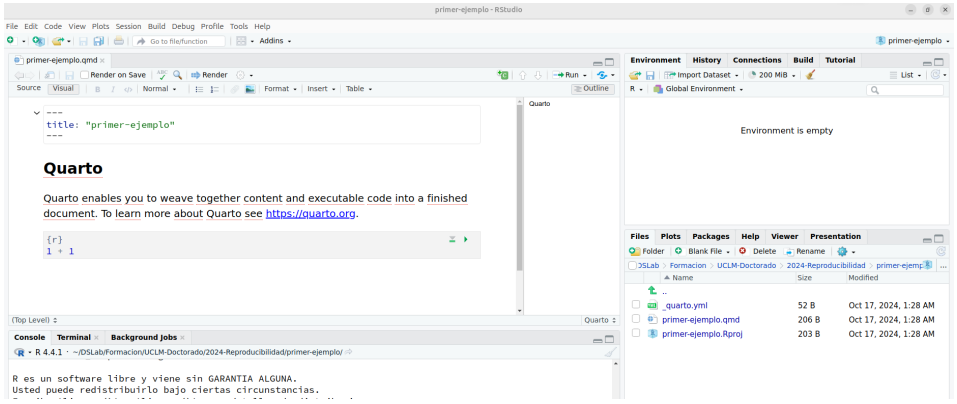


Figura 4.2: First example

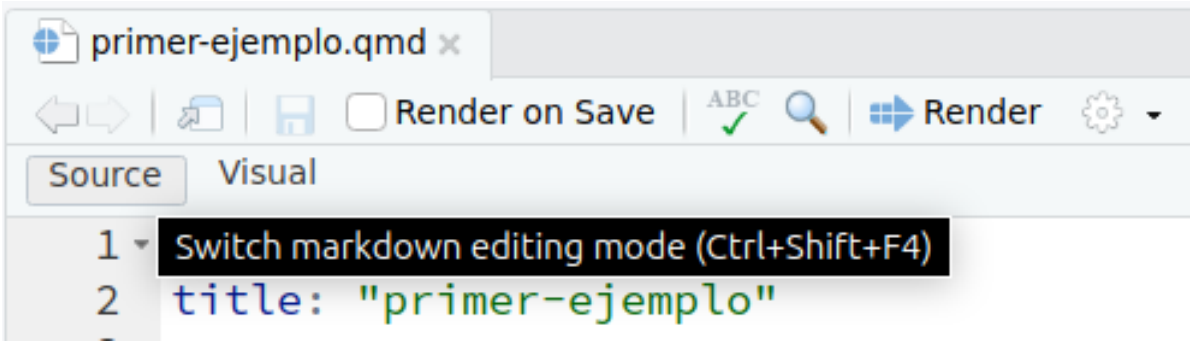


Figura 4.3: Source editor

4.2 Estructura del documento

La estructura de un documento individual en Quarto es esta.

```
---
title: "Mi primer documento"
author: John Doe
date: 2024-10-24
---

Aquí tenemos algo de contenido en formato **Markdown**.

```{r}
#| label: Etiqueta

1 + 1
```

Contenido *adicional* en Markdown.
```

El contenido del archivo consta de dos partes:

- **Preámbulo**: está delimitado por dos marcas `---`. Dentro de esta área podemos asignar valores a opciones de configuración para maquetar y crear el documento, tales como el título, autor/es, fecha, etc. También podemos configurar diversas opciones relacionadas con el formato de salida de los documentos.
- **Cuerpo del documento**: se compone de párrafos de texto formateados utilizando la sintaxis de marcado Markdown, que veremos después. Además, también se pueden intercalar en el texto fragmentos de código ejecutable o *chunks*, que se marcan siguiendo una sintaxis especial (como vemos en el ejemplo anterior).

Cada *chunk* de código ejecutable está delimitado de la siguiente manera

```
```{r}
# Código en R
```
```

> Soporte para otros lenguajes de programación
>
> Aunque en este taller nos centramos en el lenguaje R, debemos sabes que Quarto también soporta otros lenguajes de programación como Python, Julia u Observable.
> Podemos cambiar el lenguaje de programación de cada *chunk* indicando su nombre al comienzo, como por ejemplo:
>
> ```
> ```{pyton}
> # Código en Python
> ```
> ```
>
> Sin embargo, para que pueda funcionar necesitaremos realizar algunas tareas adicionales de configuración.

4.2.1 El preámbulo

4.2.2 Listado de opciones

Existe un extenso listado de opciones de configuración que podemos incluir en nuestros documentos.

- *Opciones para salida HTML*: permiten configurar diversos aspectos básicos del documento, tales como el título y subtítulo, fecha, autor (o lista de autores), resumen o DOI; opciones de formato como el tema o estilos avanzados para contenido HTML con CSS; numeración y tabla de contenidos, etc.
  - Opciones básicas para HTML con Quarto.
  - Lista completa de opciones HTML con Quarto.
- *Opciones para salida PDF*: ofrecen la posibilidad de configurar múltiples parámetros para la creación del documento en este formato, muchas de ellas similares a las de la salida en HTML. Una opción particularmente relevante es elegir el formato de documento LaTeX (opción `documentclass`), que define el aspecto general de la maquetación que se va a emplear. Por defecto, se emplean clases del metapaquete KOMA Script, como `scrartcl` o `scrbook`. También es importante indicar la opción `papersize`, en nuestro caso para garantizar que se usa un formato estándar como el A4. El formato de las

citas también es relevante, pudiendo elegir, por ejemplo, el motor BibLaTeX que es más potente, con soporte multilenguaje y para codificación de caracteres UTF-8 nativa. Por último, también es importante indicar el motor de compilación. Si queremos una flexibilidad total en la maquetación del documento, se recomienda encarecidamente usar el motor XeLaTeX (opción `pdf-engine: xelatex`), que es el *valor por defecto* que utiliza Quarto.

  - Opciones básicas para PDF con Quarto.
  - Lista completa de opciones disponibles en PDF con Quarto.

4.2.3 Sintaxis Markdown básica

En el siguiente enlace puedes encontrar un rápido tutorial básico que muestra las opciones básicas de la sintaxis Markdown aceptada en documentos Quarto para formatear el contenido textual.

- Guía básica de sintaxis markdown.

4.3 Creación del documento (*output*)

Por defecto, si no indicamos nada Quarto generará un solo formato de salida del documento en HTML. Sin embargo, es posible definir más de un formato de salida incluyendo más opciones de configuración. Por supuesto, se pueden indicar diferentes opciones para generar varios formatos de salida simultáneamente, o bien para elegir el formato de salida que queremos producir en función de nuestros intereses, seleccionando el formato que necesitamos al previsualizar o al generar el documento definitivo.

4.3.1 Previsualización

Para previsualizar el documento tenemos que pulsar el botón *Render* en el menú de herramientas de la interfaz de RStudio, tal y como se muestra en la Figura 4.4.

Figura 4.4: Botón *Render* para previsualizar el documento generado.

Por defecto, lo normal es que se abra nuestro navegador web principal o un panel en la interfaz de RStudio mostrando la página HTML con el documento ya generado. Pulsando sobre el icono con un engranaje junto al botón *Render* se puede seleccionar, entre otros aspectos, el tipo de previsualización que queremos que se lance tras completar la creación del documento o desactivar por completo dicha previsualización. La opciones disponibles se muestran en la Figura 4.5

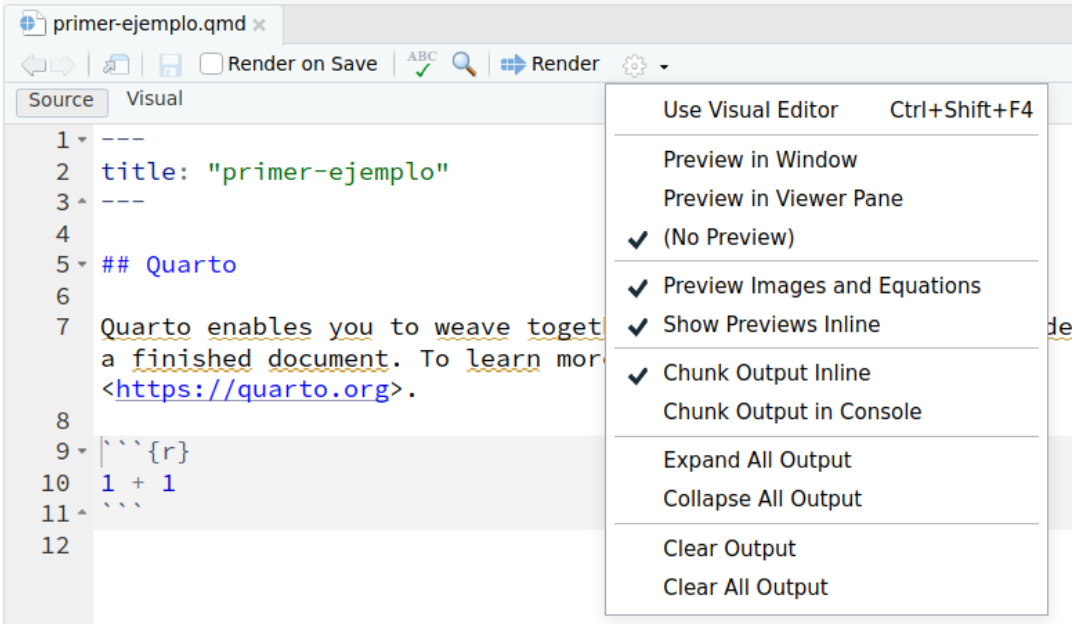


Figura 4.5: Opciones de previsualización de la operación *Render*.

4.3.2 Seleccionar el tipo de documento

Cuando tenemos varias opciones de formato de salida configuradas en nuestro documento, podemos elegir en tiempo de previsualización cuál de los formatos se elige para generar el documento. En la Figura 4.6 se puede observar un ejemplo de documento que incluye configuración para dos formatos de salida (HTML y PDF) y el cambio en el botón *Render*, en el que ahora aparece una pequeña flecha negra justo a la derecha del icono del botón para desplegar las dos opciones de salida disponibles.

4.3.3 Opciones básicas de configuración

A continuación, se presenta un ejemplo de algunas opciones básicas de configuración que suelen ser habituales en documentos con formato de salida HTML.

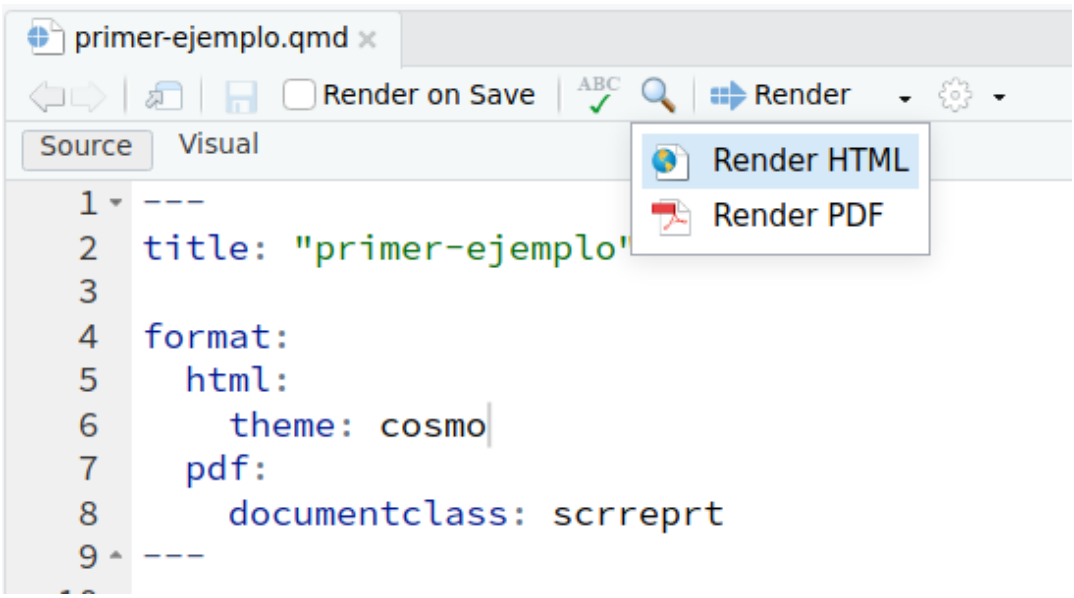


Figura 4.6: Selección de formatos de salida con la operación *Render*.

```yaml
---
title: "Mi primer documento"
author:
  - "John Doe"
  - "Mary Jane"
date: 2024-10-24

lang: es
bibliography: references.bib

format:
  html:
    theme: cosmo
    toc: true
    number-sections: true
    html-math-method: katex
    css: styles.css
  pdf:
    documentclass: scrreprt
---

RESTO DEL DOCUMENTO
```

En este ejemplo se indica, además del autor y la fecha, una lista de dos autores, el lenguaje principal del documento (español), el archivo de referencias de bibliografía (en formato `.bib`) y ya dentro de las opciones HTML, el tema de maquetación, la inclusión de una tabla de contenidos (por defecto situada en la parte superior derecha), numeración de secciones, selección del motor para renderizar ecuaciones en el documento y un archivo de estilos personalizados en formato CSS para ajustar algunas opciones finas de maquetación.

Una opción que conviene destacar es la de forzar a que todos los recursos (imágenes, información de estilos, etc.) estén integrados en el propio archivo HTML, para facilitar la compartición o publicación directa del documento sin necesidad de aportar también los archivos auxiliares necesarios para mostrarlo en el navegador. Esta opción se muestra a continuación:

```yaml
format:
  html:
    embed-resources: true
```

4.4 *Chunks* de código ejecutable

La característica más diferencial de los documentos creados con Quarto es la posibilidad de intercalar fragmentos de código ejecutable, llamadas *chunks* en el propio documento. Esto incluye también la opción de que dicho código genere diferentes resultados (numéricos, gráficos, tablas, animaciones, etc.) que se integren directamente en el documento. De este modo, si mantenemos actualizado el código siempre se generarán las versiones correcgtas de dichos resultados.

Los fragmentos de código ejecutable tienen la siguiente estructura:

````
```{r}
#| label: id-fragmento

# Aquí va el código ejecutable
a = c(1, 2, 3, 4)
b = a^2
```
````

La tripleta de caracteres ``` se denomina *fence* y delimita el comienzo y el final del fragmento de código. Justo a continuación del delimitador de apertura se escribe entre llaves el identificador del lenguaje de programación en el que está escrito el código de ese fragmento. Esa información se usa para elegir el resaltado de sintaxis apropiado para mostrar el código de ese lenguaje y para seleccionar el intérprete que ejecuta el código y produce los resultados.

En las siguientes líneas podemos incluir una o varias **opciones de configuración** específicas para ese fragmento de código, mediante la sintaxis `#| opcion: valor`. Por ejemplo, en el fragmento anterior la opción `#| label: id-fragmento` crea una etiqueta (que debe ser unívoca) para identificar a ese fragmento de código dentro del documento.

- Lista de opciones para fragmentos de código.

Algunas opciones de uso frecuente son:

- `eval: true | false | [...]`: Indica si se debe evaluar (ejecutar) el contenido de ese fragmento. Se puede pasar una lista de números de línea positivos o negativos para seleccionar explícitamente qué líneas de código se incluyen (positivos) o excluyen (negativos) de la ejecución.
- `echo: true | false | fenced | [...]`: Indica si se debe incluir el código fuente del fragmento en el documento o no. La opción `fenced` incluye también el delimitador de celda como parte de la salida. Por último, también acepta una lista de números de línea positivos o negativos para seleccionar qué líneas de código se mostrarán o no en el fragmento.
- `output: true | false | asis`: Para decidir si el resultado de la ejecución del código se incluye o no en el documento. El valor `asis`fuerza a que el resultado se trate como contenido Markdown en crudo.
- `warning: true | false`: Indica si se deben incluir los mensajes de aviso en la salida.
- `error: true | false`: Marca si los mensajes de error generados se incluyen en la salida.
- `message: true | false`: Indica si los mensajes de información generados se incluyen en la salida.

Cuando los fragmentos generan figuras, estas se insertan dentro del propio documento. Veamos un ejemplo:

````
```{r}
#| label: fig-example-cars
#| fig-cap: "Gráfico de correlación lineal positiva entre el kilometraje en ciudad y en carre

library(ggplot2)
#| label: scatterplot
#| echo: true

ggplot(mpg, aes(x = hwy, y = cty, color = cyl)) +
  geom_point(alpha = 0.5, size = 2) +
  scale_color_viridis_c() +
  theme_minimal()
```
````

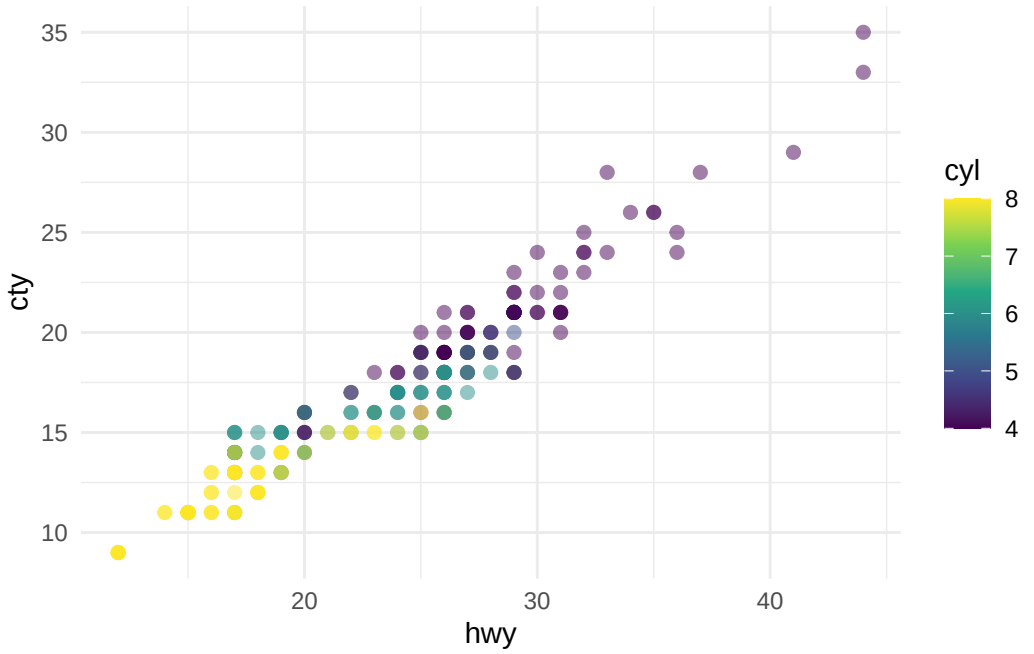


Figura 4.7: Gráfico de correlación lineal positiva entre el kilometraje en ciudad y en carretera de diferentes modelos de coches.

> **Numeración automática de figuras**
>
> Es importante que *el identificador de fragmento* que elegimos para código que genera una o varias figuras *comience por el prefijo* `fig-`. De ese modo, nos aseguramos de que Quarto le asigne automáticamente una numeración a la figura generada y que podamos crear referencias cruzadas (enlaces internos) a dicha figura en nuestro documento.
> Como veremos más adelante, otros tipos de salida como las tablas también necesitan que se les asigne un patrón concreto en su identificador de fragmento para que se numeren de forma automática y se puedan referenciar dentro del documento.

La gestión de figuras en Quarto es bastante sofisticada, hasta el punto de que se pueden organizar de forma sencilla varias subfiguras con sus respectivas descripciones individuales, como se muestra en el siguiente ejemplo usando algunas opciones adiconales.

```
```{r}
#| label: fig-mpg-subplot
#| fig-cap: "Kilometraje en ciudad y en carretera de 38 modelos populares de coches."
#|
#| fig-subcap:
#|   - "Color por núm. de cilindros."
```
```

```
#|   - "Color por cubicaje del motor, en litros."
#| layout-ncol: 1

ggplot(mpg, aes(x = hwy, y = cty, color = cyl)) +
  geom_point(alpha = 0.5, size = 2) +
  scale_color_viridis_c() +
  theme_minimal()

ggplot(mpg, aes(x = hwy, y = cty, color = displ)) +
  geom_point(alpha = 0.5, size = 2) +
  scale_color_viridis_c(option = "E") +
  theme_minimal()
```

Algunas opciones frecuentes para *chunks* que generan figuras son:

- `fig-width`: Ancho de la figura.
- `fig-height`: Alto de la figura.
- `fig-cap`: String entre comillas que se insertará como descripción al pie de la figura (*caption*).
- `fig-alt`: Mensaje de texto alternativo que rellena el atributo `alt` de la imagen HTML (por ejemplo, para mejorar la accesibilidad del contenido).
- `fig-dpi`: Ajuste de la resolución de la figura (en puntos por pulgada).

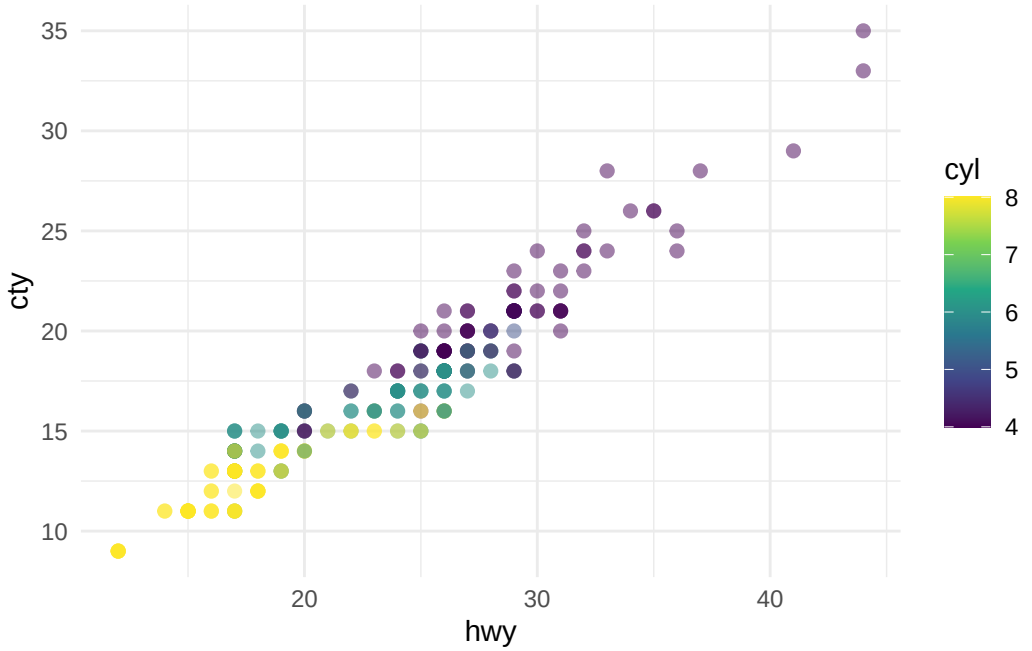
El tutorial sobre fragmentos de código ejecutables de la documentación oficial presenta más información y ejemplos sobre cómo utilizar esta potente característica de Quarto.l

4.5 Herramientas para el autor

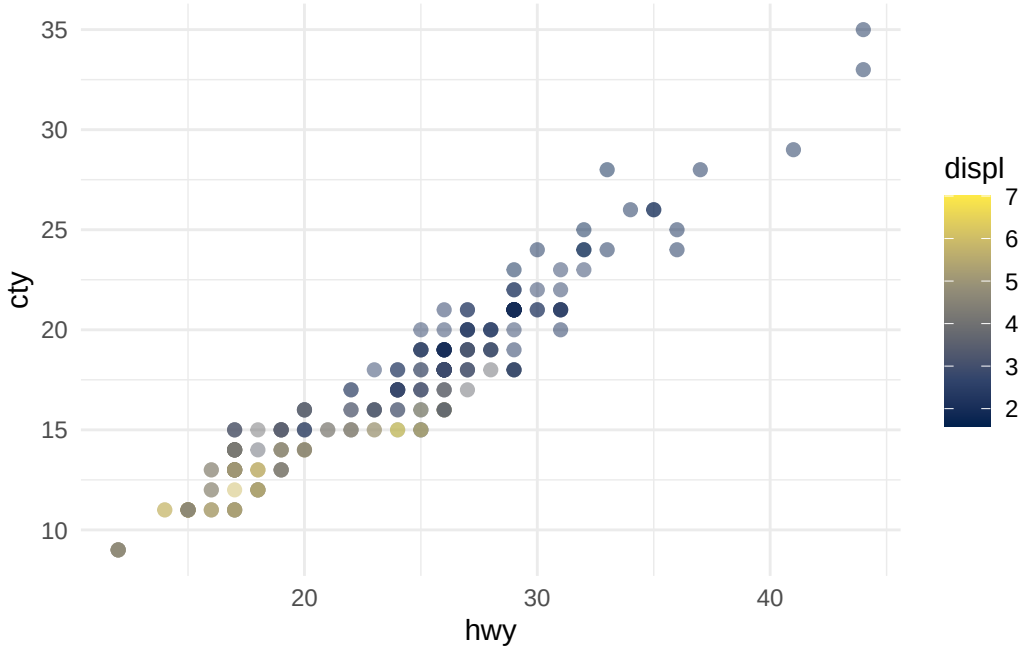
Además de la capacidad de integrar código ejecutable y sus resultados en nuestros documentos científicos, Quarto incluye un buen número de recursos y herramientas para ofrecer una experiencia de autoría completa y eficiente.

4.5.1 Secciones del documento

Como ya vimos en el ejemplo de la Sección 4.3.3, existen dos opciones de configuración del documento HTML que nos permiten numerar las secciones e incorporar una tabla de contenidos generada automáticamente en la parte superior derecha de nuestro documento.



(a) Color por núm. de cilindros.



(b) Color por cubicaje del motor, en litros.

Figura 4.8: Kilometraje en ciudad y en carretera de 38 modelos populares de coches.

```yaml
format:
  html:
    toc: true
    number-sections: true
```

Un funcionalidad importante para la creación de documentación científica es poder incluir **referencias cruzadas**, es decir, enlaces que nos lleven a otras secciones del documento. En Quarto esto se consigue siguiendo un sencillo procedimiento en dos pasos:

1. Añadimos una *etiqueta única* para identificar la sección con la sintaxis:

```
## Título de sección {#sec-etiqueta}
```

2. Referenciamos en otro lugar del texto la etiqueta que hemos creado para esa sección, de forma que Quarto crea automáticamente el enlace (referencia cruzada) a dicha sección:

```
En el texto añadimos una referencia a la @sec-etiqueta
```

Se puede ver un ejemplo de este tipo de referencias cruzadas creadas de forma automática al comienzo de esta misma sección. Por el contrario, si queremos que una sección del documento se excluya del esquema de numeración del resto de secciones, usamos en el título de esa sección la etiqueta especial:

```
## Sección no numerada {.unnumbered}
```

Existen varias opciones adicionales que controlan la forma y estilo con que se crean y numeran las secciones. Algunas de ellas son:

- `anchor-sections`: Hace que se muestre un enlace de anclado (para enlazar directamente esa sección en otro documento) cuando se pasa el ratón por encima del título de una sección.
- `toc-depth`: Especifica cuántos niveles de profundidad en la numeración de secciones aparecen en la tabla de contenidos. Por defecto se muestran 3 niveles.
- `toc-location: body | left | right | left-body | right-body`: Controla la ubicación en la que aparece la tabla de contenidos en el documento.
- `toc-title`: Cadena de caracteres con el título de la tabla de contenidos.
- `toc-expand`: Indica si se deben expandir todas las secciones de la tabla de contenidos o deben quedar colapsadas para que el usuario vaya pulsando en las que quiera expandir.
- `number-depth`: Determina la profundidad máxima a la que se numeran las secciones del documento (cuidado, debería estar en consonancia con el valor asignado a la opción `toc-depth`).

- `number-offset`: Permite ajustar el número por el que se empiezan a numerar las secciones. Si queremos que el documento comience a numerar la sección de más alto nivel como "4" entonces usamos `number-offset: 3`. Si queremos que el documento empiece en una sección de nivel 2 con numeración "1.5" debemos especificar `number-offset: [1,4]`. Definir un valor para esta opción implica que automáticamente `number-sections: true`.

4.5.2 Ecuaciones

Otro aspecto esencial de los documentos científicos es la aparición de símbolos, fórmulas y ecuaciones matemáticas. Existen varias librerías para HTML que permiten mostrar las ecuanciones debidamente formateadas en pantalla. Por su parte, LaTeX, debido a sus orígenes, siempre ha incluído potentes y versátiles herramientas para manejar este tipo de contenido, por lo que el soporte está garantizado para documentos en PDF.

En general, la sintaxis empleada para escribir las ecuaciones es muy similar a la que se utiliza en LaTeX.

- Tutorial sobre expresiones matemáticas en LaTeX.
- Resumen de sintaxis matemática en LaTeX.

Existen dos maneras de mostrar las ecuaciones en nuestro contenido, siguiendo también una filosofía similar a la de los documentos LaTeX:

- Ecuaciones en línea con el texto: para mostrar la ecuación dentro de una línea o párrafo, a la misma altura que el resto del texto.
- Ecuaciones en modo *display*: la ecuación se muestra en un espacio aparte, entre dos párrafos de texto y con cierto margen de espacio en la zona superior e inferior.

```
Ejemplo de ecuación en línea: $F = m \cdot a$
```

Que produce como resultado: ejemplo de ecuación en línea: $F = m \cdot a$.

```
Ejemplo de ecuación en modo display:

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

Que genera el siguiente resultado (ver debajo cómo añadir la numeración):

$$E = mc^2 \tag{4.1}$$

Si además queremos numerar nuestras ecuaciones, hay que acordarse de emplear el patrón de etiqueta identificadora unívoca `eq-etiqueta` para identificarla y luego poder insertar referencias internas a dicha ecuación en el texto.

```
$$ E = mc^{2} $$ {#eq-energy}
```

Como resultado, podemos insertar una a la Ecuación 4.1.

4.5.3 Tablas

Las tablas son otro contenido relevante que podemos formatear de diferentes maneras en los documentos generados con Quarto.

- Introducción a la creación de tablas en Quarto.

En este caso, el editor visual nos puede simplificar mucho esta tarea. Se aconseja probarlo para ver la diferencia, ya que es una herramienta muy intuitiva. Sin embargo, siguiendo la misma línea que el resto del taller, aquí vamos a describir los detalles para crear este contenido directamente en el código Markdown del archivo.

La forma más directa de crear una tabla en Markdown es componer una tabla `pipe`, así denominada porque su sintaxis se basa en el operador `|` de la línea de comandos. Veamos un ejemplo.

```
| Default | Left | Right | Center |
|---------|:-----|------:|:------:|
| 1       | 2    |     3 |    4   |
| 22      | 23   |    24 |   25   |
| 4       | 3    |     2 |   1    |
```

El resultado de incluir el código anterior en nuestro documento es:

Default	Left	Right	Center
1	2	3	4
22	23	24	25
4	3	2	1

Podemos observar cómo la clave para controlar el alineamiento horizontal del contenido de la tabla es situar apropiadamente el símbolo `:` en la línea justo debajo de la línea de título, que separa este del cuerpo de la tabla. Si no queremos incluir título es obligatorio que incluyamos la primera línea, pero podemos dejar las celdas en blanco.

Debajo de la tabla podemos insertar la expresión `: Caption de la tabla` para incluir un mensaje descriptivo. También es posible utilizar directamente algunos elementos de estilo incluidos en las clases de Bootstrap, el entorno de estilo web que utiliza Quarto para componer

las páginas (hemos visto antes cómo utilizar la opción de documento `theme: cosmo` para usar el tema Cosmo de Boostrap). Existen diferentes efectos, y uno de los más frecuentes es colorear en gris el fondo de las filas alternativamente así como resaltar la fila en la que está posada la flecha del ratón. Estos dos efectos son `.striped` y `.hover`, respectivamente.

```
| Default | Left | Right | Center |
|---------|:-----|------:|:------:|
| 1       | 2    |     3 |    4   |
| 22      | 23   |    24 |   25   |
| 4       | 3    |     2 |   1    |

: Caption de la tabla {.striped .hover}
```

Tabla 4.2: Caption de la tabla

Default	Left	Right	Center
1	2	3	4
22	23	24	25
4	3	2	1

Por último, de forma análoga a lo que hacemos para referenciar internamente ecuaciones y figuras en nuestro documento, también podemos etiquetar las tablas utilizando el patrón `#tbl-etiqueta` para referenciarlo como `@tbl-etiqueta` que queda formateado así: Tabla 4.3.

```
| Default | Left | Right | Center |
|---------|:-----|------:|:------:|
| 1       | 2    |     3 |    4   |
| 22      | 23   |    24 |   25   |
| 4       | 3    |     2 |   1    |

: Caption de la tabla. {#tbl-etiqueta .striped .hover}
```

Tabla 4.3: Caption de la tabla.

Default	Left	Right	Center
1	2	3	4
22	23	24	25
4	3	2	1

El mismo patrón de etiqueta se debe emplear en la opción de identificación de *chunks* de código `#| label: tbl-etiqueta` si luego queremos referenciar la tabla generada por ese fragmento de código con una referencia cruzada.

Se pueden consultar más detalles sobre la creación de subtablas, cambio de ubicación del caption, así como la creación de tablas *grid* que usan una sintaxis diferente y permiten incluir elementos de bloque arbitrarios en cada celda (múltiples párrafos, bloques de código, listas no numeradas o numeradas, etc.).

```
+-----------+-----------+--------------------+
| Fruta     | Precio    | Ventajas           |
+===========+===========+====================+
| Bananas   | $1.34     | - envoltorio       |
|           |           | - color brillante  |
+-----------+-----------+--------------------+
| Oranges   | $2.10     | - rica en vitam. C |
|           |           | - saborsa          |
+-----------+-----------+--------------------+

: Sample grid table.
```

Tabla 4.4: Tabla *grid* de ejemplo.

Fruta	Precio	Ventajas
Bananas	$1.34	• envoltorio • color brillante
Oranges	$2.10	• rica en vitam. C • saborsa

4.5.4 Llamadas

Es posible incluir bloques de llamadas de atención, para resaltar notas prácticas, advertencias o consejos de especial interés. Además, se suele poner un título a la llamada para hacerla aún más informativa.

```
::: {.callout-note}
## Título de la nota

Existen cinco tipos diferentes de llamaedas:
`note`, `tip`, `warning`, `caution`, e `important`.
:::
```

> **Título de la nota**
>
> Existen cinco tipos diferentes de llamaedas: `note`, `tip`, `warning`, `caution`, e `important`.

- Introducción al uso de llamadas en Quarto.

4.5.5 Citas bibliográficas

La gestión de referencias bibliográficas en Quarto se realiza codificando la información en formato BibTeX. Esto permite utilizar cualquiera de los formatos de citas bibliográficas soportados por este paquete, o bien incluir un fichero CLS que defina un formato estándar (APA, Chicago, IEEE, etc.).

Por ejemplo, las opciones de documento

```
---
title: "My Document"
bibliography: references.bib
csl: nature.csl
---
```

indican un fichero `references.bib` donde podemos almacenar la información sobre las referencias bibliográficas (que podemos conseguir de Google Scholar, Zotero u otras herramientas y servicios en Internet), así como un fichero de estilo de citas `nature.cls` (estilo definido por la editorial Nature).

- Repositorio CLS con estilos de citas.
- Repositorio Zotero con estilos de citas.

Dependiendo del estilo y formato de la cita, podemos utilizar una u otra sintaxis para indicar el autor y el año entre paréntesis, el autor fuera del paréntesis, números de página, capítulos, etc.

- Tabla de referencia de sintaxis para citas en Quarto.

Por último, la lista ordenada de referencias bibliográficas (según los criterios de estilo de citas que hayamos seleccionado) deben aparecer al final del documento. Para conseguirlo en un documento HTML, debemos incluir un código especial, que normalmente se coloca en una sección independiente y no numerada, tal y como se muestra en la Figura 4.9.

Cuando la salida generada es en formato PDF y se usan los motores de gestión de referencias BibLaTeX o natbib, entonces la lista de referencias siempre aparece al final de documento y la etiqueta anterior se ignora. Finalmente, en el poco frecuente caso de que no queramos incluir ninguna referencia bibliográfica en nuestro documento podemos incluir en los metadatos de la cabecera del mismo la opción `supress-bibliography: true`.

```
# References {.unnumbered}

::: {#refs}
:::
```

Figura 4.9: Sintaxis para mostrar las referencias bibliográficas al final del documento.

4.5.6 Estilo general del documento

Hasta el momento, el documento de ejemplo que hemos mostrado así como estos mismos apuntes utilizan siempre un formato de estilo o `theme` del entorno de desarrollo web Bootstrap, llamado `cosmo`. No obstante, existe una amplia lista de temas alternativos para modificar el estilo general de nuestro documento (combinación de colores, tipografía y tamaño de fuentes, organización del contenido, aspecto de los enlaces, etc.). El proyecto Quarto se encarga de combinar regularmente los temas de estilo más populares para que estén disponibles como opción del documento.

En este directorio de temas en GitHub se puede comprobar una lista actualizada de los posibles valores que podemos asignar a la opción `theme` en la cabecera del documento. Es útil experimentar con diversas opciones para encontrar la que más se ajuste al tipo de documento generado, a su contenido y a la audiencia a la que va dirigido.

En la página web https://bootswatch.com/ se puede acceder a una demo en línea de muchos de los temas disponibles.

Parte II

Libros con Quarto

5 Libros

5.1 Herramientas de redacción

5.1.1 Figuras

5.1.2 Tablas

5.1.3 Ecuaciones

5.1.4 *Callouts*

5.2 Gestión de referencias

5.2.1 Referencias cruzadas en el documento

5.2.2 Referencias bibliográficas

5.3 Plantillas y personalización

6 Taller: colección de apuntes

6.1 Plantillas y herramientas

6.2 Gestión del proyecto

6.3 Publicación

Parte III

Publicaciones

7 Artículos y publicaciones científicas

7.1 Replicabilidad en publicaciones científicas

7.2 Figuras y gráficos para publicación

7.3 EL paquete `rticles`

7.4 Ejemplos y recomendaciones

8 Principios FAIR

8.1 Visión general

8.2 Publicación del código fuente

8.3 Publicación de conjuntos de datos

8.4 Gestión de referencias

DOI

Figshare

Zenodo

arXiv

etc.

9 Recursos adicionales

Listado de recursos adicionales de interés.

See Knuth (1984) for additional discussion of literate programming.

Referencias

Barba, L. A. (2018). Terminologies for reproducible research. *arXiv preprint arXiv:1802.03311*.

Begley, C., & Ellis, L. (2012). *Drug development: Raise standards for preclinical cancer research. Nature.[Online]. 483 (7391)*.

Brainard, J., You, J., et al. (2018). What a massive database of retracted papers reveals about science publishing's «death penalty». *Science*, *25*(1), 1-5.

Burman, L. E., Reed, W. R., & Alm, J. (2010). A call for replication studies. *Public Finance Review*, *38*(6), 787-793.

Ioannidis, J. P. (2005). Why most published research findings are false. *PLoS Medicine*, *2*(8), e124.

Knuth, D. E. (1984). Literate Programming. *Comput. J.*, *27*(2), 97-111. https://doi.org/10.1093/comjnl/27.2.97

Leek, J. T., & Peng, R. D. (2015). Reproducible research can still be wrong: Adopting a prevention approach. *Proceedings of the National Academy of Sciences*, *112*(6), 1645-1646.

Peng, R. D. (2011). Reproducible research in computational science. *Science*, *334*(6060), 1226-1227.

Wicherts, J. M., Borsboom, D., Kats, J., & Molenaar, D. (2006). The poor availability of psychological research data for reanalysis. *American PAsychologist*, *61*(7), 726.

A Comandos de utilidad

A.1 Comandos Quarto

A.2 Celdas de código en R

B Entornos de desarrollo para Quarto

B.1 R Studio

B.2 Visual Studio

C Paquetes R y atribuciones

C.1 `rticles`

Shiny

IOSlides, otras...

Otros

C.2 Atribución de imágenes e iconos

- R logo by The R Foundation.
- Python file icons created by Flat Icons - Flaticon.
- Html icons created by Freepik - Flaticon.
- Microsoft word icons created by Pixel perfect - Flaticon.
- Pdf icons created by Dimitry Miroliubov - Flaticon.

D Documentos PDF con LaTeX

D.1 Salida en formato PDF

D.2 Acerca de LaTeX

D.3 Ejemplos prácticos

Referencias

Barba, L. A. (2018). Terminologies for reproducible research. *arXiv preprint arXiv:1802.03311*.

Begley, C., & Ellis, L. (2012). *Drug development: Raise standards for preclinical cancer research. Nature.[Online]. 483 (7391)*.

Brainard, J., You, J., et al. (2018). What a massive database of retracted papers reveals about science publishing's «death penalty». *Science*, *25*(1), 1-5.

Burman, L. E., Reed, W. R., & Alm, J. (2010). A call for replication studies. *Public Finance Review*, *38*(6), 787-793.

Ioannidis, J. P. (2005). Why most published research findings are false. *PLoS Medicine*, *2*(8), e124.

Knuth, D. E. (1984). Literate Programming. *Comput. J.*, *27*(2), 97-111. https://doi.org/10.1093/comjnl/27.2.97

Leek, J. T., & Peng, R. D. (2015). Reproducible research can still be wrong: Adopting a prevention approach. *Proceedings of the National Academy of Sciences*, *112*(6), 1645-1646.

Peng, R. D. (2011). Reproducible research in computational science. *Science*, *334*(6060), 1226-1227.

Wicherts, J. M., Borsboom, D., Kats, J., & Molenaar, D. (2006). The poor availability of psychological research data for reanalysis. *American PAsychologist*, *61*(7), 726.