# # Informes con R Markdown

{#reports-with-r-markdown}

R Markdown es una herramienta ampliamente utilizada para crear resultados automatizados, reproducibles y dignos de compartir, como los informes. Puedes generar resultados estáticos o interactivos, en Word, pdf, html, powerpoint y otros formatos.

Un script de R Markdown intercala el código R y el texto de tal manera que el script se convierte en su documento de salida. Puedes crear un documento completo con formato, incluyendo texto narrativo (puede ser dinámico para cambiar en función de sus datos), tablas, figuras, viñetas/números, bibliografías, etc.

Estos documentos pueden producirse para actualizarlos de forma rutinaria (por ejemplo, informes de vigilancia diarios) y/o ejecutarse sobre subconjuntos de datos (por ejemplo, informes para cada jurisdicción).

Otras páginas de este manual amplían este tema:

* La página [Organización de los informes de rutina](#organizing-routine-reports) muestra cómo rutinizar la producción de informes con carpetas autogeneradas con marca de tiempo.
* La página [Dashboardscon R Markdown](#dashboards-with-r-markdown) explica cómo formatear un informe de R Markdown como un cuadro de mando o tablero de control.

Cabe destacar que el proyecto [R4Epis](https://r4epis.netlify.app/) ha desarrollado plantillas de scripts R Markdown para los escenarios de brotes y encuestas más comunes que se encuentran en las ubicaciones de los proyectos de MSF.

## Preparación

**Antecedentes de R Markdown**

Explicar algunos de los conceptos y paquetes involucrados:

* **Markdown** es un "lenguaje" que permite escribir un documento en texto plano, que puede ser convertido a html y otros formatos. No es específico de R. Los archivos escritos en Markdown tienen una extensión '.md'.
* R **Markdown**: es una variación de markdown que es específica de R - le permite escribir un documento usando markdown para producir texto y para incrustar código R y mostrar sus resultados. Los archivos R Markdown tienen la extensión '.Rmd'.
* **rmarkdown - el paquete**: Esto es usado por R para convertir el archivo .Rmd en la salida deseada. Su objetivo es convertir la sintaxis markdown (texto), por lo que también necesitamos...
* **knitr**: Este paquete de R leerá los trozos de código, los ejecutará y los "tejerá" de nuevo en el documento. Así es como se incluyen las tablas y los gráficos junto al texto.
* **Pandoc**: Por último, pandoc convierte la salida en word/pdf/powerpoint, etc. Es un software independiente de R, pero se instala automáticamente con RStudio.

En resumen, el proceso que ocurre en segundo plano (¡no es necesario que conozcas todos estos pasos!) consiste en alimentar el archivo .Rmd a **knitr**, que ejecuta los trozos de código R y crea un nuevo archivo .md (markdown) que incluye el código R y su salida renderizada. El archivo .md es entonces procesado por pandoc para crear el producto final: un documento de Microsoft Word, un archivo HTML, un documento powerpoint, un pdf, etc.

(fuente: [https:](https://rmarkdown.rstudio.com/authoring_quick_tour.html)//rmarkdown.rstudio.com/authoring\_quick\_tour.html):

**Instalación**

Para crear una salida de R Markdown, necesita tener instalado lo siguiente:

* El paquete **rmarkdown** (**knitr** también se instalará automáticamente)
* Pandoc, que debería venir instalado con RStudio. Si no utiliza RStudio, puede descargar Pandoc aquí: [http:](http://pandoc.org/)//pandoc.org.
* Si quiere generar una salida en PDF (un poco más complicado), necesitará instalar LaTeX. Para los usuarios de R Markdown que no hayan instalado LaTeX antes, recomendamos que instalen TinyTeX [(](https://yihui.name/tinytex/)https://yihui.name/tinytex/). Puedes utilizar los siguientes comandos:

## Cómo empezar

### Instalar el paquete R rmarkdown

Instale el paquete R **rmarkdown**. En este manual destacamos p\_load() de **pacman**, que instala el paquete si es necesario y lo carga para su uso. También puede cargar los paquetes instalados con library() de **.** Consulta la página sobre [los fundamentos de R](#r-basics) para obtener más información sobre los paquetes de R.

### Iniciar un nuevo archivo Rmd

En RStudio, abra un nuevo archivo R markdown, comenzando con 'Archivo', luego 'Nuevo archivo' y luego 'R markdown...'.

R Studio le dará algunas opciones de salida para elegir. En el ejemplo siguiente seleccionamos "HTML" porque queremos crear un documento html. El título y los nombres de los autores no son importantes. Si el tipo de documento de salida que desea no es uno de estos, no se preocupe - puede elegir cualquiera y cambiarlo en el script más tarde.

Esto abrirá un nuevo script .Rmd.

### Es importante saber

**El directorio de trabajo**

El directorio de trabajo de un archivo markdown es el lugar donde se guarda el propio archivo Rmd. Por ejemplo, si el proyecto R está dentro de ~/Documents/projectX y el archivo Rmd en sí está en una subcarpeta ~/Documents/projectX/markdownfiles/markdown.Rmd, el código read.csv("data.csv") dentro del markdown buscará un archivo csv en la carpeta markdownfiles, y no en la carpeta raíz del proyecto donde los scripts dentro de los proyectos normalmente buscarían automáticamente.

Para referirse a los archivos en otro lugar, tendrá que utilizar la ruta completa del archivo o utilizar el paquete **here**. El paquete **here** establece el directorio de trabajo en la carpeta raíz del proyecto R y se explica en detalle en las páginas de [proyectos R](#r-projects) e [importación y exportación](#import-and-export) de este manual. Por ejemplo, para importar un archivo llamado "data.csv" desde la carpeta projectX, el código sería import(here("data.csv")).

Ten en cuenta que no se recomienda el uso de setwd() en los scripts de R Markdown - sólo se aplica al trozo de código en el que está escrito.

**Trabajar en una unidad frente a tu ordenador**

Debido a que R Markdown puede tener problemas con pandoc cuando se ejecuta en una unidad de red compartida, se recomienda que su carpeta esté en su máquina local, por ejemplo, en un proyecto dentro de 'Mis Documentos'. Si utilizas Git (¡muy recomendable!), esto te resultará familiar. Para más detalles, vea las páginas del manual sobre [R en unidades de red](#r-on-network-drives) y [Errores y ayuda].

## Componentes de R Markdown

Un documento R Markdown puede ser editado en RStudio igual que un script estándar de R. Cuando se inicia un nuevo script de R Markdown, RStudio intenta ser útil mostrando una plantilla que explica las diferentes secciones de un script de R Markdown.

Lo que aparece a continuación es lo que aparece al iniciar un nuevo script Rmd destinado a producir una salida html (según la sección anterior).

Como puede ver, hay tres componentes básicos en un archivo Rmd: YAML, texto Markdown y trozos de código R.

Estos crearán y se convertirán en la salida de su documento. Consulta el siguiente diagrama:

### Metadatos YAML

Denominado "metadatos YAML" o simplemente "YAML", se encuentra en la parte superior del documento R Markdown. Esta sección del script le dirá a su archivo Rmd qué tipo de salida producir, preferencias de formato y otros metadatos como el título del documento, el autor y la fecha. Hay otros usos que no se mencionan aquí (pero a los que se hace referencia en "Producción de una salida"). Ten en cuenta que la sangría es importante; los tabuladores no se aceptan, pero los espacios sí.

Esta sección debe comenzar con una línea que contenga sólo tres guiones --- y debe cerrar con una línea que contenga sólo tres guiones ---. Los parámetros YAML vienen en pares clave:valor. La colocación de los dos puntos en YAML es importante - los pares clave:valor están separados por dos puntos (¡no por signos de igualdad!).

El YAML debe comenzar con los metadatos del documento. El orden de estos parámetros YAML primarios (sin sangría) no importa. Por ejemplo:

título: "Mi documento"

autor: "Yo"

fecha: "2021-06-04"

Puedes utilizar código R en valores YAML escribiéndolo como código en línea (precedido por r entre comillas) pero también entre comillas (véase el ejemplo anterior para date:).

En la imagen de arriba, porque hemos clicado en que nuestra salida por defecto sea un archivo html, podemos ver que el YAML dice salida: html\_document. Sin embargo, también podemos cambiar esto para decir powerpoint\_presentation o word\_document o incluso pdf\_document.

### Texto

Esta es la narrativa de su documento, incluyendo los títulos y encabezados. Está escrito en el lenguaje "markdown", que se utiliza en muchos programas diferentes.

A continuación se presentan las formas principales de escribir este texto. Consulta una documentación más extensa disponible en R Markdown "cheatsheet" en el [sitio web de RStudio](https://rstudio.com/resources/cheatsheets/).

#### Nuevas líneas

Únicamente en R Markdown, para iniciar una nueva línea, introduzca \*dos espacios\*\* al final de la línea anterior y luego Enter/Return.

#### Caso

Rodee su texto normal con estos caracteres para cambiar su apariencia en la salida.

* Guiones bajos (\_texto\_) o asterisco simple (\*texto\*) para poner en cursiva
* Doble asterisco (\*\*texto\*\*) para el **texto en negrita**
* Tildes (texto) para mostrar el texto como código

El aspecto real de la fuente puede establecerse utilizando plantillas específicas (especificadas en los metadatos YAML; véase el ejemplo de las pestañas).

#### Color

No existe un mecanismo sencillo para cambiar el color del texto en R Markdown. Una solución, si su salida es un archivo HTML, es añadir una línea HTML en el texto de Markdown. El siguiente código HTML imprimirá una línea de texto en negrita roja.

<span style="color: rojo;"> \*\*\_DANGER:\_\*\* Esto es una advertencia. </span>

**PELIGRO:** Esto es una advertencia.

#### Títulos y encabezamientos

Un símbolo hash en una porción de texto de un script de R Markdown crea un encabezado. Esto es diferente que en un trozo de código R en el script, en el que un símbolo hash es un mecanismo para comentar/anotar/desactivar, como en un script normal de R.

Los distintos niveles de encabezamiento se establecen con diferentes números de símbolos de almohadilla al comienzo de una nueva línea. Un símbolo de almohadilla es un título o encabezamiento primario. Dos símbolos hash son un encabezamiento de segundo nivel. Los encabezamientos de tercer y cuarto nivel pueden hacerse con más símbolos hash sucesivamente.

# Encabezamiento / título de primer nivel

## Rúbrica de segundo nivel

### Rúbrica de tercer nivel

#### Viñetas y numeración

Utiliza asteriscos (\*) para crear una lista de viñetas. Termine la frase anterior, introduzca dos espacios, Enter/Return dos veces, y luego comience sus viñetas. Incluya un espacio entre el asterisco y el texto de su viñeta. Después de cada viñeta, introduzca dos espacios y luego Enter/Return. Las sub viñetas funcionan de la misma manera pero con sangría. Los números funcionan de la misma manera, pero en lugar de un asterisco, escriba 1), 2), etc. A continuación se muestra cómo podría ser el texto de tu script de R Markdown.

Aquí están mis viñetas (hay dos espacios después de los dos puntos):

\* Viñeta 1 (seguida de dos espacios y Enter/Return)

\* Viñeta 2 (seguida de dos espacios y Enter/Return)

\* Subbalanceo 1 (seguido de dos espacios y Enter/Return)

\* Subbalanceo 2 (seguido de dos espacios y Enter/Return)

#### Comentar el texto

Puedes "comentar" el texto de R Markdown del mismo modo que puede utilizar el "#" para comentar una línea de código R en un chunk de R. Simplemente resalta el texto y clica Ctrl+Mayús+c (Cmd+Mayús+c para Mac). El texto estará rodeado de flechas y se volverá verde. No aparecerá en su salida.

### Trozos de código

Las secciones del script que se dedican a ejecutar el código R se denominan "chunks". Aquí es donde se pueden cargar paquetes, importar datos y realizar la gestión y visualización de datos propiamente dicha. Puedes haber muchos "chunks" de código, por lo que pueden ayudarle a organizar su código R en partes, quizás intercaladas con texto. Para tener en cuenta: estos "trozos" parecerán tener un color de fondo ligeramente diferente al de la parte narrativa del documento.

Cada trozo se abre con una línea que comienza con tres marcas de retroceso y corchetes que contienen parámetros para el trozo ({ }). El chunk termina con otros tres back-ticks.

Puedes crear un nuevo fragmento escribiéndolo tú mismo, utilizando el atajo de teclado "Ctrl + Alt + i" (o Cmd + Shift + r en Mac), o clicando en el icono verde 'insertar un nuevo fragmento de código' en la parte superior de tu editor de scripts.

Algunas notas sobre el contenido de las llaves { }:

* Empiezan por "r" para indicar que el nombre del idioma dentro del chunk es R
* Después de la r puedes escribir opcionalmente un "nombre" de chunk - no es necesario pero puede ayudarte a organizar tu trabajo. Ten en cuenta que si nombras tus chunks, debes usar SIEMPRE nombres únicos o de lo contrario R se quejará cuando intentes renderizarlos.
* Las llaves pueden incluir también otras opciones, escritas como tag=value, como por ejemplo
* eval = FALSE para no ejecutar el código R
* echo = FALSE para no imprimir el código fuente de R del chunk en el documento de salida
* warning = FALSE para no imprimir las advertencias producidas por el código R
* message = FALSE para no imprimir ningún mensaje producido por el código R
* include = TRUE/FALSE si se incluyen los resultados de los trozos (por ejemplo, los gráficos) en el documento
* out.width = y out.height = - proporcionar en estilo out.width = "75%"
* fig.align = "center" ajustar cómo se alinea una figura en la página
* fig.show='hold' si tu chunk imprime múltiples figuras y quieres que se impriman una al lado de la otra (par con out.width = c("33%", "67%"). También puede establecer como fig.show='asis' para mostrarlas debajo del código que las genera, 'hide' para ocultarlas, o 'animate' para concatenar varias en una animación.
* La cabecera de un trozo debe escribirse en una sola línea
* Intente evitar los puntos, las barras bajas y los espacios. Utiliza guiones ( - ) en su lugar si necesita un separador.

Lea más extensamente sobre las opciones de **knitr** [aquí](https://yihui.org/knitr/options/).

Algunas de las opciones anteriores pueden configurarse con apuntar y clicar mediante los botones de configuración situados en la parte superior derecha del chunk. Aquí puedes especificar qué partes del chunk quieres que incluya el documento renderizado, es decir, el código, las salidas y las advertencias. Esto aparecerá como preferencias escritas dentro de los corchetes, por ejemplo, echo=FALSE si especifica que quiere 'Mostrar sólo la salida'.

También hay dos flechas en la parte superior derecha de cada trozo, que son útiles para ejecutar el código dentro de un trozo, o todo el código en trozos anteriores. Pasa el cursor por encima de ellas para ver lo que hacen.

Para que las opciones globales se apliquen a todos los chunks del script, puede configurar esto dentro de su primer chunk de código R en el script. Por ejemplo, para que sólo se muestren las salidas de cada trozo de código y no el propio código, puede incluir este comando en el trozo de código R:

#### Código R en el texto

También puede incluir un mínimo de código R dentro de los back-ticks. Dentro de los back-ticks, comience el código con "r" y un espacio, para que RStudio sepa que debe evaluar el código como código R. Vea el ejemplo siguiente.

El ejemplo siguiente muestra múltiples niveles de encabezamiento, viñetas, y utiliza el código R para la fecha actual (Sys.Date()) para evaluar en una fecha impresa.

El ejemplo anterior es sencillo (muestra la fecha actual), pero utilizando la misma sintaxis puede mostrar valores producidos por un código R más complejo (por ejemplo, para calcular el mínimo, la mediana o el máximo de una columna). También puede integrar objetos R o valores que fueron creados en trozos de código R anteriormente en el script.

Como ejemplo, el siguiente script calcula la proporción de casos que tienen menos de 18 años, utilizando funciones **tidyverse**, y crea los objetos less18, total y less18prop. Este valor dinámico se inserta en el texto posterior. Vemos cómo queda cuando se teje en un documento de Word.

### Imágenes

Puedes incluir imágenes en su R Markdown de dos maneras:

Si lo anterior no funciona, prueba a utilizar knitr::include\_graphics()

(recuerde que la ruta de su archivo puede ser escrita usando el paquete **aquí**)

### Tablas

Cree una tabla utilizando guiones ( - ) y barras ( | ). El número de guiones antes/entre las barras permite el número de espacios en la celda antes de que el texto comience a envolverse.

Columna 1 |Columna 2 |Columna 3

---------|----------|--------

Celda A |Celda B |Celda C

Celda D |Celda E |Celda F

El código anterior produce la siguiente tabla:

| **Columna 1** | **Columna 2** | **Columna 3** |
| --- | --- | --- |
| Celda A | Celda B | Celda C |
| Celda D | Celda E | Celda F |

### Secciones con pestañas

Para las salidas HTML, puede organizar las secciones en "pestañas". Basta con añadir .tabset en las llaves { } que se colocan después de un encabezamiento. Todos los subtítulos debajo de ese encabezado (hasta otro encabezado del mismo nivel) aparecerán como pestañas en las que el usuario puedes clicar. Lee más [aquí](https://bookdown.org/yihui/rmarkdown-cookbook/html-tabs.html)

Puedes añadir una opción adicional .tabset-pills después de .tabset para dar a las propias pestañas una apariencia "en forma de píldora". Ten en cuenta que al ver la salida HTML con pestañas, la funcionalidad de búsqueda Ctrl+f sólo buscará en las pestañas "activas", no en las ocultas.

## Estructura de los archivos

Hay varias maneras de estructurar su R Markdown y cualquier script de R asociado. Cada una tiene ventajas y desventajas:

* R Markdown autónomo: todo lo necesario para el informe se importa o se crea dentro de R Markdown
  + Fuente de otros archivos - Puedes ejecutar scripts R externos con el comando source() y utilizar sus salidas en el Rmd
  + Scripts hijos - un mecanismo alternativo para source()
* Utilizar un "archivo de ejecución" - Ejecutar comandos en un script de R antes de renderizar el R Markdown

### Rmd autónomo

Para un informe relativamente sencillo, puede optar por organizar tu script de R Markdown de manera que sea "autocontenido" y no implique ningún script externo.

Todo lo que necesita para ejecutar el R Markdown se importa o se crea dentro del archivo Rmd, incluyendo todos los trozos de código y la carga de paquetes. Este enfoque "autocontenido" es apropiado cuando no necesita hacer mucho procesamiento de datos (por ejemplo, trae un archivo de datos limpio o semilimpio) y la representación del R Markdown no tomará demasiado tiempo.

En este escenario, una organización lógica del script de R Markdown podría ser:

1. Establecer las opciones globales de **knitr**
2. Cargar paquetes
3. Importar datos
4. Datos del proceso
5. Producir resultados (tablas, gráficos, etc.)
6. Guarde los resultados, si procede (.csv, .png, etc.)

#### Fuente de otros archivos

Una variación del enfoque "autocontenido" es hacer que los trozos de código R Markdown "originen" (ejecuten) otros scripts R. Esto puede hacer que su script de R Markdown sea menos desordenado, más simple y más fácil de organizar. También puede ayudar si quiere mostrar las cifras finales al principio del informe. En este enfoque, el script de R Markdown final simplemente combina las salidas preprocesadas en un documento.

Una forma de hacerlo es proporcionando los scripts de R (ruta del archivo y nombre con extensión) al comando **base de** R source().

Ten en cuenta que cuando se utiliza source() dentro de R Markdown, los archivos externos se seguirán ejecutando durante el curso de la representación de su archivo Rmd. Por lo tanto, cada script se ejecuta cada vez que se renderiza el informe. Por lo tanto, tener estos comandos source() dentro del R Markdown no acelera el tiempo de ejecución, ni ayuda mucho a la depuración, ya que el error producido todavía se imprimirá al producir el R Markdown.

Una alternativa es utilizar la opción child = **knitr**. EXPLICAR MÁS PARA HACER

Debe ser consciente de los distintos entornos de R. Los objetos creados dentro de un entorno no estarán necesariamente disponibles para el entorno utilizado por R Markdown.

### Archivo de ejecución

Este enfoque implica utilizar el script de R que contiene el comando(s) render() para preprocesar los objetos que se introducen en el markdown de R.

Por ejemplo, puede cargar los paquetes, cargar y limpiar los datos, e incluso crear los gráficos de interés antes de render(). Estos pasos pueden ocurrir en el script de R, o en otros scripts que se originan. Siempre y cuando estos comandos ocurran en la misma sesión de RStudio y los objetos se guarden en el entorno, los objetos pueden ser llamados dentro del contenido de Rmd. Entonces el propio R markdown sólo se utilizará para el paso final - para producir la salida con todos los objetos pre-procesados. Esto es mucho más fácil de depurar si algo va mal.

Este enfoque es útil por las siguientes razones:

* Mensajes de error más informativos - estos mensajes serán generados por el script de R, no por el R Markdown. Los errores de R Markdown tienden a decirle qué trozo tuvo un problema, pero no le dirán qué línea.
* Si procede, puede ejecutar pasos de procesamiento largos antes del comando render() - se ejecutarán sólo una vez.

En el ejemplo siguiente, tenemos un script de R separado en el que preprocesamos un objeto de datos en el entorno de R y luego renderizamos el "create\_output.Rmd" usando render().

### Estructura de la carpeta

El flujo de trabajo también se refiere a la estructura general de las carpetas, como tener una carpeta de "salida" para los documentos y figuras creados, y carpetas de "datos" o "entradas" para los datos depurados. No entramos en más detalles aquí, pero echa un vistazo a la página de [organización de los informes de rutina](#organizing-routine-reports).

## Elaboración del documento

Puedes elaborar el documento de las siguientes maneras:

* Manualmente clicando sobre el botón "Knit" en la parte superior del editor de scripts de RStudio (rápido y fácil)
* Ejecuta el comando render() (ejecutado fuera del script de R Markdown)

### Opción 1: botón "Knit" (tejer)

Cuando tengas el archivo Rmd abierto, clica el icono/botón 'Knit' en la parte superior del archivo.

R Studio le mostrará el progreso dentro de una pestaña 'R Markdown' cerca de su consola R. El documento se abrirá automáticamente cuando esté completo.

El documento se guardará en la misma carpeta que su script de R markdown, y con el mismo nombre de archivo (aparte de la extensión). Obviamente, esto no es ideal para el control de versiones (se sobreescribirá cada vez que se haga un punto, a menos que se mueva manualmente), ya que entonces puede que tengas que renombrar el archivo (por ejemplo, añadir una fecha).

Este es el botón de acceso directo de RStudio para la función render() de **rmarkdown**. Este enfoque sólo es compatible con un R markdown autocontenido, donde todos los componentes necesarios existen o se originan dentro del archivo.

### Opción 2: comando render()

Otra forma de producir su salida de R Markdown es ejecutar la función render() (del paquete **rmarkdown**). Debe ejecutar este comando fuera del script de R Markdown, ya sea en un script de R separado (a menudo llamado "archivo de ejecución"), o como un comando independiente en la consola de R.

Al igual que con "knit", la configuración por defecto guardará la salida Rmd en la misma carpeta que el script Rmd, con el mismo nombre de archivo (aparte de la extensión del archivo). Por ejemplo, "mi\_informe.Rmd" cuando se teje creará "mi\_informe.docx" si se teje en un documento de Word. Sin embargo, al usar render() tiene la opción de usar diferentes configuraciones. render() puede aceptar argumentos que incluyen:

* output\_format = Este es el formato de salida al que se va a convertir (por ejemplo, "html\_document", "pdf\_document", "word\_document", o "all"). También puede especificar esto en el YAML dentro del script de R Markdown.
* output\_file = Este es el nombre del archivo de salida (y la ruta del archivo). Se puede crear a través de funciones de R como here() o str\_glue() como se demuestra a continuación.
* output\_dir = Este es un directorio de salida (carpeta) para guardar el archivo. Esto le permite elegir una alternativa distinta al directorio en el que se guarda el archivo Rmd.
* output\_options = Puedes proporcionar una lista de opciones que anulen las del YAML del script (por ejemplo )
* output\_yaml = Puedes proporcionar la ruta a un archivo .yml que contenga las especificaciones YAML
* params = Ver la sección de parámetros más abajo
* Vea la lista completa [aquí](https://pkgs.rstudio.com/rmarkdown/reference/render.html)

Como ejemplo, para mejorar el control de versiones, el siguiente comando guardará el archivo de salida dentro de una subcarpeta 'outputs', con la fecha actual en el nombre del archivo. Para crear el nombre del archivo, se utiliza la función str\_glue() del paquete **stringr** para "pegar" las cadenas estáticas (escritas sin formato) con el código dinámico de R (escrito entre llaves). Por ejemplo, si es 10 de abril de 2021, el nombre del archivo de abajo será "Informe\_2021-04-10.docx". Consulta la página sobre [Caracteres y cadenas](#characters-and-strings) para obtener más detalles sobre str\_glue().

A medida que el archivo se renderiza, la consola de RStudio le mostrará el progreso de la renderización hasta el 100%, y un mensaje final para indicar que la renderización se ha completado.

### Opción 3: paquete ****informativo****

El paquete de R **reportfactory** ofrece un método alternativo de organización y compilación de informes R Markdown para situaciones en las que se ejecutan informes de forma rutinaria (por ejemplo, diariamente, semanalmente...). Facilita la compilación de múltiples archivos R Markdown y la organización de sus resultados. En esencia, proporciona una "fábrica" desde la que se pueden ejecutar los informes R Markdown, obtener automáticamente carpetas con fecha y hora para los resultados, y tener un control de versiones "ligero".

Lea más sobre este flujo de trabajo en la página sobre la [organización de los informes de rutina](#organizing-routine-reports).

## Informes parametrizados

Puedes utilizar la parametrización para hacer que un informe sea dinámico, de forma que pueda ejecutarse con una configuración específica (por ejemplo, una fecha o lugar concretos o con determinadas opciones de tejido). A continuación, nos centramos en los aspectos básicos, pero hay más [detalles en línea](https://bookdown.org/yihui/rmarkdown/parameterized-reports.html) sobre los informes parametrizados.

Utilizando linelist del ébola como ejemplo, digamos que queremos ejecutar un informe de vigilancia estándar para cada hospital cada día. Mostramos cómo se puede hacer esto usando parámetros.

Importante: los informes dinámicos también son posibles sin la estructura formal de parámetros (sin *params:*), utilizando simples objetos R en un script adyacente de R. Esto se explica al final de esta sección.

### Ajuste de los parámetros

Tienes varias opciones para especificar los valores de los parámetros para tu salida de R Markdown.

#### Opción 1: Establecer parámetros dentro de YAML

Edite el YAML para incluir una opción params:, con declaraciones sangradas para cada parámetro que desee definir. En este ejemplo creamos los parámetros date y hospital, para los que especificamos valores. Estos valores están sujetos a cambios cada vez que se ejecuta el informe. Si utiliza el botón "Knit" para producir la salida, los parámetros tendrán estos valores por defecto. Del mismo modo, si utiliza render() los parámetros tendrán estos valores por defecto a menos que se especifique lo contrario en el comando render().

---

título: Informe de vigilancia

salida: documento\_html

params:

fecha: 2021-04-10

hospital: Hospital Central

---

En el fondo, estos valores de los parámetros están contenidos en una lista de sólo lectura llamada params. Así, puedes insertar los valores de los parámetros en el código de R como lo haría con otro objeto/valor de R en tu entorno. Simplemente escriba params$ seguido del nombre del parámetro. Por ejemplo params$hospital para representar el nombre del hospital ("Hospital Central" por defecto).

Ten en cuenta que los parámetros también pueden tener valores verdaderos o falsos, y por lo tanto estos pueden ser incluidos en sus opciones de **knitr** para un chunk de R. Por ejemplo, puedes establecer {r, eval=params$run} en lugar de {r, eval=FALSE}, y ahora si el chunk se ejecuta o no depende del valor de un parámetro run:.

Ten en cuenta que para los parámetros que son fechas, serán introducidos como una cadena. Por lo tanto, para que params$date se interprete en el código de R, es probable que tenga que ser envuelto con as.Date() o una función similar para convertir al tipo Date.

#### Opción 2: Establecer los parámetros dentro de render()

Como se ha mencionado anteriormente, como alternativa a clicar el botón "Knit" para producir la salida es ejecutar la función render() desde un script independiente. En este último caso, se pueden especificar los parámetros a utilizar en ese renderizado al argumento params = de render().

Ten en cuenta que los valores de los parámetros proporcionados aquí sobrescribirán sus valores por defecto si se escriben en el YAML. Escribimos los valores entre comillas ya que en este caso deben ser definidos como valores de carácter/cadena.

El siguiente comando renderiza "surveillance\_report.Rmd", especifica un nombre de archivo de salida dinámico y una carpeta, y proporciona una lista() de dos parámetros y sus valores al argumento params =.

#### Opción 3: Configurar los parámetros mediante una interfaz gráfica de usuario

Para obtener una sensación más interactiva, también puede utilizar la interfaz gráfica de usuario (GUI) para seleccionar manualmente los valores de los parámetros. Para ello, podemos clicar en el menú desplegable situado junto al botón "Tejer" y elegir "Tejer con parámetros".

Aparecerá una ventana emergente que le permitirá introducir los valores de los parámetros establecidos en el YAML del documento.

Puedes lograr lo mismo a través de un comando render() especificando params = "ask", como se demuestra a continuación.

Sin embargo, la introducción de valores en esta ventana emergente está sujeta a errores y faltas de ortografía. Es posible que prefiera añadir restricciones a los valores que se pueden introducir a través de los menús desplegables. Puedes hacerlo añadiendo en el YAML varias especificaciones para cada entrada params:.

* label: es como el título para ese menú desplegable en particular
* valor: es el valor por defecto (inicial)
* entrada: establecer la selección para el menú desplegable
* opciones: Indique los valores elegibles en el menú desplegable

A continuación, estas especificaciones están escritas para el parámetro hospitalario.

---

título: Informe de vigilancia

salida: documento\_html

params:

fecha: 2021-04-10

hospital:

etiqueta: "Ciudad:"

valor: Hospital Central

entrada: seleccionar

elecciones: [Hospital Central, Hospital Militar, Hospital del Puerto, Hospital de Maternidad de San Marcos (SMMH)]

---

Al tejer (ya sea a través del botón "tejer con parámetros" o mediante render()), la ventana emergente tendrá opciones desplegables para seleccionar.

### Ejemplo parametrizado

El siguiente código crea parámetros para la fecha y el hospital, que se utilizan en el R Markdown como params$date y params$hospital, respectivamente.

En la salida del informe resultante, vea cómo los datos se filtran al hospital específico, y el título del gráfico se refiere al hospital y a la fecha correctos. En este caso utilizamos el archivo "linelist\_cleaned.rds", pero sería especialmente adecuado que la propia lista de líneas tuviera también un sello de fecha para alinearse con la fecha parametrizada.

Si se teje esto se obtiene la salida final con la fuente y el diseño por defecto.

### Parametrización sin parámetros

Si está renderizando un archivo R Markdown con render() desde un script separado, puede realmente crear el impacto de la parametrización sin usar la funcionalidad params:.

Por ejemplo, en el script de R que contiene el comando render(), puedes simplemente definir hospital y fecha como dos objetos R (valores) antes del comando render(). En el R Markdown, no sería necesario tener una sección params: en el YAML, y nos referiríamos al objeto date en lugar de params$date y a hospital en lugar de params$hospital.

Seguir este enfoque significa que no se puede "tejer con parámetros", ni utilizar la interfaz gráfica de usuario, ni incluir opciones de tejido dentro de los parámetros. Sin embargo, permite simplificar el código, lo que puede ser ventajoso.

## Informes en bucle

Es posible que queramos ejecutar un informe varias veces, variando los parámetros de entrada, para producir un informe para cada jurisdicción/unidad. Esto puede hacerse utilizando herramientas para la iteración, que se explican en detalle en la página sobre [Iteración, bucles y listas](#iteration-loops-and-lists). Las opciones incluyen el paquete **purrr**, o el uso de un bucle for como se explica a continuación.

A continuación, utilizamos un simple bucle for para generar un informe de vigilancia para todos los hospitales de interés. Esto se hace con un solo comando (en lugar de cambiar manualmente el parámetro del hospital uno por uno). El comando para generar los informes debe existir en un script separado fuera del informe Rmd. Este script también contendrá objetos definidos para "hacer un bucle" - la fecha de hoy, y un vector de nombres de hospitales para hacer un bucle.

A continuación, introducimos estos valores uno a uno en el comando render() mediante un bucle, que ejecuta el comando una vez por cada valor del vector de hospitales. La letra i representa la posición del índice (del 1 al 4) del hospital que se está utilizando en esa iteración, de modo que hospital\_list[1] sería "Hospital Central". Esta información se suministra en dos lugares en el comando render():

1. Al nombre del archivo, de forma que el nombre del archivo de la primera iteración si se produce el 10 de abril de 2021 sería "Informe-Hospital Central\_2021-04-10.docx", guardado en la subcarpeta 'output' del directorio de trabajo.
2. A params = de manera que el Rmd utiliza el nombre del hospital internamente siempre que se llame al valor params$hospital (por ejemplo, para filtrar de los datos sólo al hospital concreto). En este ejemplo, se crearían cuatro archivos, uno por cada hospital.

## Plantillas

Utilizando un documento de plantilla que contenga cualquier formato deseado, puede ajustar la estética del aspecto de la salida Rmd. Puedes crear, por ejemplo, un archivo de MS Word o Powerpoint que contenga páginas/diapositivas con las dimensiones, marcas de agua, fondos y fuentes deseadas.

### Documentos de Word

Para crear una plantilla, inicie un nuevo documento de Word (o utiliza uno ya existente con el formato que le convenga), y edite las fuentes definiendo los Estilos. En el Estilo, los encabezados 1, 2 y 3 se refieren a los distintos niveles de encabezado de markdown (# Encabezado 1, ## Encabezado 2 y ### Encabezado 3 respectivamente). Clica con el botón derecho en el estilo y clica en "modificar" para cambiar el formato de la fuente, así como el párrafo (por ejemplo, puede introducir saltos de página antes de ciertos estilos que pueden ayudar con el espaciado). Otros aspectos del documento de Word, como los márgenes, el tamaño de la página, los encabezados, etc., pueden modificarse como un documento de Word normal en el que se trabaja directamente.

### Documentos Powerpoint

Como en el caso anterior, cree un nuevo conjunto de diapositivas o utiliza un archivo PowerPoint existente con el formato deseado. Para seguir editando, clica en "Ver" y "Patrón de diapositivas". Desde aquí puede cambiar la apariencia de la diapositiva "maestra" editando el formato del texto en los cuadros de texto, así como las dimensiones del fondo/página para la página en general.

Desgraciadamente, la edición de archivos PowerPoint es algo menos flexible:

* Una cabecera de primer nivel (# Cabecera 1) se convertirá automáticamente en el título de una nueva diapositiva,
* El texto de la Cabecera 2 no aparecerá como subtítulo, sino como texto dentro del cuadro de texto principal de la diapositiva (a menos que encuentre una manera de maniplar la vista del Patrón).
* Los gráficos y las tablas resultantes irán automáticamente a nuevas diapositivas. Tendrá que combinarlos, por ejemplo con la función patchwork para combinar ggplots, para que aparezcan en la misma página. Vea esta [entrada del blog](https://mattherman.info/blog/ppt-patchwork/) sobre el uso del paquete patchwork para poner múltiples imágenes en una diapositiva.

En el [paquete **oficial**](https://davidgohel.github.io/officer/) encontrará una herramienta para trabajar más a fondo con las presentaciones de PowerPoint.

### Integración de plantillas en el YAML

Una vez preparada la plantilla, el detalle de la misma puede añadirse en el YAML del Rmd debajo de la línea 'output' y debajo de donde se especifica el tipo de documento (que va a una línea separada en sí). Ten en cuenta que reference\_doc se puede utilizar para las plantillas de diapositivas de PowerPoint.

Lo más fácil es guardar la plantilla en la misma carpeta en la que está el archivo Rmd (como en el ejemplo siguiente), o en una subcarpeta dentro de ella.

---

título: Informe de vigilancia

de salida:

palabra\_documento:

reference\_docx: "plantilla.docx"

params:

fecha: 2021-04-10

hospital: Hospital Central

plantilla:

---

### Formateo de archivos HTML

Los archivos HTML no utilizan plantillas, pero pueden tener los estilos configurados dentro del YAML. Los HTML son documentos interactivos, y son particularmente flexibles. Aquí cubrimos algunas opciones básicas.

* Tabla de contenidos: Podemos añadir una tabla de contenidos con toc: true a continuación, y también especificar que permanezca visible ("flotante") al desplazarse, con toc\_float: true.
* Temas: Podemos referirnos a algunos temas prefabricados, que provienen de una biblioteca de temas de Bootswatch. En el siguiente ejemplo utilizamos cerulean. Otras opciones son: journal, flatly, darkly, readable, spacelab, united, cosmo, lumen, paper, sandstone, simplex y yeti.
* Resaltar: Configurando esto se cambia el aspecto del texto resaltado (por ejemplo, el código dentro de los trozos que se muestran). Los estilos soportados son default, tango, pygments, kate, monochrome, espresso, zenburn, haddock, breezedark y textmate.

He aquí un ejemplo de cómo integrar las opciones anteriores en el YAML.

---

título: "Ejemplo de HTML"

de salida:

html\_documento:

toc: true

toc\_float: true

tema: cerúleo

destacar: kate

---

A continuación se muestran dos ejemplos de salidas HTML que tienen ambas tablas de contenido flotantes, pero diferentes estilos de tema y resaltado seleccionados:

## Contenido dinámico

En una salida HTML, el contenido de tu informe puede ser dinámico. A continuación, algunos ejemplos:

### Tablas

En un informe HTML, se puede imprimir un dataframe/tablo de manera que el contenido sea dinámico, con filtros y barras de desplazamiento. Hay varios paquetes que ofrecen esta capacidad.

Para hacer esto con el paquete **DT**, como se utiliza a lo largo de este manual, se puede insertar un trozo de código como este:

La función datatable() imprimirá el dataframe proporcionado como una tabla dinámica para el lector. Puedes establecer rownames = FALSE para simplificar el extremo izquierdo de la tabla. filter = "top" proporciona un filtro sobre cada columna. En el argumento option() proporciona una lista de otras especificaciones. A continuación incluimos dos: pageLength = 5 establece el número de filas que aparecen como 5 (las filas restantes se pueden ver paginando a través de flechas), y scrollX=TRUE habilita una barra de desplazamiento en la parte inferior de la tabla (para las columnas que se extienden demasiado a la derecha).

Si tu conjunto de datos es muy grande, considere la posibilidad de mostrar sólo las X filas superiores envolviendo el dataframe en head().

### Widgets HTML

Los [widgets HTML para R](http://www.htmlwidgets.org/) son un tipo especial de paquetes de R que permiten una mayor interactividad utilizando bibliotecas de JavaScript. Puedes incrustarlos en salidas HTML R Markdown.

Algunos ejemplos comunes de estos widgets son:

* Plotly (utilizado en la página de este manual y en la página de [Plots interativos])
* visNetwork (utilizado en la página de [cadenas de transmisión de](#transmission-chains) este manual)
* Folleto (utilizado en la página de [fundamentos del SIG](#gis-basics) de este manual)
* dygraphs (útil para mostrar interactivamente los datos de las series temporales)
* DT (datatable()) (se utiliza para mostrar tablas dinámicas con filtro, ordenación, etc.)

La función ggplotly() de **plotly** es particularmente fácil de usar. Consulta la página de [gráficos interactivos](#interactive-plots-1).

## Recursos

Puedes encontrar más información en:

* <https://bookdown.org/yihui/rmarkdown/>
* <https://rmarkdown.rstudio.com/articles_intro.html>

Una buena explicación de markdown vs knitr vs Rmarkdown está aquí: <https://stackoverflow.com/questions/40563479/relationship-between-r-markdown-knitr-pandoc-and-bookdown>