Yihui Xie

bookdown: Escritura de libros y documentos técnicos con R Markdown

To ...

Contents

List of Tables

List of Figures

Prefacio

Este breve libro presenta el paquete de R, **bookdown**, que cambia la forma de escribir libros. Escribir un libro debe ser técnicamente fácil, visualmente agradable a la hora de leerlo, divertido para interactuar con él, cómodo para navegar a través de sus páginas y sencillo si los lectores desean contribuir o dejar comentarios para su(s) autor(es), y lo más importante, para que los autores no se distraigan con los detalles de composición tipográfica.

El paquete **bookdown** fue construido con base en R Markdown (`http://rmarkdown.rstudio.com`), y heredó la simplicidad en la sintaxis de R Markdown (puede aprenderse lo básico en cinco minutos; véase la sección 2.1), así como la posibilidad de múltiples tipos de formatos de salida (PDF/HTML/Word/…). Asimismo, ha añadido diferentes características como la salida de múltiples páginas HTML, numeración y referenciación cruzada a figuras/tablas/secciones/ecuaciones, inserción de partes/apéndices, e importación del estilo GitBook (`https://www.gitbook.com`) para crear páginas de libros en un formato elegante y atractivo en HTML. Este libro en sí es un ejemplo de cómo se puede producir un libro de una serie de documentos R Markdown, y tanto la versión impresa como la versión en línea puede parecer profesional. Puede encontrar más ejemplos en `https://bookdown.org`.

A pesar de que el nombre del paquete contiene la palabra "libro", **bookdown** no es sólo para libros. El "libro" puede ser cualquier cosa que se componga de varios documentos R Markdown, como documentación de un curso, notas de estudio, un manual de software, una tesis de maestría/doctorado, o incluso un diario. De hecho, muchas características de **bookdown** se aplican a documentos sencillos de R Markdown (Sección 3.4).

¿Por qué leer este libro?

¿Puede escribirse un libro en un formato de origen, y generar la salida a múltiples formatos? Tradicionalmente los libros se escriben con LaTeX o Microsoft Word. Cualquiera de estas herramientas hará que escribir libros sea un viaje de ida donde no se puede dar marcha atrás: si elige LaTeX, normalmente termina sólo con un documento PDF; si se trabaja con Word, es probable que tenga que permanecer en Word para siempre, y también se pueden desperdiciar muchas de las características útiles y hermosas de la salida PDF que sí genera LaTeX.

¿Puede enfocarse en escribir el contenido del libro sin tener que preocuparse demasiado por la composición tipográfica? Parece haber una contradicción natural entre el contenido y la apariencia, y siempre tiene que equilibrarse el tiempo que se dedica a estos dos aspectos. Se quiere que el libro luzca razonablemente bonito, a la vez que quiere centrarse en su contenido. Una posibilidad es renunciar a PDF temporalmente, y a cambio obtener una vista previa bastante bella de su libro en páginas web HTML . LaTeX es una excelente herramienta de composición, pero puede ahogarse fácilmente en las numerosas órdenes de su programación y en los detalles de composición tipográfica mientras se está trabajando en el libro. Es difícil obtener una previsualización del libro en PDF, y, por desgracia, algunas veces aparecen ciertas palabras que exceden el margen de la página, algunas figuras flotan en páginas al azar, cinco o seis palabras sueltas en el final de un capítulo llevan a una página nueva, y así sucesivamente. Si el libro se va a imprimir, tendrá que hacer frente a estos problemas eventualmente, y realmente no vale la pena distraerse una y otra vez sobre estos asuntos mientras está escribiendose el libro. El hecho de que la sintaxis de Markdown es más simple y tiene menos funciones que LaTeX también ayuda a centrarse en el contenido del libro. ¿Realmente tiene que definirse un nuevo comando como `\myprecious{}` que aplica `\textbf{\textit{\textsf{}}}` a su texto? ¿La letra "R" tiene que estar encerrada en `\proglang{}` cuando los lectores pueden

fácilmente intuir que representa el lenguaje R? No hay mucha diferencia si todo o nada, necesita la atención del lector.

¿Pueden interactuar los lectores con ejemplos del libro cuando lo leen? La respuesta es no, sin duda alguna, siempre y cuando el libro esté impreso en papel, pero es posible hacerlo si su libro tiene una versión HTML que contenga ejemplos interactivos, como las aplicaciones Shiny (`https://shiny.rstudio.com`) o los HTML widgets (`https://htmlwidgets.org`). Por ejemplo, los lectores pueden saber de inmediato qué pasa si cambian algunos parámetros en un modelo estadístico.

¿Puede obtener retroalimentación e incluso contribuciones de los lectores a medida que se escribe el libro? Tradicionalmente, el editor puede encontrar un pequeño número de revisores anónimos para corregir su libro. Los revisores suelen ser útiles, pero es posible que aún así se pierda el aporte de los lectores más representativos. Es demasiado tarde después de que se imprimió la primera edición, y el lector tiene que esperar unos años antes que la segunda edición esté lista. Hay algunas plataformas web que hacen que sea fácil para las personas proporcionar retroalimentación y contribución a sus proyectos, y GitHub (`https://github.com`) es un ejemplo prominente. Si alguien encuentra un error tipográfico en su libro, puede simplemente corregirlo en línea y enviarle el cambio para su aprobación. Es una cuestión de hacer clic en un botón para fusionar el cambio, sin hacer preguntas o correos electrónicos de ida y vuelta. Para poder utilizar estas plataformas, es necesario aprender los conceptos básicos de herramientas de control de versiones como GIT, y los archivos de origen del libro deben estar en texto plano.

La combinación de R (`https://www.r-project.org`), Markdown, y Pandoc (`http://pandoc.org`) hace que sea posible pasar de un formato de fuente simple (R Markdown) a múltiples formatos de salida posibles (PDF, HTML, EPUB y Word, etc). El paquete **bookdown** está basado en R Markdown, y proporciona formatos de salida para libros y artículos de formato largo, incluyendo el formato GitBook, que es un formato de salida HTML de varias páginas con una interfaz de usuario útil y bonita. Es mucho más

fácil componer en HTML que en LaTeX, por lo que siempre se puede obtener una previsualización de su libro en HTML, y el trabajo sobre el PDF del contenido está casi hecho. Los ejemplos interactivos se pueden incorporar fácilmente en HTML, lo que puede hacer al libro más atractivo y útil. R Markdown es un formato de texto plano, lo que también puede brindar beneficios al control de versiones, como colaborar en GitHub. También se ha tratado de poner algunas de las características importantes de LaTeX a HTML y otros formatos de salida, como enumeración de figuras/tablas y referenciación cruzada.

En resumen, únicamente prepare un par de capítulos de un libro en R Markdown, y **bookdown** puede ayudarle a convertirlos en un hermoso libro.

Estructura del libro

Los capítulos 1 y 2 introducen el uso básico y la sintaxis, lo cual debería ser suficiente para que la mayoría de los lectores empiecen a escribir un libro. Los capítulos 3 y 4 son para aquellos que quieren afinar la apariencia de sus libros. Pueden parecer muy técnicos si no se está familiarizados con HTML/CSS y LaTeX. No es necesario leer estos dos capítulos con mucho cuidado la primera vez. Puede aprender lo que posiblemente puede cambiarse, y volver después para saber cómo hacerlo. Para el capítulo 5, los detalles técnicos no son importantes a menos que no utilice el RStudio IDE (Sección 5.4). Del mismo modo, puede sentirse abrumado por los comandos presentados en el capítulo 6 para publicar su libro, pero nuevamente, hemos tratado de facilitar la publicación de su libro en línea a través de la IDE de RStudio. Los comandos y funciones personalizadas son sólo para aquellos que deciden no usar el servicio de RStudio o quieren entender los detalles técnicos.

Para resumir, este libro es una referencia completa del paquete **bookdown**. Puede seguir la regla 80/20 al leerla. Algunas sec-

ciones están allí para el bien de la integridad, y no todas las secciones son igualmente útiles al libro particular que usted piensa escribir.

Información acerca del software y convenciones

Este libro es principalmente sobre el paquete **bookdown** de R, por lo que necesita instalar al menos R y el paquete **bookdown**. Sin embargo, su libro no tiene que estar relacionado en absoluto con el lenguaje R. Se pueden usar otros lenguajes de computación (C++, SQL, Python, etc.; véase el Apéndice B), e incluso puede ser totalmente irrelevante para la computación (por ejemplo, puede escribir una novela, o una colección de poemas). Las herramientas de software necesarias para construir un libro se introducen en el Apéndice A.

La información de la sesión de R al compilar este libro es la siguiente:

```
sessionInfo()
```

```
## R version 3.3.2 (2016-10-31)
## Platform: x86_64-apple-darwin13.4.0 (64-bit)
## Running under: OS X El Capitan 10.11.6
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets
## [6] base
##
## loaded via a namespace (and not attached):
## [1] knitr_1.15.1    tools_3.3.2     miniUI_0.1.1
```

```
## [4] htmltools_0.3.5 bookdown_0.3    shiny_0.14.2
## [7] rmarkdown_1.3
```

No se añaden instrucciones (`>` y `+`) al código fuente en R en este libro, y, por defecto, se comenta la salida de texto mediante dos numerales `##`, como se pudo advertir anteriormente en la información de sesión de R . Esto es para comodidad del lector a la hora que desee copiar y ejecutar el código (el texto resultante se ignorará ya que se encuentra comentado). Los nombres de paquetes están en negrita (por ejemplo, **rmarkdown**), y el código en línea y los nombres de archivos se formatean en una fuente de máquina de escribir (por ejemplo, `knitr::knit('foo.Rmd')`). Los nombres de funciones son seguidas por paréntesis (por ejemplo, `bookdown::render_book()`). El operador doble dos puntos (`::`) significa acceder a un objeto desde un paquete.

Agradecimientos

En primer lugar me gustaría dar las gracias a mi empleador, RStudio, por darme la oportunidad de trabajar en este emocionante proyecto. Tenía la esperanza de trabajar en él cuando vi por primera vez el proyecto GitBook en 2013, ya que de inmediato me di cuenta que era un estilo hermoso de libro y había mucha más potencia que podríamos agregarle, a juzgar por mi experiencia de escribir el libro sobre **knitr** (Xie, 2015) y la lectura de otros libros. R Markdown se convirtió en un proyecto maduro dos años después, y por suerte, **bookdown** se convirtió en mi trabajo oficial a finales de 2015. No hay muchas cosas en el mundo mejores que el hecho de que su trabajo pase a ser su afición (o viceversa). Estoy totalmente complacido de jugar un poco con bibliotecas de JavaScript, paquetes de LaTeX, y un sinfín de expresiones regulares en R. Honestamente también debo agradecer a StackOverflow (`http://stackoverflow.com`), y creo que todos ustedes saben [lo que quiero decir] (`http://bit.ly/2cWbiAp`), si alguna vez han escrito algún código de programa.

Este proyecto es, sin duda, no solo el esfuerzo de una sola persona. Varios colegas en RStudio me han ayudado a lo largo del camino. Hadley Wickham proporcionó una enorme cantidad de información durante el desarrollo de **bookdown**, mientras trabajaba en su libro "R for Data Science" con Garrett Grolemund. JJ Allaire y Jonathan McPherson proporcionaron una gran cantidad de ayuda técnica directamente al paquete, así como apoyo en el IDE RStudio. Jeff Allen, Chaita Chaudhari, y el equipo de RStudio Connect han mantenido el sitio web `https://bookdown.org`. Robby Shaver diseñó una imagen de portada bonita para este libro. Tareef Kawaf hizo todo lo posible para ayudarme a crecer como un ingeniero de software profesional. Es una bendición trabajar en esta empresa con personas entusiastas e inteligentes. Recuerdo que una vez le dije a Jonathan "Hey, me encontré con un problema en el almacenamiento en caché de HTML widgets y finalmente me di cuenta de una posible solución", y Jonathan agarró su cerveza, "yo ya lo resolví" "Oh, bien, muy bien."

También he recibido un montón de comentarios de los autores de libros fuera de RStudio, incluyendo Jan de Leeuw, Jenny Bryan, Dean Attali, Rafael Irizarry, Michael Love, Roger Peng, Andrew Clark, etc. Algunos usuarios también contribuyeron con código para el proyecto y ayudaron a revisar el libro. Aquí está una lista de todos los contribuyentes: `https://github.com/rstudio/bookdown/graphs/contributors`. Se siente bien cuando uno inventa una herramienta y se da cuenta que también es el beneficiario de su propia herramienta. Como alguien que ama el modelo de solicitud de extracción de GitHub, deseaba que los lectores no tuvieran que enviarme un correo electrónico para mencionar que había un error tipográfico o un error evidente en mi libro, sino que sólo pudiera arreglarse a través de una solicitud de extracción. Esto fue posible en **bookdown**. Se puede ver el número de solicitudes de extracción de los errores tipográficos que he fusionado: `https://github.com/rstudio/bookdown/pulls` Es bueno tener tantos correctores ortográficos humanos. No es que no sepa cómo usar un verdadero corrector ortográfico, pero no quiero hacer esto antes de que el libro esté terminado, y el malévolo Yihui también quiere

dejar algunas tareas sencillas para los lectores e involucrarlos en la mejora del libro.

El paquete **bookdown** no sería posible sin un par de paquetes de software de código abierto. En particular, Pandoc, GitBook, jQuery, y los paquetes de R dependientes, por no hablar de R en sí. Doy las gracias a los desarrolladores de estos paquetes.

Me mudé a Omaha, Nebraska en 2015, y disfruté de un año en apartamentos Steeplechase, donde viví con comodidad para desarrollar el paquete **bookdown**, gracias al personal del condominio que fue muy amable y servicial. Luego me encontré con un agente inmobiliario profesional y muy listo, Kevin Schaben, que encontró una casa fabulosa para nosotros en un período de tiempo sorprendentemente corto, y terminé este libro en nuestra nueva casa.

John Kimmel, el editor de Chapman & Hall/CRC, me ayudó a publicar mi primer libro. Es un placer trabajar con él de nuevo. Él generosamente accedió a que me quedara con la versión en línea de este libro de forma gratuita, por lo que puede seguirlo para actualizar después de que se imprima y publique (es decir, usted no tiene que esperar años para la segunda edición para corregir errores e introducir nuevas caracteristicas). Me gustaría poder ser lo más abierto de mente posible como él cuando tenga su edad. Shashi Kumar resolvió algunos de mis problemas técnicos con la clase del editor de LaTeX (`krantz.cls`) cuando yo estaba tratando de integrarlo con **bookdown**. También aprecio los comentarios muy útiles de los revisores Jan de Leeuw, Karl Broma n, Brooke Anderson, Michael Grayling, Daniel Kaplan y Max Kuhn.

Por último quiero agradecer a mi familia, en especial, a mi esposa e hijo, por su apoyo. Él, de un año de edad, ha descubierto que el monitor se enciende cuando toca el teclado, así que en ocasiones simplemente gatea en mi oficina y presiona al azar el teclado. No estoy seguro de si esto cuenta como su contribución al libro … @)!%)Y@*

Yihui Xie
Elkhorn, Nebraska

Acerca del Autor

Yihui Xie (http://yihui.name) es ingeniero de software en RStudio (http://www.rstudio.com). Obtuvo su PhD del Departmento de Estadística de la Iowa State University. Se interesa por los gráficos estadísticos interactivos y la computación estadística. Como un usuario activo de R, es autor de varios paquetes, tales como **knitr**, **bookdown**, **animation**, **DT**, **tufte**, **formatR**, **fun**, **mime**, **highr**, **servr**, y **Rd2roxygen**, entre los cuales el paquete **animation** ganó el premio de Software Estadístico John M. Chambers (ASA) en 2009. También es coautor de otros pocos paquetes de R, incluyendo **shiny**, **rmarkdown**, y **leaflet**.

En 2006, fundó el "Capital of Statistics" (http://cos.name), que ha crecido hasta convertirse en una gran comunidad de estadísticos en China. Inició la conferencia china de R en 2008, y ha estado involucrado en la organización de conferencias sobre R en China desde entonces. Durante su estadía en el PhD en la Iowa State University, ganó el premio de Computación Estadística Vince Sposito (2011) y el premio Snedecor (2012) en el Departmento de Estadística.

Ocasionalmente despotrica en Twitter (https://twitter.com/xieyihui), y la mayor parte del tiempo se encuentra en GitHub (https://github.com/yihui).

Adora la comida picante tanto como la literatura china clásica.

1

Introducción

Este libro es una guía para la escritura de libros con R Markdown (Allaire et al., 2016) y con el paquete **bookdown** de R (Xie, 2016a). Se enfoca en las características específicas de la escritura de libros, artículos o reportes de gran extensión . Algunas de estas características son:

- Cómo se componen figuras y tablas y se hacen referencias cruzadas hacia ellas;
- Cómo generar múltiples formatos de salida tales como HTML, PDF y e-books para un sólo libro;
- Como personalizar las plantillas de los libros y elementos de diferentes estilos en un libro;
- El soporte de editor (en particular, el IDE RStudio);
- Cómo publicar un libro;

No es una completa introducción a R Markdown o al paquete **knitr** (Xie, 2016c), sobre el cual se construyó **bookdown**. Para obtener más información sobre R Markdown, consulte la documentación en línea `http://rmarkdown.rstudio.com`. Para **knitr**, consulte Xie (2015). No hay que ser un experto en el lenguaje R (R Core Team, 2016) para leer este libro, pero se espera que se tengan conocimientos básicos sobre R Markdown y **knitr**. Para los principiantes, puede empezar a trabajar con los Cheatsheets en `https://www.rstudio.com/resources/cheatsheets/`. El apéndice de este libro contiene breves introducciones a esos paquetes de software. Para poder personalizar las plantillas de libros y temas, debe estar familiarizado con LaTeX, HTML y CSS.

1.1 Motivación

Markdown es un maravilloso lenguaje para escribir documentos relativamente simples que contengan elementos como secciones, párrafos, listas, vínculos e imágenes, etc. Pandoc `http://pandoc.org` ha extendido enormemente la sintaxis original de Markdown[1] y ha añadido unas pequeñas nuevas características tales como notas al pie de página, citas y tablas. Lo más importante que hace Pandoc es hacer posible la generación de documentos en una amplia variedad de formatos desde Markdown, incluyendo HTML, LaTeX/PDF, MSWord y Slides.

Existen unas pocas características que Pandoc aún no permite hacer si lo que se desea es escribir un documento como un libro relativamente complicado, tales como numerar automáticamente las figuras y tablas en la salida de documentos HTML, referencias cruzadas de figuras y tablas y un control óptimo de la apariencia de figuras (e.g. actualmente es imposible especificar la alineación de imágenes usando la sintaxis de Markdown). Estos son algunos de los problemas que se han intentado resolver con la librería **bookdown**.

Bajo la restricción de que queremos producir el libro en varios formatos de salida, es casi imposible cubrir todas las posibles características específicas de estos formatos. Por ejemplo, puede ser difícil recrear un determinado ambiente complicado en LaTeX en la salida HTML utilizando la sintaxis de R Markdown. El objetivo principal no es reemplazar *todo* con Markdown, sino cubrir las funcionalidades más comunes que se requieren para escribir un documento relativamente complicado, y hacer la sintaxis de dichas funcionalidades consistentes a través de todos los formatos de salida, por lo que sólo necesita aprenderse una cosa y ésta funciona bien para todos los formatos de salida.

Otro de los objetivos de este proyecto es hacer que sea fácil

[1] `http://daringfireball.net/projects/markdown/`

producir libros que parezcan visualmente agradables. Algunos buenos ejemplos existentes incluyen Gitbook `https://www.gitbook.com`, Tufte CSS `http://edwardtufte.github.io/tufte-css/`, y Tufte-LaTeX `https://tufte-latex.github.io/tufte-latex/`. Se espera integrar estos temas y estilos en **bookdown**, por lo que los autores no tienen que sumergirse en los detalles sobre cómo utilizar una cierta clase de plantilla LaTeX o cómo configurar CSS para la salida HTML.

1.2 Comenzando

El camino más fácil para que los principiantes comiencen con la escritura de un libro con R Markdown y **bookdown** es a través de la demostración `bookdown-demo` en GitHub:

1. Descargue el repositorio de GitHub `https://github.com/rstudio/bookdown-demo` como un [archivo Zip] y luego descomprímalo localmente;
2. Instale el IDE RStudio. Observe que necesita una versión superior a la 1.0.0. Por favor descargue la última versión[2] si su versión de RStudio es menor a la 1.0.0);
3. Instale el paquete de R **bookdown**:

```
# version estable en el CRAN
install.packages('bookdown')
# o la versión de desarrollo en GitHub
# devtools::install_github('rstudio/bookdown')
```

4. Abra el repositorio `bookdown-demo` que descargó en RStudio haciendo click en `bookdown-demo.Rproj`.

[2] `https://www.rstudio.com/products/rstudio/download/preview/`

5. Abra el archivo de R Markdown `index.Rmd` y haga clic en el botón `Build Book` en la pestaña `Build` de RStudio;

Ahora debería ver la página de índice de ese libro de demostración en el visor de RStudio. Puede agregar o cambiar los archivos de R Markdown, volver a `index.Rmd`, y pulsar el botón `Knit` de nuevo para obtener una vista previa del libro. Si prefiere no utilizar RStudio, también puede compilar el libro a través de línea de comandos. Consulte la siguiente sección para más detalles.

Aunque se ve un buen número de archivos en el ejemplo `bookdown-demo`, la mayoría de ellos no son esenciales para un libro. Si se siente abrumado por el número de archivos, puede utilizar este mínimo ejemplo, en lugar del otro, que es esencialmente un archivo `index.Rmd`: `https://github.com/yihui/bookdown-minimal` El ejemplo `bookdown-demo` contiene algunas configuraciones avanzadas que es posible que desee aprender más adelante, como la forma de personalizar el preámbulo de LaTeX, modificar el CSS, y construir el libro en GitHub, etc.

1.3 Uso

Normalmente, un libro contiene varios capítulos, y un capítulo está contenido dentro de un archivo de R Markdown, con la extensión de archivo `.Rmd`. Cada archivo de R Markdown debe comenzar inmediatamente con el título del capítulo usando el encabezado de primer nivel, por ejemplo, `# Título del Capítulo`. Todos los archivos R Markdown deben ser codificados en UTF-8. Aquí hay un ejemplo (las viñetas son los nombres de archivo, seguidos por el contenido del archivo):

- index.Rmd

```
# Prefacio {-}

En este libro presentamos un método interesante.
```

- 01-intro.Rmd

```
# Introducción

Este capítulo es un panorama de los métodos con que proponemos resolver un  **problema impor
```

- 02-literatura.Rmd

```
# Literatura

Aquí hay una revisión de los métodos existentes.
```

- 03-metodos.Rmd

```
# Métodos

Describimos nuestros métodos en este capítulo.
```

- 04-aplicacion.Rmd

```
# Aplicaciones

Algunos aplicaciones _significativas_ se demuestran
en este capítulo.

## Ejemplo uno

## Ejemplo dos
```

- 05-resumen.Rmd

```
# Palabras finales

Hemos finalizado un bello libro.
```

De forma predeterminada, **bookdown** fusiona todos los archivos Rmd por el orden de los nombres de archivo, por ejemplo, `01-intro.Rmd` aparecerá antes de `02-literatura.Rmd`. Los nombres de archivo que comiencen con un guión bajo `_` se omiten. Si existe un archivo Rmd llamado `index.Rmd`, siempre va a ser tratado como el primer archivo cuando se fusionen todos los archivos Rmd. La razón de este tratamiento especial es que el archivo HTML `index.html` se genera a partir de`index.Rmd`. Este `index.html` suele ser el archivo de índice por defecto cuando se ve una página web, por ejemplo, usted está viendo `http://yihui.name/index.html` cuando se abre `http://yihui.name/`.

Puede anular el comportamiento anterior mediante la inclusión de un archivo de configuración llamado `_bookdown.yml` en el directorio del libro. Este es un archivo YAML (`https://en.wikipedia.org/wiki/YAML`), y los usuarios de R Markdown deben estar familiarizados con este formato, ya que también se utiliza para escribir los metadatos al comienzo de los documentos R Markdown (puede aprender más acerca de YAML en la Sección B.2). Puede utilizar un campo llamado `rmd_files` para definir su propia lista y el orden de los archivos Rmd para el libro. Por ejemplo,

```
rmd_files: ["index.Rmd", "abstract.Rmd", "intro.Rmd"]
```

En este caso, **bookdown** sólo tiene que utilizar lo que se ha definido en este campo YAML sin tratamientos especiales de `index.Rmd` o guiones bajos. Si desea salidas HTML y LaTeX/PDF del libro, y usar diferentes archivos Rmd para la salida HTML y LaTeX, puede especificar estos archivos para los dos formatos de salida por separado, por ejemplo,

```
rmd_files:
  html: ["index.Rmd", "abstract.Rmd", "intro.Rmd"]
  latex: ["abstract.Rmd", "intro.Rmd"]
```

A pesar de que hemos estado hablando acerca de los archivos de R Markdown, los archivos de capítulos en realidad no tienen que ser R Markdown. Pueden ser archivos sin formato Markdown (`.md`), y no tienen que contener trozos de código R (en adelante *chunks*) en absoluto. ¡Por supuesto que puede utilizar **bookdown** para componer novelas o poemas!

Por el momento, los principales formatos de salida que se pueden utilizar incluyen `bookdown::pdf_book`,`bookdown::gitbook`, `bookdown::html_book`, y `bookdown::epub_book`. Hay una función `bookdown::render_book()` similar a `rmarkdown::render()`, pero fue diseñada para hacer *múltiples* documentos Rmd en un libro utilizando las funciones de formato de salida. Es posible que se quiera llamar a esta función desde la línea de comandos directamente, o hacer clic en los botones correspondientes en el IDE RStudio. Estos son algunos ejemplos de línea de comandos:

```
bookdown::render_book('foo.Rmd', 'bookdown::gitbook')
bookdown::render_book('foo.Rmd', 'bookdown::pdf_book')
bookdown::render_book('foo.Rmd', bookdown::gitbook(lib_dir = 'libs'))
bookdown::render_book('foo.Rmd', bookdown::pdf_book(keep_tex = TRUE))
```

Para utilizar las funciones `render_book` y el formato de salida en el IDE RStudio, se puede definir un campo YAML llamado `site` que toma el valor `bookdown::bookdown_site` [3], y las funciones de formato de salida se pueden utilizar en el campo `output`, por ejemplo,

```
---
site: "bookdown::bookdown_site"
output:
```

[3]Esta función llama a `bookdown::render_book()`.

```
  bookdown::gitbook:
    lib_dir: "book_assets"
  bookdown::pdf_book:
    keep_tex: yes
---
```

A continuación, puede hacer clic en el botón `Build Book` en el panel `Build` en RStudio para compilar los archivos Rmd en un libro, o haga clic en el botón `Knit` en la barra de herramientas para obtener una vista previa del capítulo actual.

Más opciones de configuración **bookdown** en `_bookdown.yml` se explican en la sección 4.4. Además de estas configuraciones, también puede especificar otras relacionadas con Pandoc en los metadatos YAML del primer archivo Rmd del libro, como el título, autor y fecha del libro, etc. Por ejemplo:

```
---
title: "Creando un libro con R Markdown"
author: "Yihui Xie"
date: "`r Sys.Date()`"
site: "bookdown::bookdown_site"
output:
  bookdown::gitbook: default
documentclass: book
bibliography: ["book.bib", "packages.bib"]
biblio-style: apalike
link-citations: yes
---
```

1.4 Dos enfoques de representación

Fusionar todos los capítulos en un archivo Rmd y compilarlo es una manera de hacer el libro en **bookdown**. Actualmente, existe otra forma: es posible compilar cada capítulo en una sesión de R *separada*, y **bookdown** combinará los Markdown de todos los capítulos para hacer el libro. A estos dos enfoques se les llama "Merge and knit" (MK) y "knit and Merge" (KM), respectivamente. Las diferencias entre ellos parecen sutiles, pero pueden ser bastante importantes dependiendo de sus casos de uso.

- La diferencia más significativa es que el MK corre *todos* los trozos de código de todos los capítulos en la misma sesión de R, mientras que K-M utiliza sesiones de R separadas para los distintos capítulos. Para M-K, el estado de la sesión de R de los capítulos anteriores se lleva a capítulos posteriores (por ejemplo, los objetos creados en los capítulos anteriores están disponibles para los capítulos siguientes, a menos que se les elimine deliberadamente); para K-M, todos los capítulos están aislados unos de otros[4]. Si desea que cada capítulo se compile desde un estado limpio, utilice el enfoque K-M. Puede ser muy complicado y difícil restaurar una sesión de R corriendo a un estado completamente limpio si se utiliza el enfoque de M-K. Por ejemplo, incluso si desea desvincular/descargar paquetes previamente cargados en un capítulo previo, R no va a limpiar los métodos S3 registrados por los paquetes.
- Debido a que **knitr** no permite etiquetas de chunk duplicadas en un documento de origen, debe asegurarse que no haya etiquetas duplicadas en sus capítulos del libro cuando se utilice el enfoque M-K, de lo contrario **knitr** señalará un error cuando compile el archivo Rmd fusionado. Tenga en cuenta que esto significa que no debe haber etiquetas duplicadas a lo largo de todo el libro. El

[4]Obviamente, nadie puede dejar de escribir algunos archivos en un capítulo, y leerlos en otro capítulo. Es difícil aislar este tipo de efectos colaterales

enfoque K-M requiere no tener etiquetas duplicadas dentro de cualquier archivo sencillo Rmd.
- El método K-M no permite que los archivos Rmd estén en subdirectorios, mientras M-K sí.

El método por defecto en **bookdown** es M-K. Para cambiar a K-M, use o bien el argumento `new_session = TRUE` al llamar `render_book()`, o establezca `new_session: yes` en el archivo de configuración `_bookdown.yml`.

Se puede configurar `book_filename` en `_bookdown.yml` para el enfoque K-M, pero debe ser un nombre de archivo Markdown, por ejemplo, `_main.md`, aunque la extensión del archivo no importa realmente, e incluso se puede dejar de lado la extensión, por ejemplo, establezca `book_filename: _main`. Todas las demás configuraciones funcionan tanto para M-K como para K-M.

1.5 Algunos tips

Componer textos bajo la restricción de paginación (por ejemplo, para salidas LaTeX/PDF) puede ser un trabajo muy tedioso y que consume mucho tiempo. Se recomienda no ver la salida de PDF frecuentemente, ya que la mayoría de las veces es muy poco probable que quede satisfecho: el texto puede desbordar el margen de la página, las figuras pueden ubicarse demasiado lejos de donde se pensó inicialmente, etc. No trate de hacer que las cosas se vean perfectas *immediatamente*, porque puede decepcionarse una y otra vez al estar revisando el libro, y las cosas pueden desorganizarse, incluso si sólo hizo algunos cambios menores (véase `http://bit.ly/tbrLtx` para una mejor ilustración).

Si desea obtener una previsualización del libro, visualice la salida HTML. Trabaje en el libro en PDF después de haber terminado el contenido del libro, y seguramente no serán necesarias revisiones importantes.

Si ciertos chunks en los documentos de R Markdown requieren mucho tiempo para funcionar, es posible almacenarlo en caché mediante la adición de la opción `cache = TRUE` en el encabezado del chunk, y se recomienda etiquetar dichos chunks, así,

```
```{r important-computing, cache=TRUE}
```

Se hablará acerca de cómo previsualizar rápidamente los libros que se mantengan en edición en el capítulo 5. En resumen, se puede utilizar la función `preview_chapter()` para compilar un solo capítulo en lugar de todo el libro. La función `serve_book ()` hace que sea fácil previsualizar en tiempo real páginas del libro en HTML: cada vez que se modifica un archivo Rmd, el libro puede volver a compilarse y, en consecuencia, el navegador se puede actualizar automáticamente.
```

2

Componentes

En este capítulo, se muestra la sintaxis de algunos componentes básicos de un libro, incluyendo el código en R, figuras, tablas, citas, y demás. Primero empezamos con la sintaxis de Markdown Pandoc.

2.1 Sintaxis de Markdown

En esta sección se da una breve introducción a Markdown de Pandoc. Los lectores que estén familiarizados con Markdown pueden omitir esta sección. La sintaxis completa de Markdown de Pandoc se puede encontrar en el sitio web de Pandoc `http://pandoc.org`.

2.1.1 Formateo en línea

Puede producir texto en *cursiva* rodeándolo con guiones o asteriscos, por ejemplo, `_texto_` o `*texto*`. Para texto en **negrita**, utilice dos guiones bajos (`__texto__`) o asteriscos (`**texto**`). El texto rodeado por `~` se convertirá en un subíndice (por ejemplo, `H~2~SO~4~` muestra H_2SO_4), y de manera similar, dos signos de intercalación `^` producen un superíndice (por ejemplo, `ClO^-^` muestra ClO^-). Para marcar el texto como `código en línea`, utilice un par de acentos abiertos, por ejemplo, `` `code` ``[1]. Las versal-

[1]Para incluir acentos abiertos literales, sólo tiene que utilizar más acentos abiertos en el exterior, por ejemplo, se pueden utilizar dos comillas sencillas para preservar un acento grave en el interior: `` ` `code` ` ``.

itas se pueden producir con la etiqueta HTML `span`, por ejemplo, `<span style="font-variant:small-caps;">Versalitas</span>` muestra VERSALITAS. Los enlaces se crean usando `[texto](link)`, por ejemplo, `[RStudio](http://www.rstudio.com)`, y la sintaxis de las imágenes es similar: sólo tiene que añadir un signo de exclamación, por ejemplo, `![título de la imagen](ruta)`. Las notas al pie se colocan dentro de corchetes cuadrados después de un acento circunflejo `^[]`, por ejemplo,`^[Esta es una nota al pie.]`. Se hablará de las citas en la sección 2.8.

2.1.2 Elementos de nivel bloque

Los encabezados de sección se pueden escribir después de una serie de símbolos de numeral, por ejemplo,

```
# Encabezado de primer nivel

## Encabezado de segundo nivel

### Encabezado de tercer nivel
```

Si no desea que un determinado encabezado sea numerado, puede agregar `{-}` después del encabezado, por ejemplo,

```
# Prefacio {-}
```

Los elementos de la lista no ordenada comienzan con `*` `-`, o `+`, y se puede anidar una lista dentro de otra lista sangrando la sub-lista con cuatro espacios, por ejemplo,

```
- un item
- un item
- un item
    - un item
    - un item
```

La salida es:

- un item
- un item
- un item
    - un item
    - un item

Los elementos de la lista ordenada comienzan con números (la regla para las listas anidadas es el mismo que en el anterior), por ejemplo,

```
1. primer item
2. segundo item
3. tercer item
```

La salida no se ve demasiado diferente que en Markdown:

1. primer item
2. segundo item
3. tercero item

Un párrafo citado se escribe después de >, por ejemplo:

```
> "I thoroughly disapprove of duels. If a man should challenge me,
  I would take him kindly and forgivingly by the hand and lead him
  to a quiet place and kill him."
>
> --- Mark Twain
```

Y la salida es (se personalizó el estilo de las comillas en este libro):

> "I thoroughly disapprove of duels. If a man should challenge me, I would take him kindly and forgivingly by the hand and lead him to a quiet place and kill him."

— Mark Twain

Los bloques de código sin formato se pueden escribir después de tres o más acentos abiertos, y también se puede aplicar sangría a los bloques de cuatro espacios, por ejemplo,

```
```
Este texto se muestra verbatim
```


O sangrándolo con cuatro espacios:

    Este texto se muestra verbatim
```

2.1.3 Expresiones matemáticas en LaTeX

Las ecuaciones en LaTeX se pueden escribir dentro de un par de signos peso usando la sintaxis de LaTeX, por ejemplo `$f(k) = {n \choose k} p^{k} (1-p)^{n-k}$` (cuya salida es: $f(k) = \binom{n}{k} p^k (1-p)^{n-k}$); las expresiones matemáticas también pueden centrarse en la página si se ponen entre dobles signos peso, por ejemplo: `$$f(k) = {n \choose k} p^{k} (1-p)^{n-k}$$`, cuya salida se vería como:

$$f\left(k\right) = \binom{n}{k} p^k \left(1-p\right)^{n-k}$$

También puede utilizar entornos matemáticos dentro de `$ $` o `$$ $$`, por ejemplo,

```
$$\begin{array}{ccc}
x_{11} & x_{12} & x_{13}\\
x_{21} & x_{22} & x_{23}
\end{array}$$
```

$$\begin{matrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \end{matrix}$$

```
$$X = \begin{bmatrix}1 & x_{1}\\
1 & x_{2}\\
1 & x_{3}
\end{bmatrix}$$
```

$$X = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \end{bmatrix}$$

```
$$\begin{vmatrix}a & b\\
c & d
\end{vmatrix}=ad-bc$$
```

$$\begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc$$

2.2 Extensiones de Markdown en bookdown

A pesar de que Pandoc de Markdown es mucho más rico que la sintaxis de Markdown original, aún existe una serie de cosas que podemos necesitar para la escritura académica. Por ejemplo, es compatible con ecuaciones matemáticas, pero no se puede numerar y referenciar ecuaciones en varias páginas HTML o en salida EPUB. Se ha proporcionado un par de extensiones de Markdown en **bookdown** para llenar estos vacíos.

2.2.1 Numerar y referenciar ecuaciones

Para numerar y referenciar ecuaciones, póngalas dentro de entornos de ecuaciones y asígneles etiquetas mediante la sintaxis `(\#eq:label)`, por ejemplo,

```
\begin{equation}
  f\left(k\right) = \binom{n}{k} p^k\left(1-p\right)^{n-k}
  \label{eq:binom}
\end{equation}
```

Esto muestra la siguiente ecuación:

$$f\left(k\right) = \binom{n}{k} p^k\left(1-p\right)^{n-k} \tag{2.1}$$

Puede hacer referencia a la ecuación mediante `\@ref(eq:binom)`, por ejemplo, véase la ecuación (2.1).

Las etiquetas de una ecuación deben comenzar con el prefijo `eq:` en **bookdown**. Las referencias a una ecuación funcionan mejor para la producción de LaTeX/PDF, y no están bien soportadas en la producción de Word o libros electrónicos. Para la salida HTML, **bookdown** sólo puede numerar las ecuaciones con etiquetas. Por favor asegúrese de que las ecuaciones sin etiquetas no estén numeradas ya sea usando el entorno `equation*` o adicionando `\nonumber` o `\notag` a sus ecuaciones. Las mismas reglas se aplican a otros entornos de matemáticas, tales como `eqnarray`,`gather`, `align`, etc. (por ejemplo, se puede utilizar el entorno `align*`).

Demostramos unos entornos más de ecuaciones matemáticas abajo. Aquí está una ecuación sin numerar utilizando el entorno `equation*`:

```
\begin{equation*}
```

```
\frac{d}{dx}\left( \int_{a}^{x} f(u)\,du\right)=f(x)
\end{equation*}
```

$$\frac{d}{dx}\left(\int_{a}^{x} f(u)\,du\right)=f(x)$$

Abajo hay un entorno `align` (2.2):

```
\begin{align}
g(X_{n}) &= g(\theta)+g'({\tilde{\theta}})(X_{n}-\theta) \notag \\
\sqrt{n}[g(X_{n})-g(\theta)] &= g'\left({\tilde{\theta}}\right)
  \sqrt{n}[X_{n}-\theta ] (\#eq:align)
\end{align}
```

$$\begin{aligned} g(X_n) &= g(\theta) + g'(\tilde{\theta})(X_n - \theta) \\ \sqrt{n}[g(X_n) - g(\theta)] &= g'\left(\tilde{\theta}\right)\sqrt{n}[X_n - \theta] \end{aligned} \tag{2.2}$$

El entorno `split` dentro de `equation` de manera que todas las líneas comparten el mismo número (2.3). Por defecto, cada línea en el entorno `align` se le asignará un número de ecuación. Suprimimos el número de la primera línea en el ejemplo anterior usando `\notag`. En este ejemplo, a todo el entorno `split` se le asignó un único número.

```
\begin{equation}
\begin{split}
\mathrm{Var}(\hat{\beta}) & =\mathrm{Var}((X'X)^{-1}X'y)\\
 & =(X'X)^{-1}X'\mathrm{Var}(y)((X'X)^{-1}X')'\\
 & =(X'X)^{-1}X'\mathrm{Var}(y)X(X'X)^{-1}\\
 & =(X'X)^{-1}\sigma^{2}
\end{split}
(\#eq:var-beta)
\end{equation}
```

TABLE 2.1: Entornos de teoremas en **bookdown**.

Environment	Printed Name	Label Prefix
theorem	Theorem	thm
lemma	Lemma	lem
definition	Definition	def
corollary	Corollary	cor
proposition	Proposition	prp
example	Example	ex

$$\begin{aligned} \text{Var}(\hat{\beta}) &= \text{Var}((X'X)^{-1}X'y) \\ &= (X'X)^{-1}X'\text{Var}(y)((X'X)^{-1}X')' \\ &= (X'X)^{-1}X'\text{Var}(y)X(X'X)^{-1} \\ &= (X'X)^{-1}\sigma^2 \end{aligned} \tag{2.3}$$

2.2.2 Teoremas y demostraciones

Los teoremas y las demostraciones se utilizan comúnmente en artículos y libros en matemáticas. Sin embargo, no se deje engañar por los nombres: un "teorema" es sólo un entorno numerado/etiquetado, y no tiene que ser un teorema matemático (por ejemplo, puede ser un ejemplo irrelevante para las matemáticas). Del mismo modo, una "demostración" es un entorno no numerado. En esta sección, siempre usamos los significados generales de un "teorema" y "demostración" a menos que se indique explícitamente.

En **bookdown**, los tipos de entornos de teoremas soportados están en la tabla 2.1. Para escribir un teorema, puede usar la siguiente sintaxis:

````
```{theorem}
Aquí está mi teorema.
```
````

Para escribir otros entornos de teoremas, reemplace ```` ```{theorem} ```` por otros nombres de entornos en la tabla 2.1, por ejemplo, ```` ```{lemma} ````.

Un teorema puede tener una opción `name` con lo cual su nombre se imprimirá, por ejemplo,

````
```{theorem, name="Teorema de Pitágoras"}
Para un triángulo rectángulo, si $c$ denota la longitud de la hipotenusa
y $a$ y $b$ denotan las longitudes de los otros dos lados, se tiene que
$$a^2 + b^2 = c^2$$
```
````

Si desea referirse a un teorema, debe etiquetarlo. La etiqueta puede escribirse después de ```` ```{theorem} ````, por ejemplo,

````
```{theorem, label="foo"}
Un teorema etiquetado aquí.
```
````

La opción `label` puede ser implícita, por ejemplo, el siguiente teorema tiene la etiqueta `bar`:

````
```{theorem, bar}
Un teorema etiquetado aquí
```
````

Después de etiquetar un teorema, puede referirse a él usando la sintaxis `\@ref(prefix:label)`. Consulte la columna `Label Prefix` en la tabla 2.1 para el valor de `prefix` en cada entorno. Por ejemplo, tenemos un teorema etiquetado y llamado a continuación, y `\@ref(thm:pyth)` nos da su número de teorema 2.1:

````
```{theorem, pyth, name="Pythagorean theorem"}
Para un triángulo rectángulo, si $c$ denota la longitud de la hipotenusa
y $a$ y $b$ denotan las longitudes de los otros dos lados, se tiene que
````
````

```
$$a^2 + b^2 = c^2$$
```

Teorema 2.1 (Teorema de Pitágoras)**.** *Para un triángulo rectángulo, si c denota la longitud de la hipotenusa y a y b denotan las longitudes de los otros dos lados, se tiene que*

$$a^2 + b^2 = c^2$$

Los entornos de demostración que actualmente se soportan `proof` and `remark`. La sintaxis es similar a los entornos de teoremas, y los entornos de demostración también pueden ser nombrados. La única diferencia es que, puesto que no tienen número, no se puede hacer referencia a ellos.

Hemos tratado de hacer que todos estos entornos de teorema y demostración funcionen fuera de lo normal, no importa si su salida es PDF, HTML o EPUB. Si es un experto en LaTeX o HTML, quizá desee personalizar el estilo de estos entornos de todos modos (consulte el capítulo 4). La personalización en HTML es fácil con CSS, y cada entorno está encerrado en `<div></div>` con la clase CSS siendo el nombre del entorno, por ejemplo, `<div class="lemma"></div>`. Para la salida de LaTeX, hemos predefinido el estilo para que sea `definición` para entornos `definition` and `example` y `remark` para entornos `proof` and `remark`. Todos los demás entornos usan el estilo `plain`. La definición de estilo se realiza a través del comando `\theoremstyle{}` del paquete **amsthm**.

Por defecto, los teoremas están numerados por capítulos. Si no hay capítulos en su documento, en su lugar se numeran por secciones. Si todo el documento no tiene número (la opción de formato de salida `number_sections = FALSE`), todos los teoremas están numerados secuencialmente de 1, 2, …, N. LaTeX soporta numeración de un entorno de teorema después de otro, por ejemplo, teoremas y lemas comparten el mismo contador. Esto no es compatible con la salida HTML/EPUB en **bookdown**. Puede cambiar el esquema

de numeración en el preámbulo de LaTeX definiendo sus propios entornos de teorema, por ejemplo,

```
\newtheorem{theorem}{Theorem}
\newtheorem{lemma}[theorem]{Lemma}
```

Cuando **bookdown** detecta `\newtheorem{theorem}` en su preámbulo de LaTeX, no escribirá sus definiciones de teorema por defecto, lo que significa que usted tiene que definir todos los entornos de teorema por su propia cuenta. Por razones de simplicidad y consistencia, no recomendamos que lo haga. Puede ser confuso cuando el Teorema 18 en PDF se convierte en el Teorema 2.4 en HTML.

Los entornos de teorema y demostración quedarán ocultos si la opción de bloqueo `echo` se establece en `FALSE`. Para asegurarse de que siempre se muestren, puede agregar la opción de bloqueo `echo = TRUE`, por ejemplo,

````
```{theorem, echo=TRUE}
Aquí está mi teorema.
```
````

A continuación, mostramos más ejemplos [2] de los entornos teorema y demostración, para que pueda ver los estilos por defecto en **bookdown**.

Definición 2.1. La función característica de una variable aleatoria X está definida por

$$\varphi_X(t) = \mathrm{E}\left[e^{itX}\right],\ t \in \mathcal{R}$$

Ejemplo 2.1. Derivamos la función característica de $X \sim U(0,1)$ con la función de densidad de probabilidad $f(x) = \mathbf{1}_{x\in[0,1]}$.

[2] Algunos ejemplos se adaptan de la página de Wikipedia `https://en.wikipedia.org/wiki/Caracteristic_function_(probability_theory)`

$$\begin{aligned}
\varphi_X(t) &= \mathrm{E}\left[e^{itX}\right] \\
&= \int e^{itx} f(x) dx \\
&= \int_0^1 e^{itx} dx \\
&= \int_0^1 \left(\cos(tx) + i\sin(tx)\right) dx \\
&= \left(\frac{\sin(tx)}{t} - i\frac{\cos(tx)}{t}\right)\Bigg|_0^1 \\
&= \frac{\sin(t)}{t} - i\left(\frac{\cos(t)-1}{t}\right) \\
&= \frac{i\sin(t)}{it} + \frac{\cos(t)-1}{it} \\
&= \frac{e^{it}-1}{it}
\end{aligned}$$

Note que usamos el hecho $e^{ix} = \cos(x) + i\sin(x)$ dos veces.

Lema 2.1. *Para dos variables aleatorias cualquiera X_1, X_2, ambas tienen la misma distribución de probabilidad si y solo si*

$$\varphi_{X_1}(t) = \varphi_{X_2}(t)$$

Teorema 2.2. *Si X_1, …, X_n son variables aleatorias independientes, y a_1, …, a_n son constantes, entonces la función característica de la combinación lineal $S_n = \sum_{i=1}^n a_i X_i$ es*

$$\varphi_{S_n}(t) = \prod_{i=1}^n \varphi_{X_i}(a_i t) = \varphi_{X_1}(a_1 t) \cdots \varphi_{X_n}(a_n t)$$

Proposición 2.1. La distribución de la suma de variables aleatorias independientes Poisson $X_i \sim \mathrm{Pois}(\lambda_i)$, $i = 1, 2, \cdots, n$ es $\mathrm{Pois}(\sum_{i=1}^n \lambda_i)$.

Proof. La función característica de $X \sim \mathrm{Pois}(\lambda)$ es $\varphi_X(t) = e^{\lambda(e^{it}-1)}$. Sea $P_n = \sum_{i=1}^n X_i$. Se sabe del teorema 2.2 que

$$\begin{aligned}\varphi_{P_n}(t) &= \prod_{i=1}^{n} \varphi_{X_i}(t) \\ &= \prod_{i=1}^{n} e^{\lambda_i(e^{it}-1)} \\ &= e^{\sum_{i=1}^{n} \lambda_i(e^{it}-1)}\end{aligned}$$

Esta es la función característica de una variable aleatoria Poisson con parámetro $\lambda = \sum_{i=1}^{n} \lambda_i$. Del Lemma 2.1, se sabe que la distribución de P_n es Pois$(\sum_{i=1}^{n} \lambda_i)$. □

Observación. En algunos casos, es muy conveniente y fácil calcular la distribución de la suma de variables aleatorias independientes usando funciones características.

Corolario 2.1. La función característica de la suma de dos variables aleatorias independientes X_1 y X_2 es el producto de las funciones características de X_1 y X_2, por ejemplo,

$$\varphi_{X_1+X_2}(t) = \varphi_{X_1}(t)\varphi_{X_2}(t)$$

2.2.3 Encabezados especiales

Hay unos cuantos tipos especiales de encabezados de primer nivel que serán procesados de forma diferente en **bookdown**. El primer tipo es un enzacebzado sin numerar que se inicia con el token `(PART)`. Este tipo de encabezados se traducen en títulos de parte. Si está familiarizado con LaTeX, esto significa básicamente `\part{}`. Cuando su libro tenga un gran número de capítulos, es posible que desee organizarlos en partes, por ejemplo,

```
# (PART) Parte I {-}

# Capítulo Uno

# Capítulo Dos

# (PART) Parte II {-}
```

```
# Capítulo Tres
```

El segundo tipo es un encabezado sin numeración que comienza con `(APPENDIX)`, lo que indica que todos los capítulos después de este encabezado son apéndices , por ejemplo,

```
# Capítulo Uno

# Capítulo Dos

# (APPENDIX) Apéndice {-}

# Apéndice A

# Apéndice B
```

El estilo de numeración de los apéndices se cambiará automáticamente en la salida LaTeX/PDF y HTML (por lo general de la forma A, A.1, A.2, B, B.1, …). Esta función no está disponible para libros electrónicos o de salida Word

2.2.4 Referencias de textos

Se puede asignar un texto a una etiqueta y hacer referencia al texto usando la etiqueta en otro lugar del documento. Esto puede ser particularmente útil para títulos largos de figuras/cuadros (sección 2.4 y 2.5), en cuyo caso normalmente tiene que escribir toda la cadena de caracteres en el encabezado del chunk (por ejemplo, `fig.cap = "Un título de figura larguísimo"`) o su código en R (por ejemplo, `kable(caption = "Un título de tabla larguísimo.")`). También es útil cuando estos títulos contienen caracteres especiales HTML o LaTeX, por ejemplo, si el pie de figura contiene un guión bajo, que funciona en la salida HTML, pero no pueden trabajar en la producción de LaTeX ya que el guión bajo debe ser omitido en LaTeX.

La sintaxis de una referencia de texto es `(ref:label)text`, donde

`label` es una etiqueta única [3] en todo el documento para `text`. Debe estar en un párrafo separado mediante líneas vacías por encima y por debajo de ella. Por ejemplo,

```
(ref:foo) Defina una referencia de texto **aquí**.
```

A continuación, puede usar `(ref: foo)` en sus pies de figura/tabla. El texto puede contener cualquier cosa que Markdown soporte, siempre y cuando sea un solo párrafo. Aquí hay un ejemplo completo:

````
Un párrafo normal.

(ref:foo) Un diagrama de dispersión de los datos `cars` usando las gráficas de la **base** de

```{r foo, fig.cap='(ref:foo)'}
plot(cars)  # a scatterplot
```
````

2.3 Código en R

Hay dos tipos de código en R en documentos R Markdown/**knitr**: chunks de R y código en línea R. La sintaxis de este último es `` `r R_CODE` ``, y puede ser integrado en línea con otros elementos del documento. Los chunks de R parecen bloques de código plano, pero tienen `{r}` después de los tres acentos abiertos y (opcionalmente) parámetros dentro del `{}`, por ejemplo,

````
```{r chunk-label, echo = FALSE, fig.cap = 'A figure caption.'}
1 + 1
rnorm(10)  # 10 random numbers
plot(dist ~ speed, cars)  # a scatterplot
```
````

[3] Usted puede considerar el uso de las etiquetas de chunk.

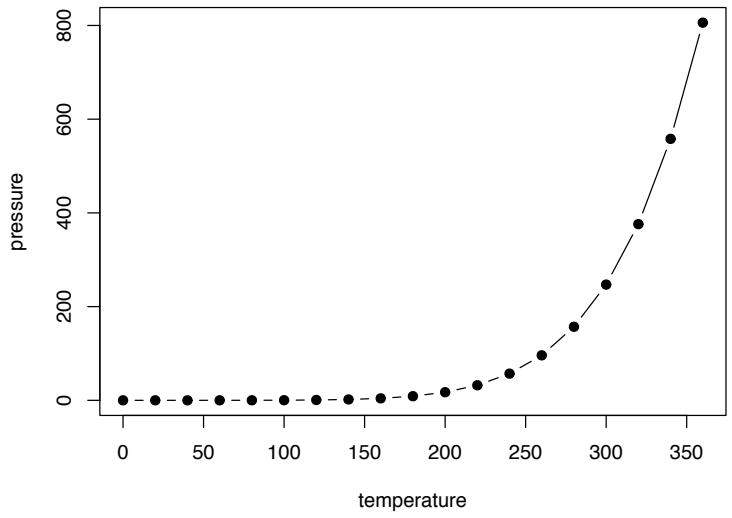
```

Para obtener más información sobre las opciones de trozos de código **knitr**, consulte Xie (2015) o `http://yihui.name/knitr/options`. Para libros, código adicional en R puede ser ejecutado antes/después de cada capítulo; véase `before_chapter_script` y `after_chapter_script` en la sección 4.4.

## 2.4 Figuras

Por defecto, las figuras no tienen títulos en la salida generada por **knitr**, lo que significa que solo pueden colocarse en el lugar en el que se generaron en el código en R. A continuación se muestra un ejemplo de ello.

```
par(mar = c(4, 4, .1, .1))
plot(pressure, pch = 19, type = 'b')
```




La desventaja de realizar figuras de esta manera es que cuando no hay suficiente espacio en la página actual para colocar una figura, puede o bien llegar a la parte inferior de la página (por lo tanto ex-
```

cede el margen de la página), o ser empujada a la página siguiente, dejando un gran margen blanco en la parte inferior de la página actual. Esto es, básicamente, porque hay "ambientes flotantes" en LaTeX: los elementos que no se pueden dividir en varias páginas (como las figuras) se ponen en entornos flotantes, para que puedan deambular a una página que tenga espacio suficiente para sostenerlos. Sin embargo, claramente es una desventaja el hecho de que floten las cosas hacia adelante o hacia atrás. Es decir, los lectores pueden tener que saltar a una página diferente para encontrar la figura mencionada en la página actual. Esto es simplemente una consecuencia natural de tener que componer las cosas en varias páginas de tamaños fijos. Este problema no existe en HTML ya que todo se puede colocar de forma continua en una sola página (presumiblemente de altura infinita), y no hay necesidad de dividir nada a través de múltiples páginas del mismo tamaño de página.

Si se asigna un pie de figura a un chunk a través de la opción `fig.cap`, los gráficos en R se pondrán dentro de entornos figura, que serán etiquetados y numerados de forma automática, y también pueden usarse como una referencia cruzada. La etiqueta de un entorno figura se genera a partir de la etiqueta del chunk, por ejemplo, si la etiqueta es `foo`, la etiqueta de la figura será`fig: foo` (el prefijo `fig` se añade antes de `foo`). Para hacer referencia a una figura, utilice la sintaxis `\@ref(etiqueta)` [4], donde `label` es la etiqueta de la figura, por ejemplo,`fig: foo`.

Para aprovechar el formato Markdown en el subtítulo de la figura, necesitará usar referencias de texto (consulte la sección @ref (referencias-de-texto)). Por ejemplo, una leyenda de la figura que contiene `_texto en bastardilla_` no funcionará cuando el formato de salida sea LaTeX/PDF, ya que el carácter de subrayado es un carácter especial en LaTeX, pero si utiliza referencias de texto,`_texto en bastardilla_` será traducido a código LaTeX cuando la salida sea LaTeX.

[4] No olvide de la barra invertida inicial! y también note el paréntesis `()` después de `ref`; no hay corchetes `{}` .

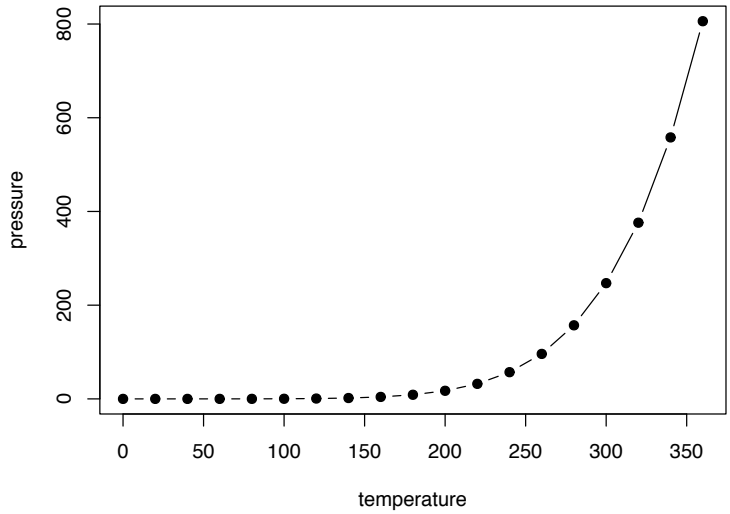


FIGURE 2.1: Un ejemplo de figura con el aspecto especificado, ancho, y alineación.

Si desea referencias cruzadas hacia figuras o tablas generadas a partir de un chunk, por favor asegúrese de que la etiqueta del chunk sólo contenga caracteres alfanuméricos (a-z, A-Z, 0-9), barras (/) o guiones (-).

La opción del chunk `fig.asp` se puede utilizar para establecer la relación de aspecto de los gráficos, es decir, la relación entre la altura/anchura. Si el ancho de la figura es de 6 pulgadas (`fig.width = 6`) y`fig.asp = 0.7`, la altura de la figura se calcula automáticamente con `fig.width * fig.asp = 6 * 0,7 = 4.2`. La figura 2.1 es un ejemplo utilizando las opciones del chunk `fig.asp = 0.7`, `fig.width = 6`, y `fig.align = 'center'`, generado a partir del código de abajo:

```
par(mar = c(4, 4, .1, .1))
plot(pressure, pch = 19, type = 'b')
```

El tamaño real de un gráfico se determina por las opciones del chunk `fig.width` y `fig.height` (el tamaño del gráfico generado desde

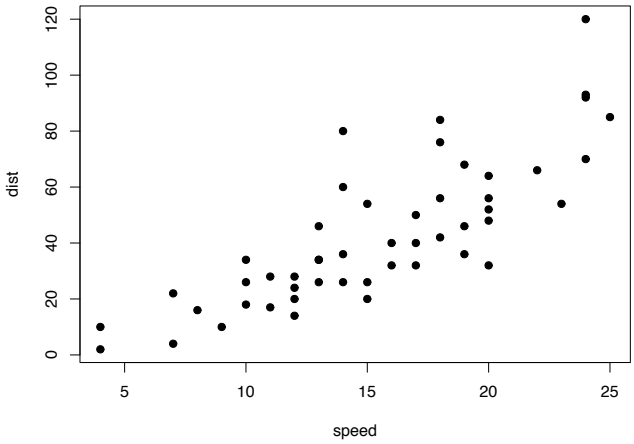


FIGURE 2.2: Una figura de ejemplo con un ancho relativo de 70%.

un dispositivo gráfico), y se puede especificar el tamaño de salida de los gráficos a través de las opciones del chunk `out.width` y `out.height`. El valor posible de estas dos opciones depende del formato de salida del documento. Por ejemplo, `out.width = '30%'` es un valor válido para la salida HTML, pero no para la producción de documentos LaTeX/PDF. Sin embargo, **knitr** convertirá automáticamente a un valor de porcentaje para `out.width` de la forma`x%` a `(x / 100) \linewidth`, por ejemplo, `out.width = '70%'` se intrpretará como `.7 \linewidth` cuando el formato de salida es LaTeX. Esto hace que sea posible especificar una anchura relativa de un gráfico de una manera consistente. La figura 2.2 es un ejemplo de `out.width = 70%`.

```
par(mar = c(4, 4, .1, .1))
plot(cars, pch = 19)
```

Si desea poner varios gráficos en un entorno de figuras, debe utilizar la opción del chunk `fig.show = 'hold'` para disponer de múltiples gráficos en un chunk e incluirlas en un único entorno. También puede poner gráficos uno al lado del otro si la suma de la anchura de todas los gráficos es menor o igual a la anchura de la línea actual. Por ejemplo, si dos gráficos tienen la misma anchura `50%`, serán

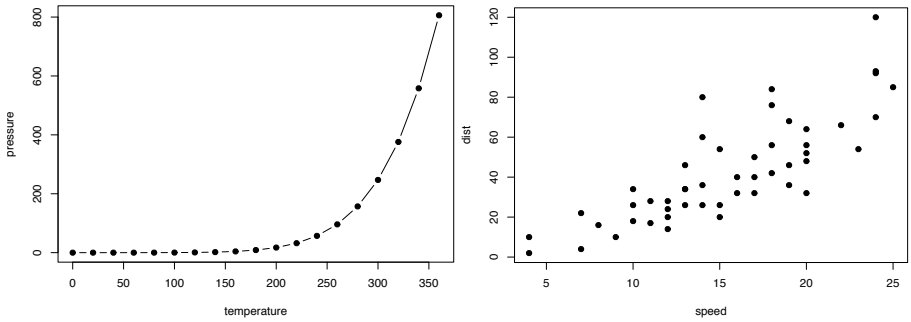


FIGURE 2.3: Dos gráficos dispuestos uno al lado del otro.

ubicados uno al lado del otro. Del mismo modo, puede especificar `out.width = '33% '` para organizar tres gráficos en una sola línea. La figura 2.3 es un ejemplo de dos gráficos, cada una con una anchura de `50%`.

```
par(mar = c(4, 4, .1, .1))
plot(pressure, pch = 19, type = 'b')
plot(cars, pch = 19)
```

A veces es posible que tenga imágenes que no se generaron a partir de código en R. A pesar de esto, puede incluirlas en R Markdown a través de la función `knitr::include_graphics()`. La figura 2.4 es un ejemplo de tres logotipos de **knitr** incluidos en un entorno gráfico. Se pueden incluir una o varias imágenes con la función `include_graphics()`, y todas las opciones del chunk que se aplican a los gráficos en R también se aplican a estas imágenes, por ejemplo, se puede usar `out.width = '33% '` para establecer las anchuras de estas imágenes en el documento de salida.

```
knitr::include_graphics(rep('images/knit-logo.png', 3))
```

Las ventajas de usar `include_graphics()` son:

1. Usted no necesita preocuparse por el formato de salida de documentos, por ejemplo, cuando el formato de salida es LaTeX, puede que tenga que utilizar el comando LaTeX

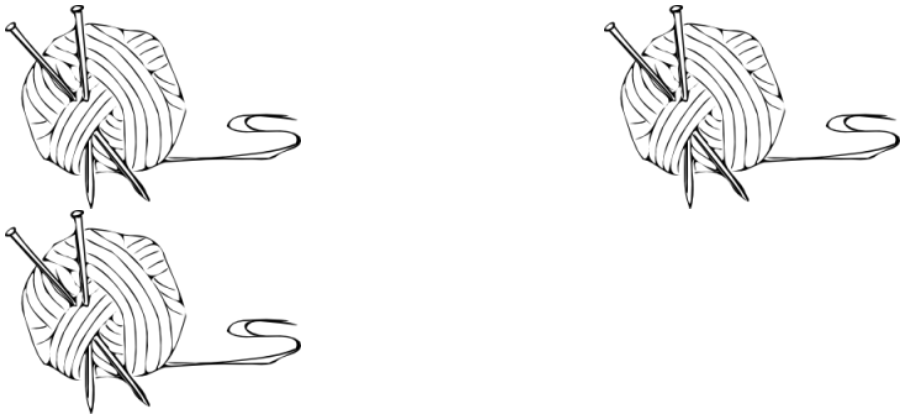


FIGURE 2.4: Tres logos de knitr incluidos en el documento desde una imagen PNG externa.

`\includegraphics{}` para incluir una imagen, y cuando el formato de salida es Markdown, usted tiene que usar `![]()`. La función `include_graphics()` en **knitr** se ocupa de estos detalles de forma automática.

2. La sintaxis para el control de los atributos de la imagen es el mismo que cuando las imágenes se generan a partir de código en R, por ejemplo, las opciones del chunk `fig.cap`,`out.width`, y `fig.show` tienen todas la misma función.
3. `include_graphics()` es lo suficientemente versátil como para utilizar gráficos PDF de forma automática cuando el formato de salida es LaTeX y existen gráficos en formato PDF, por ejemplo, una ruta de la imagen `foo/bar.png` puede sustituirse automáticamente por `foo/bar.PDF` si existe esta última. Las imágenes en PDF a menudo tienen mejores cualidades que las imágenes de mapa de bits en la producción de LaTeX/PDF. Por supuesto, se puede desactivar esta función `include_graphics(auto_pdf = FALSE)`.
4. Se pueden escalar fácilmente estas imágenes de forma proporcional utilizando la misma razón. Esto se puede hacer a través del argumento `dpi` (puntos por pulgada), que toma el valor de la opción del chunk `dpi` por defecto.

Si se trata de un valor numérico y la opción del chunk `out.width` no está establecida, el ancho de salida de una imagen será su anchura real (en píxeles) dividido por `dpi`, y sus unidades estarán en pulgadas. Por ejemplo, para una imagen de tamaño de 672 x 480, su anchura de salida será de 7 pulgadas (`7in`) cuando `dpi = 96`. Esta función requiere del paquete **png** y/o **jpeg**. Siempre se puede anular el cálculo automático del ancho en pulgadas, proporcionando un valor no nulo a la opción del chunk `out.wdith`, o usar`include_graphics(dpi = NA)`.

2.5 Tablas

Por ahora, la forma más conveniente para generar una tabla es la función `knitr::kable()`, porque hay algunos trucos internos en **knitr** para que funcione con **bookdown** y los usuarios no tienen necesidad de preocuparse por estos detalles de implementación. Adelante se explicará cómo utilizar otros paquetes y funciones más detalladamente en esta sección.

Al igual que las figuras, las tablas con títulos también estarán numeradas y podrán ser referenciadas. La función `kable()` generará automáticamente una etiqueta para un entorno de la tabla, que es el prefijo `tab:` más la etiqueta del chunk. Por ejemplo, la etiqueta de la tabla para un chunk con la etiqueta `foo` será: `tab:foo`, y usar la sintaxis `\@ref(etiqueta)` para hacer referencia a la tabla. La tabla 2.2 es un ejemplo sencillo.

```
knitr::kable(
  head(mtcars, 10), booktabs = TRUE,
  caption = 'Una tabla de las primeras 10 filas de la base de datos mtcars.'
)
```

Si desea poner varias tablas en un único entorno de tablas, simple-

TABLE 2.2: Una tabla de las primeras 10 filas de la base de datos mtcars.

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4

mente empaquete los objetos de datos (usualmente en data frames en R) dentro de una lista. Véase la tabla 2.3 para un ejemplo.

```
knitr::kable(
  list(
    head(iris[,1:2],3),
    head(mtcars[,1:3],5)
  ),
  caption = 'Una trama de dos tablas.', booktabs = TRUE
)
```

Cuando no se quiera una tabla flotante en PDF, es posible utilizar el paquete de LaTeX **longtable**[5], que puede dividir una tabla a través de múltiples páginas. Para utilizar **longtable**, sólo tiene que poner `longtable = TRUE` en `kable()`, y asegurarse de incluir `\usepackage{longtable}` en el preámbulo de LaTeX (véase la sección 4.1 para saber cómo personalizar el preámbulo LaTeX). Por supuesto, esto es irrelevante para la salida HTML, ya que las tablas de HTML no necesitan flotar.

[5] https://www.ctan.org/pkg/longtable

TABLE 2.3: Una trama de dos tablas.

Sepal.Length	Sepal.Width
5.1	3.5
4.9	3.0
4.7	3.2

	mpg	cyl	disp
Mazda RX4	21.0	6	160
Mazda RX4 Wag	21.0	6	160
Datsun 710	22.8	4	108
Hornet 4 Drive	21.4	6	258
Hornet Sportabout	18.7	8	360

```
knitr::kable(
  iris[1:100, ], longtable = TRUE, booktabs = TRUE,
  caption = 'Una tabla generada mediante el paquete `longtable`.'
)
```

TABLE 2.4: Una tabla generada mediante el paquete 'longtable'.

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa
4.6	3.4	1.4	0.3	setosa
5.0	3.4	1.5	0.2	setosa
4.4	2.9	1.4	0.2	setosa
4.9	3.1	1.5	0.1	setosa
5.4	3.7	1.5	0.2	setosa
4.8	3.4	1.6	0.2	setosa

4.8	3.0	1.4	0.1	setosa
4.3	3.0	1.1	0.1	setosa
5.8	4.0	1.2	0.2	setosa
5.7	4.4	1.5	0.4	setosa
5.4	3.9	1.3	0.4	setosa
5.1	3.5	1.4	0.3	setosa
5.7	3.8	1.7	0.3	setosa
5.1	3.8	1.5	0.3	setosa
5.4	3.4	1.7	0.2	setosa
5.1	3.7	1.5	0.4	setosa
4.6	3.6	1.0	0.2	setosa
5.1	3.3	1.7	0.5	setosa
4.8	3.4	1.9	0.2	setosa
5.0	3.0	1.6	0.2	setosa
5.0	3.4	1.6	0.4	setosa
5.2	3.5	1.5	0.2	setosa
5.2	3.4	1.4	0.2	setosa
4.7	3.2	1.6	0.2	setosa
4.8	3.1	1.6	0.2	setosa
5.4	3.4	1.5	0.4	setosa
5.2	4.1	1.5	0.1	setosa
5.5	4.2	1.4	0.2	setosa
4.9	3.1	1.5	0.2	setosa
5.0	3.2	1.2	0.2	setosa
5.5	3.5	1.3	0.2	setosa
4.9	3.6	1.4	0.1	setosa
4.4	3.0	1.3	0.2	setosa
5.1	3.4	1.5	0.2	setosa
5.0	3.5	1.3	0.3	setosa
4.5	2.3	1.3	0.3	setosa
4.4	3.2	1.3	0.2	setosa
5.0	3.5	1.6	0.6	setosa
5.1	3.8	1.9	0.4	setosa
4.8	3.0	1.4	0.3	setosa

5.1	3.8	1.6	0.2	setosa
4.6	3.2	1.4	0.2	setosa
5.3	3.7	1.5	0.2	setosa
5.0	3.3	1.4	0.2	setosa
7.0	3.2	4.7	1.4	versicolor
6.4	3.2	4.5	1.5	versicolor
6.9	3.1	4.9	1.5	versicolor
5.5	2.3	4.0	1.3	versicolor
6.5	2.8	4.6	1.5	versicolor
5.7	2.8	4.5	1.3	versicolor
6.3	3.3	4.7	1.6	versicolor
4.9	2.4	3.3	1.0	versicolor
6.6	2.9	4.6	1.3	versicolor
5.2	2.7	3.9	1.4	versicolor
5.0	2.0	3.5	1.0	versicolor
5.9	3.0	4.2	1.5	versicolor
6.0	2.2	4.0	1.0	versicolor
6.1	2.9	4.7	1.4	versicolor
5.6	2.9	3.6	1.3	versicolor
6.7	3.1	4.4	1.4	versicolor
5.6	3.0	4.5	1.5	versicolor
5.8	2.7	4.1	1.0	versicolor
6.2	2.2	4.5	1.5	versicolor
5.6	2.5	3.9	1.1	versicolor
5.9	3.2	4.8	1.8	versicolor
6.1	2.8	4.0	1.3	versicolor
6.3	2.5	4.9	1.5	versicolor
6.1	2.8	4.7	1.2	versicolor
6.4	2.9	4.3	1.3	versicolor
6.6	3.0	4.4	1.4	versicolor
6.8	2.8	4.8	1.4	versicolor
6.7	3.0	5.0	1.7	versicolor
6.0	2.9	4.5	1.5	versicolor
5.7	2.6	3.5	1.0	versicolor

5.5	2.4	3.8	1.1	versicolor
5.5	2.4	3.7	1.0	versicolor
5.8	2.7	3.9	1.2	versicolor
6.0	2.7	5.1	1.6	versicolor
5.4	3.0	4.5	1.5	versicolor
6.0	3.4	4.5	1.6	versicolor
6.7	3.1	4.7	1.5	versicolor
6.3	2.3	4.4	1.3	versicolor
5.6	3.0	4.1	1.3	versicolor
5.5	2.5	4.0	1.3	versicolor
5.5	2.6	4.4	1.2	versicolor
6.1	3.0	4.6	1.4	versicolor
5.8	2.6	4.0	1.2	versicolor
5.0	2.3	3.3	1.0	versicolor
5.6	2.7	4.2	1.3	versicolor
5.7	3.0	4.2	1.2	versicolor
5.7	2.9	4.2	1.3	versicolor
6.2	2.9	4.3	1.3	versicolor
5.1	2.5	3.0	1.1	versicolor
5.7	2.8	4.1	1.3	versicolor

Si decide utilizar otros paquetes para generar tablas, tiene que asegurarse de que la etiqueta para el ambiente de la tabla aparezca al comienzo de la leyenda de la tabla de la forma `(\#label)`, donde `label` debe tener el prefijo `tab:`. Tiene que tenerse mucho cuidado con la *portabilidad* de la función que genera la tabla: deberá trabajar tanto para la salida HTML como para LaTeX de forma automática, por lo que debe tener en cuenta el formato de salida internamente (revise `knitr::opts_knit$get('pandoc.to')`). Al escribir una tabla HTML, el título debe ser escrito en la etiqueta `<caption></caption>`. Para las tablas simples, `kable()` debería ser suficiente. Si usted tiene que crear tablas complicadas (por ejemplo, con ciertas celdas que atraviesen múltiples columnas/filas), tendrá que tomar las cuestiones mencionadas en consideración.

2.6 Referencias cruzadas

Se ha explicado cómo funcionan las referencias cruzadas para ecuaciones (Sección 2.2.1), teoremas (Sección 2.2.2), figuras (Sección 2.4) y tablas (Sección 2.5). De hecho, también se puede hacer referencia a secciones utilizando la misma sintaxis `\@ref(label)`, donde `label` es el identificador de una sección. De forma predeterminada, Pandoc generará un ID para todos los encabezados de sección, por ejemplo, una sección `# Hola Mundo` tendrá un ID `hola-mundo`. Se recomienda asignar manualmente un identificador para un encabezado de sección con el fin de asegurarse de que no se olvide actualizar la etiqueta de referencia después de cambiar el encabezado de sección. Para asignar un ID a un encabezado de sección, simplemente añada `{#id}` hasta el final del encabezado de sección.

Cuando una etiqueta referenciada no se puede encontrar, verá dos signos de interrogación como **??**, así como un mensaje de advertencia en la consola de R cuando se compila el libro.

También pueden crearse enlaces basados en texto usando identificadores de sección explícita o automáticos e incluso el texto del encabezado de sección actual.

- Si usted está satisfecho con el encabezado de sección como texto de enlace, úselo dentro de un único conjunto de paréntesis cuadrados:
  - `[Texto en la sección de encabezado]`: ejemplo "Un documento sencillo" a través de `[Un documento sencillo]`
- Hay dos formas de especificar el texto del vínculo personalizado:
  - `[Texto del vínculo][Sección texto de encabezado]`: los libros que no están en inglés a través de `[libros no están en inglés] [Internacionalización]`
  - `[Texto del enlace] (#ID)`: "Tabla cosas" a través de `[Tabla cosas](#tablas)`

La documentación acerca de Pandoc proporciona más detalles

sobre identificación automática de secciones[6] y referencias de cabecera implícitas[7].

Las referencias cruzadas siguen funcionando incluso cuando se refiere a un item que no esté en la página actual de la salida PDF o HTML. Por ejemplo, véase la ecuación (2.1) y la figura 2.4

2.7 Bloques personalizados

Puede generar bloques personalizados utilizando el motor de `block` en **knitr** , es decir, la opción del chunk `engine = 'block'`, o la sintaxis más compacta ```` ```{block} ````. Este motor se debe utilizar en combinación con la opción del chunk `type`, que tiene una cadena de caracteres. Cuando se utiliza el motor de `block`, genera un `<div>` para envolver el contenido del chunk si el formato de salida es HTML, y un entorno de LaTeX si la salida es ésta. La opción `type` especifica la clase del `<div>` y el nombre del entorno de LaTeX. Por ejemplo, la salida HTML de este chunk

````
```{block, type='FOO'}
Some text for this block.
```
````

sería:

```
<div class="FOO">
Some text for this block.
</div>
```

y la salida en LaTeX sería:

[6] http://pandoc.org/README.html#extension-auto_identifiers

[7] http://pandoc.org/README.html#extension-%20implicit_header_references

```
\begin{FOO}
Some text for this block.
\end{FOO}
```

Depende del autor del libro establecer cómo definir el estilo del bloque. Puede definir el estilo de `<div>` en CSS e incluirlo en la salida a través de la opción `includes` en los metadatos YAML. Del mismo modo, puede definir el entorno de LaTeX a través de `\newenvironment` e incluir la definición en la salida de LaTeX a través de la opción `includes`. Por ejemplo, podemos guardar el siguiente estilo en un archivo CSS, digamos, `style.css`:

```
div.FOO {
  font-weight: bold;
  color: red;
}
```

Y los metadatos YAML del documento R Markdown pueden ser:

```
---
output:
  bookdown::html_book:
    includes:
      in_header: style.css
---
```

Hemos definido algunos tipos de bloques para que este libro muestre notas, consejos y advertencias, etc. A continuación se presentan algunos ejemplos:

R es software libre y viene con ABSOLUTAMENTE NINGUNA GARANTÍA. Le invitamos a redistribuirlo bajo los términos de la GNU General Public License versiones 2 o 3. Para más información sobre estos asuntos, `http://www.gnu.org/licenses/`.

R es software libre y viene con ABSOLUTAMENTE NINGUNA GARANTÍA. Le invitamos a redistribuirlo bajo los términos de la GNU General Public License versiones 2 o 3. Para más información sobre estos asuntos, `http://www.gnu.org/licenses/`.

R es software libre y viene con ABSOLUTAMENTE NINGUNA GARANTÍA. Le invitamos a redistribuirlo bajo los términos de la GNU General Public License versiones 2 o 3. Para más información sobre estos asuntos, `http://www.gnu.org/licenses/`.

R es software libre y viene con ABSOLUTAMENTE NINGUNA GARANTÍA. Le invitamos a redistribuirlo bajo los términos de la GNU General Public License versiones 2 o 3. Para más información sobre estos asuntos, `http://www.gnu.org/licenses/`.

R es software libre y viene con ABSOLUTAMENTE NINGUNA GARANTÍA. Le invitamos a redistribuirlo bajo los términos de la GNU General Public License versiones 2 o 3. Para más información sobre estos asuntos, `http://www.gnu.org/licenses/`.

El motor **knitr** `block` fue diseñado para mostrar contenido simple (normalmente un párrafo de texto sin formato). Puede utilizar una sintaxis de formato simple, como hacer que ciertas palabras sean negritas o cursivas, pero la sintaxis más avanzada, como las citas y las referencias cruzadas, no funcionarán. Sin embargo, hay un motor alternativo denominado `block2` que soporta la sintaxis arbitraria de Markdown, por ejemplo,

```
```{block2, type='FOO'}
Some text for this block [@citation-key].

- a list item
```
```

```
- another item

More text.
```

El motor `block2` también debería ser más rápido que el motor `block` si tiene muchos bloques personalizados en el documento, pero su implementación se basó en un hack,[8] por lo que no estamos 100% seguros de si siempre va a funcionar en el futuro. No hemos visto problemas con Pandoc v1.17.2 todavía.

Una advertencia más para el motor `block2`: si el último elemento del bloque no es un párrafo ordinario, debe dejar una línea en blanco al final, por ejemplo,

```
```{block2, type='FOO'}
Some text for this block [@citation-key].

- a list item
- another item
- end the list with a blank line

```
```

El teorema y los entornos de prueba en la sección 2.2.2 se implementan en realidad a través del motor `block2`.

Para todos los bloques personalizados basados en el motor `block` o `block2`, hay una opción de bloqueo `echo` que puede usar para mostrar (`echo = TRUE`) u ocultar (`echo = FALSE`) los bloques.

[8] https://github.com/jgm/pandoc/issues/2453

2.8 Citas

Aunque Pandoc soporta múltiples formas de escribir las citas, se recomienda que utilice las bases de datos BibTeX porque trabajan mejor con la salida de LaTeX/PDF. Pandoc puede procesar otros tipos de bases de datos bibliográficas con la utilidad `pandoc-citeproc` (`https://github.com/jgm/pandoc-citeproc`), pero no pone ciertos elementos de bibliografía correctamente (especialmente en el caso de múltiples autores). Con las bases de datos BibTeX, podrá definirse el estilo bibliográfico si se requiere por un determinado editor o revista.

Una base de datos BibTeX es un archivo de texto (con la extensión de nombre de archivo convencional `.bib`) que consta de entradas bibliográficas como:

```
@Manual{R-base,
  title = {R: A Language and Environment for Statistical Computing},
  author = {{R Core Team}},
  organization = {R Foundation for Statistical Computing},
  address = {Vienna, Austria},
  year = {2015},
  url = {https://www.R-project.org/},
}
```

Una entrada bibliográfica comienza con `{@type`, donde `type` puede ser `article`, `book`,`manual`, etc.[9]. A continuación hay una clave de citación, como `R-base` en el ejemplo anterior. Para citar una entrada, use `@ key` o `[@key]` (este último pone la cita entre paréntesis), por ejemplo, `@R-base` se compila como R Core Team (2016), y `[@ R-base]` genera "(R Core Team, 2016)". Si no está familiarizado con el paquete **natbib** de LaTeX, `@key` es básicamente `\citet{key}` y `[@key]` es equivalente a `\citep{key}`.

[9]El nombre del tipo no diferencia mayúsculas y minúsculas, por ende no importa si es `manual`, `Manual`, o `MANUAL`.

Hay una serie de campos en una entrada de bibliografía, tales como `title`, `author`, y `year`, etc. Puede visitar `https://en.wikipedia.org/wiki/BibTeX` para saber más sobre posibles tipos de entradas y campos en BibTeX.

Hay una función auxiliar `write_bib()` en **knitr** para generar entradas BibTeX automáticamente para paquetes en R. Tenga en cuenta que sólo genera una entrada de BibTeX para el propio paquete en el momento, mientras que un paquete puede contener múltiples entradas en el archivo `CITATION`, y algunas entradas versan sobre las publicaciones relacionadas con el paquete. Estas entradas son ignoradas por `write_bib()`.

```
# el segundo argumento puede ser un archivo .bib
knitr::write_bib(c('knitr', 'stringr'), '')
```

```
@Manual{R-knitr,
  title = {knitr: A General-Purpose Package for Dynamic Report Generation in R},
  author = {Yihui Xie},
  year = {2016},
  note = {R package version 1.15.1},
  url = {https://CRAN.R-project.org/package=knitr},
}
@Manual{R-stringr,
  title = {stringr: Simple, Consistent Wrappers for Common String Operations},
  author = {Hadley Wickham},
  year = {2016},
  note = {R package version 1.1.0},
  url = {https://CRAN.R-project.org/package=stringr},
}
```

Una vez que tenga uno o varios archivos `.bib`, puede utilizar el campo `bibliography` en los metadatos del documento YAML del documento R Markdown, y también se puede especificar el estilo de bibliografía a través de `biblio-style` (esto sólo se aplica a las salidas en PDF), por ejemplo,

```
---
bibliography: ["one.bib", "another.bib", "yet-another.bib"]
biblio-style: "apalike"
link-citations: true
---
```

El campo `link-citations` se puede utilizar para añadir enlaces internos a partir del texto de la cita del estilo autor-año de la entrada bibliográfica en la salida HTML.

Cuando el formato de salida es LaTeX, las citas se colocarán automáticamente en un capítulo o sección. Para la salida no LaTeX, puede agregar un capítulo vacío como el último capítulo de su libro. Por ejemplo, si su último capítulo es el archivo Rmd `06-references.Rmd`, su contenido puede ser una expresión R en línea:

```
`r if (knitr:::is_html_output()) '# References {-}'`
```

2.9 Índices

Actualmente el índice sólo se admite para la producción de LaTeX/PDF. Para imprimir un índice después del libro, puede utilizar el paquete de LaTeX **makeindex** en el preámbulo (véase la sección 4.1):

```
\usepackage{makeidx}
\makeindex
```

A continuación, inserte `\printindex` al final de su libro a través de la opción YAML `includes -> after_body`. Una entrada de índice se puede crear a través del comando `\{index}` en el cuerpo del libro, por ejemplo, `\index{GIT}`.

2.10 HTML Widgets

Aunque una de las mayores fortalezas de R es la visualización de datos, hay un gran número de librerías de JavaScript para la visualización de datos mucho más rica. Estas librerías se pueden utilizar para crear aplicaciones interactivas que pueden procesarse fácilmente en los navegadores web, por lo que los usuarios no necesitan instalar ningún paquete de software adicional para ver las visualizaciones. Una forma de llevar estas librerías JavaScript a R es a través del paquete **htmlwidgets**[10] (Vaidyanathan et al., 2016).

Los HTML widgets pueden representarse como una página web independiente (al igual que un gráfico de R), o incrustarse en los documentos de R Markdown y aplicaciones Shiny. Fueron diseñados originalmente para salidas HTML solamente, y requieren la disponibilidad de JavaScript, por lo que no van a trabajar en formatos que no son HTML de salida, tales como LaTeX/PDF. Antes de la v1.13 de **knitr**, se obtenía un error cuando se procesaban widgets de HTML a un formato de salida que no era HTML. A partir de esta versión, los widgets de HTML se procesan automáticamente como imágenes tomadas a través del paquete **webshot** (Chang, 2016). Por supuesto, es necesario instalar el paquete **webshot**. Además, debe instalarse PhantomJS (`http://phantomjs.org`), ya que es lo que **webshot** utiliza para capturar imágenes. Tanto **WebShot** como PhantomJS se pueden instalar de forma automática desde R:

```
install.packages('webshot')
webshot::install_phantomjs()
```

La función `install_phantomjs()` funciona para Windows, OSX, y Linux. También puede optar por descargar e instalar PhantomJS

[10] `http://htmlwidgets.org`

por sí mismo, si está familiarizado con la modificación del entorno del sistema variable de PATH.

Cuando **knitr** detecta un objeto widget de HTML en un chunk, o compila el widget normalmente cuando el formato de salida actual es HTML, o guarda el widget como una página HTML y llama a **webshot** para capturar la pantalla de la página HTML cuando el formato de salida no es HTML. He aquí un ejemplo de una tabla creada a partir del paquete **DT** (Xie, 2016b):

```
DT::datatable(iris)
```

```
## PhantomJS not found. You can install it with webshot::install_phantomjs(). If it is install
```

Una tabla widget compilada a través del paquete DT.

Si usted está leyendo este libro como un página web ahora mismo, debería ver una tabla interactiva generada a partir del chunk anterior, por ejemplo, puede ordenar las columnas y generar la búsqueda en la tabla. Si está leyendo una versión que no sea HTML de este libro, debería ver una captura de pantalla de la tabla. La captura de pantalla puede tener un aspecto un poco diferente con el widget real mostrado en el navegador web, debido a la diferencia entre un navegador web real y navegador virtual PhantomJS '.

Hay un número de opciones de chunk de **knitr** relacionados con la captura de pantalla. En primer lugar, si usted no está satisfecho con la calidad de las capturas de pantalla automáticas, o desea una captura de pantalla del widget en un estado en particular (por ejemplo, después de hacer clic y ordenar una determinada columna de una tabla), puede capturar la pantalla de forma manual, y proporcionar su propia captura a través de la opción del chunk `screenshot.alt` (capturas de pantalla alternativas). Esta opción toma las ubicaciones de las imágenes. Si tiene varios widgets en un chunk, se puede proporcionar un vector con las rutas de las imágenes. Cuando esta opción está presente, **knitr** ya no llamará a **webshot** para tomar capturas de pantalla automáticas.

En segundo lugar, a veces es posible que desee forzar a **knitr** a

utilizar imágenes estáticas en lugar de compilar los widgets incluso en páginas HTML. En este caso, se puede establecer la opción del chunk `screenshot.force = TRUE`, y los widgets siempre serán mostrados como imágenes estáticas. Debe tenerse en cuenta que aún puede optar por utilizar capturas de pantalla automáticas o personalizadas.

En tercer lugar, **webshot** tiene algunas opciones para controlar las capturas de pantalla automáticas, y es posible especificar estas opciones a través de la opción en el chunk `screenshot.opts`, que toma una lista como `list(delay = 2, cliprect = 'viewport')`. Ver la página de ayuda `? webshot::webshot` para la lista completa de posibles opciones, y la viñeta del paquete[11] `vignette('intro', package = 'webshot')` ha ilustrado el efecto de estas opciones. Aquí la opción `delay` puede ser importante para los widgets que toman mucho tiempo en cargarse: `delay` especifica el número de segundos de espera antes de que PhantomJS capture la pantalla. Si ve una pantalla incompleta, es posible que desee especificar un retraso más largo (el valor predeterminado es 0,2 segundos).

En cuarto lugar, si usted siente que esto es lento para capturar las imágenes, o no quieren hacerlo cada vez que ejecuta el chunk, se puede utilizar la opción del chunk `cache = TRUE` para almacenar en caché el chunk. El almacenamiento en caché funciona tanto para HTML como para formatos de salida que no son HTML.

Las imágenes se comportan como gráficos normales en R en el sentido de que muchas opciones del chunk relacionadas con figuras también se aplican a las capturas, incluyendo `fig.width`,`fig.height`, `out.width`, `fig.cap`, etc. Así pues, puede especificarse el tamaño de las capturas de pantalla en el documento de salida, y asignar los pies de figura a ellos también. El formato de imagen de las capturas de pantalla automáticas puede especificarse mediante la opción `dev`, y los posibles valores son `PDF`, `png`, y `jpeg`. El valor predeterminado para la salida PDF es `PDF`, y `png` para otros tipos de salida. Nótese que `PDF` puede no funcionar tan fielmente como `png`: a veces hay ciertos elementos en una página HTML que no puede represen-

[11] `https://cran.rstudio.com/web/packages/webshot/vignettes/intro.html`

tarse en la captura de pantalla en PDF, por lo que es posible que desee utilizarse `dev = 'png'` incluso para la salida PDF. Depende de los casos específicos de widgets de HTML, y se puede intentar tanto `PDF` como `png` (o `jpeg`) antes de decidir qué formato es el más deseable.

2.11 Páginas web y aplicaciones Shiny

Al igual que en los HTML widgets, los sitios web se pueden incrustar en el libro. Puede utilizar la función `knitr::include_url()` para incluir una página web a través de su URL. Cuando el formato de salida es HTML, se usa un `iframe`[12]; en otros casos, **knitr** trata de tomar una captura de pantalla de la página web (o utilizar una captura de pantalla personalizada en la medida en que se haya pensado). Todas las opciones del chunk son las mismas que para los HTML widgets. Una opción que puede requerir atención especial es la opción `delay`: los HTML widgets se compilan a nivel local, por lo que por lo general son rápidos para cargar por PhantomJS para tomar capturas de pantalla, pero una URL arbitraria pueden tardar más tiempo en cargarse, por lo que es posible que desee utilizar un valor mayor de `delay`, por ejemplo, utilizar la opción del chunk `screenshot.opts = list(delay = 5)`.

Una función relacionada es `knitr::include_app()`, que es muy similar a `include_url()`, y fue diseñada para incrustar aplicaciones shiny a través de sus direcciones URL en la salida. Su única diferencia con `include_url()` es que añade automáticamente un parámetro de consulta `?showcase=0` a la URL, si no hay otros parámetros de consulta que estén presentes en la URL, desactiva el modo showcase de Shiny, lo que es poco probable que sea útil para las capturas de pantalla o iframes. Si se desea solo el modo showcase, única-

[12]Un `iframe` es básicamente una caja en una página Web para incrustar otra página Web.

mente use `include_url()` en lugar de `include_app()`. A continuación se muestra un ejemplo de aplicación Shiny (Figura 2.11):

```
knitr::include_app('https://yihui.shinyapps.io/miniUI/', height = '600px')
```

```
## PhantomJS not found. You can install it with webshot::install_phantomjs(). If it is install
```

Una aplicación en Shiny creada mediante el paquete miniUI; puede ver una versiòn en vivo en: `https://yihui.shinyapps.io/miniUI/`.

Una vez más, verá una aplicación en vivo si está leyendo una versión HTML de este libro, y una captura de pantalla estática si está leyendo otros tipos de formatos. La aplicación Shiny anterior se ha creado usando el paquete **miniUI** (Cheng, 2016), que proporciona funciones de diseño que son particularmente agradables para aplicaciones Shiny en pantallas pequeñas. Si utiliza funciones de diseño Shiny normales, es probable que vea las barras de desplazamiento verticales y/o horizontales en los marcos flotantes debido a que el tamaño de página es demasiado grande para caber un marco flotante. Cuando el ancho predeterminado del marco flotante es demasiado pequeño, puede utilizar la opción del chunk `out.width` para cambiarlo. Para la altura del iframe, utilice el argumento `height` característica de `include_url()` / `include_app()`.

Las aplicaciones Shiny pueden tardar más tiempo en cargar que las URL habituales. Es posible que desee utilizar un valor conservador para la opción `delay`, por ejemplo, 10. Es evidente que tanto `include_url()` como `include_app()` requieren conexión a Internet, a menos que haya almacenado en caché previamente el chunk (pero páginas Web dentro de iframes aún no funcionan sin una conexión a Internet).

3

Formatos de salida

El paquete **bookdown** tablita es compatible principalmente con tres tipos de formatos de salida: HTML, LaTeX/PDF y e-books. En este capítulo presentamos las opciones posibles para estos formatos. Los formatos de salida pueden especificarse en los metadatos YAML del primer archivo Rmd del libro o en un archivo YAML por separado llamado `_output.yml` que se encuentra en el directorio raíz del libro. Un breve ejemplo de los primeros (los formatos de salida que se especifican en el campo `output` de los metadatos YAML) es:

```
---
title: "Un libro impresionante"
author: "Martín Macías y Prof. Daniel Rodríguez"
output:
  bookdown::gitbook:
    lib_dir: assets
    split_by: section
    config:
      toolbar:
        position: static
  bookdown::pdf_book:
    keep_tex: yes
  bookdown::html_book:
    css: toc.css
documentclass: book
---
```

Un ejemplo del archivo `_output.yml` es:

```
bookdown::gitbook:
  lib_dir: assets
  split_by: section
  config:
    toolbar:
      position: static
bookdown::pdf_book:
  keep_tex: yes
bookdown::html_book:
  css: toc.css
```

En este caso, todos los formatos deben estar en el nivel superior, en vez de estar bajo un campo `output`. En el archivo `_output.yml` no necesita usar tres guiones `---` .

3.1 HTML

La principal diferencia entre compilar un libro (usando **bookdown**) y compilar un simple documento R Markdown (utilizando **rmarkdown**) a HTML es que un libro generará múltiples páginas HTML de forma predeterminada — normalmente un archivo HTML por capítulo. Esto hace que sea más fácil de señalar un determinado capítulo o compartir su URL con otras personas a medida que se lee el libro, además de ser más rápido a la hora de cargar el libro en el navegador web. Actualmente se ha proporcionado un número de estilos diferentes para la salida HTML: el estilo GitBook, el estilo Bootstrap, y el estilo Tufte.

3.1.1 El estilo GitBook

El estilo GitBook fue tomado de GitBook, un proyecto puesto en marcha por Friendcode, Inc (`https://www.gitbook.com`) y se dedica a ayudar a los autores a escribir libros con Markdown. Propor-

ciona un estilo bonito, con un diseño que consiste en una barra lateral que muestra la tabla de contenido en la parte izquierda de la pantalla, y el cuerpo principal del libro a la derecha. El diseño es sensible al tamaño de la ventana, por ejemplo, los botones de navegación se muestran a la izquierda/derecha del cuerpo del libro cuando la ventana es lo suficientemente ancha, y colapsa en la parte inferior cuando la ventana es estrecha para dar a los lectores más espacio horizontal para leer el cuerpo del libro.

Se han hecho varias mejoras con respecto al proyecto original GitBook. El más significativo es que se ha sustituido el motor de Markdown con R Markdown v2 basado en Pandoc, por lo que hay muchas más características para utilizar cuando se escribe un libro. Por ejemplo,

- Puede incorporar chunks en R y expresiones en línea de R en Markdown, y esto hace que sea fácil crear documentos reproducibles y lo libera de sincronizar su cómputo con la salida actual (**knitr** se encargará de eso automáticamente);
- La sintaxis de Markdown es mucho más rica: se puede escribir cualquier cosa que Markdown de Pandoc soporte, como las expresiones matemáticas de LaTeX y citas;
- Puede incrustar contenido interactivo en el libro (para la salida HTML únicamente), tales como HTML widgets y aplicaciones Shiny;

También se han añadido algunas características útiles en la interfaz de usuario que se introducirán en detalle a continuación. La función de formato de salida para el estilo GitBook en **bookdown** es `gitbook()`. A continuación se presentan sus argumentos:

```
gitbook(fig_caption = TRUE, number_sections = TRUE, self_contained = FALSE,
  lib_dir = "libs", ..., split_by = c("chapter", "chapter+number",
    "section", "section+number", "rmd", "none"), split_bib = TRUE,
  config = list())
```

La mayoría de los argumentos se pasan a `rmarkdown::html_document()`, incluyendo `fig_caption`, `lib_dir`,

y ... . Puede comprobarse en la página de ayuda de `rmarkdown::html_document()`, la lista completa de opciones posibles. Se recomienda encarecidamente utilizar `fig_caption = TRUE` por dos razones: 1) Es importante explicar las figuras con etiquetas; 2) permitir que los pies de figura representen figuras que se colocarán en entornos flotantes cuando la salida sea LaTeX, de lo contrario puede terminar con una gran cantidad de espacio en blanco en ciertas páginas. El formato de los números de figura/tabla depende de si las secciones están numeradas o no: si `number_sections = TRUE`, estos números serán del formato`X.i`, donde `X` es el número del capítulo, e `i` es un incremento numérico; si las secciones no están numeradas, todas las figuras/tablas serán numeradas secuencialmente a través del libro de 1, 2, ..., N. Note que en cualquiera de los casos, las figuras y tablas se numerarán de forma independiente.

Entre todos los argumentos posibles en `...` , es muy probable que utilice el argumento `css` para proporcionar uno o más archivos CSS personalizados para modificar el estilo CSS por defecto. Hay algunos argumentos de `html_document()` que se han codificado en `gitbook()` y no se pueden cambiar: `toc = TRUE` (debe haber una tabla de contenidos),`theme = null` (no usar ningún tema Bootstrap), y `template` (habrá una plantilla interna GitBook).

Tenga en cuenta que si se cambia `self_contained = TRUE` para hacer páginas HTML independientes, el tamaño total de todos los archivos HTML puede aumentar de manera significativa, ya que hay muchos archivos JS y CSS que se incorporarán en cada archivo HTML.

Además de estas opciones `html_document()`, `gitbook()` tiene otros dos argumentos: `split_by` y `config`. El argumento `split_by` especifica la forma en que desea dividir la salida HTML en varias páginas, y sus posibles valores son:

- `rmd`: utiliza los nombres de archivo base de los archivos de entrada Rmd para crear los archivos HTML, por ejemplo `chapter3.html` para `chapter3.Rmd`;

- `none`: no divide el archivo HTML (el libro será de un sólo archivo HTML);
- `chapter`: divide el archivo por los encabezados de primer nivel;
- `section`: divide el archivo por los encabezados de segundo nivel;
- `chapter+number` y `section+number`: similar a `chapter` y `section`, pero los archivos se numerarán;

Para `chapter` y `section`, los nombres de archivo HTML serán determinados por los identificadores de encabezado, por ejemplo, el nombre de archivo para el primer capítulo con un título del capítulo `# Introducción` será `introducción.html` por defecto. Para `chapter+number` y `section+number`, los números de capítulo/sección se antepondrá a los nombres de archivo HTML, por ejemplo, `1-introduction.html` y`2-1-literature.html`. El identificador del encabezado se genera automáticamente a partir del texto del encabezado por defecto[1], y puede especificar manualmente un identificador utilizando la sintaxis `{#su-propio-id}` después del texto del encabezado, por ejemplo,

```
# Una introducción {#introduccion}

El identificador por defecto es `una-introduccion` pero se cambió a `introduccion`.
```

Por defecto, la bibliografía se divide y los artículos de citas relevantes se ponen en la parte inferior de cada página, para que los lectores no tengan que desplazarse a una página de bibliografía diferente para ver los detalles de las citas. Esta característica se puede desactivar usando `split_bib = FALSE`, en cuyo caso todas las citas se colocan en una página separada.

Hay varias sub-opciones en la opción `config` para poder ajustar algunos detalles en la interfaz de usuario. Vale recordar que todas las opciones de formatos de salida (no sólo para `bookdown::gitbook`) pueden transmitirse a la función de formato si se utiliza la interfaz de línea de comandos `bookdown::render_book()`, o escritos en los

[1]Para ver más detalles sobre cómo se genera automáticamente un identificador, ver la extensión `auto_identifiers` en la documentación del Pandoc `http://pandoc.org/README.html#header-identifiers`

metadatos YAML. A continuación se muestran las sub-opciones predeterminadas de `config` en el formato `gitbook` como metadatos YAML (tenga en cuenta que se inserta debajo de la opción `config`):

```
bookdown::gitbook:
  config:
    toc:
      collapse: subsection
      scroll_highlight: true
      before: null
      after: null
    toolbar:
      position: fixed
    edit:
      link: null
      text: null
    download: null
    search: true
    fontsettings:
      theme: white
      family: sans
      size: 2
    sharing:
      facebook: yes
      twitter: yes
      google: no
      weibo: no
      instapper: no
      vk: no
      all: ['facebook', 'google', 'twitter', 'weibo', 'instapaper']
```

La opción `toc` controla el comportamiento de la tabla de contenido (TOC, por sus siglas en inglés). Puede contraer algunos items inicialmente cuando una página se carga a través de la opción `collapse`. Sus valores posibles son `subsection`,`section`, `none` (o `null`). Esta opción puede ser útil si la TOC es muy larga y tiene más de tres niveles de títulos: `subsection` para colapsar de todos los items del índice

para las subsecciones (X.X.X), `section` entenderá que colapse los items para las secciones (X.X) por lo que sólo los encabezados de nivel superior se muestran inicialmente, y `none` significa que no colapse ningún ítem en la tabla de contenido. Para aquellos ítems de la TOC colapsados, puede alternar su visibilidad haciendo clic en los ítems de jerarquía superior. Por ejemplo, puede hacer clic en un título de capítulo en la tabla de contenido para mostrar/ocultar sus secciones.

La opción `scroll_highlight` en `toc` se utiliza para activar el resaltado de elementos de la TOC a medida que se desplaza el cuerpo del libro (por defecto, esta función está activada). Cada vez que un nuevo encabezado entra en la ventana gráfica actual a medida que se desplaza hacia abajo/arriba, se resaltará el elemento correspondiente en la tabla de contenido de la izquierda.

Como la barra lateral tiene una anchura fija, cuando un elemento en la tabla de contenido se trunca porque el texto del encabezado es demasiado amplio, puede pasar el cursor sobre él para ver una información sobre herramientas que muestran el texto completo.

Es posible añadir más ítems antes y después del TOC utilizando la etiqueta HTML `<li>`. Estos ítems se separarán de la tabla de contenido utilizando un divisor horizontal. Se puede utilizar el carácter de barra vertical `|` por lo que no es necesario omitir ningún carácter en estos ítems siguientes de la sintaxis YAML, por ejemplo,

```
toc:
  before: |
    <li><a href="...">My Awesome Book</a></li>
    <li><a href="...">John Smith</a></li>
  after: |
    <li><a href="https://github.com/rstudio/bookdown">
    Proudly published with bookdown</a></li>
```

A medida que navega a través de diferentes páginas HTML, se preservará la posición de desplazamiento de la TOC. Normalmente verá la barra de desplazamiento en la tabla de contenido en una posición fija, incluso si se desplaza a la siguiente página. Sin embargo, si el ítem de la TOC para el capítulo/sección actual no es

FIGURE 3.1: La barra de herramientas de GitBook.

visible cuando se carga la página, se desplazará automáticamente la tabla de contenido para que sea visible.

El estilo GitBook tiene una barra de herramientas en la parte superior de cada página que le permite cambiar dinámicamente la configuración de los libros. La opción `toolbar` tiene una subopción `position`, que puede tomar valores `fixed` o `static`. El valor por defecto es que la barra de herramientas se mantenga fija en la parte superior de la página, por lo que incluso si se desplaza hacia abajo de la página, la barra de herramientas es aún visible allí. Si se trata de `static`, la barra de herramientas no se desplazará con la página, es decir, una vez que se desplaza lejos, ya no podrá verla.

El primer botón de la barra de herramientas puede cambiar la visibilidad de la barra lateral. También puede pulsar la tecla `s` en el teclado para hacer lo mismo. El estilo GitBook puede recordar el estado de visibilidad de la barra lateral, por ejemplo, si se ha cerrado la barra lateral, permanecerá cerrada la próxima vez que abra el libro. De hecho, el estilo GitBook recuerda muchas otras configuraciones, así como la palabra clave de búsqueda y la configuración de la tipografía.

El segundo botón en la barra de herramientas es el botón de búsqueda. Su combinación de teclas es `F` (Buscar). Cuando se hace clic en el botón, verá un cuadro de búsqueda en la parte superior de la barra lateral. A medida que escribe en el cuadro, la TOC se filtra para mostrar las secciones que coincidan con la palabra clave de búsqueda. Ahora bien, puede utilizar las teclas de flecha `Up`/`Down`

para resaltar la siguiente palabra clave en la página actual. Al hacer clic en el botón de búsqueda de nuevo (o digitar la tecla `F` fuera del cuadro de búsqueda), la palabra clave de búsqueda se vacía y la caja de búsqueda se oculta. Para deshabilitar la búsqueda, establezca la opción `search: no` en `config`.

El tercer botón es para la configuración de fuente/tema. Se puede cambiar el tamaño de la fuente (aumentar o reducir), la familia de fuentes (serif o sans serif), y el tema (`White`, `Sepia`, o `Night`). Estos ajustes se pueden modificar a través de la opción `fontsettings`.

La opción `edit` es la misma que la opción que se mencióno en la sección 4.4. Si no está vacío, un botón de edición se añadirá a la barra de herramientas. Esto fue diseñado para posibles contribuyentes del libro para su edición en GitHub después de hacer clic en el botón y enviar solicitudes de extracción.

Si el libro tiene otros formatos de salida para que los lectores puedan descargarlo, es posible proporcionar la opción `download` para que un botón de descarga se pueda agregar a la barra de herramientas. Esta opción tiene ya sea un vector de caracteres, o una lista de vectores de caracteres con la longitud de cada vector. Cuando se trata de un vector de caracteres, debe ser o bien un vector de nombres de archivo, o las extensiones nombre de archivo, por ejemplo, los dos siguientes ajustes están bien:

```
download: ["book.pdf", "book.epub"]
download: ["pdf", "epub", "mobi"]
```

Cuando sólo proporciona las extensiones del nombre de archivo, el nombre del archivo se deriva del nombre del archivo libro del archivo de configuración `_bookdown.yml` (Sección 4.4). Cuando `download` es `null`, `gitbook()` buscará PDF, EPUB y MOBI en el directorio de salida del libro, y añadirá automáticamente la opción `download`. Si lo que desea es suprimir el botón de descarga, usar `download: no`. Todos los archivos que los lectores descarguen se muestran en un menú desplegable, y las extensiones de nombre de archivo se utilizan como en el texto del menú. Cuando el

único formato disponible para los lectores para descargar es PDF, el botón de descarga será un solo botón PDF en lugar de un menú desplegable.

Una forma alternativa para el valor de la opción `download` es una lista de vectores de longitud 2, por ejemplo,

```
download: [["book.pdf", "PDF"], ["book.epub", "EPUB"]]
```

También puede escribirse como:

```
download:
  - ["book.pdf", "PDF"]
  - ["book.epub", "EPUB"]
```

Cada vector en la lista consiste en el nombre del archivo y el texto que se mostrará en el menú. En comparación con la primera forma, esta le permite personalizar el texto del menú, por ejemplo, puede tener dos copias diferentes de PDF para los lectores para descargar y tendrá que hacer los elementos de menú diferente.

A la derecha de la barra de herramientas, hay algunos botones para compartir el enlace en sitios web de redes sociales como Twitter, Facebook y Google+. Puede utilizar la opción `sharing` para decidir qué botones activar. Si desea deshacerse de estos botones en su totalidad, sólo tiene que utilizar `sharing: null` (o `no`).

Por último, hay algunas opciones de más alto nivel en los metadatos YAML que se puede utilizar en la plantilla GitBook HTML a través de Pandoc. Puede que no tengan efectos visibles claros sobre la salida HTML, pero pueden ser útiles cuando se implementa la salida HTML como una página web. Estas opciones incluyen:

- `description`: Una cadena de caracteres que se escribe en el atributo `content` del tag `<meta name="description" content="">` en el encabezado del HTML (si falta, se usará el título del libro). Esto puede ser útil para efectos de optimización de motores de

búsqueda (SEO). Debe ser texto plano sin ningún formato Markdown talcomo *itálica* o **negrita**;
- `url`: La URL de la página web del libro, por ejemplo, `https\://bookdown.org/yihui/bookdown/`[2];
- `github-repo`: El repositorio GitHub del libro de la forma: `user/repo`;
- `cover-image`: la ruta de la imagen de la carátula del libro;
- `apple-touch-icon`: Una ruta a un ícono (e.g., una imagen PNG). Esto funciona sólo para iOS: cuando la página web se añade a la pantalla de inicio, el vínculo se representa por este ícono.
- `apple-touch-icon-size`: El tamaño del ícono (por defecto, 152 x 152 pixels).
- `favicon`: Una ruta al "ícono favorito". Típicamente este ícono se muestra en la barra de dirección del buscador, o al frente de la página del título, si el buscador lo soporta.

Abajo se muestra un ejemplo de metadatos YAML (de nuevo note que existen opciones *top-level*):

```
---
title: "Un libro impresionante"
author: "Daniel Rodríguez"
description: "Este libro introduce la teoría ABC, y ..."
url: "https\://bookdown.org/john/awesome/"
github-repo: "john/awesome"
cover-image: "images/cover.png"
apple-touch-icon: "touch-icon.png"
apple-touch-icon-size: 120
favicon: "favicon.ico"
---
```

Un buen efecto para establecer `description` y `cover-image` es que cuando se comparte el enlace de su libro sobre algunos sitios web de redes sociales como Twitter, el enlace se puede ampliar de forma

[2]El backslash antes de : se debe a errores técnicos: se quiere prevenir que Pandoc traduzca el vínculo a código HTML `<a href="..."></a>`.

automática a una tarjeta de la imagen del libro y la descripción del libro.

3.1.2 El estilo Bootstrap

Si ha utilizado R Markdown antes, debe estar familiarizado con el estilo Bootstrap (`http://getbootstrap.com`), que es el estilo por defecto de la salida HTML de R Markdown. La función del formato de salida en **rmarkdown** es `html_document()`, y se tiene un formato correspondiente `html_book()` en **bookdown** usando `html_document()` como formato de base. De hecho, hay un formato más general `html_chapters()` en **bookdown** y `html_book()` es sólo su caso especial:

```
html_chapters(toc = TRUE, number_sections = TRUE, fig_caption = TRUE,
  lib_dir = "libs", template = bookdown_file("templates/default.html"),
  ..., base_format = rmarkdown::html_document, split_bib = TRUE,
  page_builder = build_chapter, split_by = c("section+number",
    "section", "chapter+number", "chapter", "rmd", "none"))
```

Tenga en cuenta que tiene un argumento `base_format` que tiene una función base de formato de salida, y `html_book()` es, básicamente, `html_chapters(base_format = rmarkdown::html_document)`. Todos los argumentos de `html_book()` se pueden utilizar en `html_chapters()`:

```
html_book(...)
```

Esto significa que puede utilizar la mayoría de los argumentos de `rmarkdown::html_document`, como `toc` (si desea mostrar la tabla de contenidos), `number_sections` (si quiere numerar encabezados de sección), y así sucesivamente. Una vez más, vale la pena visitar la página de ayuda de `rmarkdown::html_document` para ver la lista completa de posibles opciones. Tenga en cuenta que el argumento `self_contained` no es modificable a `FALSE` internamente, por lo que no puede cambiar el valor de este argumento. Ya hemos explicado el argumento `split_by` en la sección anterior.

La argumentos `template` y `page_builder` son para usuarios avanzados, y no es necesario entenderlos a menos que tenga una imperiosa necesidad de personalizar la salida HTML, y la gran variedad de opciones de que dispone `rmarkdown::html_document()` no le muestren el resultado deseado.

Si desea pasar una plantilla HTML diferente al argumento `template`, la plantilla debe contener tres pares de comentarios HTML, y cada comentario tiene que estar en una línea separada:

- `<!--bookdown:title:start-->` y `<!--bookdown:title:end-->` para marcar la sección de título del libro. Esta sección se pondrá en la primera página del libro compilado.
- `<!--bookdown:toc:start-->` y `<!--bookdown:toc:end-->` para marcar la sección de tabla de contenidos. Esta sección se pondrá en todas las páginas HTML.
- `<!--bookdown:body:start-->` y `<!--bookdown:body:end-->` para marcar el cuerpo HTML del libro, y el cuerpo HTML se dividirá en múltiples páginas separadas. Recuerde que se combinan todos los archivos Mrkdown y R Markdown, se compilan en un solo archivo HTML y se dividen.

Puede abrir la plantilla HTML por defecto para ver donde se insertaron estos comentarios:

```
bookdown:::bookdown_file('templates/default.html')
# puede usar file.edit() para abrir este archivo
```

Una vez que sepa cómo **bookdown** trabaja internamente para generar varias páginas HTML de salida, será más fácil entender el argumento `page_builder`, que es una función para componer cada página HTML individual utilizando los fragmentos HTML extraídos de la ficha de comentario anterior. El valor por defecto de `page_builder` es una función `build_chapter` en **bookdown** , y su código fuente es relativamente simple (ignore esas funciones internas como `button_link()`):

```
build_chapter = function(
  head, toc, chapter, link_prev, link_next, rmd_cur, html_cur, foot
) {
  # add a has-sub class to the <li> items that has sub lists
  toc = gsub('^(<li>)(.+<ul>)$', '<li class="has-sub">\\2', toc)
  paste(c(
    head,
    '<div class="row">',
    '<div class="col-sm-12">',
    toc,
    '</div>',
    '</div>',
    '<div class="row">',
    '<div class="col-sm-12">',
    chapter,
    '<p style="text-align: center;">',
    button_link(link_prev, 'Previous'),
    edit_link(rmd_cur),
    button_link(link_next, 'Next'),
    '</p>',
    '</div>',
    '</div>',
    foot
  ), collapse = '\n')
}
```

Básicamente, esta función toma un número de componentes como el encabezado de HTML, la tabla de contenido, el cuerpo del capítulo, etc., y se espera que devuelva una cadena de caracteres que es el código fuente HTML de una página HTML completa. Es posible manipular todos los componentes de esta función utilizando las funciones de procesamiento de textos como `gsub()` y `paste()`.

Lo que el constructor de página por defecto hace es poner la tabla de contenido en la primera fila, el cuerpo en la segunda fila, los botones de navegación en la parte inferior del cuerpo, y concatenarlos con el encabezado y el final del HTML. Aquí hay un boceto

del código fuente HTML que pueden ayudarle a entender la salida de `build_chapter()`:

```
<html>
  <head>
    <title>A Nice Book</title>
  </head>
  <body>

    <div class="row">TOC</div>

    <div class="row">
      CHAPTER BODY
      <p>
        <button>PREVIOUS</button>
        <button>NEXT</button>
      </p>
    </div>

  </body>
</html>
```

Para todas las páginas HTML, la principal diferencia es el cuerpo del capítulo, y la mayoría del resto de elementos son los mismos. La salida por defecto de `html_book()` incluirá la Bootstrap CSS y los archivos JavaScript en la etiqueta `<head>`.

La tabla de contenidos se utiliza a menudo para fines de navegación. En el estilo GitBook, el índice se mostrará en la barra lateral. Para el estilo Bootstrap, no se aplica un estilo especial a ella, por lo que se muestra como una lista desordenada normal (en la etiqueta HTML `<ul>`). Es fácil dar vuelta a esta lista en una barra de navegación con algunas técnicas de CSS. Se ha proporcionado un archivo CSS `toc.css` en este paquete que se puede utilizar, y se puede encontrar aquí: `https://github.com/rstudio/bookdown/blob/master/inst/examples/css/toc.css`

Puede copiar este archivo en el directorio raíz de su libro, y aplicarla a la salida HTML a través de la opción `css`, por ejemplo,

```
---
output:
  bookdown::html_book:
    toc: yes
    css: toc.css
---
```

Hay muchas maneras posibles de convertir listas `<ul>` a menús de navegación si hace una pequeña búsqueda en la web, puede elegir un estilo de menú que le guste. El archivo `toc.css` que se mencionó anteriormente es un estilo de menú con textos blancos sobre un fondo negro, y es compatible con los submenús (por ejemplo, los títulos de las secciones se muestran como menús desplegables bajo los títulos de los capítulos).

De hecho, usted puede deshacerse del estilo Bootstrap en `html_document()` si se establece la opción `theme` como `null`, y queda libre de aplicar estilos arbitrarios en la salida HTML utilizando la opción `css` (y posiblemente la opción `includes` si desea incluir contenido arbitrario en el encabezado/pie del HTML).

3.1.3 El estilo Tufte

Al igual que el estilo Bootstrap, el estilo Tufte es proporcionado por un formato de salida `tufte_html_book()`, que es también un caso especial de `html_chapters()` usando `tufte::tufte_html()` como el formato de base. Por favor, vea el paquete **Tufte** (Xie and Allaire, 2016) si no está familiarizado con el estilo Tufte. Básicamente, es un diseño con una columna principal a la izquierda y una columna al margen, dispuesta a la derecha. El cuerpo principal está en la columna principal, y la columna al margen se utiliza para poner notas al pie, notas al margen, referencias y figuras al margen, etc.

Todos los argumentos de `tufte_html_book()` tienen exactamente el mismo significado que en `html_book()`, por ejemplo, también se

puede personalizar el CSS a través de la opción `css`. Hay algunos elementos que son específicos del estilo Tufte, tales como notas al margen, figuras de margen, y figuras de ancho completo. Estos elementos requieren una sintaxis especial para su generación, por lo que se recomienda consultar la documentación del paquete **Tufte**. Note que no es necesario hacer nada especial para las notas al pie y las referencias (sólo tiene que utilizar la sintaxis Markdown normal `^[nota]` y `[@citas]`), ya que éstas se ponen automáticamente al margen. Un breve ejemplo YAML del formato `tufte_html_book`:

```
---
output:
  bookdown::tufte_html_book:
    toc: yes
    css: toc.css
---
```

3.2 LaTeX/PDF

Se recomienda encarecidamente el uso de un formato de salida HTML en lugar de LaTeX cuando se hace un libro, ya que no tendrá que estar distrayéndose por los detalles de composición tipográfica que pueden molestar mucho si se revisa constantemente la salida PDF de un libro. Deje el trabajo de cuidar la composición tipográfica sólo para el final (idealmente después de que realmente haya terminado el contenido del libro).

El formato de salida de LaTeX/PDF es proporcionado por `pdf_book()` en **bookdown**. No hay una diferencia significativa entre `pdf_book()` y el `pdf_document()` del formato **rmarkdown**. El objetivo principal de `pdf_book()` es aclarar las etiquetas y referencias cruzadas escritas usando la sintaxis descrita en las secciones 2.4, 2.5, y 2.6. Si el único formato de salida que desea para un libro es el LaTeX/PDF, puede utilizar la sintaxis específica de LaTeX,

por ejemplo `\label{}` para etiquetar las figuras/tablas/secciones, y `\ref{}` para sus referencias cruzadas, puesto que Pandoc admite comandos de LaTeX en Markdown. Sin embargo, la sintaxis LaTeX no es portátil a otros formatos de salida, tales como HTML y libros electrónicos. Es por eso que se ha introducido la sintaxis `(\#label)` para etiquetas y `\@ref(label)` para referencias cruzadas.

Hay algunas opciones YAML de alto nivel que se aplicarán a la salida LaTeX. Para un libro, puede cambiar la clase de documento predeterminado a `book` (el valor predeterminado es`article`), y especificar un estilo de bibliografía requerida por su editor. Un breve ejemplo YAML:

```
---
documentclass: book
bibliography: [book.bib, packages.bib]
biblio-style: apalike
---
```

El formato `pdf_book()` es un formato general como `html_chapters`, que también tiene un argumento `base_format`:

```
pdf_book(toc = TRUE, number_sections = TRUE, fig_caption = TRUE,
  ..., base_format = rmarkdown::pdf_document, toc_unnumbered = TRUE,
  toc_appendix = FALSE, toc_bib = FALSE, quote_footer = NULL,
  highlight_bw = FALSE)
```

Puede cambiar la función `base_format` a otras funciones de formato de salida, y **bookdown** ha proporcionado una sencilla función de contenedor `tufte_book2()`, que es básicamente `pdf_book(base_format = Tufte::tufte_book)`, para producir un libro PDF utilizando el estilo PDF Tufte (de nuevo, ver el **tufte** paquete).

3.3 E-Books

Actualmente **bookdown** proporciona dos formatos de libros electrónicos, EPUB y MOBI. Los libros en estos formatos se pueden leer en dispositivos como teléfonos inteligentes, tabletas o lectores electrónicos especiales como el Kindle.

3.3.1 EPUB

Para crear un libro EPUB, puede utilizar el formato `epub_book()`. Este formato tiene algunas opciones en común con `rmarkdown::html_document()`:

```
epub_book(fig_width = 5, fig_height = 4, dev = "png", fig_caption = TRUE,
  number_sections = TRUE, toc = FALSE, toc_depth = 3,
  stylesheet = NULL, cover_image = NULL, metadata = NULL,
  chapter_level = 1, epub_version = c("epub3", "epub"),
  md_extensions = NULL, pandoc_args = NULL)
```

La opción `toc` se deshabilita debido a que el lector de libros electrónicos puede determinar a menudo una tabla de contenido de forma automática desde el libro, por lo que no es necesario añadir unas pocas páginas para la tabla de contenido. Hay algunas opciones específicas para EPUB:

- `stylesheet`: Es similar a la opción `css` en formatos de salida HTML, y se puede personalizar la apariencia de los elementos usando CSS;
- `cover_image`: La ruta de la imagen de la portada del libro;
- `metadata`: La ruta de acceso a un archivo XML para los metadatos del libro (ver documentación Pandoc para más detalles);
- `chapter_level`: Internamente un libro EPUB es una serie de archivos "capítulo", y esta opción determina el nivel por el cual el libro se divide en estos archivos. Esto es similar al argumento `split_by` de formatos de salida HTML que hemos mencionado en

la sección 3.1, con la diferencia de que un libro EPUB es un solo archivo, y no se verán estos archivos "capítulo" directamente. El nivel predeterminado es el primer nivel, y si lo ajusta a 2, significa que el libro será organizado por la sección de archivos internos, lo que puede hacer al lector más rápido para cargar el libro;

- `epub_version`: Versión 3 o 2 de EPUB;

Un libro EPUB es esencialmente una colección de páginas HTML, por ejemplo, se pueden aplicar reglas CSS a sus elementos, incrustar imágenes, insertar expresiones matemáticas (debido a MathML está soportado parcialmente), etc. Las leyendas de las figuras/tablas, referencias cruzadas, bloques personalizados, y las citas mencionadas en el capítulo 2 también deben trabajar en EPUB. Es posible comparar la salida EPUB de este libro a la salida HTML, con la única diferencia importante del aspecto visual.

Hay varios lectores EPUB disponibles, incluyendo Calibre (`https://www.calibre-ebook.com`), iBooks de Apple y Google Play Books.

3.3.2 MOBI

Los libros electrónicos MOBI pueden leerse en dispositivos Kindle de Amazon. Pandoc no es compatible con la salida MOBI de forma nativa, pero Amazon ha proporcionado una herramienta llamada KindleGen (`https://www.amazon.com/gp/feature.html?docId=1000765211`) para crear libros MOBI de otros formatos, incluyendo EPUB y HTML. Hemos proporcionado una simple función de contenedor `KindleGen()` en **bookdown** para llamar KindleGen con el fin de convertir un libro EPUB a MOBI. Esto requiere que se descargue primero KindleGen, y asegurarse de que el ejecutable KindleGen se pueda encontrar a través de la variable de entorno de sistema `path`.

Otra herramienta para convertir EPUB a MOBI es proporcionada por Calibre. A diferencia de KindleGen, Calibre es de código abierto y libre, y soporta la conversión entre muchos más for-

matos. Por ejemplo, se puede convertir HTML a EPUB, documentos de Word a MOBI, etc. La función `calibre()` en **bookdown** es una función de contenedor de la utilidad de línea de comandos `ebook-convert` en Calibre. Del mismo modo, es necesario asegurarse de que el ejecutable `ebook-convert` se pueda encontrar a través de la variable de entorno `path`. Si utiliza OS X, puede instalar tanto KindleGen como Calibre a través de Homebrew-Cask (`https://caskroom.github.io`), por lo que no necesita preocuparse por el problema `path`.

3.4 Un documento sencillo

A veces, puede que no desee escribir un libro, sino un simple artículo de formato largo o un reporte. Por lo general, lo que se hace es llamar a `rmarkdown::render()` con un determinado formato de salida. Las principales características que faltan son la numeración automática de títulos de figuras/tablas y referenciación cruzada de figuras/tablas/secciones. Se han descartado estas características en **bookdown** para que puedan utilizarse sin tener que preparar un libro de varios archivos Rmd.

Las funciones `html_document2()`, `tufte_html2()`, `pdf_document2()`, `word_document2()`, `tufte_handout2()` y `tufte_book2()` están diseñadas para estos propósitos. Si se procesa un documento R Markdown con el formato de salida, por ejemplo, `bookdown::html_document2`, se obtendrán números de figuras/tablas y podrán referenciarse cruzadamente en una simple página HTML usando la sintaxis descrita en el capítulo 2.

Las funciones de formato de salida HTML y PDF anteriores son básicamente envolturas de formatos de salida `bookdown::html_book` y `bookdown::pdf_book`, en el sentido de que cambiaron el argumento `base_format`. Por ejemplo, puede echar un vistazo al código fuente de `pdf_document2`:

```
bookdown::pdf_document2
```

```
## function (...)
## {
##     pdf_book(..., base_format = rmarkdown::pdf_document)
## }
## <environment: namespace:bookdown>
```

Después de conocer este hecho, puede aplicar la misma idea a otros formatos de salida mediante el `base_format` apropiado. Por ejemplo, puede portar las características **bookdown** al formato `jss_article` en el paquete **rticles** (**?**) usando los metadatos YAML:

```
output:
  bookdown::pdf_book:
    base_format: rticles::jss_article
```

A continuación, podrá utilizar todas las funciones que hemos introducido en el capítulo 2.

Aunque el formato `gitbook()` fue diseñado principalmente para libros, en realidad se puede también aplicar a un simple documento R Markdown. La única diferencia es que no habrá ningún botón de búsqueda en la salida de una página, ya que simplemente puede utilizar la herramienta de búsqueda de su navegador web para buscar texto (por ejemplo, pulse `Ctrl + F` o `Command + F`). También puede configurar la opción `split_by` con `none` para generar una página de salida única, en cuyo caso no habrá ningún botón de navegación, ya que no hay otras páginas para navegar. Aun es posible generar archivos HTML de varias páginas si se desea. Otra opción posible es utilizar ‘self_contained = TRUE cuando sólo es una página de salida.

4

Personalización

Como se mencionó al comienzo de este libro, se asume que el usuario tiene algunos conocimientos básicos sobre R Markdown, para centrarse en la introducción de las características de **bookdown** en lugar de las de **rmarkdown**. De hecho, R Markdown es altamente personalizable, y hay muchas opciones que se pueden utilizar para personalizar el documento de salida. Dependiendo de qué tanto quiera personalizar la salida, es posible utilizar algunas opciones simples en los metadatos YAML, o simplemente reemplazar toda la plantilla Pandoc.

4.1 Opciones YAML

Para la mayoría de los tipos de formatos de salida, puede personalizar los estilos de resaltado de sintaxis utilizando la opción `highlight` del formato específico. En la actualidad, los estilos posibles son `default`, `tango`, `pygments`, `kate`, `monochrome`, `espresso`, `zenburn`, and `haddock`. Por ejemplo, se puede elegir el estilo `tango` para el formato de `gitbook`:

```
---
output:
  bookdown::gitbook:
    highlight: tango
---
```

Para formatos de salida HTML, es preferible usar la opción `css`

con el fin de proporcionar sus propias hojas de estilo CSS para personalizar la apariencia de los elementos HTML. Hay una opción `includes` que se aplica a más formatos, incluyendo HTML y LaTeX. La opción `includes` permite insertar contenido personalizado antes y/o después del cuerpo de salida. Tiene tres sub-opciones: `in_header`, `before_body`, y `after_body`. Es necesario conocer la estructura básica de un documento HTML o LaTeX para entender estas opciones. El código fuente de un documento HTML luce de la siguiente forma:

```
<html>

  <head>
  <!-- head content here, e.g. CSS and JS -->
  </head>

  <body>
  <!-- body content here -->
  </body>

</html>
```

La opción `in_header` toma una ruta de archivo y lo inserta en la etiqueta `<head>`. Se insertará el archivo `before_body` justo debajo de la apertura `<body>`, y `after_body` se inserta antes de la etiqueta de cierre `</body>`.

Un documento de origen LaTeX tiene una estructura similar:

```
\documentclass{book}

% LaTeX preamble
% insert in_header here

\begin{document}
% insert before_body here
```

```
% body content here

% insert after_body here
\end{document}
```

La opción `includes` es muy útil y flexible. Para la salida HTML, quiere decir que se puede insertar código HTML arbitrario a la salida. Por ejemplo, cuando se tienen expresiones matemáticas de LaTeX compiladas a través de la biblioteca MathJax en la salida HTML, y desea enumerar las ecuaciones en el entorno `equation`, puede crear un archivo de texto que contenga el siguiente código:

```
<script type="text/x-mathjax-config">
MathJax.Hub.Config({
  TeX: { equationNumbers: { autoNumber: "AMS" } }
});
</script>
```

Asumiendo que el archivo se denomina `mathjax-number.html`, y que está en el directorio raíz de su libro (el directorio que contiene todos los archivos Rmd). Puede insertar este archivo en el encabezado del HTML a través de la opción `in_header`, por ejemplo,

```
---
output:
  bookdown::gitbook:
    includes:
      in_header: mathjax-number.html
---
```

Si se utiliza el formato de salida HTML `html_book` o `gitbook` en **bookdown**, esto ya se ha establecido en las plantillas HTML, por lo que no es realmente necesario insertar un archivo de este tipo en el encabezado HTML. Sólo es necesario que lo haga para otros formatos de salida HTML.

Del mismo modo, si usted está familiarizado con LaTeX, puede

agregar código LaTeX arbitrariamente al preámbulo. Esto significa que puede utilizar cualquier paquete de LaTeX y configurar las opciones de cualquier paquete para su libro. Por ejemplo, este libro utiliza la opción `in_header` para utilizar unos pocas paquetes de LaTeX adicionales, tales como **booktabs** (para mejorar el aspecto de las tablas) y **longtable** (para las tablas que se extienden a través de múltiples páginas), y arreglar un problema XeLaTeX en el que el vínculo de los gráficos no funcionen:

```
\usepackage{booktabs}
\usepackage{longtable}

\ifxetex
  \usepackage{letltxmacro}
  \setlength{\XeTeXLinkMargin}{1pt}
  \LetLtxMacro\SavedIncludeGraphics\includegraphics
  \def\includegraphics#1#{% #1 catches optional stuff (star/opt. arg.)
    \IncludeGraphicsAux{#1}%
  }%
  \newcommand*{\IncludeGraphicsAux}[2]{%
    \XeTeXLinkBox{%
      \SavedIncludeGraphics#1{#2}%
    }%
  }%
\fi
```

El código LaTeX anterior se guarda en un archivo `preamble.tex`, y los metadatos YAML se ven así:

```
---
output:
  bookdown::pdf_book:
    includes:
      in_header: preamble.tex
---
```

4.2 Temas

A veces es posible que desee cambiar el tema general de la salida, y por lo general esto se puede hacer a través de la opción `in_header` descrita en el apartado anterior, o mediante la opción `css` si la salida es HTML. Algunos formatos de salida tienen sus temas únicos, tales como `gitbook`, `tufte_html_book`, y `tufte_book2`, y es preferible no modificar estos temas demasiado. En comparación, los formatos de salida `html_book()` y `pdf_book()` no están vinculados a temas particulares y son más personalizables.

Como se mencionó en la sección 3.1.2, el estilo predeterminado de `html_book()` es el estilo Bootstrap. El estilo Bootstrap en realidad tiene varios temas incorporados que se pueden utilizar, incluyendo `default`, `cerulean`, `journal`, `flatly`, `readable`, `spacelab`, `united`, `cosmo`, `lumen`, `paper`, `sandstone`, `simplex`, and `yeti`. Puede establecer el tema a través de la opción `theme`, por ejemplo,

```
---
output:
  bookdown::html_book:
    theme: united
---
```

Si no le gusta ninguno de estos estilos de Bootstrap, se puede establecer `theme` como `null`, y aplicar su propio CSS a través de la opción `css` o `includes`.

Para `pdf_book()`, además de la opción `in_header` mencionada en el apartado anterior, otra posibilidad es cambiar la clase de documento. Hay muchos posibles clases de LaTeX para libros, tales como **memoir** (`https://www.ctan.org/pkg/memoir`), **amsbook** (`https://www.ctan.org/pkg/` amsbook), KOMA-script (`https://www.ctan.org/pkg/koma-script`), etc. Una breve muestra de los metadatos YAML especificando la clase `scrbook` del paquete KOMA-Script es:

```
---
documentclass: scrbook
output:
  bookdown::pdf_book:
    template: null
---
```

Algunos editores (por ejemplo, Springer y Chapman & Hall/CRC) tienen sus propios archivos de estilo de LaTeX o de class. Puede intentar cambiar la opción `documentclass` para utilizar sus clases de documento, aunque generalmente no es tan simple como eso. Puede terminar usando `in_header`, o incluso diseñar una plantilla personalizada Pandoc LaTeX para dar cabida a estas clases de documentos.

Tenga en cuenta que cuando se cambia `documentclass`, es posible especificar un argumento adicional Pandoc`--chapters` de modo que Pandoc reconozca que los encabezados de primer nivel deben ser tratados como capítulos en lugar de secciones (este es el valor predeterminado cuando `documentclass` es `book`), por ejemplo,

```
documentclass: krantz
output:
  bookdown::pdf_book:
    pandoc_args: --chapters
```

4.3 Plantillas

Cuando Pandoc convierte Markdown a otro formato de salida, utiliza una plantilla bajo la manga. La plantilla es un archivo de texto plano que contiene algunas variables de la forma `$variable$`. Estas variables se reemplazarán por sus valores generados mediante Pan-

doc. A continuación se muestra una breve plantilla para la salida HTML:

```
<html>
  <head>
    <title>$title$</title>
  </head>

  <body>
  $body$
  </body>
</html>
```

Tiene dos variables `title` y `body`. El valor de `title` viene del campo `title` de los metadatos YAML, y `body` es el código HTML generado por el cuerpo del documento Markdown. Por ejemplo, supongamos que tenemos un documento Markdown:

```
---
title: Un libro chévere
---

# Introducción

Este es una **berraquera** de libro!
```

Si usamos la plantilla anterior para generar un documento HTML, su código fuente sería así:

```
<html>
  <head>
    <title>Una berraquera de libro</title>
  </head>

  <body>
```

```
  <h1>Introducción</h1>

  <p>Este es una <strong>berraquera</strong> de libro!</p>

  </body>
</html>
```

Las plantillas actuales HTML, LaTeX, y EPUB son más complicadas pero la idea es la misma. Usted sólo necesita saber qué variables están disponibles: algunas variables están incorporados en las variables Pandoc, y algunas se puede definir por los usuarios en los metadatos YAML, o se pasan de la opción de línea de comandos `-V` o `--variable`. Algunas variables sólo tienen sentido para determinados formatos de salida, por ejemplo, la variable `documentclass` sólo se utiliza en la producción de documentos en LaTeX. Consulte la documentación de Pandoc para aprender más sobre estas variables, y puede encontrar todas las plantillas Pandoc por defecto en el repositorio GitHub `https://github.com/jgm/pandoc-templates`.

Tenga en cuenta que para la salida HTML, **bookdown** requiere algunas fichas adicionales de comentarios en la plantilla que se explicó en la sección 3.1.2.

4.4 Configuración

Se han mencionado los `rmd_files` en la Sección 1.3, pero hay más cosas que se pueden configurar para un libro en `_bookdown.yml`:

- `book_filename`: el nombre del archivo principal Rmd, es decir, el archivo Rmd a partir del cual se fusionan todos los capítulos; por defecto, se llama `_main.Rmd`.
- `before_chapter_script`: una o varias secuencias de comandos en R para ser ejecutadas antes de cada capítulo, por ejemplo, es posible que desee limpiar el área de trabajo antes de compilar cada

capítulo, en cuyo caso se puede usar `rm(list = ls(all = TRUE))` en el script de R.

- `after_chapter_script`: similar a `before_chapter_script`, y el script de R se ejecuta después de cada capítulo.
- `edit`: un enlace que los colaboradores pueden utilizar para editar el documento original Rmd de la página actual; esto fue diseñado principalmente para los repositorios de Github, ya que es fácil editar archivos arbitrarios de texto sin formato en Github incluso en los repositorios de otras personas (si usted no tiene acceso a escritura en el repositorio, Github se trinchará automáticamente y le permitirá enviar una solicitud de extracción después de haber terminado de editar el archivo). Este enlace debe tener `s%` dentro de él, que será sustituido por el nombre de archivo actual Rmd para cada página.
- Opcionalmente, puede tener un campo `text` bajo el campo `edit` para especificar el texto al que está unido el enlace.
- `rmd_subdir`: si desea buscar archivos de origen del libro de Rmd en subdirectorios (por defecto, sólo el directorio raíz se busca).

- `output_dir`: el directorio de salida del libro; esta configuración es leída y utilizada por `render_book()`.
- `clean`: un vector de archivos y directorios a ser limpiados por la función `clean_book()`.

Aquí hay un ejemplo de `_bookdown.yml`:

```
book_filename: "mi-libro.Rmd"
chapter_name: "CAPÍTULO "
before_chapter_script: ["script1.R", "script2.R"]
after_chapter_script: "script3.R"
edit:
  link: https://github.com/rstudio/bookdown/edit/master/inst/examples/%s
  text: "Edit"
```

```
output_dir: "book-output"
clean: ["my-book.bbl", "R-packages.bib"]
```

4.5 Internacionalización

Si el idioma de su libro no es Inglés, se necesita traducir ciertas palabras y frases de inglés a su lengua materna, tales como "Figure" y "Table" cuando las figuras/tablas se enumeren automáticamente en la salida HTML. La internacionalización puede no ser un problema para la salida de LaTeX, ya que algunos paquetes de LaTeX pueden traducir automáticamente estos términos al idioma local, tal como el paquete **ctexcap** al chino.

Para la salida que no sea de látex, se puede establecer el campo `language` en el archivo de configuración_`bookdown.yml`. Actualmente los ajustes por defecto son:

```
language:
  label:
    fig: 'Figure '
    tab: 'Table '
    eq: 'Equation '
    thm: 'Theorem '
    lem: 'Lemma '
    def: 'Definition '
    cor: 'Corollary '
    prp: 'Proposition '
    ex: 'Example '
    proof: 'Proof. '
    remark: 'Remark. '
  ui:
```

```
    edit: Edit
    chapter_name: ''
```

Por ejemplo, si desea `FIGURE x.x` en lugar de`Figure x.x`, puede cambiar `fig` a `"Figure"`:

```
language:
  label:
    fig: "FIGURE "
```

Los campos bajo `ui` se utilizan para especificar algunos términos en la interfaz de usuario. El campo `edit` especifica el texto asociado con el enlace `edit` en `_bookdown.yml` (Sección 4.4). El campo `chapter_name` puede ser una cadena de caracteres que se antepone a los números de capítulos en títulos de los capítulos (por ejemplo, "Capítulo"), o una función de R que toma el número de capítulo como entrada y devuelve una cadena como el nuevo número de capítulo (por ejemplo, `!expr function(i) paste('Capítulo', i)`). Si se trata de un vector de caracteres de longitud 2, el prefijo del título de capítulo será `paste0(chapter_name[1], i, chapter_name[2])`, donde `i` es el número de capítulo.

Hay una advertencia cuando se escribe en un idioma que utiliza caracteres de varios bytes, como el chino, japonés y coreano (CJK): Pandoc no puede generar identificadores de los encabezados de las secciones que son caracteres CJK puros, por lo que no será capaz de hacer referencias cruzadas a las secciones (no tienen etiquetas), a menos que asigne manualmente identificadores, añadiendo `{#identifier}` a la cabecera de sección, donde `identifier` es un identificador arbitrario que elija.

5

Edición

En este capítulo, se explica cómo editar, crear, previsualizar y presentar el libro localmente. Se puede utilizar cualquier editor de texto para editar el libro, y mostraremos algunos consejos sobre la IDE RStudio. Vamos a introducir las funciones de R subyacentes para la construcción, previsualización, y presentación del libro antes de presentar al editor, quien realmente entiende lo que sucede detrás de cámaras cuando se hace clic en un botón determinado en la IDE RStudio, y también puede personalizar otros editores llamando estas funciones.

5.1 Construyendo el libro

Para conformar todos los archivos Rmd en un libro, puede llamar a la función `render_book()` en **bookdown**. A continuación se presentan los argumentos de `render_book()`:

```
render_book(input, output_format = NULL, ..., clean = TRUE,
  envir = parent.frame(), clean_envir = !interactive(),
  output_dir = NULL, new_session = NA, preview = FALSE,
  encoding = "UTF-8")
```

El argumento más importante es `output_format`, que puede tomar una cadena de caracteres del formato de salida (por ejemplo,`'bookdown::gitbook'`). Puede dejar este argumento vacío, y el formato de salida por defecto será el primer formato de salida especificado en los metadatos YAML del primer archivo Rmd o un

archivo YAML por separado `_output.yml`, como se mencionó en la sección 4.4. Si va a generar múltiples formatos de salida para un libro, se recomienda especificar todos los formatos en `_output.yml`.

Una vez que todos los formatos se especifican en el `_output.yml`, es fácil escribir un script en R o en Shell o Makefile para compilar el libro. A continuación se muestra un ejemplo sencillo del uso de un script de Shell para compilar un libro a HTML (con el estilo GitBook) y PDF:

```
#!/usr/bin/env Rscript

bookdown::render_book("index.Rmd", "bookdown::gitbook")
bookdown::render_book("index.Rmd", "bookdown::pdf_book")
```

El script de Shell no funciona en Windows (no es estrictamente cierto), pero espero que se entienda la idea.

El argumento `...` se pasa a la función de formato de salida. Los argumentos `clean` y `envir` se pasan a `rmarkdown::render()`, para decidir si se deben limpiar los archivos intermedios, y especificar el entorno para evaluar el código en R, respectivamente.

El directorio de salida del libro se puede especificar a través del argumento `output_dir`. Por defecto, el libro se genera en el directorio `_book`. Esto también se puede ajustar por medio del campo `output_dir` en el archivo de configuración `_bookdown.yml`, de modo que no se tiene que especificar varias veces para la presentación de un libro a múltiples formatos de salida. El argumento `new_session` se explicó en la sección @ref(#new-session). Al configurar `preview = TRUE`, sólo los archivos Rmd especificados en el argumento `input` se compilan, lo que puede ser conveniente al previsualizar un determinado capítulo, ya que no se vuelve a compilar el libro entero, sino cuando se publica un libro, este argumento sin duda se debe establecer a FALSE.

Al reproducir el libro en múltiples formatos en la misma sesión de R, hay que tener cuidado porque el siguiente formato podrá tener acceso a los objetos R creados a partir del formato anterior. Se

recomienda reproducir el libro con un ambiente limpio para cada formato de salida. El argumento `clean_envir` se puede utilizar para limpiar todos los objetos en el entorno especificado por `envir`. De forma predeterminada, es `TRUE` para las sesiones de R no interactivos (por ejemplo, en modo batch). Note que incluso `clean_envir = TRUE` en realidad no garantiza que la sesión R esté limpia. Por ejemplo, los paquetes cargados cuando se reproduce el formato anterior permanecerán en la sesión para el siguiente formato de salida. Para asegurarse de que cada formato se reproduce en una sesión de R completamente limpia, debe iniciar una nueva sesión de R para generar cada formato, por ejemplo, utilizar la línea de comandos

```
Rscript -e "bookdown::render_book('index.Rmd', 'bookdown::gitbook')"
Rscript -e "bookdown::render_book('index.Rmd', 'bookdown::pdf_book')"
```

Una serie de archivos de salida se generarán mediante `render_book()`. A veces es posible que desee limpiar el directorio del libro y empezar todo de nuevo, por ejemplo, eliminar los archivos de caché que se generaron de forma automática desde **knitr**. La función `clean_book()` fue diseñada para este propósito. Por defecto, esta le indica qué archivos posiblemente son los de salida y que pueden borrarse. Si usted ha revisado esta lista de archivos, y está seguro de que ningún archivo se identificó erróneamente como archivo de salida (claramente no desea borrar un archivo de entrada que haya creado manualmente), puede eliminar todos ellos usando `bookdown::clean_book(TRUE)`. Como la eliminación de archivos es una operación relativamente peligrosa, le recomendamos que conserve su libro a través de herramientas de control de versiones como GIT, o un servicio que soporte copias de seguridad y restauración, con el fin de no perder algunos archivos para siempre una vez los elimine por error.

5.2 Previsualizar un capítulo

La conformación de todo el libro puede ser lenta cuando el tamaño del libro es grande. Hay dos cosas que pueden afectar la velocidad de la conformación de un libro: el cómputo de chunks en R, y la conversión de Markdown a otros formatos a través de Pandoc. El primero se puede mejorar permitiendo el almacenamiento en caché en **knitr** usando la opción de chunk `cache = TRUE`, y no hay mucho que se pueda hacer para que el segundo sea más rápido. Sin embargo, se puede optar por compilar un sólo capítulo utilizando la función `preview_chapter()` en **bookdown** , y, por lo general, esto será mucho más rápido que compilar todo el libro. Sólo los archivos Rmd pasados a `preview_chapter()` se reproducirán.

La previsualización del capítulo actual es muy útil cuando sólo se está enfocando en ese capítulo, ya que se puede ver rápidamente la salida actual a medida que agrega más contenido o revisar el capítulo. Aunque la vista previa funciona para todos los formatos de salida, se recomienda previsualizar la salida HTML.

Una desventaja de la vista previa de un capítulo es que las referencias cruzadas a otros capítulos no funcionarán, ya que **bookdown** en ese momento desconoce la situación de otros capítulos. Ese es un precio razonablemente pequeño a pagar por la ganancia en velocidad. Como la vista previa de un capítulo sólo muestra el resultado de ese capítulo específico, no se debe esperar que el contenido de otros capítulos se procese también de forma correcta. Por ejemplo, cuando se navega a un capítulo diferente, en realidad está viendo la antigua salida de ese capítulo (que incluso puede no existir).

5.3 Presentar el libro

En lugar de ejecutar `render_book()` o `preview_chapter()` una y otra vez, se puede previsualizar el libro "en vivo" en el navegador web, y lo único que hay que hacer es guardar el archivo Rmd. La función `serve_book()` en **bookdown** puede iniciar un servidor web local para presentar la salida HTML basada en el paquete **servr** (Xie, 2016d).

```
serve_book(dir = ".", output_dir = "_book", preview = TRUE,
  in_session = TRUE, daemon = FALSE, ...)
```

Sólo tiene que pasar el directorio raíz del libro al argumento `dir`, y esta función iniciará un servidor web local para que pueda ver la salida del libro usando el servidor. La dirección URL predeterminada para acceder a la salida del libro es `http://127.0.0.1:4321`. Si ejecuta esta función en una sesión interactiva de R, este URL se abrirá automáticamente en su navegador web. Si se encuentra en la IDE RStudio, el visor de RStudio se utilizará como el navegador web por defecto, por lo que se podrá escribir en los archivos de origen Rmd y ver la salida en el mismo entorno (por ejemplo, la fuente en la izquierda y la salida en el derecha).

El servidor atenderá a los cambios en el directorio raíz del libro: siempre que modifique los archivos en el directorio de libro, `serve_book()` puede detectar los cambios, volver a compilar los archivos Rmd, y refrescar el navegador web de forma automática. Si los archivos modificados no incluyen archivos Rmd, sólo se actualiza el navegador (por ejemplo, si sólo actualizó un determinado archivo CSS). Esto significa que una vez que se inicie el servidor, todo lo que tiene que hacer es simplemente escribir el libro y guardar los archivos. La compilación y previsualización se llevarán a cabo automáticamente al guardar archivos.

Si en realidad no se toma demasiado tiempo para volver a compilar todo el libro, es posible establecer el argumento `preview = FALSE`, de

modo que cada vez que se actualice el libro, todo el libro se vuelve a compilar, de lo contrario sólo los capítulos modificados se vuelven a compilar a través de `preview_chapter()`.

Los argumentos `...` se pasan a `servr::httw()`, y debe consultarse la página de ayuda para conocer todas las opciones posibles, como `port` y `daemon`. Hay ventajas y desventajas al utilizar `in_session = TRUE` o `FALSE`:

- Para `in_session = TRUE`, tendrá acceso a todos los objetos creados en el libro en la sesión actual de R: si utiliza un servidor endemoniado (a través del argumento `daemon = TRUE`), puede revisar los objetos en cualquier momento cuando la sesión de R actual no esté ocupado; de lo contrario tendrá que detener el servidor antes de poder revisar los objetos. Esto puede ser útil cuando necesite explorar de forma interactiva los objetos de R en el libro. La desventaja de `in_session = TRUE` es que la salida puede ser diferente con el libro compilado a partir de una sesión de R fresca, porque el estado de la sesión de R actual puede no estar limpio.
- Para `in_sesion = FALSE`, no tiene acceso a los objetos del libro en la sesión actual de R, pero es más probable que la salida sea reproducible, ya que todo se crea a partir de las nuevas sesiones de R. Como esta función es sólo para propósitos de previsualización, la limpieza de la sesión de R no puede es una gran preocupación.

Usted puede elegir `in_session = TRUE` o `FALSE` dependiendo de sus casos de uso específicos. Con el tiempo, se debe ejecutar `render_book()` desde una sesión de R fresca para generar una copia fiable de la salida del libro.

5.4 IDE RStudio

Se recomienda utilizar la versión preliminar de RStudio[1] si su versión RStudio es inferior a 1.0.0. Como se mencionó en la sección 1.3, todos los archivos R Markdown deben ser codificados en UTF-8. Esto es importante sobre todo cuando sus archivos contienen caracteres de varios bytes. Para guardar un archivo con la codificación UTF-8, puede utilizar el menú `Archivo -> Guardar con Encoding`, y elija `UTF-8`.

Cuando se hace click en el botón `knit` para compilar un documento R Markdown en el IDE RStudio, la función por defecto llamada por RStudio es `rmarkdown::render()`, que no es la que se quiere para los libros. Para llamar a la función `bookdown::render_book()` en lugar de la anterior, se puede establecer el campo `site` a `bookdown::bookdown_site` en los metadatos YAML del documento R Markdown llamado 'index.Rmd, por ejemplo,

```
---
title: "Un libro chévere"
knit: bookdown::render_book
output:
  bookdown::gitbook: default
---
```

Cuando haya configurado `site:bookdown::bookdown_site` en`index.Rmd`, RStudio será capaz de encontrar el directorio como un directorio de origen del libro [2], y verá una botón `Build Book` en el panel`Build`. Puede hacer clic en el botón para construir todo el libro en diferentes formatos, y si hace clic en el botón `Knit` en la barra de herramientas, RStudio va a obtener una vista previa de forma automática del capítulo actual, sin tener que usar `preview_chapter()` explícitamente.

[1] `https://www.rstudio.com/products/rstudio/download/preview/`

[2] Este directorio tiene que ser un proyecto RStudio.

El paquete **bookdown** viene con unos pocos addins para RStudio. Si no está familiarizado con los addins de RStudio, es posible echarle un vistazo a la documentación en `http://rstudio.github.io/rstudioaddins/`. Después de haber instalado el paquete **bookdown** y utilizar RStudio v0.99.878 o versiones anteriores, verá un menú desplegable en la barra de herramientas denominado "Complementos" y elementos en el menú tales como "Preview Book" y "Input LaTeX Math" después de abrir el menú.

El complemento "Preview Book" llama a `bookdown::serve_book()` para compilar y presentar el libro. Se bloqueará la sesión actual de R, es decir, cuando `serve_book()` se esté ejecutando, no se podrá ejecutar ninguna operación en la consola de R. Para evitar el bloqueo de la sesión de R, se puede demonizar el servidor usando `bookdown::serve_book (daemon = TRUE)`. Tenga en cuenta que este complemento debe ser usado cuando el documento actual abierto en RStudio esté bajo el directorio raíz de su libro, de lo contrario `serve_book()` puede que no encuentre el origen del libro.

El complemento de "entrada de LaTeX Math" es esencialmente una pequeña aplicación Shiny que proporciona un cuadro de texto para ayudarle a escribir expresiones matemáticas de LaTeX (Figure 5.1). A medida que escribe, verá la vista previa de la expresión matemática y su código fuente LaTeX. Esto hará que sea mucho menos propenso a errores para escribir expresiones matemáticas — cuando se escribe una expresión matemática LaTeX sin previsualizar, es fácil cometer errores como `X_ij` cuando la intención era `X_ {ij}`, u omitiendo un paréntesis de cierre. Si ha seleccionado una expresión matemática de LaTeX en el editor RStudio antes de hacer clic en el complemento, la expresión se carga y representa en el cuadro de texto de forma automática. Este complemento fue construido con base en la biblioteca MathQuill (`http://mathquill.com`). No está destinado a proporcionar apoyo total a todos los comandos de látex para las expresiones matemáticas, sino que debe ayudarle a escribir algunas expresiones matemáticas comunes.

También existen otros paquetes que proporcionan complementos de R para ayudarle al autor de libros. El paquete **citr** (**?**) proporciona un complemento llamado "Insertar citas", lo que resuelve fá-

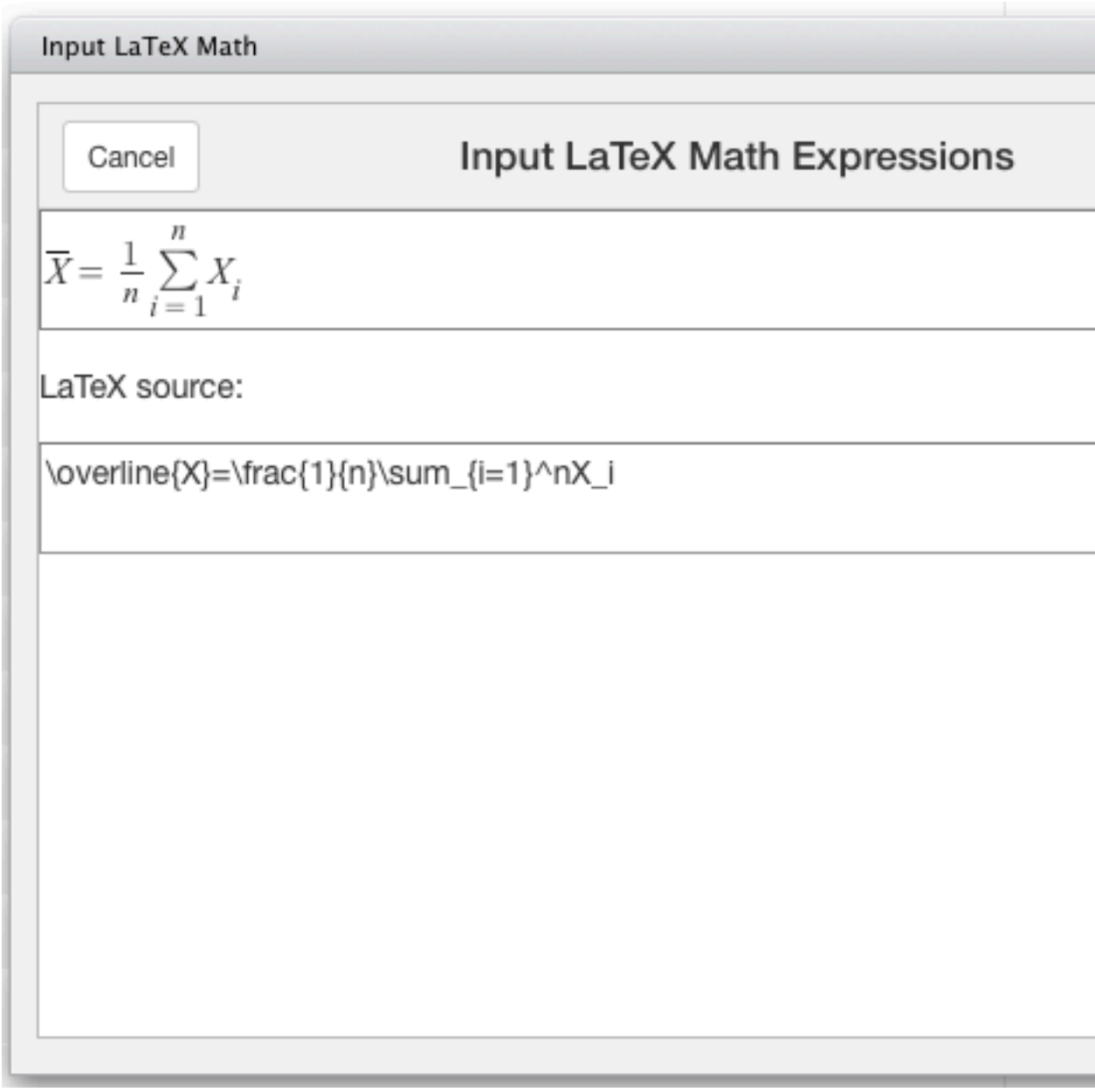


FIGURE 5.1: The RStudio addin to help input LaTeX math.

cilmente la inserción de citas en documentos de R Markdown. Este complemento analiza las bases de datos bibliográficas, y muestra todos las citas en un menú desplegable, para que se pueda elegir de la lista, recordando qué cita corresponde a qué elemento de la cita (figure 5.2).

5.5 Colaboración

Escribir un libro casi de seguro involucrará a más de una sola persona. Usted puede tener co-autores, y los lectores que le dan retroalimentación de vez en cuando.

Dado que todos los capítulos de libros son archivos de texto sin formato, son perfectos para herramientas de control de versiones, lo que significa que si todos sus coautores y colaboradores tienen conocimientos básicos de una herramienta de control de versiones como GIT, pueden colaborar con ellos en el contenido del libro usando estas herramientas. De hecho, la colaboración con GIT es posible incluso si no saben cómo usar GIT, porque GitHub ha hecho posible crear y editar archivos en línea directamente en su navegador web. La persona sólo tiene que estar familiarizada con GIT, y puede configurar el repositorio de libros. El resto de los colaboradores pueden aportar contenido en línea, aunque tendrán más libertad si conocen el uso básico de GIT para trabajar localmente.

Los lectores pueden contribuir de dos maneras. Una forma es aportar contenido directamente, y la forma más fácil, es a través de GitHub pull requests[3] si la fuente de su libro está alojada en GitHub. Básicamente, cualquier usuario de GitHub puede hacer clic en el botón de edición en la página de un archivo de origen Rmd, editar el contenido y enviar los cambios a su aprobación. Si está satisfecho con los cambios propuestos (puede ver claramente lo

[3] https://help.github.com/articles/about-pull-requests/

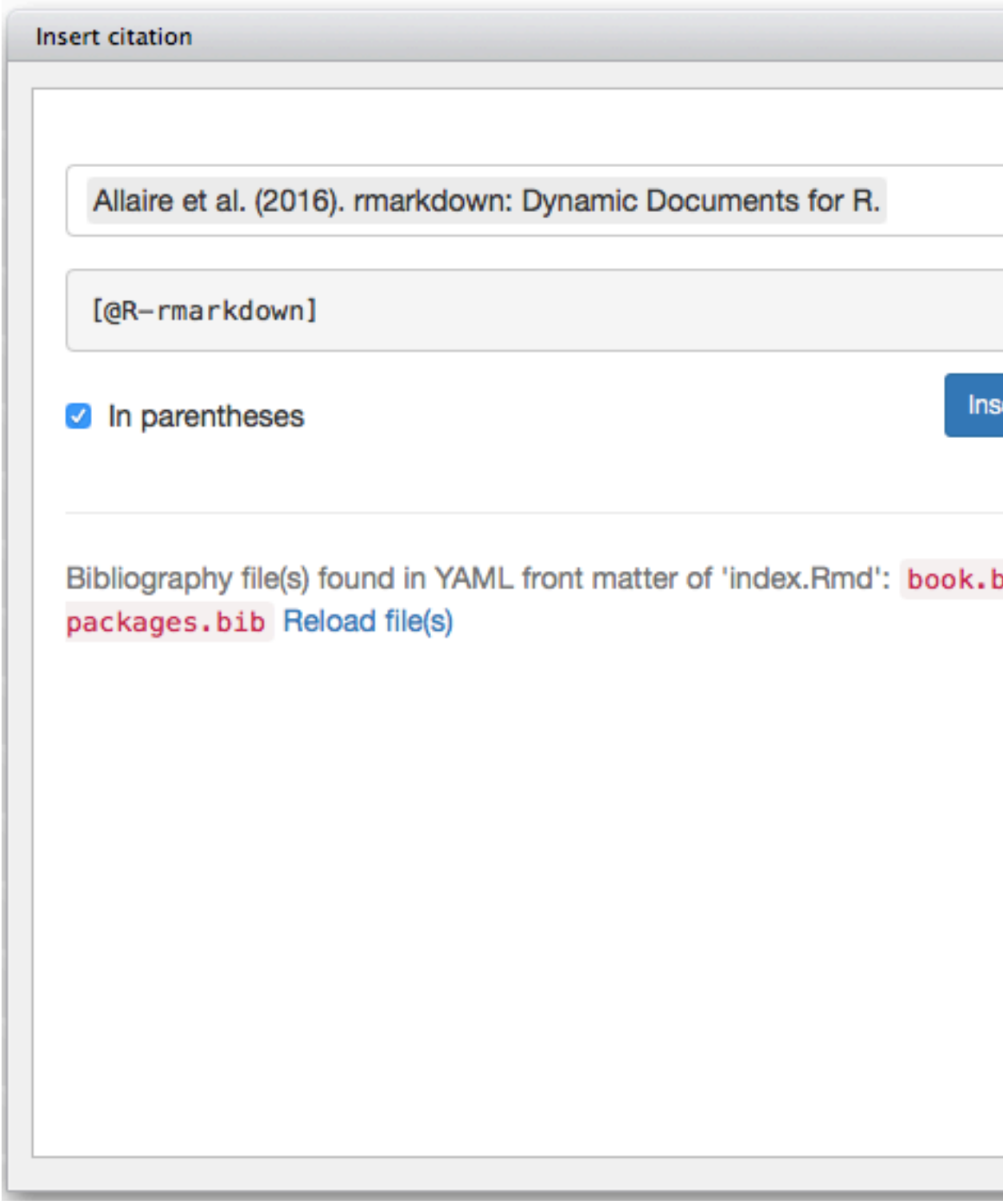


FIGURE 5.2: The RStudio addin to help insert citations.

que se cambió exactamente), puede hacer clic en un botón "Combinar" para combinar los cambios. Si no está satisfecho, puede proporcionar su retroalimentación en la solicitud de extracción, por lo que el lector puede revisarla de acuerdo a sus necesidades. Hemos mencionado el botón de edición en el estilo de GitBook en la sección 3.1.1. Este botón está vinculado a la fuente Rmd de cada página y puede guiarlo para crear la solicitud de extracción. No hay necesidad de escribir correos electrónicos de ida y vuelta para comunicar cambios sencillos, como arreglar un error tipográfico.

Otra forma de que los lectores contribuyan a su libro es dejar comentarios. Los comentarios se pueden dejar en múltiples formas: correos electrónicos, problemas de GitHub o comentarios de páginas HTML. Aquí utilizamos Disqus (ver Sección 4.1) como ejemplo. Disqus es un servicio para integrar un área de discusión en sus páginas web, y se puede cargar a través de JavaScript. Puede encontrar el código JavaScript después de registrarte y crear un nuevo foro en Disqus, que tiene este aspecto:

```
<div id="disqus_thread"></div>
<script>
(function() { // DON'T EDIT BELOW THIS LINE
var d = document, s = d.createElement('script');
s.src = '//yihui.disqus.com/embed.js';
s.setAttribute('data-timestamp', +new Date());
(d.head || d.body).appendChild(s);
})();
</script>
<noscript>Please enable JavaScript to view the
<a href="https://disqus.com/?ref_noscript">
  comments powered by Disqus.</a></noscript>
```

Tenga en cuenta que tendrá que reemplazar el nombre `yihui` con su propio nombre de foro (este nombre debe proporcionarse al crear un nuevo foro de Disqus). Puede guardar el código en un archivo HTML denominado, por ejemplo, `disqus.html`. A continuación, puede incrustarlo al final de cada página a través de la

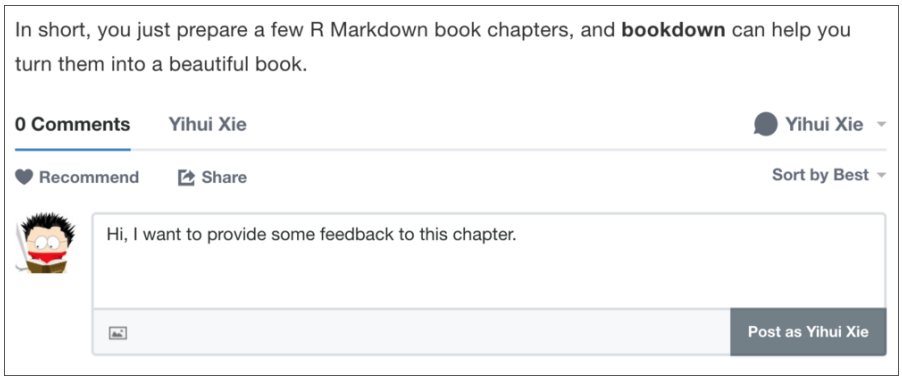


FIGURE 5.3: Una página del ibro con un área de discusión.

opción `after_body` (Figura @ref(fig: disqus) muestra cómo se ve el área de discusión):

```
---
output:
  bookdown::gitbook:
    includes:
      after_body: disqus.html
---
```

6

Publicación

A medida que desarrolla el libro, puede incluir un borrador del libro públicamente para obtener retroalimentación temprana de los lectores, por ejemplo, publicarla en un sitio web. Después de terminar de escribir el libro, es necesario pensar acerca de las opciones para publicar formalmente como copias impresas o libros electrónicos.

6.1 RStudio Connect

En teoría, se puede compilar el libro por sí mismo y publicar la salida de cualquier lugar que desee. Por ejemplo, puede alojar los archivos HTML en su propio servidor web. Se ha proporcionado una función `publish_book()` en **bookdown** para que sea muy simple cargar su libro en `https://bookdown.org`, que es un sitio web proporcionado por RStudio para alojar sus libros de forma gratuita. Este sitio web está construido con base en "RStudio Connect"[1] , un producto de RStudio que le permite implementar una variedad de aplicaciones relacionadas con R hacia un servidor, incluyendo documentos R Markdown, aplicaciones Shiny, gráficos de R, etc.

No tiene que saber mucho sobre RStudio Connect para publicar su libro en bookdown.org. Básicamente, sólo tiene que registrarse en `https://bookdown.org/connect/`, y la primera vez que intente ejecutar `bookdown::publish_book()`, se le pedirá autorizar **bookdown**

[1] `https://www.rstudio.com/products/connect/`

publicar en su cuenta bookdown.org. En el futuro, sólo tiene que llamar `publish_book()` de nuevo y **bookdown** ya no pedirá nada.

```
publish_book(name = NULL, account = NULL, server = NULL,
  render = c("none", "local", "server"))
```

El único argumento de `publish_book()` que es posible que desee tocar es `render`. Este determina si desea procesar el libro antes de publicar. Si ha ejecutado `render_book()` antes, no es necesario cambiar este argumento, de lo contrario puede configurarlo para que quede `'local'`:

```
bookdown::publish_book(render = 'local')
```

Si ha establecido su propio servidor RStudio Connect, sin duda puede publicar el libro a ese servidor en lugar de en bookdown.org.

6.2 GitHub

Puede alojar su libro en GitHub de forma gratuita a través de GitHub Pages (`https://pages.github.com`). GitHub sopporta Jekyll (`http://jekyllrb.com`), un constructor de sitios web estático, para construir un sitio web a partir de archivos Markdown. Ese puede ser el caso de uso más común de GitHub pages, pero GitHub también es compatible con archivos HTML estáticos arbitrarios, por ende puede albergar los archivos de salida HTML de su libro en GitHub.

Un enfoque para publicar su libro como un sitio de GitHub Pages de una carpeta `/docs` en su rama `master` como se describe en [GitHub Ayuda] (`http://bit.ly/2cvloKV`). En primer lugar establecer el directorio de salida de su libro para que sea `/docs` añadiendo la línea `output_dir: "docs"` al archivo de configuración `_bookdown.yml`. A continuación, después de empujar los cambios en GitHub ir a

la configuración de su repositorio y en "GitHub Pages" cambiar la "Fuente" por "rama maestra / carpeta de documentos".

Un enfoque alternativo es crear una rama `gh-pages` en su repositorio, construir el libro, poner la salida HTML (incluyendo todos los recursos externos, como imágenes, CSS y archivos JavaScript) en esta rama, y empujar la rama al repositorio remoto. Si el repositorio de libro no tiene la rama `gh-pages`, puede utilizar los siguientes comandos para crear una:

```
# assume you have initialized the git repository,
# and are under the directory of the book repository now

# create a branch named gh-pages and clean up everything
git checkout --orphan gh-pages
git rm -rf .

# create a hidden file .nojekyll
touch .nojekyll
git add .nojekyll

git commit -m"Initial commit"
git push origin gh-pages
```

El archivo oculto `.nojekyll` le dice a GitHub que su sitio web no debe ser construido a través de Jekyll, ya que la salida **bookdown** HTML ya es un sitio web independiente. Si está en Windows, puede que no tenga el comando `touch`, y sólo se pueda crear el archivo en R usando `file.create('.nojekyll')`.

Después de haber configurado GIT, el resto del trabajo puede ser automatizado a través de una secuencia de comandos (Shell, R o Makefile, dependiendo de su preferencia). Básicamente, se compila el libro a HTML, a continuación, ejecuta los comandos git para empujar los archivos en GitHub, pero es probable que no quiera hacer esto una y otra vez de forma manual y localmente. Puede ser muy útil para automatizar el proceso de publicación por completo en la nube, por lo que una vez que se haya configurado correc-

tamente, todo lo que tiene que hacer ahora es escribir el libro y empujar los archivos de origen Rmd a GitHub, y su libro siempre será construido de forma automática y publicado hacia el servidor.

Un servicio que se puede utilizar es Travis CI (`https://travis-ci.org`). . Es libre para repositorios públicos en GitHub, y fue diseñado para la integración continua (CI) de los paquetes de software. Travis CI se puede conectar a GitHub en el sentido de que cada vez que se empuja a GitHub, Travis puede ser activado para ejecutar ciertos comandos/scripts en la versión más reciente de su repositorio [2]. Estos comandos se especifican en un archivo YAML llamado `.travis.yml` en el directorio raíz de su repositorio, y están por lo general con el propósito de pruebas de software, pero en realidad son bastante abiertos, lo que significa que puede ejecutar arbitrariamente programas en una máquina Travis (virtual). Esto significa que sin duda puede ejecutar sus propios scripts para construir su libro sobre Travis. *Nota*: Travis sólo es compatible con Ubuntu y Mac OS X en este momento, por lo que debe tener algunos conocimientos básicos acerca de los comandos de Linux/Unix.

La siguiente pregunta es, cómo publicar el libro construido sobre Travis a GitHub? Básicamente tiene que conceder acceso de escritura a Travis a su repositorio GitHub. Esta autorización se puede hacer a través de varias maneras, y la más fácil a los principiantes puede ser una señal de acceso personal. A continuación se presentan algunos pasos que puede seguir:

1. Crear un token de acceso personal[3] para su cuenta en GitHub (asegúrese de que sea posible el funcionamiento "repo", por lo que el uso de esta muestra permitirá escrito a sus repositorios de GitHub);
2. Cifrarlo en la variable de entorno `GITHUB_PAT` a través de la línea de comandos `travis encrypt` y almacenarlo en `.travis.yml`, por ejemplo `travis encrypt GITHUB_PAT=TOKEN`.

[2] Debe autorizar el servicio Travis CI para su repositorio en GitHub primero . Ver `https://docs.travis-ci.com/user/getting-started/` de cómo empezar con Travis CI.

[3] `http://bit.ly/2cEBYWB`

Si usted no sabe cómo instalar o utilizar la herramienta de línea de comandos Travis, simplemente guarde esta variable de entorno a través de `https://travis-ci.org/user/repo/settings` donde `user` es su ID de GitHub, y `repo` es el nombre del repositorio;

3. Puede clonar esta rama `gh-pages` en Travis utilizando el token de GitHub, añadir los archivos HTML de salida de R Markdown (no se olvide de añadir figuras y archivos de estilo CSS, también), y empujar al repositorio remoto.

Suponga que usted está en la rama `master` en este momento (donde pone los archivos de origen Rmd), y ha compilado el libro en el directorio `_book`. Lo que puede hacer al lado de Travis es:

```
# configure your name and email if you have not done so
git config --global user.email "you@example.com"
git config --global user.name "Your Name"

# clone the repository to the book-output directory
git clone -b gh-pages \
  https://${GITHUB_PAT}@github.com/${TRAVIS_REPO_SLUG}.git \
  book-output
cd book-output
cp -r ../_book/* ./
git add --all *
git commit -m"Update the book"
git push origin gh-pages
```

El nombre de la variable `GITHUB_PAT` y el nombre del directorio `book-output` son arbitrarios, y se puede usar cualquier nombre que desee, siempre y cuando los nombres no entren en conflicto con los nombres de variables de entorno existentes o nombres de directorio. Esta secuencia de comandos, junto con la escritura de la estructura mencionamos en la sección **??**, se puede poner en la rama `master` como scripts de shell, por ejemplo, puede nombrarlos a ellos como `_build.sh` y `_deploy.sh`. Luego, su `.travis.yml` puede tener este aspecto:

```
language: r

env:
  global:
    - secure: A_LONG_ENCRYPTED_STRING

before_script:
  - chmod +x ./_build.sh
  - chmod +x ./_deploy.sh

script:
  - ./_build.sh
  - ./_deploy.sh
```

La clave `language` le dice a Travis que utilice una máquina virtual que ha instalado R. La clave `secure` es el token de acceso personal encriptada. Si ya ha guardado la variable `GITHUB_PAT` utilizando la interfaz Web Travis en lugar de la herramienta de línea de comandos `travis encrypt`, puede dejar de lado esta clave.

Como este servicio Travis es principalmente para el control de paquetes en R, también se necesita un (falso) archivo de `DESCRIPTION` como si el repositorio de libro fuera un paquete R. La única cosa en este archivo que realmente importa es la especificación de las dependencias. Todas las dependencias se instalarán a través deL paquete **devtools**. Si una dependencia está en CRAN o BioConductor, sólo tiene que incluirla en el campo `Imports` del archivo `DESCRIPTION`. Si está en GitHub, puede utilizar el campo `Remotes` para listar el nombre de su repositorio. A continuación se muestra un ejemplo:

```
Package: placeholder
Title: Does not matter.
Version: 0.0.1
Imports: bookdown, ggplot2
Remotes: rstudio/bookdown
```

Si se utiliza la infraestructura basada en contenedores[4] de Travis, puede habilitar el almacenamiento en caché mediante el uso de `sudo:false` en `.travis.yml`. Normalmente debería almacenar en caché al menos dos tipos de directorios: el directorio de la figura (por ejemplo, `_main_files`) y el directorio de caché (por ejemplo,`_main_cache`). Estos nombres de los directorios también pueden ser diferentes si se ha especificado las opciones del chunk **knitr** ``` `fig.path` y `cache.path`, pero se recomienda encarecidamente que no cambie estas opciones. Los directorios de las figuras y de caché se almacenan en el directorio `_bookdown_files` del directorio raíz del libro. Un archivo `.travis.yml` que ha permitido el almacenamiento en caché de la figura **knitr** los y directorios de caché pueden tener configuraciones adicionales`sudo` y `cache` como este:

```
sudo: false

cache:
  packages: yes
  directories:
    - $TRAVIS_BUILD_DIR/_bookdown_files
```

Si el libro consume mucho tiempo para construir, puede utilizar las configuraciones anteriores de Travis para ahorrar tiempo. Note que `paquetes: yes` significa que los paquetes instalados en R Travis también están almacenados en caché.

Todas las secuencias de comandos y configuraciones anteriores se pueden encontrar en el repositorio `bookdown-demo`: `https://github.com/rstudio/bookdown-demo/`. Si los copia a su propio repositorio, por favor recuerde que debe cambiar la tecla `secure` en `.travis.yml` usando su propia variable de cifrado `GITHUB_PAT`.

GitHub y Travis CI ciertamente no son las únicas opciones para construir y publicar su libro. Usted es libre para almacenar y publicar el libro en su propio servidor.

[4] `https://docs.travis-ci.com/user/workers/container-based-infrastructure/`

6.3 Editores

Además de publicar su libro en línea, sin duda puede considerar la publicación con un editor index{publisher}. Por ejemplo, este libro fue publicado con Chapman & Hall/CRC, y también hay una versión en línea gratis en `https://bookdown.org/yihui/bookdown/` (con un acuerdo con la editorial). Otra opción que puede considerar es la auto-publicación (`https://en.wikipedia.org/wiki/Self-publishing`) si no desea trabajar con una editorial establecida.

Será mucho más fácil publicar un libro escrito con **bookdown** si el editor que elija es compatible con LaTeX. Por ejemplo, Chapman & Hall proporciona una clase de LaTeX llamado `krantz.cls`, y Springer ofrece`svmono.cls`. Para aplicar estas clases de látex para sus libros en PDF, sólo establezca `documentclass` en los metadatos de YAML en el `index.Rmd` al nombre del archivo de clase (sin la extensión `.cls`).

La clase LaTeX es el escenario más importante de los metadatos YAML. Controla el estilo general del libro PDF. A menudo hay otros ajustes que desea modificar, y se mostrarán algunos detalles sobre este libro a continuación.

Los metadatos YAML de este libro contiene los siguientes ajustes:

```
documentclass: krantz
lot: yes
lof: yes
fontsize: 12pt
monofont: "Source Code Pro"
monofontoptions: "Scale=0.7"
```

El campo `lote: yes` significa que queremos la “Lista de tablas”, y de manera similar, `lof` significa “Lista de figuras”. El tamaño de la fuente base es `12pt`, y usamos” Source Code Pro[5] “como el tipo

[5] `https://www.fontsquirrel.com/fonts/source-code-pro`

de letra monoespaciada (ancho fijo), la cual es aplicada a todo el código del programa en este libro.

En el preámbulo LaTeX (Sección 4.1), se tienen unos cuantos ajustes. En primer lugar, se propuso la principal fuente como "Alegreya[6]", y dado que esta fuente no tiene la función SMALL CAPITALS, se utilizó la fuente "Alegreya SC".

```
\setmainfont[
  UprightFeatures={SmallCapsFont=AlegreyaSC-Regular}
]{Alegreya}
```

Los siguientes comandos hacen entornos flotantes con menos probabilidades de flotar al permitirles ocupar grandes fracciones de páginas sin flotar.

```
\renewcommand{\textfraction}{0.05}
\renewcommand{\topfraction}{0.8}
\renewcommand{\bottomfraction}{0.8}
\renewcommand{\floatpagefraction}{0.75}
```

Como `krantz.cls` proporcionar un entorno`VF` para las citas, hay que redefinir el entorno `quote` estándar a `VF`. Se puede ver su estilo en la sección 2.1.

```
\renewenvironment{quote}{\begin{VF}}{\end{VF}}
```

Entonces volvemos a definir hipervínculos como notas al pie, porque cuando el libro está impreso en papel, los lectores no son capaces de hacer clic en algunos enlaces de texto. Las notas al pie les dirán que los enlaces son reales.

```
\let\oldhref\href
\renewcommand{\href}[2]{#2\footnote{\url{#1}}}
```

[6] `https://www.fontsquirrel.com/fonts/alegreya`

También tenemos algunos ajustes para el formato `bookdown::pdf_book` en `_output.yml`:

```
bookdown::pdf_book:
  includes:
    in_header: latex/preamble.tex
    before_body: latex/before_body.tex
    after_body: latex/after_body.tex
  keep_tex: yes
  dev: "cairo_pdf"
  latex_engine: xelatex
  citation_package: natbib
  template: null
  pandoc_args: "--chapters"
  toc_unnumbered: no
  toc_appendix: yes
  quote_footer: ["\\VA{", "}{}"]
```

Todos los ajustes de preámbulo que hemos mencionado anteriormente están en el archivo `latex/preamble.tex`. En `latex/before_body.tex`, insertamos algunas páginas en blanco que requeridas por el editor, escribimos la página dedicación, y especificamos que el asunto delante comienza:

```
\frontmatter
```

Antes del primer capítulo del libro, insertamos

```
\mainmatter
```

por lo que el látex sabe cambiar el estilo de numeración de las páginas de los números romanos (por el asunto frontal) para números arábigos (para el cuerpo del libro).

Imprimimos el índice en `latex/after_body.tex` (Sección 2.9).

El dispositivo gráfico (`dev`) para guardar gráficos se establece en

`cairo_pdf` de manera que las fuentes están incrustadas en los gráficos, ya que el dispositivo predeterminado `PDF` no incrusta fuentes. Su corrector de estilo es probable que requiera incrustar todas las fuentes utilizadas en el PDF, por lo que el libro se puede imprimir exactamente como lo que se ve, de lo contrario algunas fuentes pueden ser sustituidos y el tipo de letra puede ser impredecible.

El campo `quote_footer` aseguraba que los pies de página de la cita estaban alineados a la derecha: el comando LaTeX `\{VA}` fuera proporcionado por `krantz.cls` para incluir el pie de cita.

La opción `highlight_bw` se estableció en true para que los colores en los bloques de código resaltados por sintaxis se convirtieran en escala de grises, ya que este libro se imprimirá en blanco y negro.

El libro fue compilado a PDF a través de `xelatex` para que sea más fácil para nosotros usar fuentes personalizadas.

Todos los ajustes anteriores, excepto el entorno `VF` y el comando `\{VA}` se pueden aplicar a cualquier otra clase de documentos LaTeX.

En caso de que quiera trabajar con Chapman & Hall, es posible comenzar con la copia de `krantz.cls` en nuestro repositorio (`https://github.com/rstudio/bookdown/tree/master/inst/examples`) en lugar de la copia que recibe de su editor. Hemos trabajado con la recepción de LaTeX ayudan a fijar un buen número de problemas con esta clase de LaTeX, así que espero que funcione bien para su libro si utiliza **bookdown**

A

Herramientas de Software

Para aquellos que no están familiarizados con los paquetes de software necesarios para el uso de R Markdown, a continuación se presenta una breve introducción a la instalación y mantenimiento de estos paquetes.

A.1 R y sus paquetes

R puede ser descargado e instalado desde cualquier repositorio CRAN (the Comprehensive R Archive Network, en inglés), por ejemplo, `https://cran.rstudio.com`. Tenga en cuenta que habrá algunas nuevas versiones de R cada año, y es posible que desea actualizar R ocasionalmente.

Para instalar el paquete **bookdown** , puede escribir esto en R:

```
install.packages("bookdown")
```

Esto instala todos los paquetes de R necesarios. También puede optar por instalar todos los paquetes opcionales, si no le importa demasiado si estos paquetes se van a usar para compilar su libro (como **htmlwidgets**):

```
install.packages("bookdown", dependencies = TRUE)
```

Si desea probar la versión de desarrollo de **bookdown** en GitHub, es necesario instalar primero **devtools** :

```
if (!requireNamespace('devtools')) install.packages('devtools')
devtools::install_github('rstudio/bookdown')
```

Los paquetes de R son constantemente actualizados en el CRAN o en GitHub, por lo que es posible que desee actualizarlos de vez en cuando:

```
update.packages(ask = FALSE)
```

Aunque no se requiere, el IDE RStudio puede hacer un montón de cosas mucho más fáciles cuando se trabaja en proyectos relacionados con R. El IDE RStudio se puede descargar desde `https://www.rstudio.com`.

A.2 Pandoc

Un documento de R Markdown (`*.Rmd`) se compila primero Markdown (`*.md`) a través del paquete **knitr**, y después Markdown se compila a otros formatos de salida (como el LaTeX o HTML) a través de Pandoc. Este proceso está automatizado por el paquete **rmarkdown**. No es necesario instalar **knitr** o **rmarkdown** por separado, ya que son los paquetes necesarios de **bookdown** y se instalarán automáticamente al instalar **bookdown**. Sin embargo, Pandoc no es un paquete de R, por lo que no se instalará automáticamente al instalar **bookdown**. Puede seguir las instrucciones de instalación en la página Pandoc (`http://pandoc.org`) para instalar Pandoc, pero si se utiliza el IDE RStudio, realmente no se necesita instalar Pandoc por separado, puesto que RStudio ha incluido una copia de Pandoc en él. El número de versión Pandoc puede obtenerse a través de:

```
rmarkdown::pandoc_version()
## [1] '1.17.2'
```

Si cree que esta versión es demasiado baja y hay características de Pandoc que aparecen en una versión posterior, puede instalar una versión posterior de Pandoc, y **rmarkdown** va a llamar a la versión más reciente en lugar de su versión incorporada.

A.3 LaTeX

LaTeX sólo es necesario si desea convertir su libro en PDF. La elección típica de la distribución de LaTeX depende de su sistema operativo. Los usuarios de Windows pueden considerar MiKTeX (`http://miktex.org`), los usuarios de Mac OS X pueden instalar MacTeX (`http://www.tug.org/mactex/`), y los usuarios de Linux pueden instalar TeXLive (`http://www.tug.org/texlive`). Ver `https://www.latex-project.org/get/` para obtener más información sobre LaTeX y su instalación.

La mayoría de las distribuciones de LaTeX proporcionan un paquete mínimo/básico y un paquete completo. Puede instalar el paquete básico si tiene espacio en disco limitado y conoce cómo instalar paquetes de LaTeX más adelante. El paquete completo es, a menudo, mucho más grande en tamaño, ya que contiene todos los paquetes de LaTeX, y es poco probable encontrarse con el problema de que falten paquetes en LaTeX.

Los mensajes de error en LaTeX pueden ser oscuros para los principiantes, pero es posible encontrar soluciones mediante la búsqueda de un mensaje de error en línea (tiene buenas posibilidades de acabar en StackExchange[1]). De hecho, el código LaTeX convertido desde R Markdown debe ser lo suficientemente seguro y con frecuencia no se deben tener problemas en LaTeX a menos

[1] `http://tex.stackexchange.com`

que se introduzca contenido de LaTeX en lenguaje natural en sus documentos Rmd. El problema más común en LaTeX debería ser la falta de paquetes, y el error puede tener este aspecto:

```
! LaTeX Error: File `titling.sty' not found.

Type X to quit or <RETURN> to proceed,
or enter new name. (Default extension: sty)

Enter file name:
! Emergency stop.
<read *>

l.107 ^^M

pandoc: Error producing PDF
Error: pandoc document conversion failed with error 43
Execution halted
```

Esto quiere decir que utilizó un paquete que contiene `titling.sty`, pero no se ha instalado. Los nombres de los paquetes de LaTeX son a menudo los mismos que los nombres de fichero `*.sty`, por lo que en este caso, se puede tratar de instalar el paquete `titling`. Tanto MiKTeX como MacTeX proporcionan una interfaz gráfica de usuario para administrar paquetes. Puede encontrar el gestor de paquetes MiKTeX desde el menú de inicio, y el gestor de paquetes de MacTeX de la aplicación "Utilidad de TeX Live". Escriba el nombre del paquete, o el nombre del archivo para buscar el paquete e instalarlo. TeXLive puede ser un poco más complicado: si utiliza los paquetes pre-construidos TexLive de la distribución de Linux, es necesario buscar en el repositorio de paquetes y sus palabras clave pueden coincidir con otros paquetes que no contengan LaTeX. En lo personal resulta frustrante usar las colecciones preconstruidos de paquetes en Linux, y mucho más fácil de instalar sólo TeXLive de la fuente, en cuyo caso se pueden gestionar los paquetes utilizando el comando `tlmgr`. Por ejemplo, puede buscar `titling.sty` desde el repositorio de paquetes TeXLive:

```
tlmgr search --global --file titling.sty
# titling:
#    texmf-dist/tex/latex/titling/titling.sty
```

Una vez que haya averiguado el nombre del paquete, se puede instalar mediante:

```
tlmgr install titling  # quizás requiera sudo
```

Las distribuciones de látex y los paquetes también se actualizan de vez en cuando, y es posible considerar este proceso, especialmente cuando se encuentra con problemas en LaTeX. Puede averiguar la versión de su distribución LaTeX mediante:

```
system('pdflatex --version')
## pdfTeX 3.14159265-2.6-1.40.17 (TeX Live 2016)
## kpathsea version 6.2.2
## Copyright 2016 Han The Thanh (pdfTeX) et al.
## There is NO warranty.  Redistribution of this software is
## covered by the terms of both the pdfTeX copyright and
## the Lesser GNU General Public License.
## For more information about these matters, see the file
## named COPYING and the pdfTeX source.
## Primary author of pdfTeX: Han The Thanh (pdfTeX) et al.
## Compiled with libpng 1.6.21; using libpng 1.6.21
## Compiled with zlib 1.2.8; using zlib 1.2.8
## Compiled with xpdf version 3.04
```

B

Uso de Software

Como se ha mencionado en el capítulo 1, este libro no es una guía completa **knitr** o **rmarkdown** . En este capítulo, se explican brevemente algunos conceptos básicos y la sintaxis en **knitr** y **rmarkdown**. Si tiene alguna duda, puede preguntar en StackOverflow (`https://stackoverflow.com`) y etiquetar las preguntas con `R`,`knitr`, `rmarkdown`, y/o `bookdown`, según sea el caso.

B.1 knitr

El paquete **knitr** fue diseñado basado en la idea de "programación literaria" (Knuth, 1984), que le permite combinar código de programa con el texto en un documento de origen. Cuando **knitr** compila un documento, el código del programa (en chunks) se extrae y se ejecuta, y la salida del programa se muestra junto con el texto original en el documento de salida. Se ha introducido la sintaxis básica de la sección 2.3.

R Markdown no es el único formato de origen que **knitr** soporta. La idea básica se puede aplicar a otros idiomas de computación y de creación. Por ejemplo, **knitr** también es compatible con la combinación de R y látex (documentos `*.Rnw`), y R + HTML (`*.Rhtml`), etc. Se pueden utilizar otros lenguajes de computación con **knitr**, como C++, Python, SQL, etc. A continuación se muestra un ejemplo sencillo y se puede ver más en `http://rmarkdown.rstudio.com/authoring_knitr_engines.html`.

```{python}
```

```
x = 'Hello, Python World!'
print(x.split(' '))
```

Los usuarios de Python pueden estar familiarizados con los portátiles IPython o Jupyter (`https://jupyter.org`). De hecho, R Markdown también se puede utilizar como notebooks, y tiene algunos beneficios adicionales; Vea esta entrada del blog para más información: `https://blog.rstudio.org/2016/10/05/r-notebooks/`.

Si desea mostrar un fragmento literal de chunk en su documento, se puede añadir una expresión en línea que genera una cadena vacía (`` `r ''` ``) antes de la cabecera del chunk, y sangrar el trozo de código cuatro espacios [1], e.g.,

````
    `r ''`````{r}
    # a literal code chunk
    ```
````

Después de que se compila el documento, la expresión en línea desaparecerá y se verá:

````
```{r}
# a literal code chunk
```
````

Normalmente no es necesario llamar directamente funciones **knitr** al compilar un documento, ya que **rmarkdown** llamará **knitr**. Si desea compilar un documento de origen sin la conversión a otros formatos, es posible utilizar la función `knitr::knit)`.

B.2 R Markdown

Gracias a la potencia de R y Pandoc, usted puede fácilmente hacer el cómputo de documentos en R Markdown, y convertirlos

[1] Sigue la regla de cuatro espacios si el chunk literal se muestra en otros entornos como una lista: `http://pandoc.org/MANUAL.html#the-four-space-rule`
````

en una variedad de formatos de salida, incluyendo documentos de Word/HTML/PDF/, diapositivas HTML5/Beamer, dashboards, y sitios web, etc. Un documento en R Markdown, por lo general, consiste en metadatos YAML (opcional) y el cuerpo del documento. Se ha introducido la sintaxis para escribir diversos componentes del cuerpo del documento en el capítulo 2, y se explica más sobre los metadatos YAML en esa sección.

Los metadatos para R Markdown se puede escribir al principio de un documento, empezando y terminando con tres guiones ---, respectivamente. Los metadatos YAML consisten, generalmente, en pares de etiquetas de valor separados por dos puntos, por ejemplo,

```
---
title: "Un documento en R Markdown"
author: "Yihui Xie"
---
```

Para los valores de caracteres, es posible que omita las comillas cuando los valores no contienen caracteres especiales, pero es más seguro para citarlos si se espera que sean los valores de caracteres.

Además de los caracteres, otro tipo común de valores son valores lógicos. Tanto `yes` como `true` significan verdad, y `no` / `false` significan falso, por ejemplo,

```
link-citations: yes
```

Los valores pueden ser vectores, y hay dos formas de escribir vectores. Las dos formas siguientes son equivalentes:

```
output: ["html_document", "word_document"]
```

```
output:
  - "html_document"
  - "word_document"
```

Los valores también pueden ser listas de valores. Sólo tiene que aplicar sangría de dos espacios más a los valores, por ejemplo,

```
output:
  bookdown::gitbook:
    split_by: "section"
    split_bib: no
```

Es un error común olvidar sangrar los valores. Por ejemplo, los siguientes datos

```
output:
html_document:
toc: yes
```

significaría

```
output: null
html_document: null
toc: yes
```

en lugar de lo que probablemente habría esperado:

```
output:
  html_document:
    toc: yes
```

El formato de salida R Markdown se especifica en el campo `output` de los metadatos YAML, y hay que consultar las páginas de ayuda de R para las opciones posibles, por ejemplo,`?rmarkdown::html_document`, o `?bookdown::gitbook`. Los significados de la mayoría de los otros campos en YAML se pueden encontrar en la documentación de Pandoc.

El paquete **rmarkdown** ha porporcionado estos formatos de salida R Markdown:

- `beamer_presentation`
- `github_document`
- `html_document`
- `ioslides_presentation`
- `md_document`
- `odt_document`
- `pdf_document`
- `rtf_document`
- `slidy_presentation`
- `word_document`

Hay muchos posibles formatos de salida en otros paquetes de R, incluyendo **bookdown**, **tufte**, **rticles**, **flexdashboard**, **revealjs**, y **rmdformats**, etc.

C

FAQ

A continuación se muestra la lista *completa* de preguntas frecuentes (FAQ). Sí, sólo hay una pregunta aquí. Personalmente no me gustan las preguntas frecuentes. A menudo significan sorpresas, y las sorpresas no son buenas para los usuarios de software.

1. P: **bookdown** tendrá características X, Y, y Z?

   R: La respuesta corta es no, pero si usted se ha hecho esta pregunta tres veces: "¿Realmente lo necesito", y la respuesta sigue siendo "sí", no dude en presentar una solicitud de función en `https://github.com/rstudio/bookdown/issues`. Los usuarios que solicitan más funciones a menudo provienen del mundo LaTeX. Si ese es el caso suyo, la respuesta a esta pregunta es sí, porque Markdown Pandoc apoya código LaTeX sin procesar. Siempre que sienta que Markdown no puede hacer el trabajo por usted, siempre tiene la opción de aplicar un código LaTeX sin procesar en su documento de Markdown. Por ejemplo, puede crear glosarios utilizando el paquete **glossaries** o incrustar una tabla LaTeX complicada, siempre y cuando conozca la sintaxis de LaTeX. Sin embargo, tenga en cuenta que el contenido de LaTeX no es portátil. Sólo funcionará para la salida LaTeX/PDF, y se ignorará en otros tipos de salida. Dependiendo de la solicitud, podemos introducir algunas características más de LaTeX en **bookdown** en el futuro, pero nuestra filosofía general es que Markdown debe mantenerse tan simple como sea posible.

Lo más difícil en el mundo no es aprender tecnologías fantásticas, sino controlar su propio corazón salvaje.

Bibliography

Allaire, J., Cheng, J., Xie, Y., McPherson, J., Chang, W., Allen, J., Wickham, H., Atkins, A., and Hyndman, R. (2016). *rmarkdown: Dynamic Documents for R*. R package version 1.3.

Chang, W. (2016). *webshot: Take Screenshots of Web Pages*. R package version 0.3.2.

Cheng, J. (2016). *miniUI: Shiny UI Widgets for Small Screens*. R package version 0.1.1.

Knuth, D. E. (1984). Literate programming. *The Computer Journal*, 27(2):97–111.

R Core Team (2016). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Vaidyanathan, R., Xie, Y., Allaire, J., Cheng, J., and Russell, K. (2016). *htmlwidgets: HTML Widgets for R*. R package version 0.8.

Xie, Y. (2015). *Dynamic Documents with R and knitr*. Chapman and Hall/CRC, Boca Raton, Florida, 2nd edition. ISBN 978-1498716963.

Xie, Y. (2016a). *bookdown: Authoring Books and Technical Documents with R Markdown*. R package version 0.3.

Xie, Y. (2016b). *DT: A Wrapper of the JavaScript Library 'DataTables'*. R package version 0.2.

Xie, Y. (2016c). *knitr: A General-Purpose Package for Dynamic Report Generation in R*. R package version 1.15.1.

Xie, Y. (2016d). *servr: A Simple HTTP Server to Serve Static Files or Dynamic Documents*. R package version 0.5.

Xie, Y. and Allaire, J. (2016). *tufte: Tufte's Styles for R Markdown Documents*. R package version 0.2.

Index