Índice general

Tema 4

Metodología del análisis estadístico con R

4.3. Creación de informes estadísticos en R

En R disponemos de herramientas para combinar texto y código en R, creando de forma automática un informe estadístico. Esta es la base para conseguir un documento reproducible que se regenere automáticamente cuando cambia el código o los datos.
En general se trata de crear un tipo especial de documento que contiene trozos de código en R (denominados *chunks*) y documentación. Cuando este documento es procesado (desde herramientas adecuadas de R) se produce la ejecución del código en R y los resultados se incorporan al documento resultante[1]. En este proceso habitualmente se crea un fichero intermedio que permite la conversión.
A continuación vamos a describir herramientas de R que permiten crear este tipo de documentos.

4.3.1. *knitr* y R Markdown

Actualmente la forma más sencilla de construir un informe estadístico reproducible en R es usando R Markdown R Markdown es una variante de Markdown[2] que interacciona con R[3] para poder incluir la ejecución de trozos de código (*code chunks*) . Para ello utiliza el paquete *knitr* de R.
Veremos después que *knitr* incluye otros tipos de documentos además de Markdown entre los cuales está LaTeX. Sin embargo la simplicidad de Markdown se impone en la práctica y se ha convertido en los últimos años en uno de los más utilizados.

[1] Esto es lo que comúnmente en inglés se dice *"knitting" or "weaving" the R code chunks*, lo que motiva el nombre de dos paquetes de R para esta tarea: *Sweave* y *knitr*.

[2] Markdown es un lenguaje creado inicialmente para la creación de páginas web a partir de documentos de texto de forma sencilla y rápida.

[3] Como LaTeX con Sweave como veremos después.

En pocas palabras podemos decir que R Markdown es la conjunción de dos herramientas: el paquete *knitr* y Pandoc[4]. El primero permite la ejecución del código R incrustado en el documento y convierte R Markdown a Markdown. Después Pandoc convierte (*render*) Markdown en el formato deseado (PDF, HTML, Word, etc.). En este proceso se crea un fichero intermedio (pandoc, con extensión `md`) que permite la conversión.

Figura 4.1: Proceso de conversión de un documento R Markdown.

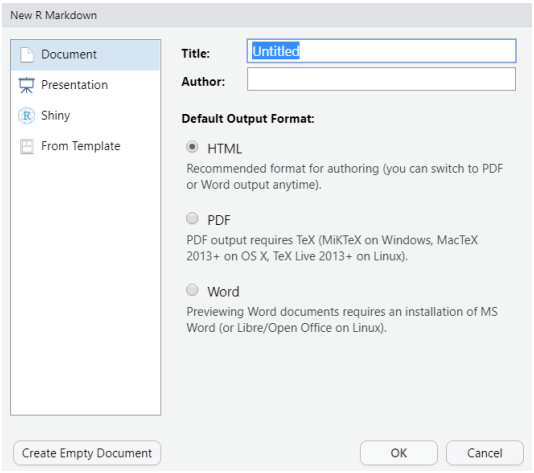
RStudio facilita la tarea de crear, editar y procesar documentos de R Markdown. Desde Rstudio haremos a continuación nuestra primera aproximación a R Markdown[5], no obstante veremos que también es posible hacerlo directamente en R, instalando y cargando los paquetes *rmarkdown* y *knitr*, y evaluando directamente funciones apropiadas de los mismos.

Crear un documento R Markdown

Para crear un documento de R Markdown en RStudio elegimos en el menú:

```
File > New File > R Markdown...
```

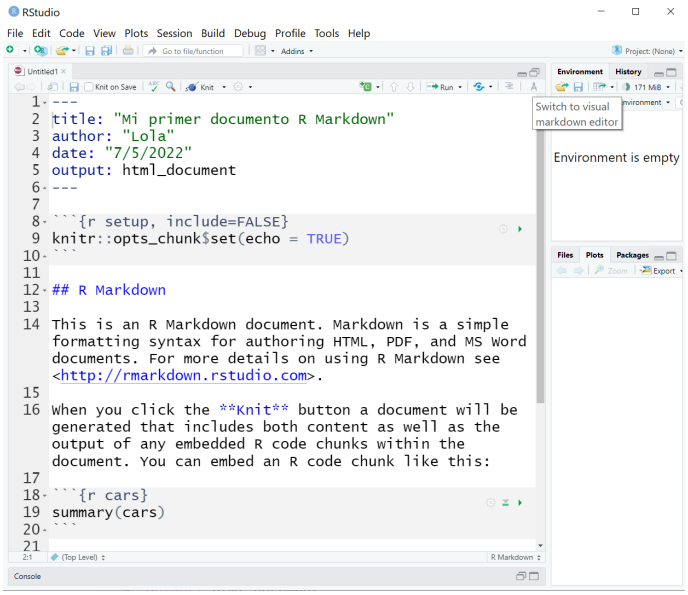
lo que nos muestra la siguiente caja de diálogo donde podemos especificar opciones básicas del documento:



y, una vez definidas podemos ver el documento creado en una pestaña de la ventana principal:

[4] `http://pandoc.org`

[5] Te recomendamos visitar `https://rmarkdown.rstudio.com/lesson-1.html` donde podrás una buena introducción que hemos utilizado para escribir estas notas.



Se trata de un documento R Markdown que RStudio identifica con un icono particular y que guardamos con extensión `Rmd`. El documento tiene una estructura básica y proporciona algunas indicaciones y sugerencias para adaptarlo a nuestras necesidades.
Podemos observar que el documento R Markdown es un fichero de texto que consta de tres componentes básicas:

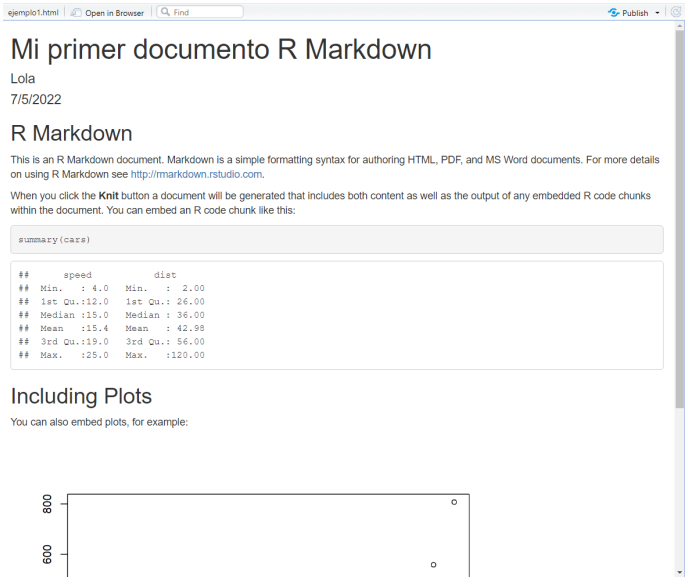
- Una cabecera delimitada por dos marcadores del tipo `---`.
- *R code chunks* delimitados por dos marcadores del tipo ```.
- Texto.

La cabecera es opcional, constituye los metadatos del documento y se escribe usando sintaxis YAML[6]. El texto incluido en el cuerpo del documento se escribe usando sintaxis Markdown. Desde la web de RStudio podemos descargar una "chuleta" (`https://raw.githubusercontent.com/rstudio/cheatsheets/main/rmarkdown.pdf`) de R Markdown que incluye opciones básicas y la sintaxis para cada componente del documento. En estos apuntes también proporcionamos más abajo algunas notas básicas.

Compilar un documento R Markdown

Para compilar un documento R Markdown en RStudio podemos usar el botón `Knit` de la barra de herramientas. Compilando el documento que hemos creado antes, RStudio abre una ventana que muestra el siguiente resultado:

[6] Ain't Markup Language, `https://en.wikipedia.org/wiki/YAML`.



Para obtener este resultado RStudio ha evaluado la función `render()` del paquete *rmarkdown*. Vamos a comprobar que en efecto se obtiene el mismo resultado desde la consola de R evaluando directamente esta función. Para ello cargamos primero el paquete y luego evaluamos la función indicando como primer argumento el nombre del fichero. En este caso lo hemos guardado con el nombre `ejemplo1.Rmd` y está en el directorio de trabajo con lo que escribiríamos:

```
library(rmarkdown)
render('ejemplo1.Rmd')
```

Una vez obtengamos la confirmación de que el proceso se ha realizado correctamente, podemos comprobar que se ha creado un fichero HTML. Este fichero lo podemos abrir directamente en el navegador o hacerlo desde R escribiendo:

```
browseURL(url=render('ejemplo1.Rmd'))
```

El formato de salida por defecto en este caso es un fichero HTML (`html_document`), esto es lo que está indicado en la cabecera del documento R Markdown. No obstante es posible especificar otros formatos[7] como segundo argumento de la función (`output_format`) por ejemplo `output_format='pdf_document'`[8]. Esto también era posible desde el botón `Knit` en RStudio.
Es posible comprobar con más detalle cómo se realiza el proceso de conversión que ilustrábamos en la Figura 4.1. Por defecto la función `render()` borra los ficheros intermedios pero podemos indicar que no lo haga usando el argumento `clean`. Comprobamos el resultado de:

[7] Consulta sobre esto por ejemplo en `https://rmarkdown.rstudio.com/lesson-9.html`
[8] El formato pdf requiere tener instalado LaTeX en el ordenador.

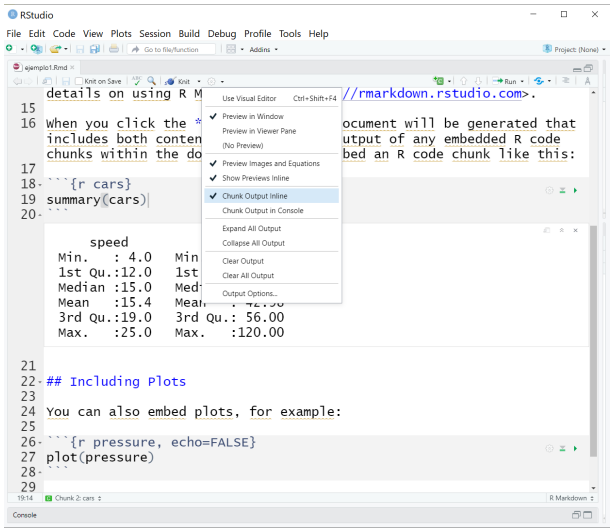
```
render('ejemplo1.Rmd',clean=FALSE)
```

Observamos que el resultado es similar salvo que se ha creado un fichero intermedio `ejemplo1.knit.md`. Se trata del fichero pandoc que permite la conversión al formato final. Si lo abrimos podemos ver que es similar al fichero `ejemplo1.Rmd` original salvo que incluye el resultado de la ejecución de los *chunks*.
Un comentario importante a tener en cuenta en el proceso de "renderización" es que R crea una nueva sesión para ello. Esto significa que ninguno de los objetos que tengamos disponibles en el espacio de trabajo lo estará en la ejecución de los *chunks* del documento. Esto es un elemento esencial para conseguir un documento reproducible. En la sesión creada, los objetos que se vayan creando en los distintos *chunks* estarán disponibles para los siguientes.

Notebook interface

Cuando se abre un documento R Markdown en RStudio, la ventana donde se muestra se convierte en lo que se denomina un *notebook interface*. Se trata de un interfaz muy intuitivo y que facilita tanto el trabajo en el documento como su procesamiento. Entre sus funcionalidades destacamos:

- Cada *chunk* se muestra en un recuadro en el que podemos encontrar dos iconos, uno que permite modificar sus opciones y otro que permite su ejecución.
- Cuando RStudio ejecuta el código muestra los resultados a continuación (*chunk output inline*), lo que nos permite de una manera conveniente inspeccionar nuestro trabajo.
- La ejecución en línea de los *chunks* así como otras opciones se pueden cambiar desde el menú de configuración del notebook al que accedemos desde la barra de herramientas de las pestaña:



- Entre las opciones anteriores la primera permite cambiar el editor a un modo de edición visual (*Visual editor*).

Inclusión y opciones de los *chunks*

Un *chunk* comienza con ```` ```{r} ````, donde `r` indica el lenguaje del código y termina con ```` ``` ````. Es posible especificar un nombre para el chunk además de diversas opciones relativas a la ejecución y resultados del código de la forma ```` ```{r, nombre, opción1, opción2,...} ````). Por ejemplo, en el documento que hemos creado antes podemos ver que el tercer *chunk* comienza con ```` ```{r pressure, echo=FALSE} ````, esto indica que el *chunk* tiene nombre `pressure` y que cuando se ejecute el código durante la compilación no debe mostrar la ejecución línea a línea del código (lo que se haría por defecto si no indicamos esta opción). Esto puede ser adecuado cuando queremos incrustar gráficos pero no nos interesa mostrar el código que los genera.
En la tabla 4.1 mostramos algunas de las opciones disponibles[9]:

Opción	Descripción
`eval=FALSE`	no se ejecuta el código
`include=FALSE`	el código y los resultados no se muestran pero el código se ejecuta
`echo=FALSE`	se muestran los resultados pero no el código
`warning=FALSE`	no muestra posibles advertencias generadas por el código
`fig.align`	alineación del gráfico (`'left'`, `'right'` o `'center'`)
`fig.cap='...'`	añade un texto al gráfico
`fig.height`, `fig.width`	Ancho y alto del gráfico (en pulgadas)

Tabla 4.1: Opciones de los *chunks* en *knitr*.

Para especificar opciones globales para todo el documento se evalúa `knitr::opts_chunk$set` en un *chunk*. Observa por ejemplo el documento que hemos creado antes donde se incluye al comienzo este tipo de opciones. *Knitr* utilizará las que indiquemos aquí para todos los *chunks* individuales, salvo que se especifique lo contrario en alguno de ellos.
También es posible escribir código dentro de una línea de texto. Para ello escribiremos la expresión como `` `r expresión` ``. Por ejemplo podemos escribir la siguiente línea en nuestro documento y observar el resultado en la compilación:

```
The mean of *speed* is `r mean(cars$speed)`.
```

Insertar tablas

Un data frame se puede imprimir en formato de tabla en el documento usando por ejemplo la función *kable()* del paquete *knitr*. Por ejemplo podemos incluir en nuestro documento una tabla con la cabecera de las 5 primeras columnas del data frame `mtcars` escribiendo:

[9]Puedes consultarlas todas en `http://yihui.name/knitr/options`.

```
kable(head(mtcars[,1:5]),caption='Una tabla con kable()')
```

Si ejecutamos esta función en la consola podemos ver que escribe la tabla usando sintaxis Markdown:

```
Table: Una tabla con kable()

|                  |  mpg| cyl| disp|  hp| drat|
|:-----------------|----:|---:|----:|---:|----:|
|Mazda RX4         | 21.0|   6|  160| 110| 3.90|
|Mazda RX4 Wag     | 21.0|   6|  160| 110| 3.90|
|Datsun 710        | 22.8|   4|  108|  93| 3.85|
|Hornet 4 Drive    | 21.4|   6|  258| 110| 3.08|
|Hornet Sportabout | 18.7|   8|  360| 175| 3.15|
|Valiant           | 18.1|   6|  225| 105| 2.76|
```

lo que permite después la conversión a través de pandoc al renderizar el documento.
Hay otros paquetes que permiten escribir tablas con otras opciones como por ejemplo *xtable*, el cual resulta muy útil para trabajar con documentos de LaTeX.

Sintaxis básica Markdown

El texto del documento R Markdown se escribe usando la sintaxis del lenguaje de marcas (Pandoc) Markdown. Este lenguaje transforma el texto marcado en texto formateado en el documento final. A continuación mostramos algunos elementos básicos de esta sintaxis:

- Estilo del texto:

  Cursiva se obtiene escribiendo `*cursiva*`.

  Negrita escribiendo `**negrita**`.

  `Verbatim` escribiendo `` `Verbatim` ``.

  - Superíndice^2 escribiendo `Superíndice^2^`

- Encabezados de secciones:

  ```
  # Encabezado nivel 1

  ## Encabezado nivel 2

  ###### Encabezado nivel 6
  ```

- Párrafos separados se crean dejando líneas en blanco.

- Dos espacios blancos al final de una línea provocan un salto de línea sin empezar un nuevo párrafo.

- Listas:

  Construir una lista numerada comenzando una nueva línea con `1.`, seguida de espacio en blanco.

  Construir una lista no numerada comenzando una nueva línea con `*` o `-`, seguida de espacio en blanco.

  Construir listas anidadas comenzando la línea después de 4 espacios en blanco.

- Hipervínculos con `<url>` o `[texto](url)` (e.g. `http://www.ugr.es`>, `[ugr](www.ugr.es)`).

- Imágenes con `![](imagen)` e.g. `![](logo.png)`

- Citar texto empezando una nueva línea con > seguida de espacio en blanco.

- Tablas:

```
Derecha | Izquierda | Por defecto | Centrado |
-------:|:--------- | ----------- |:--------:|
celda 1 | celda 2 | celda 3 | celda 4
celda 5 | celda 6 | celda 7 | celda 8
```

Insertar expresiones matemáticas

Es posible incluir ecuaciones y expresiones matemáticas en documentos R Markdown. Para ello se utiliza la sintaxis de LaTeX. Por ejemplo podemos escribir en el texto ecuaciones centradas usando `\[...\]`, por ejemplo:

```
\[
 \bar x = \sum_{i=1}^n x_i
\]
```

También ecuaciones en una línea como: `$\hat\beta_0=\bar y-\hat\beta_1\bar x$`.

Conversión entre scripts (`.R`) y R Markdown (`.Rmd`)

A partir de un fichero R Markdown podemos generar un script de R con el código que tenga incrustado usando la función `purl()` del paquete *knitr*. Por ejemplo podemos hacerlo con el documento que hemos creado antes escribiendo

```
purl('ejemplo1.Rmd','script1_ejemplo1.R')
```

La ejecución nos muestra un mensaje indicando que se ha creado el script con el nombre que hemos proporcionado en el segundo argumento, `script1_ejemplo1.R`. Si queremos que el texto del documento se incluya como comentarios podemos escribir:

```
purl('ejemplo1.Rmd','script2_ejemplo1.R',documentation=2)
```

Abriendo los dos scripts creados podemos ver el resultado, donde los comentarios corresponden a líneas que comienzan con `#'`.
De forma similar, a partir de un script de R podemos generar de forma automática un informe en HTML usando la función `spin()` del paquete *knitr*. Para ello evaluaríamos la función indicado como primer argumento el nombre del script, por ejemplo para el script `script2_ejemplo1.R` que hemos creado antes podemos crear el informe escribiendo:

```
spin('script2_ejemplo1.R')
```

Esto creará el informe como un documento HTML, y si queremos además mostrarlo escribimos:

```
browseURL(url=knitr::spin('script2_ejemplo1.R'))
```

Con las sentencias anteriores se crea el documento HTML junto con el fichero pandoc intermedio (`.md`), pero no se crea el documento R Markdown (`.Rmd`). Para hacerlo especificaríamos el argumento `knit=FALSE` en la función `spin`, y podríamos renderizarlo para obtener el resultado en HTML usando la función `render()` de *rmarkdown*:

```
spin('script2_ejemplo1.R',knit=FALSE)
# la sentencia de arriba crearía el fichero Rmd
# la sentencia de abajo usaría markdown para renderizarlo a HTML
render(spin('script2_ejemplo1.R',knit=FALSE))
# si queremos además mostrar el documento HTML escribimos:
browseURL(url=render(spin('script2_ejemplo1.R',knit=FALSE)))
```

Ejercicio 1. Usando la función `spin()` crea un documento HTML a partir del script que escribiste para la última sesión práctica (Práctica 8). Hazlo de modo que se cree además un documento R Markdown.

4.3.2. *knitr* y LaTeX

Aunque R Markdown nos permite crear informes estadísticos bastante completos, puede no ser suficiente en muchos casos. El paquete *bookdown* proporciona una extensión que permite escribir libros en R Markdown. Muchos de los manuales que hemos referenciado a lo largo del curso están escritos usando esta herramienta.
No obstante si conocemos LaTeX, el paquete *knitr* nos permite utilizar todo el potencial de LaTeX en nuestros informes estadísticos. De esto nos vamos a ocupar en este apartado.

knitr y *Sweave*

El paquete *knitr* está inspirado en *Sweave*. Este último, incluido en el sistema base de R, es un paquete que permite crear informes estadísticos con la filosofía que hemos descrito antes pero usando documentos de LaTeX. Esto se consigue incrustando el código de R en un documento de LaTeX usando sintaxis `noweb`[10].
Sweave sin embargo presenta varios problemas y carencias lo que ha motivado la creación de otros paquetes que permiten extensiones y mejoras (*cacheSweave*, *pgfSweave*, etc.). El paquete *knitr* está inspirado en *Sweave* y su creador, Yihui Xie, ha conseguido con él unificar estas extensiones, logrando resolver muchas de los problemas y limitaciones de *Sweave*. En la mente del creador de *knitr* ha estado desde el principio mantener la compatibilidad con *Sweave*, a la vez que conseguir una herramienta muy versátil que interacciona con diversos tipos de documentos, como hemos visto antes, y lenguajes (no sólo R, sino también Python y otros).

Ejemplo de un documento `Sweave`

Los ficheros `Sweave` son muy similares a los fichero de LaTeX salvo que incluyen *chunks* de código. Habitualmente tienen extensión `.Rnw` y se pueden crear y editar también de forma sencilla en RStudio (aunque no está limitado a este editor).
La Figura 4.2 muestra un ejemplo muy básico de este tipo de fichero que se puede descargar de `https://yihui.org/knitr/demo/minimal/` (*002-minimal.Rnw*).

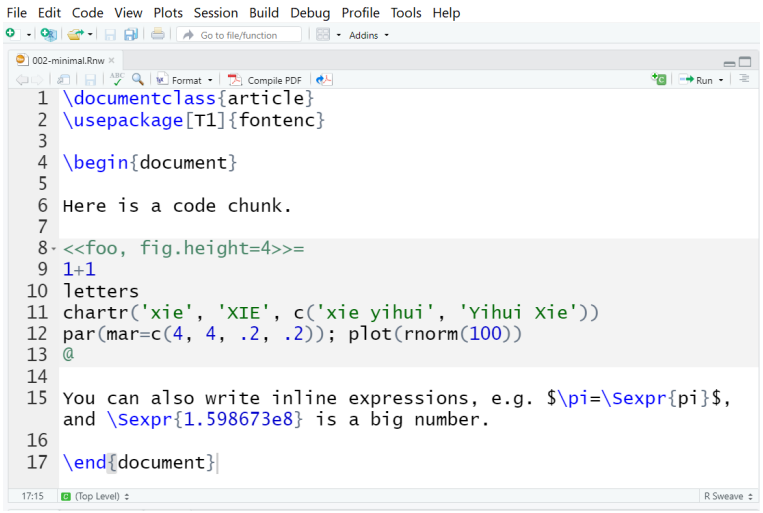


Figura 4.2: Fichero *002-minimal.Rnw*

El fichero contiene un sólo *chunk* que en este caso comienza con `<<nombre, opciones>>=`

[10] Herramienta de programación literal (*literate-programming*) que permite combinar código fuente y documentación en un sólo fichero. Un fichero `noweb` es un fichero de texto consistente en una secuencia de trozos de código y documentación. Habitualmente con extensión `.nw`.

y termina con `@`. Las opciones son del tipo que describíamos antes con R Markdown[11]. Además es posible escribir y evaluar código en una línea, lo que en este caso se hace con `Sexpr()`. El resto corresponde por completo a lo que sería un documento de LaTeX.

Compilar un documento `.Rnw`

Para compilar el documento podemos elegir entre hacerlo desde RStudio, de una forma muy sencilla usando el botón `Compile PDF`[12], o tomar el control del proceso y hacerlo evaluando funciones apropiadas de *knitr*. Siguiendo esta última opción usaríamos la función *knit()* del paquete *knitr*. Al evaluar dicha función se ejecutarían los chunks incluidos en el fichero y se incrustaría el resultado creando un documento de LaTeX. Para el ejemplo anterior lo haríamos escribiendo:

```
knit('002-minimal.Rnw')
```

Una vez que R nos muestra la confirmación de que la compilación se ha ejecutado satisfactoriamente, podemos observar que se ha creado en el directorio de trabajo un nuevo fichero `002-minimal.tex`. Este documento se podría ahora abrir y compilar desde nuestro editor de LaTeX preferido para crear un documento PDF, o bien hacerlo directamente desde R usando la función *texi2pdf()* del paquete *tools* que permite ejecutar `pdflatex`:

```
tools::texi2pdf('002-minimal.tex')
```

Ejercicio 2. Descarga de PRADO el documento fuente de la Práctica 7 (`Practica7.Rnw`) y completa las soluciones creado un informe completo en PDF.

4.3.3. Referencias y enlaces relevantes

1. El paquete *knitr*:
   - Xie, Y. (2015). *Dynamic Documents with R and Knitr*. 2nd ed. Boca Raton. Chapman, Florida. Se puede descargar completo de `https://duhi23.github.io/Analisis-de-datos/Yihue.pdf`.
   - `https://yihui.org/knitr/`)
2. El paquete *Sweave*:
   - Breve manual por su creador en `https://stat.ethz.ch/R-manual/R-devel/library/utils/doc/Sweave.pdf`

[11] No todas son compatibles con *Sweave*.

[12] En el menú de opciones `Tools >Global Options` especificamos antes si la compilación se hace con *Sweave* o *knitr*.

- Tutorial en español desde CRAN: https://cran.r-project.org/doc/contrib/Rivera-Tutorial_Sweave.pdf

3. R Markdown:

   - *R Markdown: The definitive guide*, disponible en https://bookdown.org/yihui/rmarkdown/
   - Introducción sencilla desde RStudio en https://rmarkdown.rstudio.com/
   - Una "chuleta" con lo esencial en https://raw.githubusercontent.com/rstudio/cheatsheets/main/rmarkdown.pdf
   - Algo más completa en https://www.rstudio.com/wp-content/uploads/2015/03/rmarkdown-reference.pdf.

4. El paquete *bookdown*:

   - Xie, Y. (2018). *Bookdown: Authoring Books and Technical Documents with R Markdown*. https://CRAN.R-project.org/package=bookdown
   - Versión en español en https://rubenfcasal.github.io/bookdown_intro/