

Publicación de textos científicos con R

Julio M. Sevilla Sánchez

2021-06-18

Índice general

1	Introducción al lenguaje R	5
1.1	¿Que es R?	5
1.2	¿Por qué usar R?	5
1.3	Recursos sobre R	6
2	Instalación de R, RStudio y librerías	9
2.1	Instalación de R	9
2.2	Actualización de R	9
2.3	El Sistema R	10
2.4	Las librerías de R	11
2.5	Interfaces gráficas de R	12
3	Markdown, LaTeX y Pandoc	15
3.1	La escritura con Markdown	15
3.2	Rmarkdown	17
3.3	Pandoc y la conversión entre formatos.	18
3.4	Instalación de los elementos necesarios	19
3.5	El lenguaje LaTeX	20
4	Otras de formas de publicación en R	23
4.1	Notebooks	23
4.2	Presentaciones	23
4.3	Páginas web y libros	24
4.4	Cuadros de mando	26

5	Gestión de bibliografías con Bibtex	29
5.1	Sobre Bibtex	29
5.2	Referencias bibliográficas en Rmarkdown.	30
5.3	La documentación de los paquetes	31
6	Sobre el autor	33
7	Bibliografía	35

Capítulo 1

Introducción al lenguaje R

1.1 ¿Que es R?

R o The R Project for Statistical Computing (R Core Team 2020) es un lenguaje y entorno de programación de alto nivel, orientado a objetos y fundamentalmente centrado al análisis estadístico y gráfico, distribuido bajo la licencia GNU GPL y está disponible para los sistemas operativos Windows, Macintosh, Unix y GNU/Linux.

El origen de R se remonta a 1993 cuando Robert Gentleman y Ross Ihaka del Departamento de Estadística de la Universidad de Auckland adaptan el lenguaje de programación S, propio de los famosos Bell Laboratories de AT&T, el cual es un sistema para el análisis de datos desarrollado por John Chambers y Rick Becker fundamentalmente y usado desde finales de 1970 (Ross y Robert comienzan a llamar “R” al lenguaje que implementaron, por las iniciales de sus nombres). Es más, muchos de los libros y manuales sobre S son útiles para R.

1.2 ¿Por qué usar R?

R tiene varias ventajas respecto a otros lenguajes de programación de fines estadísticos o softwares dedicados. Entre otras podemos citar:

- Sus posibilidades gráficas son excelentes y muy versátiles. Esto llega hasta el punto de que muchos que R tiene dos aspectos: el de software, vamos a llamarlo, analítico y el gráfico, pudiendo incluso independizarse su aprendizaje.
- Cuenta con una elevada flexibilidad. Los trabajos más sencillos o complejos algoritmos pueden ejecutarse escribiéndose una palabra.

- No es “una caja negra” y en todo momento sabemos que estamos haciendo. Este es un aspecto que no se valora como se debería cuando hacemos nuestros análisis estadísticos: a costa de la facilidad de hacer todo con un par de clicks perdemos fácilmente el control de lo que estamos haciendo (o incluso llegamos a desconocerlo), algo que no debería ser permitido, ya que quien hace un análisis busca una conclusión y no saber cómo se obtiene debería invalidar por sí solo dicha conclusión.
- En R, un análisis estadístico se realiza por pasos que arrojan resultados que se van almacenando en objetos. De esta forma, estos pueden recuperarse y usarse en cualquier momento.
- Al contrario que otros software, permite salidas mínimas de resultados, fácilmente legibles y analizables, y no una salida copiosa de datos y gráficos.
- Salvo que utilicemos una GUI, su aspecto es minimalista y espartano a conciencia (hasta su web sigue esta filosofía), siguiendo la filosofía de no perder el foco de atención de lo que estamos haciendo con el “marco” en el que lo hacemos.
- Es libre, con filosofía y objetivos del proyecto GNU, con lo que podemos acceder al código escrito por otros usuarios y modificarlo libremente y evidentemente, podemos programar nuestras propios procedimientos y aplicaciones.
- Es un proyecto vivo con dos millones de usuarios. Seguramente a nuestro problema ya se habrá enfrentado alguien y casi con total seguridad no deberemos avanzar más de “tres páginas de Google” para encontrar la solución.
- Existen multitud de librerías (paquetes) programadas por los usuarios de todo el mundo para llevar a cabo procedimientos específicos (¡más de 10.000!).
- Y si con todo ello, no tenemos suficiente, es totalmente gratuito, tanto el como todo su ecosistema.

1.3 Recursos sobre R

Para aprender sobre R y estar al corriente de las últimas novedades existen una serie de recursos que deberemos añadir a nuestros Favoritos. Entre otros:

- Página oficial de R: <https://www.r-project.org/>
- Fundación R: <https://www.r-project.org/foundation/>
- Manuales iniciales: <https://cran.r-project.org/manuals.html>
- Revista R: <https://journal.r-project.org/>
- R-Bloggers: <http://www.r-bloggers.com/>
- Comunidad R-Hispano: <http://r-es.org/Comunidad>
- Recursos MOOC como Coursera: www.coursera.org

- Bibliografía en papel: “R for Everyone: Advanced Analytics and Graphics” (<http://www.jaredlander.com/r-for-everyone/>), “R Cookbook” (<http://shop.oreilly.com/product/9780596809164.do>) y el clásico “Modern Applied Statistics with S” (<http://www.springer.com/us/book/9780387954578>)

Capítulo 2

Instalación de R, RStudio y librerías

2.1 Instalación de R

R funciona bajo Windows, Linux, Mac o Solaris. Para instalar R accederemos a su página (en este caso, un mirror de España): <http://cran.es.r-project.org/> y descargaremos la versión adecuada a nuestro sistema operativo.

En el caso de Windows, la instalación es muy sencilla y solo deberemos seguir los pasos que se nos indica. Si todo ha ido correctamente, cuando pulsemos sobre el icono de R, se abrirá su espartana interfaz en la que, además de la barra de herramientas, destaca la consola donde escribiremos y ejecutaremos nuestro código.

Algo bastante interesante de R es su portabilidad. R, una vez instalado en un PC puede funcionar de forma autónoma al mismo. Por ello, una simple copia del directorio donde lo hayamos instalado y su posterior “pegado” en una memoria flash, nos permitirá ejecutarlo en cualquier lugar, llevándonos así nuestra configuración personal, paquetes, entornos y demás.

2.2 Actualización de R

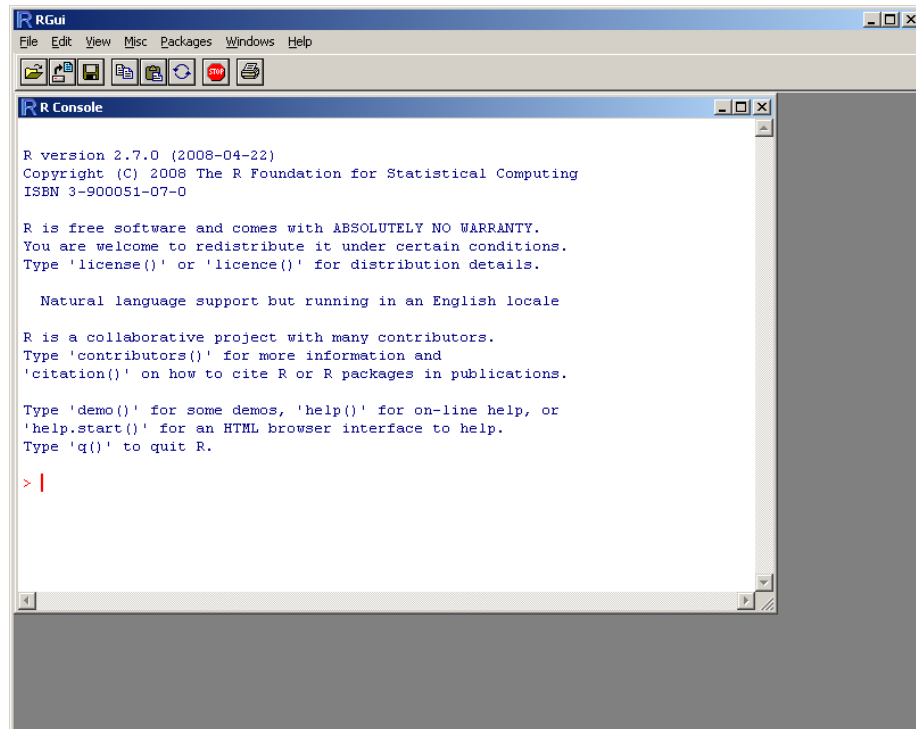
Cada cierto tiempo es recomendable realizar la actualización de R. Generalmente, un par de veces al año, el sistema es actualizado, misión que lleva a cabo un reducido grupo de especialistas (conocido como el R Core team) y los paquetes (que en próximos capítulos explicaremos) se adaptan a la nueva actualización, dejando, algunas veces de ser útiles en las versiones obsoletas.

La actualización de R puede ser automatizada o realizarse de forma manual si bien este último recurso no es la mejor opción debido a que se actualizará creando una nueva instalación manteniendo la anterior. Este echo se debe a que en la actualidad es la única forma de que no perdamos nuestras configuraciones y, sobre todo, la posible incompatibilidad con alguno de los paquetes que tengamos instalados. Para realizar una actualización automática del sistema, podemos usar la siguiente sentencia usando el paquete `installr` (Galili 2018) (despues veremos el lugar de ejecutarla):

```
if(!require(installr)) {  
  install.packages("installr"); require(installr)}  
  
updateR()
```

2.3 El Sistema R

Una vez instalamos R, nos encontramos con una interfaz gráfica de usuario que nos puede resultar “anticuada” sobre todo si trabajamos con los sistemas operativos más recientes. Este aspecto no debe confundirnos de cuál es la visión del sistema R: un lenguaje de programación con una clara vocación estadístico-matemática.



Dentro de R encontramos una pantalla situada a la izquierda (por defecto) en la que en el prompt, escribiremos los comandos a ejecutar. Los resultados de estos comandos aparecerán a continuación de los mismos en este mismo sector. En esta pantalla, aparece también la versión de R que tenemos instalada, y ciertos comandos como la forma de consultar la ayuda (`help()`) o ejemplos básicos de trabajo con R (`demo()`).

A la derecha, en su momento, aparecerá la pantalla de visualización de los gráficos resultantes de los comandos que programemos.

Finalmente, en la pantalla superior, se nos muestra la línea de herramientas, con una cantidad muy reducida de opciones para un software actual. Entre estas opciones, las más usuales son las de abrir y guardar nuestros códigos (en un formato nativo de extensión `.R`) o nuestros espacios de trabajo, para que la próxima vez que iniciemos R lo hagamos con la sesión anterior (se guarda en un formato propio con extensión `.RData`). Evidentemente también existen las opciones de utilización de portapapeles (copiar y pegar) y podremos definir el aspecto visual de nuestro software.

Asimismo, existe una opción de una importancia capital en R: Packages y que requiere un capítulo aparte.

2.4 Las librerías de R

Vamos a banalizar el sistema R: cuando instalamos R tenemos una cajón “vacío,” si podremos guardar cosas, pero poco más. Ahora bien, para nuestro cajón existen accesorios que los completan y lo hacen más útil y versátil, por ejemplo un separador para bolígrafos. Pues bien, este separador “especializado” sería una librería o paquete. Y aquí radica la potencia de R: cuando nos enfrentemos a un problema, seguramente alguien ya se habrá enfrentado y encontraremos un paquete que realice la función que necesitemos (actualmente existen más de 7.500 disponibles). Se pueden consultar todos los disponibles en el repositorio oficial de paquetes de R conocido por el nombre CRAN (Comprehensive R Archive Network): <https://cran.r-project.org/web/packages/>.

El sistema base de R contiene el paquete básico que se requiere para su ejecución y la mayoría de las funciones fundamentales. Los otros paquetes contenidos en la “base” del sistema incluye a `utils`, `stats`, `datasets`, `graphics`, `grDevices`, `grid`, `tools`, `parallel`, `compiler`, `splines`, `tcltk` y `stats4`.

Para comenzar, vamos a ver la lista de paquetes que tenemos instalados escribiendo `library()` en la pantalla de código. Si deseamos instalar un paquete que no tengamos tenemos la opción de hacerlo desde `Packages>Install Package(s)` eligiendo previamente un mirror de descarga, o bien escribiendo en la consola el comando `install.packages("nlme", dep = T)`. `nlme` es un fantástico paquete dedicado a modelos mixtos lineales y no lineales (Pinheiro, Bates, and R-core 2018) y `dep=T`, le indica a R que instale cualquier

otro paquete que sea necesario para el correcto funcionamiento del paquetes principal a instalar).

Finalmente, existe una opción más tediosa, a través de un zip local. Si nuestro paquete, por la razón que sea, no está en un repositorio web de R, podremos instalarlo desde nuestro ordenador mediante `Packages>Install package(s) from local zip file...` Finalmente no queda más que cargar el paquete deseado y empezar a trabajar con él. Para ello, escribiremos el comando `library(nlme)` y, automáticamente se cargará (y sus paquetes dependientes si lo necesita).

Todos los paquetes de R están perfectamente documentados según un estándar de la comunidad y si esto no es poco, los paquetes más importantes disponen de incluso bibliografía específica que en muchas ocasiones los creadores del paquete escriben para facilitar su aprendizaje. Estos manuales están a la venta en las librerías en formato papel y las más de las veces son descargables de forma gratuita (como ejemplo valga <http://www-bcf.usc.edu/~gareth/ISL/>).

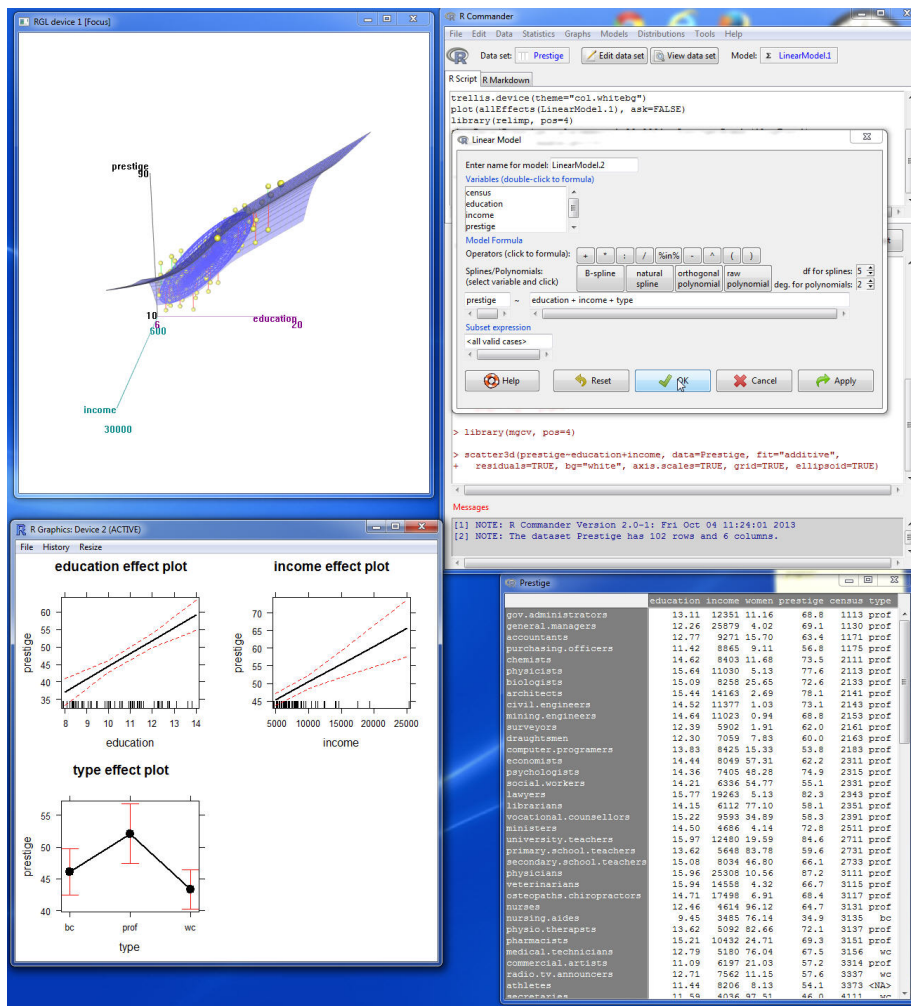
2.5 Interfaces gráficas de R

Como vemos, R no destaca por el aspecto visual de su interfaz gráfica de usuario nativa. Por ello existen toda una serie de “capas” para el mismo que enriquecen la experiencia del usuario tanto desde el punto de vista de su aspecto visual como desde el punto práctico. En este punto, encontramos ejemplos como RKWard Rattle o Deducer, pero quizás los más destacados sean R Commander y sobre todo RStudio que se ha convertido casi en el hermano gemelo de R.

2.5.1 Rcommander

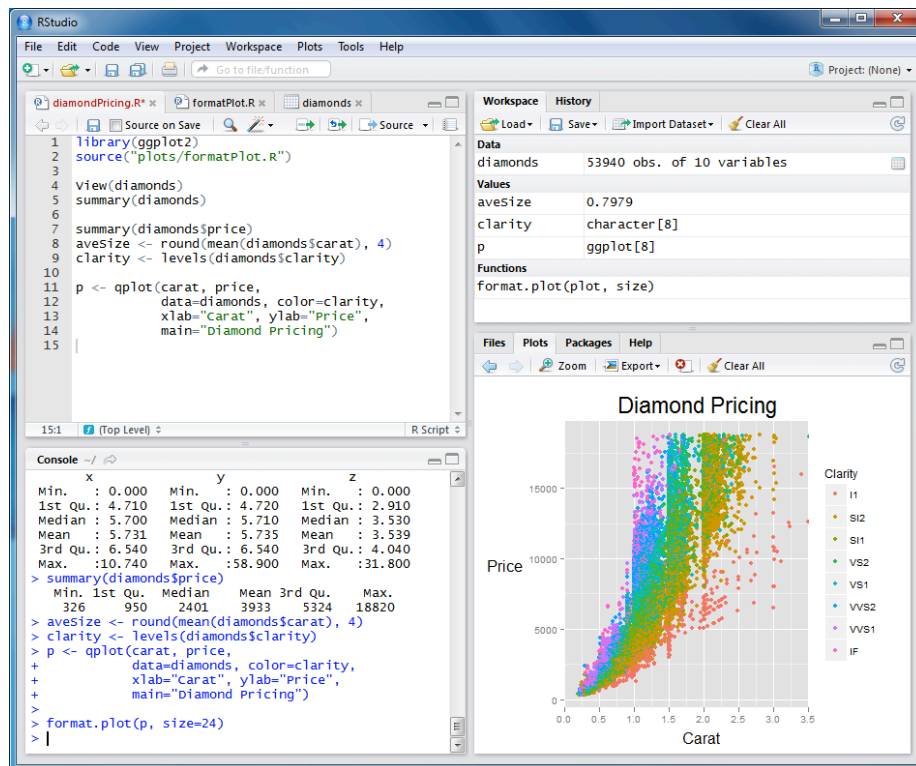
Rcommander (Fox and Bouchet-Valat 2019) es una interfaz gráfica de R, diseñada para realizar los trabajos estadísticos y analíticos más comunes sin la necesidad de dominar el lenguaje R. En simples palabras convierte el sistema R en un software “estadístico” tradicional. Sin duda, si nuestro propósito es realizar análisis sencillos o buscamos algo rápido, Rcommander es el ganador.

Rcommander se instala como cualquier paquete, por ejemplo mediante el comando `install.packages("Rcmdr")` y lo cargaremos con `library(Rcmdr)`.



2.5.2 RStudio

RStudio es un entorno de desarrollo integrado (IDE) para R y es un paso más allá de Rcmdr. En concreto y según la definición de sus autores se trata de una consola, editor de sintaxis que apoya la ejecución de código, así como herramientas para el trazado, la depuración y la gestión del espacio de trabajo, es decir un completo gestor del entorno R que se ha convertido en parte inseparable del mismo: si alguien usa R seguro, que con un 95% de probabilidad lo hace con RStudio.



La instalación de RStudio se realiza desde su página web <https://www.rstudio.com/products/RStudio/> y no difiere de una instalación estándar de los sistemas operativos habituales.

Su uso es muy sencillo y lo iremos descubriendo a lo largo de estas notas. Básicamente se compone de cuatro ventanas configurables por el usuario donde se escriben todas las operaciones, se muestran nuestras “tablas de datos,” los resultados gráficos y numéricos, se facilita la gestión del histórico, instalación y carga de paquetes,... en fin un largo etcétera que hace más sencilla la creación de soluciones en el sistema R. Una de las grandes ventajas de RStudio es la existencia de un editor de sintaxis que apoya la ejecución de código, facilitando la escritura del mismo y ayudando a recordar los innumerables comandos y funciones. Por otro lado, el trabajar con los proyectos de Rstudio, nos facilitará tener almacenado todo nuestro trabajo para recuperarlo cada vez que queramos, despreocupándonos de los ficheros origen de datos.

Capítulo 3

Markdown, LaTeX y Pandoc

3.1 La escritura con Markdown

Markdown es un lenguaje de marcado ligero ideado en 2004 por John Gruber, con la visión de crear un lenguaje sencillo, fácil de leer y de escribir, aprovechando las ventajas del lenguaje HTML pero eliminando sus inconvenientes, como por ejemplo las etiquetas, que tanto engorran el código. Pongamonos en situación: si estamos escribiendo en HTML, para añadir una palabra en negrita deberíamos escribir `importante` mientras que si estamos escribiendo en Markdownnos bastaría poner `importante` para que cuando lo compilemos, se muestre de esta forma. Huelga decir, que en un procesador de textos al uso deberemos dejar de escribir, coger el ratón, pulsar en el botón de la negrita, volver a escribir, de nuevo coger el ratón, ir a la barra de herramientas, desactivar el formato de negrita y continuar escribiendo (...). A continuación podemos ver algunos ejemplos de la notación de este lenguaje:

sintaxis

```

Texto plano
Termina línea con dos espacios para nuevo párrafo.
*cursivo* y _cursivo_
**negrita** y __negrita__
superíndice^2^
~~tachado~~
[eslabón](www.rstudio.com)

# Encabezado 1

## Encabezado 2

### Encabezado 3

#### Encabezado 4

##### Encabezado 5

##### Encabezado 6

raya em: --
raya em: ---
elipsis: ...
ecuación en línea: $A = \pi * r^2$

imagen: 

regla horizontal (o nueva diapositiva):

***

> cita en bloque

* lista sin orden
* elemento 2
  + sub-elemento 1
  + sub-elemento 2

1. lista ordenada
2. elemento 2
  + sub-elemento 1
  + sub-elemento 2

Encabezado Tabla | Segundo Encabezado
-----|-----
Celda de tabla | Celda 2
Celda 3 | Celda 4

```

resulta en

Texto plano
 Termina línea con dos espacios para nuevo párrafo.
cursivo y *cursivo*
negrita y **negrita**
 superíndice²
 tachado
 eslabón

Encabezado 1

Encabezado 2


Encabezado 3

Encabezado 4

Encabezado 5

Encabezado 6

raya em: –
 raya em: —
 elipsis: ...
 ecuación en línea: $A = \pi * r^2$

imagen: 

regla horizontal (o nueva diapositiva):

cita en bloque

- lista sin orden
 - elemento 2
 - sub-elemento 1
 - sub-elemento 2
1. lista ordenada
 2. elemento 2
 - sub-elemento 1
 - sub-elemento 2

Encabezado Tabla	Segundo Encabezado
Celda de tabla	Celda 2
Celda 3	Celda 4

En este punto nos puede asaltar la duda ¿Markdown cubre todo el espectro que cubre HTML? Sí y no, porque si no lo cubre, tan solo tienes que escribir la parte de HTML separado una línea del siguiente párrafo en Markdown.

Trabajar con Markdown, tiene infinidad de ventajas, pero es especialmente adecuado para aquellas personas que publican contenido en la web de manera regular. Entre otras ventajas cabe citar que cuando te acostumbras a la sintaxis la generación de texto es extremadamente rápido, es muy sencillo de leer, no tendremos el problema de errores por no cerrar adecuadamente las etiquetas HTML y su compilación con software como **Pandoc** permite la exportación simple a infinidad de formatos basados en texto.

3.2 Rmarkdown

Rmarkdown es un framework derivado de markdown que nos va a permitir, a la vez, ejecutar líneas de código de R y generar multitud de formatos de publicación, listos para imprimir y/o compartir con los interesados.

Para trabajar con estos tipos de documentos, lo más recomendable es el uso de RStudio, que en sus últimas versiones, viene equipado con todos los elementos necesarios para ejecutar y compilar correctamente estos documentos.

La composición de un documento en Rmarkdown es muy sencilla. Para ello tendremos la opción simple, si estamos en RStudio de File>New File>R Markdown, que nos generará una plantilla con los elementos básicos para la confección del documento. En ella, destacan los tres elementos fundamentales:

- El encabezado YAML, donde se definen las características del documento:

```
---
title: "Untitled"
author: "Julio M. Sevilla"
date: "20/6/2019"
output: html_document
---
```

- Todos los elementos de texto del documento, escritos en markdown:

```
## R Markdown
```

```
This is an R Markdown document.
```

- Los fragmentos de código o chunks de R, iniciados con ````${r}```` y finalizados con `````. Dentro de ````${r}```` podremos incluir desde el nombre del chunk (que nos permitirá identificarlo rápidamente) a acciones como que no se ejecute, no se muestre el código, etc. tal que, por ejemplo ````${r} coches, eval=FALSE``. En concreto:
 - `include = FALSE` impide que el código y los resultados aparezcan en el archivo terminado, si bien los resultados existirán en la memoria.
 - `echo = FALSE` impide que el código, pero no los resultados, aparezcan en el archivo terminado. Esta es una forma útil de incrustar figuras y datos donde no importa tanto como se construye si no el resultado final
 - `message = FALSE` impide que los mensajes generados por el código aparezcan en el archivo terminado.
 - `warning = FALSE` impide que las advertencias generadas por el código aparezcan en el terminado.

– `fig.cap = "..."` agrega un título a los resultados gráficos.

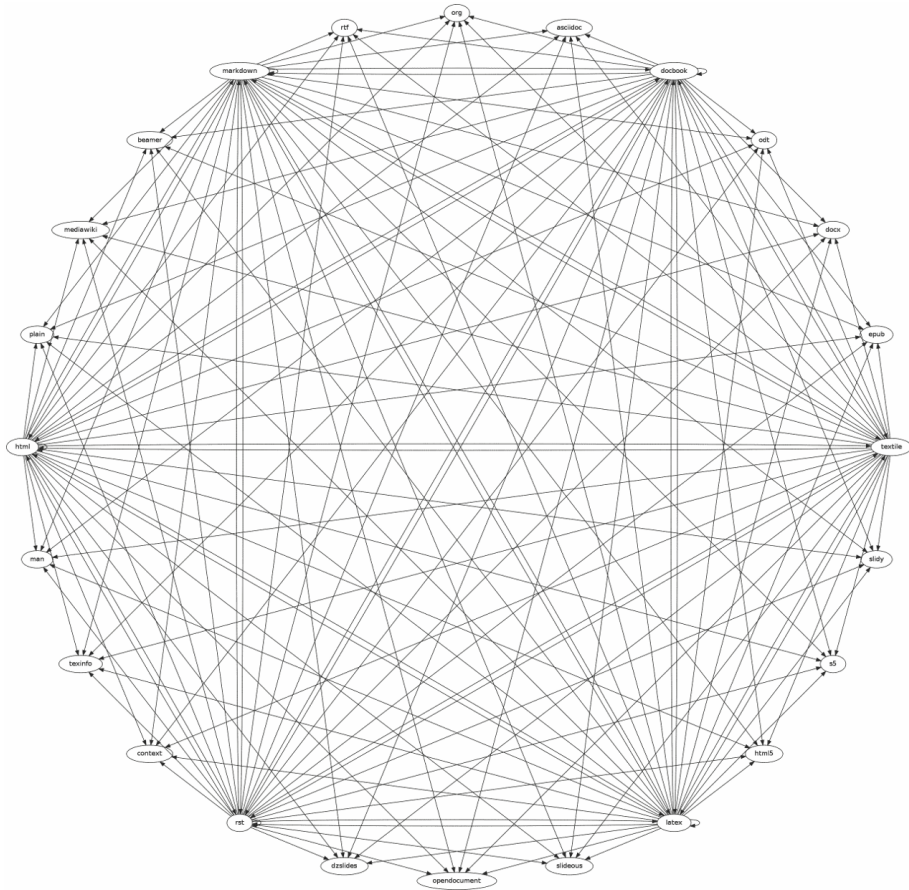
En relación a esto último, si deseamos establecer opciones globales que se aplican a cada fragmento del archivo, podemos crear un fragmento de código inicial en el que introduciremos `knitr::opts_chunk$set()` y añadiendo dentro del paréntesis las distintas acciones que deseemos (cuando creamos un nuevo script de RMarkdown veremos que este fragmento inicial se crea por defecto con `echo = TRUE`)

Cuando compilemos el documento, previo guardado en nuestra computadora, a través de pulsar en el botón Knit de RStudio, se generará un documento, en este caso en formato HTML, listo para impresión y con los resultados de nuestro análisis.

En <https://www.rstudio.com/wp-content/uploads/2015/03/rmarkdown-reference.pdf> Tenemos una guía completa de lenguaje (también podemos acceder a ella en RStudio desde Help>Cheatsheets)

3.3 Pandoc y la conversión entre formatos.

Pandoc es una herramienta de libre y de código abierto, creado por John MacFarlane y que permite la conversión entre múltiples formatos de marcado a partir de un motor de renderizado común:



Basicamente, el trabajo con pandoc consiste en que, una vez tengamos creado el documento, para lo cual recomendamos se haga en Markdown, el motor de renderizado de Pandoc lo convertirá en el formato que deseemos.

3.4 Instalación de los elementos necesarios

Para comenzar deberemos instalar todas las librerías que a priori vamos a necesitar para gestionar nuestros documentos científicos:

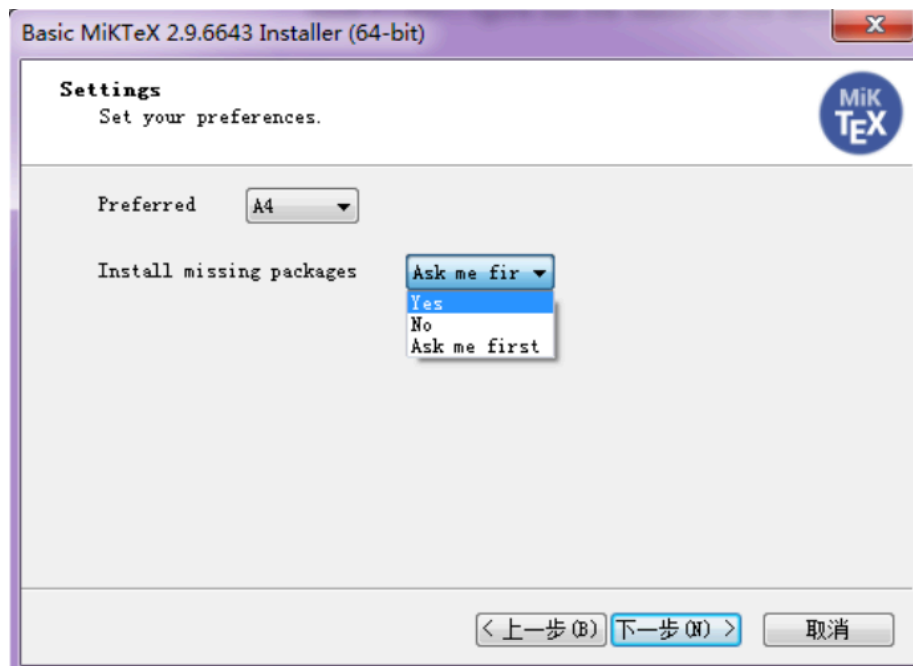
- En el caso de que usemos RStudio no será necesario, pero en caso contrario es obligatorio la instalación de Pandoc
- Con RStudio, solo deberemos instalar la librería `rmarkdown` sin tener que ejecutar el paso anterior

```
install.packages('rmarkdown')
```

- En general queremos derivar el resultado final en pdf, pero para ello, antes deberemos instalar LaTeX. La forma más sencilla de ejecutar este paso es mediante la instalación de la librería **TinyTeX**. Debemos tener en cuenta que esta librería, para instalarse necesita de Rtools en nuestro Pc, por lo que si no lo tenemos deberemos instalarlo.

```
tinytex::install_tinytex()
```

En ocasiones, la instalación de **TinyTeX** deriva en problemas y no se pueden compilar los documentos. Si esto ocurre, deberemos instalar **MiKTeX** (en el caso de que nuestro sistema operativo sea Windows). Al hacerlo, deberemos indicarle que instale todas las aplicaciones necesarias:



Si ya tenemos este software instalado en nuestro PC, deberemos ir a la configuración y en Settings, elegir la opción **Always install missing packages on-the-fly**.

3.5 El lenguaje LaTeX

Anteriormente hemos apuntado que para generar el formato pdf es necesario pasar por un paso previo, instalar LaTeX, en concreto el sistema que nos

permite su compilación. LaTeX es un sistema de preparación de documentos por comandos de bajo nivel, usado para escribir grandes textos científicos y técnicos, que fue desarrollado en 1985 por Leslie Lamport y que a diferencia de los procesadores de texto actuales WYSIWYG, permite al escritor centrarse en el contenido sin tener que preocuparse por el formato, algo que realizará el sistema LaTeX.

A grandes rasgos, un documento en LaTeX se compone de la siguiente estructura:

```
\documentclass[opciones]{clase}  
Preámbulo  
\begin{document}  
Documento  
\end{document}
```

En las opciones se definirán al menos el tamaño del papel (por ejemplo, `a4paper`) y el tamaño de letra (por ejemplo, `12pt`).

Si queremos ahondar más en la escritura con LaTeX recomendamos la lectura del documento <https://www.uv.es/~ivorra/Latex/LaTeX.pdf> de Carlos Ivorra.

En un documento de Rmarkdown, la generación de un documento en LaTeX es muy sencillo si tenemos ya instaladas las librerías citadas antes. Con ello, solo deberemos configurar el encabezado de nuestro documento como:

```
---  
title: "Untitled"  
author: "Julio M. Sevilla"  
date: "20/6/2019"  
output:  
  html_document:  
    toc: true  
  pdf_document:  
    keep_tex: true  
---
```

De esta forma, donde se haya guardado el documento `Rmd` también se generarán los documentos compilados en `HTML`, `pdf` y `tex`.

Capítulo 4

Otras de formas de publicación en R

En este capítulo, veremos otras formas de publicar en R pero no las únicas. Si es de interés, recomendamos visitar la web [R Markdown: The Definitive Guide](#), donde hay abundante material sobre el tema.

4.1 Notebooks

Un notebook es un documento de RMarkdown que puede ejecutar trozos de código de forma independiente y de forma interactiva. Para crear un documento, lo podremos hacer fácilmente desde RStudio en `File>New File>R Notebook`. Una vez creado contaremos con un fichero `*.rmd` que contendrá todas nuestras partes de código y texto y que, una vez lo guardemos, nos generará un nuevo fichero, con extensión `*.nb.html` que contendrá el notebook en un formato amigable para ejecutar.

4.2 Presentaciones

Además de generar reportes en pdf, html, docx,..., cuando generamos nuestro fichero `.Rmd` podemos indicarle que la salida sea del tipo presentación. En este caso, cada uno de los títulos que definamos, se convertirá en un slide de la presentación, la cual puede, evidentemente, contener fragmentos de códigos ejecutables, tal y como se han descrito en estos apuntes. Podemos ver un ejemplo de una presentación en este enlace.

4.3 Páginas web y libros

Con Rmarkdown y RStudio, podremos desarrollar páginas web estáticas muy fácilmente. Ya hemos visto lo sencillo que es generar un fichero HTML por lo que si disponemos de un hospedaje web podremos subirlo y disponibilizarlo de forma pública.

4.3.1 Bookdown

Una opción que da un paso más allá es la librería **bookdown**. Mediante esta librería, podremos escribir libros y publicarlos online empleando R Markdown de forma rápida y sencilla, por lo que se convierte en la forma ideal de generar el documento y, a su vez, publicarlo. Para publicarlo, podemos usar repositorios como Github o Gitlab, que nos ofrecen un almacenamiento gratuito.

Existen varias formas para generar un documento de este tipo, pero sin duda la más sencilla es instalar la librería con `devtools::install_github('rstudio/bookdown')`, crear un proyecto en RStudio, clonando el repositorio <https://github.com/yihui/bookdown-minimal.git> y, además de generar nuestro contenido deberemos editar la cabecera del fichero `index.Rmd`:

```
---
title: "A Minimal Book Example"
author: "Yihui Xie"
date: "2019-03-25"
site: bookdown::bookdown_site
output: bookdown::gitbook
documentclass: book
bibliography: [book.bib, packages.bib]
biblio-style: apalike
link-citations: yes
github-repo: rstudio/bookdown-demo
description: "This is a minimal example of using the bookdown package to write a book."
---
```

Para publicar el libro en Gitlab a modo de página web, dentro del proyecto, a su vez deberemos crear un fichero `.gitlab-ci.yml` con, al menos el siguiente contenido:

```
image: rocker/verse:3.4.1

before_script:

pages:
  stage: deploy
```



```
script:
- Rscript -e "bookdown::render_book('index.Rmd', 'all', output_dir = 'public')"
artifacts:
  paths:
  - public
only:
- master
```

Si deseamos hacer en Github, deberemos realizar lo siguiente:

- Crear el directorio `docs` dentro del repositorio creado
- Incluir dentro del directorio `docs` incluir el contenido de nuestro libro
- Crear dentro del directorio `docs` un archivo `.nojekyll`, que, si estamos en Windows, podremos hacer incluso con el comando de R `file.create('.nojekyll')`.
- Finalmente, en Github, tendremos que decirle que nuestro master branch es `\docs\`:

GitHub Pages

GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository.

✓ Your site is published at <https://fmr1789.github.io/>

Source
Your GitHub Pages site is currently being built from the `/docs` folder in the `master` branch. [Learn more.](#)

master branch /docs folder Save

Custom domain
Custom domains allow you to serve your site from a domain other than `fmr1789.github.io`. [Learn more.](#)

Save

☒ **Enforce HTTPS**
— Required for your site because you are using the default domain (`fmr1789.github.io`)

HTTPS provides a layer of encryption that prevents others from snooping on or tampering with traffic to your site. When HTTPS is enforced, your site will only be served over HTTPS. [Learn more.](#)

Otra opción, quizás más sencilla es usar las posibilidades del paquete RStudio Connect `rsconnect`, 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. Instalando previamente esta librería, podremos subir nuestro nuevo libro digital a los servidores de bookdown.org mediante el comando `bookdown::publish_book("usuario")`, disponiendo en `usuario`, nuestro identificador (si no tenemos cuenta aún, en el proceso, nos facilitará crearla).

4.3.2 Distill

Si bien Bookdown está muy centrado en la redacción científica, el paquete Distill es multipropósito, y nos servirá para crear fácilmente blogs y webs estáticas.

Para ello, instalaremos la susodicha librería (`install.packages("distill")`) y, una vez instalado, desde RStudio, podremos genera un nuevo proyecto de web o blog, mediante `File>New Project>New Directory` y, de las opciones elegimos `Distill Blog` o `Distill Website`

Una vez lo nombremos y lo guardemos en la ruta que elijamos, nos generará varios ficheros tales como el encabezado (`_site.yml`), la portada (`index.Rmd`), poágina de presentación (`about.Rmd`),... así como entradas de blog ficticias o páginas web, que podremos en todos los casos cambiar y reconfigurar.

LA creación o modificación de páginas es muy sencilla, ya que en todo momento usaremos `Rmarkdown` para escribir y posteriormente, para crear la página en lenguaje `html` deberemois “tejer” la misma, mediante el comando `knitr`. Para finalizar y montar la web completa, procedemos a pulsar sobre la pestaña `Build` de RStudio que se encargará de montarlo todo.

Para añadir nuevas páginas o post, en la consola de comandos, deberemos ejecutar `distill::create_post()` con el nombre del post dentro del paréntesis (encerrado en comillas).

4.4 Cuadros de mando

4.4.1 Flexdashboard

La libreria `flexdashboard` consiste en un conjunto de utilidades con las que generaremos facilmente cuadros de mando dinámicos. Su uso es bien sencillo; una vez cargada la libreria, cada título que definamos se convertirá en un marco donde podremos depositar la información a mostrar:

```

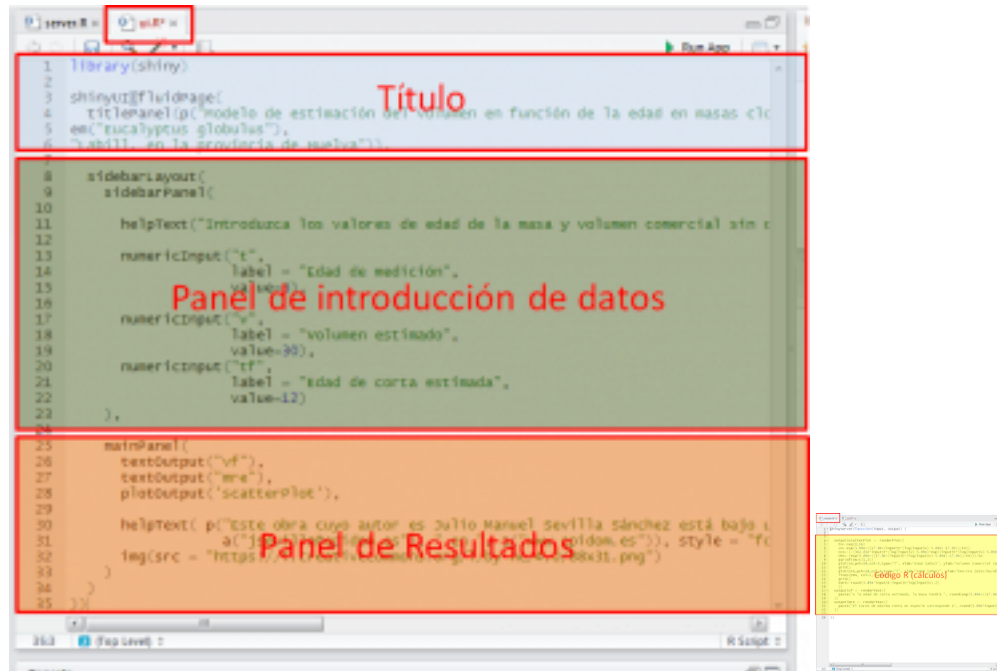
1 ---
2 title: "Multiple Columns"
3 output: flexdashboard::flex_dashboard
4 ---
5
6 Column {data-width=600}
7 -----
8
9 ## Chart 1
10
11 {r}
12
13
14
15 Column {data-width=400}
16 -----
17
18 ## Chart 2
19
20 {r}
21
22
23
24 ## Chart 3
25
26 {r}
27
28
29

```



4.4.2 Shiny

Shiny se basa en el uso de Programación Reactiva (Reactive Programming), esto es: existen unos valores que pueden variar en el tiempo y que son registrados por unas expresiones que posteriormente las reproducen. Junto a este motor de razonamiento, incorpora adaptadas los clásicos elementos HTML (botones, cuadros de texto, barras de desplazamiento,...) junto con las potentes gráficas de R y su estructura de datos. Básicamente se compone de dos ficheros que se crean en nuestro directorio local: `ui.R`, que gestiona la salida visual y el aspecto de la aplicación y `server.R` con las instrucciones para que la misma función, siendo en esta última donde se embebe el motor de cálculo en realizado en R. En las últimas versiones del paquete, ambos ficheros vienen embebidos en uno solo `app.R`. Estos archivos son publicados en nuestro servidor web o en las distintas plataformas que lo interpreten como GitHub.



Las posibilidades de Shiny se amplían al admitir incorporaciones al código nativo de todos los elementos que actualmente conforman los ecosistemas HTML, CSS y Javascript, lo que implica que los desarrollos que se puedan hacer con esta herramienta son casi ilimitados. Esto la convierte en una herramienta muy poderosa para la construcción de cuadros de mando y para mostrar los resultados de nuestros desarrollos usando las capacidades analíticas de R, de forma que el receptor de nuestra información pueda interactuar fácilmente con ella (por ejemplo como en este caso).

Capítulo 5

Gestión de bibliografías con Bibtex

5.1 Sobre Bibtex

Bibtex es una herramienta auxiliar de LaTeX, diseñada para facilitar el manejo de la bibliografía y es un estandar de almacenamiento de referencias. Su uso es muy sencillo: a modo de base de datos, en un fichero con extensión `bib` almacenaremos todas nuestras referencias bibliográficas, independientemente de que las usemos o no en el documento actual, algo que es una gran ventaja, pues podremos crear nuestro fichero, único, de referencias bibliográficas, que usaremos en todos nuestros proyectos.

La cita de la fuente es muy sencilla: simplemente, en nuestro documento LaTeX, deberemos llamar al identificador de la referencia, para que el compilador la muestre de forma adecuada. Veamos unos ejemplos:

```
@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 = {2019},
  url = {https://www.R-project.org/},
}

@article{article12,
  author = {Peter Adams},
  title = {The title of the work},
  journal = {The name of the journal},
```

```

year    = 1993,
number  = 2,
pages   = {201-213},
month   = 7,
note    = {An optional note},
volume  = 4
}

```

En estos casos, el identificador de cada referencia es el texto que va a continuación de `{`, `R-base` o `article12`.

En internet, podremos encontrar software específico para gestionar nuestros ficheros `bib` como <https://truben.no/latex/bibtex/> o <http://www.jabref.org/> aunque, si bien, el fichero `bibno` es más que un fichero de texto, por lo que cualquier editor (hasta el simple notepad del sistema nos servirá).

5.2 Referencias bibliográficas en Rmarkdown.

Existen varias formas de gestionar las referencias bibliográficas en el documento, pero lo más recomendable es trabajar con el estilo de citas BiBtex y el almacenamiento en este formato, especialmente si el resultado buscado es `pdf` y/o `tex`.

```

@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 = {2019},
  url = {https://www.R-project.org/},
}

```

Para que Rmarkdown reconozca la cita, deberemos añadir a nuestro encabezado YAML la ruta donde almacenaremos todas nuestras referencias bibliográficas (nuestra base de datos en formato `bib`). Si este se ubica en el mismo lugar que el fichero `Rmd` podremos usar simplemente:

```

---
title: "Untitled"
author: "Julio M. Sevilla"
date: "20/6/2019"
output: html_document
bibliography: bibliografia.bib
---

```

Por otro lado, en el documento que estemos escribiendo, será tan simple como usar la llamada que hayamos definido con anterioridad en nuestra base de datos: en este ejemplo, `@R-base` que generará “R Core Team (2020)” o `[@R-base]` que creará “(R Core Team 2020).” Pandoc, a su vez, creará un listado de referencias bibliográficas al final del documento. Este listado, se basa en el manual de estilo de Chicago por lo que si deseamos otro, lo deberemos definir a partir de un fichero `csl` (un manual muy útil es <http://docs.citationstyles.org/en/1.0.1/primer.html>), el cual debe también incluirse en el YAML:

```
---
title: "Untitled"
author: "Julio M. Sevilla"
date: "20/6/2019"
output: html_document
bibliography: bibliografia.bib
csl: biomed-central.csl
---
```

5.3 La documentación de los paquetes

Como sabemos todos los paquetes o librerías que usamos en R deben ser correctamente referidos. Evidentemente, esta gestión dentro del propio R es muy sencilla. Para ello, al comenzar nuestro documento Rmarkdown, podemos definir un comando para que nos cree automáticamente las referencias bibliográficas adecuadas de cada paquete mediante la sintaxis:

```
knitr::write_bib(c(.packages(), 'ggplot2', 'nlme'), 'packages.bib')
```

Por supuesto, ahora, tendremos que decirle al compilador que debe “buscar” en dos ficheros `bib` para lo cual deberemos extender el YAML de la siguiente forma:

```
---
title: "Untitled"
author: "Julio M. Sevilla"
date: "20/6/2019"
output: html_document
bibliography: ["packages.bib", "bibliografia.bib"]
---
```

De esta forma, se generará, automáticamente, un fichero específico, en formato `bib` con las referencias bibliográficas de todos los paquetes que le digamos y después será tan simple, como, de nuevo, escribir en el texto markdown la referencia al paquete a modo `@R-nlme` o `@R-ggplot2` generando, en el primer caso: Pinheiro, Bates, and R-core (2018)

Capítulo 6

Sobre el autor

Mi nombre es Julio M. Sevilla y soy Ingeniero Forestal por la Universidad de Huelva con formación de postgrado en Métodos Avanzados de Estadística Aplicada y Máster en Técnicas Actuales de Estadística Aplicada. A raíz de iniciar mi vida profesional en el área de la investigación, comienzo a apasionarme por la Estadística Aplicada y los Sistemas de Información Geográfica, especializándome en técnicas de muestreo, análisis espacial mediante Sistemas de Información Geográfica y modelización estadística. En los últimos años mi formación va encaminada por las emergentes técnicas de aprendizaje automático estadístico orientadas al análisis espacial, la teledetección, geoestadística y LiDAR. Asimismo, complemento esta formación con preparación en gestión y administración de empresas, inteligencia de negocio y gestión y valoración de proyectos. Por otro lado, soy autor y co-autor de varias publicaciones científicas.

Si quieres saber más, puedes ponerte en contacto a través del mail y las redes sociales:

- jsevilla.es
- julio@jsevilla.es
- [Linkedin](#)

Capítulo 7

Bibliografía

- Fox, John, and Milan Bouchet-Valat. 2019. *Rcmdr: R Commander*. <https://CRAN.R-project.org/package=Rcmdr>.
- Galili, Tal. 2018. *Installr: Using r to Install Stuff (Such as: R, 'Rtools', RStudio, 'Git', and More!)*. <https://CRAN.R-project.org/package=installr>.
- Pinheiro, José, Douglas Bates, and R-core. 2018. *Nlme: Linear and Nonlinear Mixed Effects Models*. <https://CRAN.R-project.org/package=nlme>.
- R Core Team. 2020. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. <https://www.R-project.org/>.