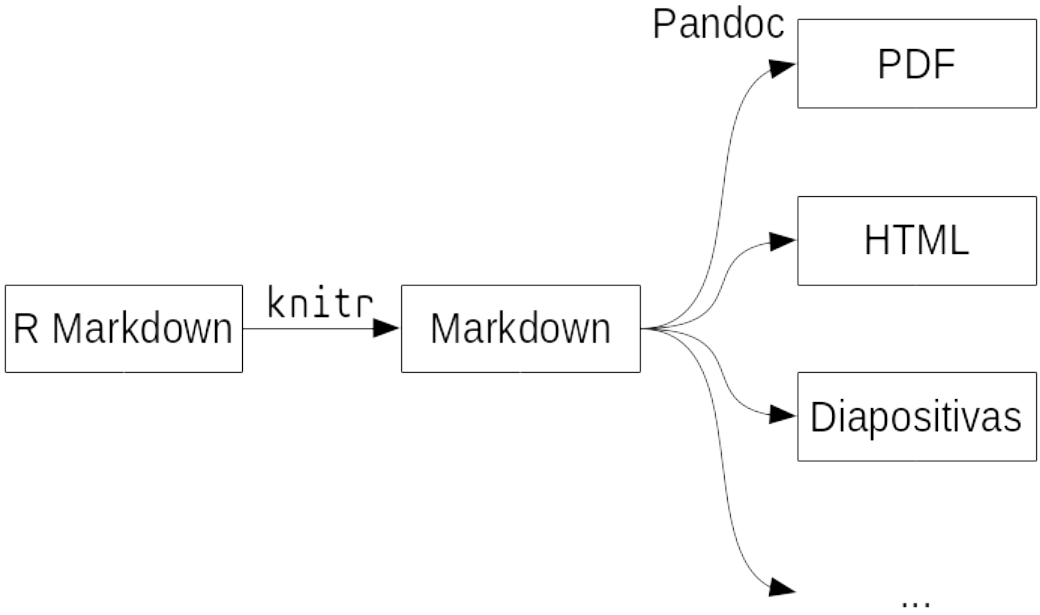
Guía de R Markdown

Santiago Gualchi

4 de abril de 2019

¿Qué es R Markdown?

R Markdown es una tecnología para la edición de documentos que se apoya sobre el paquete `knitr` y el convertidor **Pandoc**. `knitr` se ocupa de traducir el código R Markdown a **Markdown** (sin R), un **lenguaje de marcado** creado por John Gruber ampliamente usado por distintos proyectos como, por ejemplo, Wikipedia. Por su parte, Pandoc se ocupa de convertir el código Markdown al formato de salida deseado.



Markdown y los lenguajes de marcado

Podemos pensar los lenguajes de marcado como sistemas de anotación de textos que, justamente, permiten *marcar* el contenido con distintos propósitos sin necesidad de alterarlo. Un ejemplo de lenguaje de marcado con siglos de historia y desarrollo son los sistemas de puntuación. La puntuación es un conjunto de marcas con significado convencional. El punto por ejemplo puede indicar el fin de una oración, mientras que el espaciado puede marcar el fin de una palabra.

Otros ejemplos de lenguajes de marcado son LaTeX, que es usado especialmente para la confección de documentos científicos por su flexibilidad a la hora de insertar caracteres y formatos especiales:

```
Este texto en \LaTeX{} soporta un amplio número de \textbf{opciones de
formato} así como de caracteres especiales como $\alpha$ o $\preceq$.
```

> Este texto en LaTeXsoporta un amplio número de **opciones de formato** así como de caracteres especiales como α o $\preceq$.

y HTML (*HyperText Markup Language*), que es la principal tecnología detrás de los sitios web:

```
<div>
  <h2>Junio</h2>
  <ul>
    <li>
      <a href="https://gesel.github.io/cron/20190612.html">
        miércoles 12 de junio
      </a>
    </li>
    <li>
      <a href="https://gesel.github.io/cron/20190626.html">
        miércoles 26 de junio
      </a>
    </li>
  </ul>
</div>
```

> ## Junio
>
> - miércoles 12 de junio
> - miércoles 26 de junio

Markdown pertenece a un tipo de lenguajes de marcado llamado **lenguajes de marcado ligero**. Se los conoce de este modo porque usan un sistema de marcas que pretende ser mínimamente intrusivo y sencillo de aprender. El código en Markdown que produce los mismos resultados que el código HTML encima es:

```
## Junio

 * [miércoles 12 de junio](https://gesel.github.io/cron/20190612.html)
 * [miércoles 26 de junio](https://gesel.github.io/cron/20190626.html)
```

En Markdown, dos hashtags antes de una línea de texto espcifican un encabezado de nivel 2, los asteriscos en posición inicial definen una lista no ordenada y la sintaxis `[nombre](URL)` marca un hipervínculo.

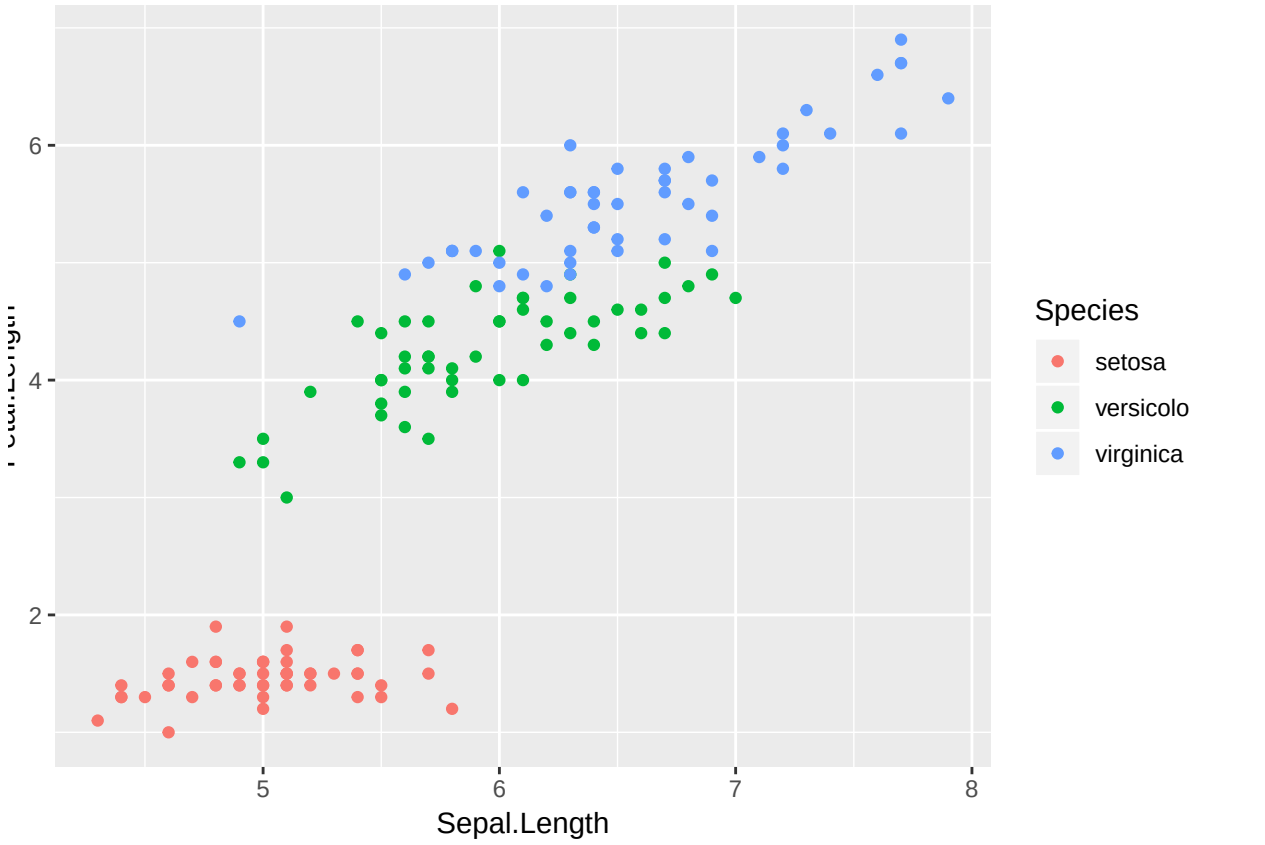
¿Qué es Pandoc?

Pandoc es un convertidor de documentos desarrollado por John MacFarlane y puede, por ejemplo, convertir documentos en Markdown a otros formatos como HTML, MS Open XML (.docx), OpenDocument (.odt), EPUB, LaTeX, PDF, otros lenguajes de marcado ligero y más. Así, Pandoc nos permite escribir documentos portables independientemente del formato de salida. Además, Pandoc extiende la sintaxis básica de Markdown permitiendo la inserción de notas al pie, tablas, listas de definiciones, superíndices y subíndices, tachado, fórmulas de LaTeXy manejo de bibliografía.

¿Por qué R Markdown?

Podríamos usar directamente Markdown y convertirlo usando Pandoc, pero la ventaja de R Markdown es que nos permite escribir documentos que incluyen código en R fácilmente, como puede ser el reporte de un experimento, la documentación de un programa, un notebook con fines educativos, programación literaria o, incluso, la creación de un sitio web. Así, podemos hacer cosas muy copadas como incluir gráficos o tablas muy fácilmente:

```
ggplot(iris, aes(Sepal.Length, Petal.Length)) +
  geom_point(aes(colour = Species))
```



Disclaimer

No me voy a detener en las diferencias entre Markdown, la expansión de Pandoc sobre la sintaxis de Markdown, ni las características específicas de R Markdown, pero sí vale decir que cada uno es (casi) un superconjunto del anterior. O sea, (casi) todo código Markdown es a su vez "Markdown-Pandoc", y (casi) todo código "Markdown-Pandoc" es código R Markdown. Si se quieren ver las diferencias específicas se puede consultar la documentación de cada lenguaje:

- documentación de Markdown;
- documentación de Pandoc; y
- documentación de `knitr`

Más adelante voy a introducir algunos elementos de la sintaxis de R Markdown, pero quiero hacer énfasis en que esta **no es una guía exhaustiva** de todas sus características.

Instalación de R Markdown

Para poder usar R Markdown es necesario seguir estos pasos:

1) Instalar Pandoc:
   - En Debian/Ubuntu y derivados:
     ```
     # apt install pandoc
     ```
   - En Fedora y derivados:
     ```
     # dnf install pandoc
     ```
   - En Arch Linux y derivados:
     ```
     # pacman -S pandoc
     ```
   - En Windows: descargar de acá
2) Instalar el paquete `tinytex` en R:
   ```
   install.packages("tinytex", dependencies = TRUE)
   ```
3) Instalar el paquete `knitr` en R:
   ```
   install.packages("knitr", dependencies = TRUE)
   ```
4) Instalar el paquete `rmarkdown` en R:
   ```
   install.packages('rmarkdown')
   ```

Primeros pasos

Ahora que ya tenemos R Markdown funcionando, lo primero que tenemos que hacer es crear un proyecto nuevo. Para eso, en RStudio, nos tenemos que dirigir a *File > New Project > New Directory > New Project*. Ahí elegimos un nombre para el proyecto y seleccionamos la carpeta donde queremos crearlo. Después de hacer esto, RStudio genera un archivo *.RProj* que vamos a abrir cada vez que queramos cargar el proyecto y va a iniciar un nuevo entorno de R.

El siguiente paso consiste en ir a *File > New File > R Markdown*, y se nos abre una ventana donde podemos elegir el título del documento, los/las autores/as y el formato de salida (esta configuración puede ser modificada luego en el bloque YAML, que es el que está arriba de todo encerrado entre guiones). Como resultado RStudio nos genera un documento R Markdown con texto de prueba. Después de borrar todo el texto que está por fuera del bloque YAML podemos empezar a escribir nuestro documento.

Sintaxis básica de R Markdown

A continuación voy a explicar algunos elementos básicos que podemos usar para escribir documentos con R Markdown.

Párrafos

Los párrafos en Markdown se delimitan mediante dos fines de línea consecutivos. Un solo fin de línea es interpretado por Markdown como un fin de palabra e inserta un espacio si es necesario. Este comportamiento es útil para organizar el texto en el editor. Se considera una buena práctica insertar fines de línea en la posición 80 porque evita que la visualización exceda el espacio visible en monitores de baja resolución.[1]

```
Esto es un párrafo. Markdown va a comenzar un nuevo párrafo solo cuando
encuentre dos fines de línea seguidos y considerará los fines de línea
simples como un fin de palabra.

Esto permite dividir el texto en líneas de 80 caracteres sin tener que
recurrir a sintaxis complejas. No obstante,
podemos
insertar
fines
de
líneas
simples
en cualquier posición y siempre se interpretará como un fin de palabra.
```

[1] Podemos activar una barra indicadora de la posición 80 yendo a *Tools > Global options > Code > Display* y activando la opción *Show margin*.

Esto es un párrafo. Markdown comenzará un nuevo párrafo solo cuando encuentre dos fines de línea seguidos y considerará los fines de línea simples como un fin de palabra.

Esto permite dividir el texto en líneas de 80 caracteres sin tener que recurrir a sintaxis complejas. No obstante, podemos insertar fines de líneas simples en cualquier posición y siempre se interpretará como un fin de palabra.

Encabezados

Markdown soporta 6 niveles de encabezados. El nivel 1, el más alto, se define mediante un hashtag, el nivel 2 mediante 2 y así sucesivamente hasta el nivel 6 que se define mediante 6 hashtags. Después de los hashtags se debe dejar un espacio antes del título.

```
# Lenguaje

[Texto]

## Características de las lenguas naturales

[Texto]

### Diversas definiciones

[Texto]

### Lenguaje humano

[Texto]

#### Origen evolutivo

[Texto]

#### Neurolingüística

[Texto]

#### Patologías

[Texto]

### Doble articulación del lenguaje

[Texto]
```

```
[//]: # (Adaptado de https://es.wikipedia.org/wiki/Lenguaje)
```

Lenguaje

[Texto]

Características de las lenguas naturales

[Texto]

Diversas definiciones

[Texto]

Lenguaje humano

[Texto]

Origen evolutivo

[Texto]

Neurolingüística

[Texto]

Patologías

[Texto]

Doble articulación del lenguaje

[Texto]

Blockquotes

Los blockquotes definen secciones en el documento que están citadas de otras fuentes.

```
El Universal 5 de Greenberg (1966) establece que:

> If a language has dominant SOV order and the genitive follows the
> governing noun, then the adjective likewise follows the noun (p.45).
```

El Universal 5 de Greenberg (1966) establece que:

> If a language has dominant SOV order and the genitive follows the governing noun, then the adjective likewise follows the noun (p.45).

Además, adentro de un blockquote se puede incluir código Markdown o anidar otros blockquotes. Esto es muy común en los correos electrónicos, cuya sintaxis fue la inspiración sobre la que se basaron los blockquotes de Markdown.

```
El dom., 17 mar. 2019 a las 4:32, Santiago (<santi@gesel.github.io>)
escribió:

> Perfecto! Gracias!!
>
> El dom., 17 mar. 2019 a las 3:24, Laura (<lau@gesel.github.io>)
> escribió:
>
>> Dale! Ya te lo mando
>>
>> El dom., 17 mar. 2019 a las 1:55, Santiago (<santi@gesel.github.io>)
>> escribió:
>>
>>> Lau, me mandás el handout de la reunión pasada?
>>>
```

> El dom., 17 mar. 2019 a las 4:32, Santiago (santi@gesel.github.io) escribió:
>
> > Perfecto! Gracias!!
> >
> > El dom., 17 mar. 2019 a las 3:24, Laura (lau@gesel.github.io) escribió:
> >
> > > Dale! Ya te lo mando
> > >
> > > El dom., 17 mar. 2019 a las 1:55, Santiago (santi@gesel.github.io) escribió:
> > >
> > > > Lau, me mandás el HO de la reunión pasada?

Formato de fuente

Énfasis

Podemos usar dos tipos de énfasis. El primero se define encerrando el texto entre asteriscos o guiones bajos simples. El segundo se define encerrando el texto entre asteriscos o guiones bajos dobles. El primer tipo se corresponde con la etiqueta HTML `em` y el segundo con la etiqueta `strong`. Por defecto, la sintaxis simple genera texto en itálicas y la doble en negritas.

```
_Este texto está en cursiva_. *Este también*. **Pero este está en
negritas**. __Igual que este__. *Separar el texto encerrado en asteriscos
o guiones bajos en líneas distintas puede producir resultados
```

```
inesperados*.

**Lo mejor es definir el énfasis en cada renglón. Sí, tenemos que**
**definir el énfasis en cada renglón**.
```

> *Este texto está en cursiva. Este también.* **Pero este esta en negritas**. **Igual que este**. *Separar el texto encerrado en asteriscos o guiones bajos en líneas distintas puede producir resultados inesperados.*
>
> **Lo mejor es definir el énfasis en cada renglón. Sí, tenemos que definir el énfasis en cada renglón**.

Tachado

Podemos marcar texto tachado entre dos virgulillas.

```
Dos eventos son ~~independientes~~ disjuntos si y solo si la
intersección entre ambos está vacía.
```

> Dos eventos son ~~independientes~~ disjuntos si y solo si la intersección entre ambos está vacía.

Subíndices y superíndices

Podemos subindizar encerrando entre virgulillas y superindizar encerrando entre acentos circunflejos.

```
La H~1~ es que *x*~1~ = *x*~2~^2^
```

> La H_1 es que $x_1 = {x_2}^2$

Listas

R Markdown soporta tres tipos de listas: ordenadas, no ordenadas y de definiciones. Recomiendo usar listas ordenadas cuando existe un orden lógico entre los elementos que se quieren listar (como en los pasos de una receta), y listas no ordenadas cuando dicho orden no existe (como en los ingredientes de una receta). Las listas de definiciones pueden ser útiles para introducir términos al estilo de un diccionario.

Listas no ordenadas

Las listas no ordenadas se generan anteponiendo un asterisco, signo menos o signo más a los elementos que se quiere listar, siempre dejando por lo menos un espacio a cada lado del símbolo:

```
 * frecuencias;
 * promedios;
 * dispersiones;
```

```
* correlaciones; o
* distribuciones.
```

```
- frecuencias;
- promedios;
- dispersiones;
- correlaciones; o
- distribuciones.
```

```
+ frecuencias;
+ promedios;
+ dispersiones;
+ correlaciones; o
+ distribuciones.
```

- frecuencias;
- promedios;
- dispersiones;
- correlaciones; o
- distribuciones.

Listas ordenadas

Las listas ordenadas se generan mediante números seguidos por punto.

```
1. Elegir el test correcto de acuerdo con la naturaleza de las
   variables con las que trabajamos.
2. Establecer la hipótesis nula.
3. Calcular la probabilidad teórica de los resultados observados
   considerando que la hipótesis nula es cierta (valor de _p_).
4. Evaluar la significancia estadística de los resultados.
5. Interpretar el nivel de significancia de los resultados.
```

1. Elegir el test correcto de acuerdo con la naturaleza de las variables con las que trabajamos.
2. Establecer la hipótesis nula.
3. Calcular la probabilidad teórica de los resultados observados considerando que la hipótesis nula es cierta (valor de p).
4. Evaluar la significancia estadística de los resultados.
5. Interpretar el nivel de significancia de los resultados.

Realmente no importa qué números sean siempre y cuando el primer número sea 1 y todos estén seguidos por un punto.

```
1. Este es el primer ítem.
1. Este es el segundo.
100. Este es el tercero.
```

1. Este es el primer ítem.
2. Este es el segundo.
3. Este es el tercero.

Listas de definiciones

Existen distintas formas de incluir listas de definiciones, pero la más sencilla consiste en escribir el nombre del término en una línea y en la línea inmediatamente siguiente introducir las definiciones con una virgulilla e indentación de dos espacios.

```
Hipótesis alternativa
  ~ En la propuesta de Neyman-Pearson, la hipótesis alternativa
  representa una segunda población que acompaña a la población de la
  hipótesis principal en el mismo continuum de valores.

Hipótesis nula
  ~ En la propuesta de Fisher, es la hipótesis que se espera falsar con
  los datos.
  ~ En la prueba de significancia de la hipótesis nula (NHST), es una
  población teórica que se postula para ser comparada con la muestra
  observada y decidir acerca de la hipótesis alternativa.
```

Hipótesis alternativa En la propuesta de Neyman-Pearson, la hipótesis alternativa representa una segunda población que acompaña a la población de la hipótesis principal en el mismo continuum de valores.

Hipótesis nula En la propuesta de Fisher, es la hipótesis que se espera falsar con los datos.

En la prueba de significancia de la hipótesis nula (NHST), es una población teórica que se postula para ser comparada con la muestra observada y decidir acerca de la hipótesis alternativa.

Numeración de ejemplos

Podemos introducir ejemplos numerados mediante `(@)` y la numeración se va a mantener durante todo el documento. También podemos incluir etiquetas para referir a ejemplos puntuales en otras secciones.

```
(@LCxCPC) Me conoció a mí y al ingeniero juntos.
(@) Me conoció a mí ayer y al ingeniero hoy.
(@LCxCUC) *Lo conoció a mí y al ingeniero juntos.
(@) *Lo conoció a mí ayer y al ingeniero hoy.
(@LCxCT) Nos conoció a mí y a al ingeniero juntos.
(@) *Nos conoció a mí ayer y al ingeniero hoy.

Mientras que (@LCxCPC), (@LCxCUC) y (@LCxCT) presentan lectura
colectiva, las otras oraciones citadas presentan lectura distributiva.
```

(1) Me conoció a mí y al ingeniero juntos.
(2) Me conoció a mí ayer y al ingeniero hoy.
(3) *Lo conoció a mí y al ingeniero juntos.
(4) *Lo conoció a mí ayer y al ingeniero hoy.
(5) Nos conoció a mí y a al ingeniero juntos.
(6) *Nos conoció a mí ayer y al ingeniero hoy.

Mientras que (1), (3) y (5) presentan lectura colectiva, las otras oraciones citadas presentan lectura distributiva.

Ítems multilíneas

Podemos generar listas con elementos de varios párrafos usando indentación de 4 espacios o una tabulación.

```
1.    Elegir el test correcto de acuerdo con la naturaleza de las
      variables con las que trabajamos.

      El tipo de test determina la distribución elegida, otras
      características vienen con la muestra.

2.    Establecer la hipótesis nula.

      Puede ser _direccional_ (si se espera un resultado determinado)
      o _no direccional_ (no hay predicciones sobre los datos). La
      hipótesis nula no tiene que ser siempre cero. Puede ser que una
      diferencia no supere un valor específico.
```

1. Elegir el test correcto de acuerdo con la naturaleza de las variables con las que trabajamos.

   El tipo de test determina la distribución elegida, otras características vienen con la muestra.

2. Establecer la hipótesis nula.

   Puede ser *direccional* (si se espera un resultado determinado) o *no direccional* (no hay predicciones sobre los datos). La hipótesis nula no tiene que ser siempre cero. Puede ser que una diferencia no supere un valor específico.

Líneas horizontales

Podemos insertar líneas horizontales poniendo 3 o más asteriscos/guiones en una línea vacía.

```
* * *
```

Figura 1: Un conejo

Hipervínculos e imágenes

Para insertar hipervínculos vamos a usar la sintaxis `[nombre](URL)` y podemos agregar un título (un texto que se muestra cuando posicionamos el cursor sobre el enlace) mediante `[nombre](URL "título")`.

```
Entrá al sitio del Grupo de Estadística para el Estudio del Lenguaje
haciendo [clic acá](https://gesel.github.io/)!

Entrá al sitio del Grupo de Estadística para el Estudio del Lenguaje
haciendo [clic acá](https://gesel.github.io/ "Sitio del GESEL")!
```

> Entrá al sitio del Grupo de Estadística para el Estudio del Lenguaje haciendo clic acá!
>
> Entrá al sitio del Grupo de Estadística para el Estudio del Lenguaje haciendo clic acá!

La sintaxis para las imágenes es muy similar pero tenemos que anteponer un símbolo de cierre de exclamación: `![Texto alternativo](URL "Título")`. El texto alternativo aparece como epígrafe de la foto.

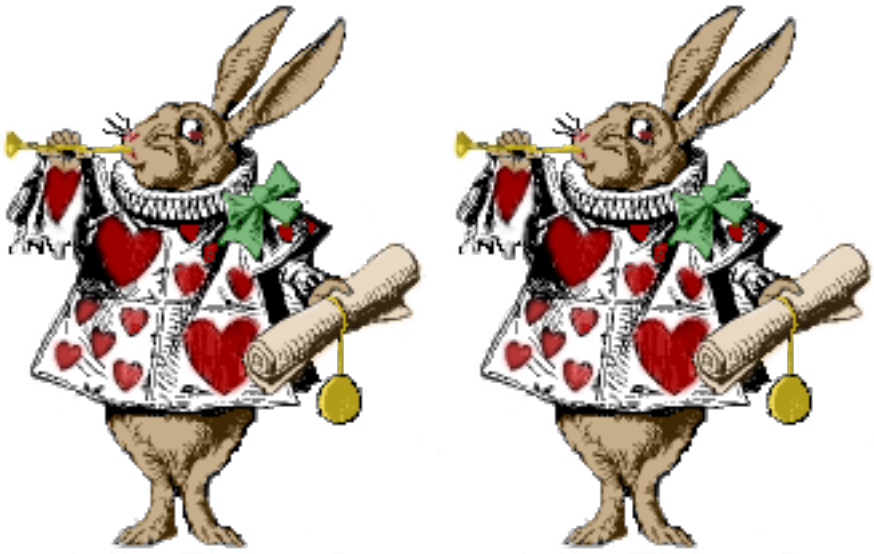
```
![Un conejo](rs/rabbit.png)
```

Referencias

Tanto para los enlaces como para las imágenes podemos usar referencias para las URL y los títulos. La sintaxis es `[nombre][identificador]` para los enlaces y `![texto alternativo][identificador]` para las imágenes. Nótese que se remplazan los paréntesis que encierran la URL por corchetes. Los identificadores se definen usando `[identificador]: URL "Título"` en cualquier parte del código, y no son sensibles al uso de mayúsculas.

```
![Otro conejo][CONEJO]
![Un conejo más][CoNeJo]

[conejo]: rs/rabbit.png
```



Enlaces automáticos

Una forma sencilla de crear hipervínculos es encerrando la dirección entre corchetes angulares. De esta forma, podemos insertar tanto URLs como direcciones de e-mail.

```
Mi sitio web personal: <https://santeh.yoo>

Mi mail: <santeh@santeh.yoo>
```

Mi sitio web personal: https://santeh.yoo

Mi mail: santeh@santeh.yoo

Enlaces a encabezados

R Markdown nos permite asignar IDs a los encabezados, que después podemos usar para, por ejemplo, establecer enlaces entre las secciones. Para hacer esto escribimos el identificador seguido de un hashtag entre llaves al final del encabezado.

```
# Encabezado nivel 1 {#nombre-significativo}

[Texto]

# Otro encabezado nivel 1

Como vimos en la [sección anterior](#nombre-significativo)...

Como vimos en la [sección anterior][Encabezado nivel 1]...
```

Encabezado nivel 1

[Texto]

Otro encabezado nivel 1

Como vimos en la sección anterior...

Como vimos en la sección anterior...

Enlaces cortos

También podemos usar la sintaxis de enlaces por referencia en forma abreviada si la etiqueta de la referencia es igual al texto que queremos enlazar.

```
Toda la información la podés encontrar en [nuestra página].

[nuestra página]: https://gesel.github.io/
```

Toda la información la podés encontrar en nuestra página.

Caracteres de escape

En lo que presenté hasta ahora hay algunos caracteres que tienen un significado especial en R Markdown, por ejemplo, el hashtag en posición inicial define un título y la secuencia espacio-asterisco-espacio inserta un ítem de una lista no numerada. La pregunta que surge es cómo usar esos caracteres con su valor original. Para poder hacer esto tenemos que *escapar* nuestros caracteres especiales usando el carácter de escape que es la barra descendente o *backslash*:.

```
Estos son los caracteres que podemos escapar:
\\ \` \* \_ \{\} \[\] \(\) \# \+ \- \. \!
```

Estos son los caracteres que podemos escapar: \ ‘ * _ {} [] () # + - . !

Código

Existen varias formas de incluir código en R Markdown. La más sencilla consiste en poner 3 acentos graves de apertura y cierre (`). Además podemos definir un identificador y el lenguaje del código.

````
Acá hay una sección de código en Python que imprime en pantalla las segundas
potencias de los números desde el 1 hasta el 9:

```python
for n in range(1, 10):
    print(n**2)
```

Acá está el código equivalente en C:

```{#codigo-en-c .c}
#include <stdio.h>

int main(int argc, const char* argv[])
{
    int n;

    for(n = 1; n < 10; n++) {
        printf("%i\n", n * n);
    }

    return 0;
}
```
````

Acá hay una sección de código en Python que imprime en pantalla las segundas potencias de los números desde el 1 hasta el 9:

```python
for n in range(1, 10):
    print(n**2)
```

Acá está el código equivalente en C:

```c
#include <stdio.h>

int main(int argc, const char* argv[])
{
    int n;

    for(n = 1; n < 10; n++) {
        printf("%i\n", n * n);
```

```
    }

    return 0;
}
```

Tablas

Existen distintas formas de incluir tablas. Una de ellas consiste en "dibujar" una tabla usando guiones, signos de más, signos de igual y *pipes*. Los signos de más marcan las esquinas, los guiones las líneas horizontales, los *pipes* las verticales, y los signos de igual separan, si lo hay, el encabezado de las demás filas. La alineación se define por columna como sigue:

- un símbolo dos puntos a la derecha más en la línea de signos de igual (si la hay) o de la primera línea de guiones, indica alineación a la izquierda;
- un símbolo dos puntos a la izquierda en la línea de signos de igual (si la hay) o de la primera línea de guiones, indica alineación a la derecha;
- un símbolo dos puntos a la izquierda y otro signo dos puntos a la derecha en la línea de signos de igual (si la hay) o de la primera línea de guiones, indican alineación a la derecha; y
- ningún signo dos puntos indica alineación por defecto.

Además podemos añadir una descripción anteponiendo a la tabla un signo dos puntos seguido de un comentario.

```
: Demostración del uso de tablas tipo grilla con encabezado.

+---------------+---------------+--------------------+
| Derecha       | Centro        | Izquierda          |
+==============:+:=============:+:===================+
| fila 1        | celda         | - ítem 1           |
|               |               | - ítem 2           |
+---------------+---------------+--------------------+
| fila 2        | otra celda    | - ítem 3           |
|               |               | - ítem 4           |
+---------------+---------------+--------------------+

: Demostración del uso de tablas tipo grilla sin encabezado.

+--------------:+:-------------:+:-------------------+
| fila 1        | celda         | - ítem 1           |
|               |               | - ítem 2           |
+---------------+---------------+--------------------+
| fila 2        | otra celda    | - ítem 3           |
|               |               | - ítem 4           |
+---------------+---------------+--------------------+
```

Cuadro 1: Demostración del uso de tablas tipo grilla con encabezado.

Derecha	Centro	Izquierda
fila 1	celda	▪ ítem 1 ▪ ítem 2
fila 2	otra celda	▪ ítem 3 ▪ ítem 4

Cuadro 2: Demostración del uso de tablas tipo grilla sin encabezado.

fila 1	celda	▪ ítem 1 ▪ ítem 2
fila 2	otra celda	▪ ítem 3 ▪ ítem 4

Fórmulas matemáticas

Pandoc permite el uso de la sintaxis de TeX para la inserción de fórmulas matemáticas.

```
$P(B- A) = P(B \cap A') = P(B) - P(A \cap B)$
```

$$P(B - A) = P(B \cap A') = P(B) - P(A \cap B)$$

Notas al pie

Si el identificador de una referencia comienza con un circunflejo `^`, definimos una nota al pie.

```
[Texto]

El testeo de hipótesis nos permite decidir si un efecto observado se
debe a relaciones reales entre las variables o al
azar[^aclaracion-testeo].

[^aclaracion-testeo]: Realmente el testeo de hipótesis no siempre nos
    permite *decidir*. Algunos procedimientos solo nos permiten calcular
    la probabilidad de que una observación se dé por azar.

[Texto]
```

[Texto]

El testeo de hipótesis nos permite decidir si un efecto observado se debe a relaciones reales entre las variables o al azar[1].

[Texto]

---

1 Realmente el testeo de hipótesis no siempre nos permite decidir. Algunos procedimientos solo nos permiten calcular la probabilidad de que una observación se dé por azar.

Metadatos

Podemos incluir metadatos en un bloque YAML definido entre dos líneas de tres guiones[2]. En esta sección podemos definir los/las autores/as del documento, el título, el formato de salida. Por ejemplo, podemos incluir el siguiente bloque YAML al inicio de nuestro documento para definir el título, los autores, la fecha, el *abstract* y el formato de salida de nuestro documento.

```
---
title:  'El título'
author:
- Hernán Hernández
- Martín Martínez
date: 16 de abril de 2019
abstract: |
  Este es el _abstract_

  Y tiene dos párrafos.
output_format: pdf_document
---
```

Referencias bibliográficas

Para incluir citas y referencias bibliográficas lo primero que tenemos que hacer es decirle a R Markdown de dónde sacar la información bibliográfica. Eso lo vamos a hacer incluyendo un campo `bibliography` en nuestro encabezado YAML.

```
---
title: "Ejemplo con bibliografía"
bibliography: "bibliografia.bib"
---
```

En el ejemplo, la información bibliográfica se encuentra en un archivo "bibliografia.bib". La extensión *.bib* se corresponde con el formato BibLaTeX.

[2]YAML es un estándar para la serialización de datos. En los comienzos, de su desarrollo, YAML era el acrónimo de *Yet Another Markup Language*, pero para distinguir su propósito centrado en datos se optó por resignificarlo como *YAML Ain't Markup Language*.

```
% Ejemplo de una entrada para un libro en formato BibLaTeX

@book{lenneberg67,
    title     = {Biological Foundations of Language},
    author    = {Eric H. Lenneberg},
    publisher = {John Wiley \& Sons},
    address   = {Nueva York},
    year      = {1967},
}
```

Después de definir nuestro archivo de bibliografía, podemos incluir referencias en el texto y Pandoc se va a encargar de producir la lista bibliográfica.

```
El enfoque biolingüístico estudia la relación entre el lenguaje y
la biología, como propuso @lenneberg67, entendiendo al primero como una
facultad de base biológica y dominio específico, un «órgano» mental
ubicado en la cabeza de los seres humanos y determinado por su dotación
genética.

# Referencias
```

> El enfoque biolingüístico estudia la relación entre el lenguaje y la biología, como propuso Lenneberg (1967), entendiendo al primero como una facultad de base biológica y dominio específico, un «órgano» mental ubicado en la cabeza de los seres humanos y determinado por su dotación genética.

Referencias

Lenneberg, Eric H. 1967. *Biological Foundations of Language*. Nueva York: John Wiley & Sons.

Con arroba indicamos qué entrada queremos referir, pero podemos formatear la referencia de distintas formas para adecuarla al contexto.

```
Autor en texto y año entre paréntesis: @lenneberg67

Autor en texto y, año y localizador entre paréntesis: @lenneberg67
[p. 67]

Autor y año entre paréntesis: [@lenneberg67]

Autor y año entre paréntesis con prefijo: [véase @lenneberg67]

Autor y año entre paréntesis con localizador: [@lenneberg67, p. 67]

Autor y año entre paréntesis con prefijo y localizador:
```

```
[véase @lenneberg67, pp. 67-78]

Autor y año entre paréntesis con prefijo, localizador y sufijo:
[véase @lenneberg67, pp. 67-78, para una discusión al respecto]

Solo año entre paréntesis: [-@lenneberg67]
```

> Autor en texto y año entre paréntesis: Lenneberg (1967)
>
> Autor en texto y, año y localizador entre paréntesis: Lenneberg (1967, 67)
>
> Autor y año entre paréntesis: (Lenneberg 1967)
>
> Autor y año entre paréntesis con prefijo: (véase Lenneberg 1967)
>
> Autor y año entre paréntesis con localizador: (Lenneberg 1967, 67)
>
> Autor y año entre paréntesis con prefijo y localizador: (véase Lenneberg 1967, 67–78)
>
> Autor y año entre paréntesis con prefijo, localizador y sufijo: (véase Lenneberg 1967, 67–78, para una discusión al respecto)
>
> Solo año entre paréntesis: (1967)

Por último, si queremos incluir entradas en la referencia bibliográfica sin tener que incluirlas en el cuerpo del texto, podemos agregar un campo `nocite` al bloque YAML.

```
nocite: @lenneberg67
```

Salida del código de R

R Markdown nos va a permitir incluir fácilmente código de R con su salida. Para esto vamos a usar la sintaxis de código (encerramos el bloque de código entre tres acentos agudos) e indicamos que es un fragmento de código de R como hacemos a continuación. Podemos insertar rápidamente un fragmento de código con la combinación de teclas Ctrl + Alt + I.

````
```{r}
summary(cars)
```
````

```
summary(cars)
```

```
##      speed           dist
##  Min.   : 4.0   Min.   :  2.00
##  1st Qu.:12.0   1st Qu.: 26.00
##  Median :15.0   Median : 36.00
##  Mean   :15.4   Mean   : 42.98
##  3rd Qu.:19.0   3rd Qu.: 56.00
##  Max.   :25.0   Max.   :120.00
```

La sintaxis de llaves le indica al paquete `knitr` que tiene que ejecutar el código a través del intérprete de R y traducir la salida a Markdown para que Pandoc pueda compilar el documento.

Opciones

Además, R Markdown nos permite definir ciertos parámetros acerca de cómo se va a presentar el código en el documento:

`echo` Define si el código se incluye o no en el documento.

````
```{r, echo = FALSE}
summary(cars)
```
````

```
##      speed           dist
##  Min.   : 4.0   Min.   :  2.00
##  1st Qu.:12.0   1st Qu.: 26.00
##  Median :15.0   Median : 36.00
##  Mean   :15.4   Mean   : 42.98
##  3rd Qu.:19.0   3rd Qu.: 56.00
##  Max.   :25.0   Max.   :120.00
```

````
```{r, echo = TRUE}
summary(cars)
```
````

```
summary(cars)
```

```
##      speed           dist
##  Min.   : 4.0   Min.   :  2.00
##  1st Qu.:12.0   1st Qu.: 26.00
##  Median :15.0   Median : 36.00
##  Mean   :15.4   Mean   : 42.98
##  3rd Qu.:19.0   3rd Qu.: 56.00
##  Max.   :25.0   Max.   :120.00
```

`eval` Determina si el código debe ser corrido o no.

````
```{r, eval = FALSE}
summary(cars)
```
````

```
summary(cars)
```

````
```{r, eval = TRUE}
summary(cars)
```
````

```r
summary(cars)
```

```
##      speed           dist
##  Min.   : 4.0   Min.   :  2.00
##  1st Qu.:12.0   1st Qu.: 26.00
##  Median :15.0   Median : 36.00
##  Mean   :15.4   Mean   : 42.98
##  3rd Qu.:19.0   3rd Qu.: 56.00
##  Max.   :25.0   Max.   :120.00
```

include Decide si el código y su salida son incluidos en el código o no.

```
```{r, include = FALSE}
summary(cars)
```
```

```
```{r, include = TRUE}
summary(cars)
```
```

```r
summary(cars)
```

```
##      speed           dist
##  Min.   : 4.0   Min.   :  2.00
##  1st Qu.:12.0   1st Qu.: 26.00
##  Median :15.0   Median : 36.00
##  Mean   :15.4   Mean   : 42.98
##  3rd Qu.:19.0   3rd Qu.: 56.00
##  Max.   :25.0   Max.   :120.00
```

Todas estas opciones pueden combinarse separándolas por coma. Además hay otras opciones en las que no me detuve pero que están disponibles acá.

Cierre

Espero que esta guía haya servido para que se den una idea de como funciona R Markdown, pero cualquier cosa pueden escribirme. Si quieren profundizar en algún aspecto les vuelvo a dejar la documentación de las distintas herramientas:

- documentación de Markdown;
- documentación de Pandoc; y
- documentación de `knitr`.

Además, hay un cheatsheet de R Markdown súper útil y práctico que pueden descargar de acá.