Tutorial de`Quarto`

Generación de reportes

Pablo López Landeros

2022-10-10

Mostramos algunas funciones avanzadas de `ggplot2`: cómo formatear el texto con otros colores usando `gridtext`, cómo agregar en la misma gráfica datos de dos fuentes distintas, cómo juntar gráficas usando `cowplot` y cómo agregar logotipos a gráficas.

Generación de Reportes en Quarto.

Como ya vimos, Quarto provee una interfaz para generar reportes mediante el uso de texto y código. Consiste de tres tipos de archivo:

- Texto (Markdown)
- Chunks o pedazos de código (R, Python o SQL)
- YAML metadata (opciones de generación)

Como funciona?

Crash Course de Markdown

( )/.

Markdown es una forma de escribir contenido mediante **plain-text**. A diferencia de los procesadores tradicionales de texto, los archivos en Markdown tienen la ventaja de poder ser compartidos e interpretados fácilmente entre distintas sistemas operativos como macOS, Windows, Linux, iOS e incluso Android.

La mejor guía-resumen de Markdown se encuentra aquí.

La documentación completa para R Markdown (ahora Quarto). Se encuentra aquí.

- Títulos y subtítulos: Deberás ocupar el símbolo *hash* #, dependiendo del número de *hashes* que pongas el header se irá haciendo más pequeño. Por ejemplo: `### Algo aquí`

Un hash

Dos Hashes

Tres Hashes

Cuatro Hashes

- Palabras en **negritas** e *itálicas*.
  - Para poner palabras en negritas, tienes que rodear la(s) palabra(s) de 2 astericos. Por ejemplo: `**palabras en negritas**`.
  - Para poner palabras en itálicas, tienes que rodear la(s) plabra(s) de 1 asterisco. Por ejemplo: `*palabras en itálicas*`.

- Palabras ~~rayadas~~: Si quieres rayas una palabra, tienes que rodearla(s) de tildes. Por ejemplo `~~palabras rayadas~~`.
- Insertar imágenes: Para insertar una imágen en el documento necesitas ocupar la combinación `![tag](path/donde/está/la/imagen)`, el tag es cualquier nombre que le quieras poner a tu imagen y que después puedas referirte a ella más adelante en tu texto, el *path* en tu computadora donde se encuentra la imagen. Por ejemplo: `![](./docs/images/pointer.png)`.
- Insertar ligas: Para agregar una liga necesitas ocupar la combinación `[palabra que aparece con el link](URL/de/la/pagina)`. Por ejemplo: `[Git tutorial](https://try.github.io/levels/1/challenges/1)`
- Generar listas: Para enlistar puedes ocupar asteriscos o el signo de más. Por ejemplo:

```
+ lista de
+ cosas
+ en markdown
```

que se vería así:

  - lista de
  - cosas
  - en markdown

- *Quote*: Puedes ocupar una identación especial para indicar un *quote* o una nota importante en tu documento al utilizar el mayor que >. Por ejemplo:

```
> Algo importante aquí
```

Generaría esto:

> Algo importante aquí

- Tablas: Puedes generar tablas utilizando el *pipe* |. Puedes definir la alineación de cada columna con los dos puntos :, para justificar a la izquierda tendrás que poner los dos puntos al inicio seguido de guiones -el número de guiones

indica qué tan ancha debe ser la columna, para justificar a la derecha tendrás que poner los puntos al final de los guiones, para dejar centrado tendrás que ocupar dos puntos al inicio y al final. Por ejemplo:

```
|Clase|Fecha|Tema|
|:------:|:---------:|:--------------------------------|
|1|20 enero 2020|Introducción, Markdown, Git|
|2|22 enero 2020|Git, branching|
|3|27 enero 2020|Python|
|4|29 enero 2020|Python|
|5|3 febrero 2020|ASUETO|
|6|5 febrero 2020|Python|
|7|10 febrero 2020|Python, Pandas|
```

Que se verá así:

Clases

Clase	Fecha	Tema
1	20 enero 2020	Introducción, Markdown, Git
2	22 enero 2020	Git, branching
3	27 enero 2020	Python
4	29 enero 2020	Python
5	3 febrero 2020	ASUETO
6	5 febrero 2020	Python
7	10 febrero 2020	Python, Pandas

Como combinar código + Markdown?

Ya vimos la parte de la escritura. Ahora veamos como podemos combinar lo aprendido en la sección pasada con lo que vimos la semana pasada (`R`).

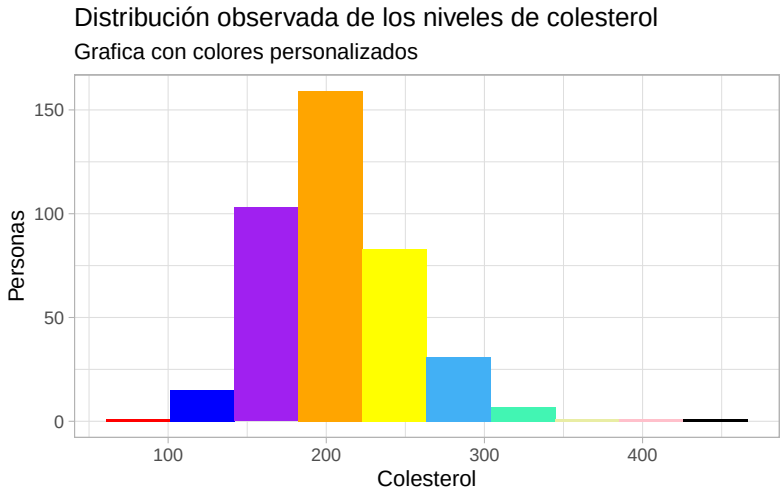
Para poder insertar (y correr) códigos de R dentro de nuestro documento de Quarto, necesitamos encerrar nuestras líneas de código dentro de dos pares de “‘.

```r
library(ggplot)
a <- 10
b <- 5

a+b
```

Entre los corchetes podemos poner múltiples comandos para modificar el comportamiento de nuestro código. Algunos de los más comunes serían:

- `include = FALSE` evita que el código y los resultados aparezcan en el archivo final aunque el código **sí se ejecuta** y los resultados pueden ser usados en otros pedazos de código.
- `echo = FALSE` **no imprime el código pero sí los resultados** en nuestro documento final. Se usa mucho para generar gráficas.

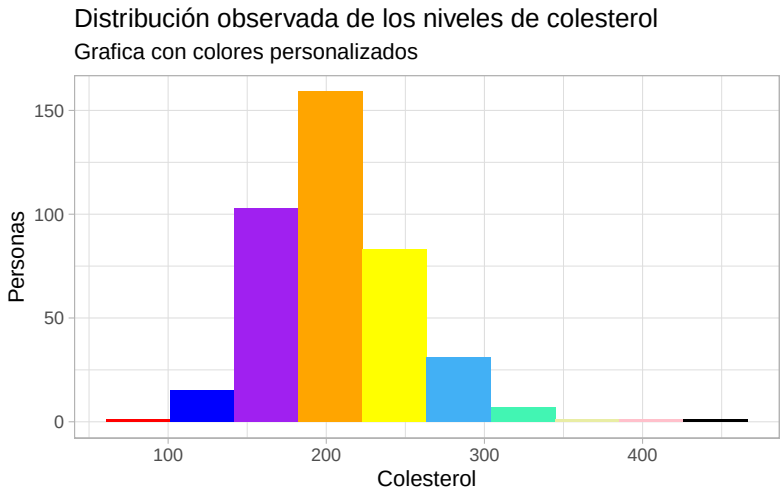


- `message = FALSE` evita que mensajes generados por código aparezcan en el documento final.
- `warning = FALSE` evita que las advertencias o "warnings" generados por código aparezcan en el documento final.

```r
colores <- c('red',
             'blue',
             'purple',
             'orange',
```

```
              'yellow',
              '#42b0f5',
              '#42f5b3',
              '#e9eda4',
              'pink',
              'black')

ggplot(diabetes)+
  geom_histogram(aes(x=chol),bins = 10, fill = colores)+
  labs(
    x='Colesterol',
    y='Personas',
    title = "Distribución observada de los niveles de colesterol",
    subtitle = 'Grafica con colores personalizados'
  )+
  scale_fill_manual()+
  theme_light()
```



- `fig.cap = "..."` agrega una nota al pie del código.

```
diabetes <- read.csv('data/diabetes.csv')

colores <- c('red',
             'blue',
```

```
              'purple',
              'orange',
              'yellow',
              '#42b0f5',
              '#42f5b3',
              '#e9eda4',
              'pink',
              'black')

ggplot(diabetes)+
  geom_histogram(aes(x=chol),bins = 10, fill = colores)+
  labs(
    x='Colesterol',
    y='Personas',
    title = "Distribución observada de los niveles de colesterol",
    subtitle = 'Grafica con colores personalizados'
  )+
  scale_fill_manual()+
  theme_light()
```

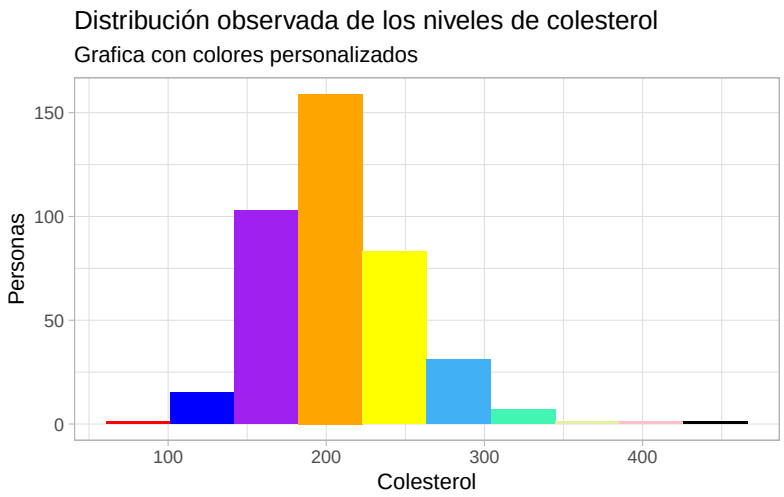


Figure 1: Una bonita gráfica generada con ggplot :D