Notas clase 4: visualización de datos

Visualización de datos

From data to Viz | Find the graphic you need

Colores

función **colors() en Rstudio**

```
# Librería ColorBrewer en R
library(RColorBrewer)
display.brewer.all(type = "qual")
display.brewer.all(type = "seq")
```

Colors in R
ColorBrewer: Color Advice for Maps
VIZ PALETTE
Datawrapper Blog Which color scale to use when visualizing data
ggplot2 Quick Reference: colour (and fill)

📓 Bibliografía

Para la elaboración tanto del material teórico como de los ejercicios prácticos se utilizaron como referencia, los libros descriptos a continuación:

- R para Ciencia de Datos de Hadley Wickham - Capítulo 3: Visualización de datos y Capítulo 28: Comunicar con gráficos.

- R Graphics Cookbook: Practical Recipes for Visualizing Data de Winston Chang - Capítulo 1: Bar and line graphs.
- A Data Visualization: a practical introduction de Kieran Healy.
- Fundamentals of Data Visualization de Claus Wilke.

Tutoriales.

- A ggplot2 Tutorial for Beautiful Plotting in R por Cédric Scherer
- The Grammar and Graphics of Data Science por Rstudio.
- Tutorial interactivo de gráficos con el paquete learnr por Yanina Bellini Saibene

Presentaciones en español.

- Taller Introductorio a ggplot2 por Gaby Sandoval para R-Ladies Santiago.
- Visualización de datos espaciales por Steph Orellana Bello para R-Ladies Santiago.

Presentaciones en inglés.

- Take A Sad Plot & Make It Better por Alison Presmanes Hill
- Plot Twist: 10 Bake Offs, Visualized 11 Ways por Alison Presmanes Hill
- Data Visualization in the Tidyverse por Alison Presmanes Hill
- Data Visualisation in R por Danielle Navarro
- Robust-tools: data visualization por Danielle Navarro

Dudas planteadas en clase:

¿Qué función cumple el argumento *mapping* en la función ggplot()?

El argumento `mapping` nos permite definir los argumentos estéticos (eje x/y, color, tamaño, relleno, etc) que variarán en función de una de las variables de nuestros datos en el gráfico. Para definir estos valores debemos usar la función `aes()`. Aquellos argumentos que estén definidos fuera de mapping serán tomados como constantes y no podrán depender de una de las variables:

```
library(tidyverse)
ggplot(mtcars, mapping = aes(x = cyl, y = hp, color = cyl) ) +
    geom_point(alpha = 0.6)
```

¿Por qué a veces no se colocan las palabras “data” y “mapping” para definir los argumentos de gglot?

Esto sucede porque, si no definimos el nombre del parámetro de la función, R toma estos en el orden en que están definidos en la función. En el caso de la función ggplot() el primer parámetro es data y el segundo es mapping, por lo cual estas dos formas de escribir son idénticas:

```
p <- ggplot(data = mtcars, mapping = aes(x = cyl)) +
          geom_histogram()
p <- ggplot(mtcars, aes(x = cyl)) +
          geom_histogram()
```

En el caso de las funciones geom_* este orden se invierte:

```
p <- ggplot() +
          geom_histogram(mapping = aes(x = cyl),
                         data = mtcars)
p <- ggplot() +
          geom_histogram(aes(x = cyl),
                         mtcars)
```

De todas formas es recomendable usar con cuidado esta característica de R y sólo hacerlo con el primero (y a lo sumo el segundo) parámetro de una función. Esto es importante porque nuevas versiones de estas funciones pueden hacer que nuestros códigos den errores inesperados.

¿Cómo hacer para ordenar de modo diferente las categorías de una variable?

Una posibilidad es, sobre la base construida en clase, utilizar el siguiente código:

```
tabla_provincia %>%
  ggplot(aes(x = fct_reorder(provincia, -n),
             y = n,
             fill = provincia)) +
  geom_col() +
  coord_flip()
```

¿Cómo se guardan los gráficos?

Si estamos utilizando ggplot una manera de guardar los gráficos es agregar una nueva “capa” con `ggsave()`:

```
library(tidyverse)
ggplot(mtcars, mapping = aes(x = cyl) ) +
```

```
    geom_histogram() +
    ggsave("grafico.png")
```

La función `ggsave()` tiene varios argumentos para definir el tamaño y la calidad del gráfico. Es posible guardar el gráfico con extensión png, jpeg o pdf.

Ej: ggsave("grafico_mtcars.png", width = 30, height = 20, units = "cm")

¿Cómo puedo dibujar una pirámide de población?

Hay varias maneras de hacer esto. Veamos dos ejemplos usando nuestra base de COVID (no es la mejor base para ver pirámides, pero nos sirve de ejemplo).
Una posibilidad es usando el paquete `epiDisplay`:

```
# install.packages("epiDisplay")
library(epiDisplay)

# Requiere que la variable sexo sea dicotómica
base_covid2 <- base_covid %>%
    filter(sexo != "NR" )

pyramid(base_covid2$edad,
                    base_covid2$sexo,
                    binwidth = 10,
                    decimal = 2,
                    col.gender = c("pink", "lightblue"))
```

Una versión con `ggplot` (quizás poco elegante):

```
library(tidyverse)

base_covid %>%
  filter(sexo != "NR" ) %>%
  select(sexo, edad) %>%
  # Primero transformamos nuestra variable edad en
  #    grupos de 5 años y recodificamos sexo para que
  #    no haya una columna con un nombre no sintáctico (F)
  mutate(EDADQUI = cut(edad,
                       include.lowest = T, right  = F,
                       breaks = c(seq(0, 80, 5), 130),
                       labels = c(paste0(seq(0, 75, 5), "-",
                                         seq(5, 80, 5) ),
                                  "+80") ),
```

```
        sexo = ifelse(sexo == "M", "V", "M" ) ) %>%
  # Contamos por grupo de edad y sexo
  count(EDADQUI, sexo) %>%
  # Hacemos un pequeño truco para que los datos
  #    de varones vayan hacia el eje izquierdo
  mutate(n = ifelse(sexo == "V", -1*n , n)) %>%
  pivot_wider(id_cols = EDADQUI,
              names_from = sexo, values_from = n) %>%
  # Gráfico (por fin!)
  ggplot(mapping = aes(x = EDADQUI)) +
  geom_bar(aes(y = V),
           stat = "identity",
           fill = "blue",
           alpha = 0.4) +
  geom_bar(aes(y = M),
           stat = "identity",
           fill = "red",
           alpha = 0.4) +
  coord_flip() +
  labs(x = NULL,
       y = NULL,
       title = "Pirámide poblacional") +
  theme_minimal() +
  scale_y_continuous(labels =  function(br)
paste0(abs(br)/1000, "k"))
```

Links de interés sobre R

Sobre la nueva pipa nativa

New features in R 4.1.0
La nueva pipa
R devel - New pipe operator
Debate en Stackoverflow: What are the differences between R's new native pipe `|>` and the magrittr pipe `%>%`?
The new R pipe

Páginas para estar informados sobre novedades en R

Home | RWeekly.org - Blogs to Learn R from the Community
The R Journal
Blog de Florencia D'Andre
Consejos para visualizar datos sobre género

Misceláneas

¿Cómo usar tidyverse::count con pesos (ponderación)?

La función count() tiene un argumento para utilizar pesos o factores de expansión (*weight*) cuando nuestras bases lo necesitan:

```
# Cuenta de casos (sin ponderar)
base_covid %>%
     rename(provincia = residencia_provincia_nombre) %>%
     count(sexo, edad)

# Cuenta ponderada (por una variable ficticia llamada
#    "pond" creada para el ejemplo con números aleatorios)
base_covid %>%
     mutate(pond = rnorm(nrow(.))) %>%
     rename(provincia = residencia_provincia_nombre) %>%
     count(sexo, edad, wt = pond)
```

¿Cómo seleccionar las filas de una variable para que me retenga los elementos entre dos números?

Una forma de hacer esto es definiendo dos condiciones (mayor que un número, menor que otro)

```
data.frame(a = rnorm(100)) %>% filter(a >= -2 & a <= 2))
```

Otra manera es usando la función between:

```
data.frame(a = rnorm(100)) %>% filter(between(a, -2, 2 ))
```

¿Cómo saber el tipo de un objeto y en particular de un gráfico? ¿Cuál es la diferencia con class?

A diferencia de la clase (class) de un objeto, el tipo (typeof) refiere a cómo es entendido internamente por R. Para saber el tipo de un objeto podemos usar:

```
typeof(objeto)
```

Esta función es la que podemos usar para saber cuándo tenemos una lista:

```
typeof(list())
class(list())
```

Si bien en principio la clase y el tipo pueden ser similares, esto puede traer algunas dificultades en algunas ocasiones:

```
typeof(ggplot())
class(ggplot())

# Esto nos dará que la clase es "gg" y "ggplot", pero que es
de tipo "list"!
```

También pueden verificar si un objeto es una lista con la familia de funciones `is.*`:

```
is.list(list())
is.list(ggplot())
```