7. Visualización de Datos

Paula Vargas Pellicer

24/02/2022

Visualización de datos usando `ggplot` (las gg de ggplot significan: "gramática" y "gráficas")

Hasta ahora hemos visto cómo importar datos y manipularlos, ahora veremos cómo comunicar los resultados de nuestros análisis. Vamos a aprender a hacer graficas lindas e informativas.

Primero repasemos que es lo que tienen en común las graficas buenas:

1. Presentan un grafico apropiado para el tipo de resultado (barras, lineal, histograma, etc.)

2. Están bien organizadas (la variable independiente está en el eje x y la dependiente en el eje de y)

3. Las unidades son las correctas

4. Los textos son fáciles de leer y son claros

5. Usan colores fáciles de distinguir y consistentes

6. Exponen, cuando son apropiadas, medidas de incertidumbre (barras de error, rangos de confidencia, etc.)

Escribir el código de tu gráfica es como construir una oración hecha de diferentes partes que se prosiguen de manera lógica. De una manera más visual, imagínate que agregas capas que se encargan de diferentes elementos de la grafica. Por lo tanto, el flujo de trabajo será algo así: 1. crear un gráfico vacío, 2. agregar una capa con sus puntos de datos, 3. luego su medida de incertidumbre, 3. las etiquetas de los ejes, etc.

Decidir el tipo de grafica adecuada

Una parte muy importante de hacer cualquier visualización de datos es asegurarse de que sea apropiada para tu tipo de datos (por ejemplo, discretos o continuos) y que se ajuste a tu propósito (¿Qué es lo que estás tratando de comunicar?)

Puedes visitar La galeria de R para darte inspiración.

Hacer diferentes gráficos

```
# Escribe tu ruta de archivo
setwd("tu-ruta")

# Paquetes
```

```r
library(tidyr)
library(dplyr)

library(ggplot2)
library(readr)
library(gridExtra)
```

Usaremos datos del Living Planet Index, que puedes descargar del repositorio de Github.

```r
# Importa los datos de Living Planet Index - tendencias poblacionales de
vertebrados entre 1970 y 2014


LPI <- read.csv("LPIdata_CC.csv")
```

Los datos están en formato ancho, esto quiere decir que los diferentes años son nombres de columnas, cuando en realidad deberían ser filas en la misma columna. Remodelaremos los datos usando la función `gather()` del paquete `tidyr`.

```r
LPI2 <- gather(LPI, "year", "abundance", 9:53)

# 9:53, significa que estamos seleccionando las columnas 9 a 53 (checa en la
base de datos que sean las que corresponden a los años)


View(LPI2)
```

Hay una 'X' delante de todos los años porque cuando importamos los datos, todos los nombres de las columnas deben estar en caracteres. (La X es la forma que tiene R de convertir números en caracteres). Ahora que los años son filas, no columnas, necesitamos que sean números de verdad (sin la X), así que los transformaremos usando `parse_number()` del paquete `readr`.

```r
LPI2$year <- parse_number(LPI2$year)

str(LPI2)

# La abundancia también es caracter, debería de ser numérica
LPI2$abundance <- as.numeric(LPI2$abundance)
```

Este es un conjunto de datos muy grande, por lo que, para los primeros pasos, nos centraremos en cómo ha cambiado la población de una sola especie.


Elije una especie y asegúrate de escribir el nombre exactamente como se ingresó en la base de datos. Yo voy a usar al "buitre leonado" (Griffon vulture), pero puedes usar cualquiera. Para ver qué especies están disponibles, usa el siguiente código para obtener una lista:

```r
unique(LPI2$Common.Name)
```

Ahora, filtra solo los registros de esa especie (sustituye `Common.Name` por la de tu elección)

```
vulture <- filter(LPI2, Common.Name == "Griffon vulture / Eurasian griffon")

head(vulture)

# Hay un montón de NAs en mis datos, eliminemos las filas vacías (no informat
ivas) usando na.omit()
vulture <- na.omit(vulture)
```

Histogramas para visualizar la distribución de datos

Haremos una comparación rápida entre los gráficos de base R y `ggplot2`; por supuesto, ambos pueden generar buenos gráficos cuando se usan bien, pero a mi me gusta trabajar con ggplot2 debido a sus poderosas capacidades de personalización.

```
# Con Rbasic:
base_hist <- hist(vulture$abundance)
```

Para hacer lo mismo con `ggplot`, necesitamos especificar el tipo de gráfico usando `geom_histogram()`.

```
# Creas el gráfico
vulture_hist <- ggplot(vulture, aes(x = abundance))  +
  geom_histogram()

# Lo llamas
vulture_hist
```

- Otra forma de verificar si los datos se distribuyen de manera (estadísticamente hablando) normal, puedes crear diagramas de densidad usando el paquete `ggpubr` y el comando `ggdensity()`, O usar las funciones `qqnorm()` y `qqline()`

La configuración predeterminada de `ggplot` no es ideal: hay mucho espacio gris innecesario detrás del histograma, las etiquetas de los ejes son muy pequeñas y las barras se mezclan entre sí. Veamos qué se puede hacer para mejorar

```
vulture_hist <- ggplot(vulture, aes(x = abundance)) +
  geom_histogram(binwidth = 250, colour = "#8B5A00", fill = "#CD8500") +
  # cambiar el ancho de cada "caja" y colores

  geom_vline(aes(xintercept = mean(abundance)),
  # agregar una línea para la abundancia media

  colour = "red", linetype = "dashed", size=1) +
  # cambiar la apariencia de la línea

    theme_bw() +
```

```
  # cambiar el tema (el fondo)

  ylab("Conteo\n") +
  # cambiar el texto del eje x. '\n' crea un espacio entre el texto y el eje

  xlab("\nAbundancia de Buitre leonado")  +
  # cambiar el texto del eje y. '\n' crea un espacio entre el texto y el eje

  theme(axis.text = element_text(size = 12),
  # cambiar el tamaño de la fuente

  axis.title.x = element_text(size = 14, face = "bold"),
  # cambiar el tipo de letra del eje

  panel.grid = element_blank(),
  # quitar las líneas grises

  plot.margin = unit(c(1,1,1,1), units = , "cm"))

  # poner 1cm de margen alrededor de la grafica
```

Entendiendo el dialecto de `ggplot2`

Quizás lo más complicado al comenzar con `ggplot2` es comprender qué tipo de elementos son responsables del contenido (datos) versus el contenedor (aspecto general).

- **geom**: es un objeto geométrico que define el tipo de gráfico que estás haciendo. Este comando va a leer tus datos en el mapeo estético para saber qué variables usar y crea el gráfico en consecuencia. Algunos tipos comunes son `geom_point()`, `geom_boxplot()`, `geom_histogram()`, `geom_col()`, etc.

- **aes**: abreviatura de estética. Generalmente ubicado dentro de un `geom_`, aquí es donde especificas tu fuente de datos y variables, Y las propiedades del gráfico que dependen de esas variables. Por ejemplo, si deseas que todos los puntos de datos sean del mismo color, debes definir el argumento `colour =` fuera de la función `aes()`; si quieres que los puntos de datos estén coloreados por los niveles de un factor (por ejemplo, por sitio o especie), especifica el argumento `colour =` dentro de `aes()`.

- **stat**: una capa `stat` aplica alguna transformación estadística a los datos subyacentes: por ejemplo, `stat_smooth(method = 'lm')` muestra una línea de regresión lineal y una cinta de intervalo de confianza encima de un diagrama de dispersión (definido con `geom_point()`).

- **theme**: un tema está compuesto por un conjunto de parámetros visuales que controlan el fondo, los bordes, las líneas de cuadrícula, los ejes, el tamaño del texto, la posición de la leyenda, etc. Puedes usar temas predefinidos, crear los tuyos o usar un tema y sobrescribir solo los elementos que no te gustan. Ejemplos de elementos dentro de los temas son `axis.text`, `panel.grid`, `legend.title`, etc. Tu defines las

propiedades con las funciones `elements_...()`: `element_blank()` devolvería algo vacío (ideal para eliminar el color de fondo), mientras que `element_text(size = ..., face = ..., angle = ...)` permite controlar todo tipo de propiedades de texto.

También es útil recordar que las capas se agregan una encima de la otra a medida que avanza en el código, lo que significa que los elementos escritos más tarde pueden sobrescribir elementos anteriores.

Diagrama de dispersión para examinar el cambio de población a lo largo del tiempo

Digamos que estamos interesados en cómo han cambiado las poblaciones de buitres entre 1970 y 2017 en Croacia e Italia.

```
# Filtrar la base de datos con `filter()`
vultureITCR <- filter(vulture, Country.list %in% c("Croatia", "Italy"))

# usando Rbase
plot(vultureITCR$year, vultureITCR$abundance, col = c("#1874CD", "#68228B"))

# Usando ggplot2
(vulture_scatter <- ggplot(vultureITCR, aes(x = year, y = abundance, colour =
Country.list)) +  # si ponermos el color dentro de aes() asegura que los punt
o see colorean de acuerdo con los niveles de ese factor
    geom_point())
```

Editemos la gráfica anterior:

```
vulture_scatter <- ggplot(vultureITCR, aes (x = year, y = abundance, colour =
Country.list)) +
    geom_point(size = 2) +
  # cambiar el tamaño del punto

    geom_smooth(method = "lm", aes(fill = Country.list)) +
  # agregar un modelo lineal, colorear por país

    theme_bw() +
    scale_fill_manual(values = c("#EE7600", "#00868B")) +
  # seleccionar colores para los "listones"

    scale_colour_manual(values = c("#EE7600", "#00868B"),
  # agregar colores para las líneas y puntos

  labels = c("Croacia", "Italia")) +
  # Agregar etiquetas en la leyenda

    ylab("Abundancia Buitre leonado\n") +
    xlab("\nAño")  +
```

```
    theme(axis.text.x = element_text(size = 12, angle = 45, vjust = 1, hjust
= 1),
  # Año en un ángulo

          axis.text.y = element_text(size = 12),
          axis.title = element_text(size = 14, face = "plain"),
          panel.grid = element_blank(),
  # quitar las líneas del fondo

          plot.margin = unit(c(1,1,1,1), units = , "cm"),
  # margen 1cm

          legend.text = element_text(size = 12, face = "italic"),
  # la fuente de la leyenda

          legend.title = element_blank(),
  # leyenda sin titulo

          legend.position = c(0.9, 0.9))

# posición de la leyenda - 0 es izquierda/abajo, 1 es arriba/derecha
vulture_scatter
```

Diagrama de cajas para examinar si la abundancia de buitres difiere entre Croacia e Italia

Los diagramas de caja son muy informativos, ya que muestran la mediana y la dispersión de sus datos, y permiten comparar valores rápidamente entre grupos. Si las casillas no se superponen entre sí, probablemente hay diferencias significativas; sin embargo, siempre vale la pena investigar más a fondo con pruebas estadísticas.

```
vulture_boxplot <- ggplot(vultureITCR, aes(Country.list, abundance)) + geom_b
oxplot()

vulture_boxplot

# Embellecerla

(vulture_boxplot <- ggplot(vultureITCR, aes(Country.list, abundance)) +
    geom_boxplot(aes(fill = Country.list)) +
    theme_bw() +
    scale_fill_manual(values = c("#EE7600", "#00868B")) +
    scale_colour_manual(values = c("#EE7600", "#00868B")) +
    ylab("Abundancia Buitre leonado\n") +
    xlab("\nPais")  +
    theme(axis.text = element_text(size = 12),
          axis.title = element_text(size = 14, face = "plain"),
```

```
          panel.grid = element_blank(),
          plot.margin = unit(c(1,1,1,1), units = , "cm"),
          legend.position = "none"))
```

¿Te queda claro que hace cada línea aquí?

Gráfica de barras para comparar la riqueza de especies de algunos países europeos

Ahora vamos a calcular cuántas especies se encuentran en el conjunto de datos LPI para algunos países europeos y trazar la riqueza de especies.

```
# Calcular riqueza de especies (esto ya lo hemos hecho)
richness <- LPI2 %>% filter (Country.list %in% c("United Kingdom", "Germany",
"France", "Netherlands", "Italy")) %>%
            group_by(Country.list) %>%
            mutate(richness = (length(unique(Common.Name)))) # aquí creamos
una nueva columna basada en cuantas especies únicas hay en cada país

# Graficar
(richness_barplot <- ggplot(richness, aes(x = Country.list, y = richness)) +
    geom_bar(position = position_dodge(), stat = "identity", colour = "black"
, fill = "#00868B") +
    theme_bw() +
    ylab("Riqueza de especies\n") +
    xlab("Pais")  +
    theme(axis.text.x = element_text(size = 12, angle = 45, vjust = 1, hjust
= 1),
          axis.text.y = element_text(size = 12),
          axis.title = element_text(size = 14, face = "plain"),
          panel.grid = element_blank(),
          plot.margin = unit(c(1,1,1,1), units = , "cm")))
```

Usar facetas y crear paneles

A veces, mostrar todos los datos en un gráfico lo hace demasiado desordenado. Si quisiéramos examinar el cambio de población de buitres en todos los países, en lugar de Italia y Croacia, tendríamos 10 poblaciones en el mismo gráfico:

```
(vulture_scatter_all <- ggplot(vulture, aes (x = year, y = abundance, colour
= Country.list)) +
      geom_point(size = 2) +
      geom_smooth(method = "lm", aes(fill = Country.list)) +
      theme_bw() +
      ylab("Abundancia Buitre leonado\n") +
      xlab("\nAño")  +
```

```
      theme(axis.text.x = element_text(size = 12, angle = 45, vjust = 1, hjus
t = 1),
            axis.text.y = element_text(size = 12),
            axis.title = element_text(size = 14, face = "plain"),
            panel.grid = element_blank(),
            plot.margin = unit(c(1,1,1,1), units = , "cm"),
            legend.text = element_text(size = 12, face = "italic"),
            legend.title = element_blank(),
            legend.position = "right"))
```

¡No se entiende nada en esta gráfica!

Al agregar una capa de facetas, podemos dividir los datos en múltiples paneles que representan los diferentes países. Esto se hace usando `facet_wrap()`.

```
(vulture_scatter_facets <- ggplot(vulture, aes (x = year, y = abundance, colo
ur = Country.list)) +
      geom_point(size = 2) +
      geom_smooth(method = "lm", aes(fill = Country.list)) +
      facet_wrap(~ Country.list, scales = "free_y") +
   # Esto crea los paneles
      theme_bw() +
      ylab("Abundancia Buitre leonado\n") +
      xlab("\nAño")  +
      theme(axis.text.x = element_text(size = 12, angle = 45, vjust = 1, hjus
t = 1),
            axis.text.y = element_text(size = 12),
            axis.title = element_text(size = 14, face = "plain"),
            panel.grid = element_blank(),
            plot.margin = unit(c(1,1,1,1), units = , "cm"),
            legend.text = element_text(size = 12, face = "italic"),
            legend.title = element_blank(),
            legend.position = "right"))
```

** NOTA importante: Como podrás ver, algunos países no siguen una sola línea recta (por ejemplo, Italia), esto quiere decir que en la realidad tendríamos que ajustar varios modelos (tal vez tres para las tres poblaciones que aparecen), pero eso lo dejamos para una clase más adelante.

Algunos argumentos útiles para incluir en `facet_wrap()` son `nrow =` o `ncol =`, que especifican el número de filas o columnas, respectivamente. También podrás ver que usamos `scales = "free_y"`, para permitir diferentes valores del eje y debido a la amplia gama de valores de abundancia en los datos. Puedes usar `"fixed"` cuando quieras restringir todos los valores de los ejes.

A veces, tal vez quieras colocar varias graficas en el mismo panel (sobretodo cuando en las revistas te restringen el numero de imágenes que puedes publicar), entonces puedes hacer lo siguiente:

```r
grid.arrange(vulture_hist, vulture_scatter, vulture_boxplot, ncol = 1)

# Y se puede mejorar:
(panel <- grid.arrange(
      vulture_hist + ggtitle("(a)") + ylab("Conteo") + xlab("Abundancia") +
      theme(plot.margin = unit(c(0.2, 0.2, 0.2, 0.2), units = , "cm")),

      vulture_boxplot + ggtitle("(b)") + ylab("Abundancia") + xlab("Pais") +
      theme(plot.margin = unit(c(0.2, 0.2, 0.2, 0.2), units = , "cm")),

      vulture_scatter + ggtitle("(c)") + ylab("Abundancia") + xlab("Año") +
      theme(plot.margin = unit(c(0.2, 0.2, 0.2, 0.2), units = , "cm")) +
      theme(legend.text = element_text(size = 12, face = "italic"),
            legend.title = element_blank(),
            legend.position = c(0.85, 0.85)),
      ncol = 1))
```

Si quieres cambiar el ancho o el alto de cualquiera de tus imágenes, puedes agregar `widths = c(1, 1, 1)` o `heights = c(2, 1, 1)`, por ejemplo, al final del Comando de organización de cuadrícula. En estos ejemplos, esto crearía tres graficas de igual ancho, y la primera parcela sería el doble de alta que las otras dos. Esto es útil cuando tienes figuras de diferentes tamaños o si quieres resaltar la figura más importante en tu panel.

Ejercicio

Vuelve al conjunto de datos LPI original y

1 - Elije DOS especies de los datos de LPI y muestra sus tendencias de población a lo largo del tiempo, utilizando un diagrama de dispersión (pista: debe de ser una gráfica de puntos) y un ajuste de modelo lineal.

2 - Usando las mismas dos especies, filtra los datos para incluir solo registros de CINCO países de tu elección y haz un diagrama de cajas para comparar cómo varía la abundancia de esas dos especies entre los cinco países.