8. Visualización de datos

Paula Vargas Pellicer

01/03/2022

¡Más gráficas!

Hoy usaremos datos inventados que consisten en la abundancia y la altura de diferentes especies de plantas que se encuentran en dos tierras mágicas: Hogsmeade y Narnia. Los datos se pueden descargar desde el repositorio de la clase. Crea un nuevo Script, asígnale un título y la información pertinente del ejercicio.

```
# Carga los paquetes ----
library(dplyr)  # manipulación

library(ggplot2)  # visualización


setwd("ruta-de-archivo")

# lee la base de datos
magic_veg <- read.csv("magic_veg.csv")

# Explora la base de datos con el comando str()
```

Histogramas

Vamos a calcular cuántas especies hay en cada parcela

```
species_counts <- magic_veg %>%
  group_by(land, plot) %>%
  summarise(Species_number = length(unique(species)))
```

Hagamos un histograma de esto

```
hist <- ggplot(species_counts, aes(x = plot)) +
  geom_histogram()

hist
```

Ok, aquí hay algo raro. Esta es la forma común de hacer un histograma, cuando tienes una observación por fila y el histograma las cuenta por ti. Pero en este caso estamos trabajando con datos resumidos; por lo tanto, debemos decirle a R que ya sabemos cuántas especies hay en cada parcela. Lo haces especificando el argumento `stat`:

```
hist <- ggplot(species_counts, aes(x = plot, y = Species_number)) +
    geom_histogram(stat = "identity")
```

```
hist
```

Todavía parece tener demasiadas especies. Eso es porque las parcelas de cada tierra se agrupan. Podemos separarlos introduciendo un código de color y hacer un gráfico de barras apiladas como este:

```
hist <- ggplot(species_counts, aes(x = plot, y = Species_number, fill = land)
) +
  geom_histogram(stat = "identity")

hist
```

Y si quisiéramos que las columnas aparezcan una al lado de la otra en lugar de estar apiladas, agrega `position = "dodge"` a los argumentos de geom.

```
hist <- ggplot(species_counts, aes(x = plot, y = Species_number, fill = land)
) +
    geom_histogram(stat = "identity", position = "dodge")

hist
```

Observa cómo nuestra figura solo muestra los números de parcela 2, 4 y 6. Si quieres que el eje muestre cada número de parcela, 1 - 6, puedes ejecutar el siguiente código usando breaks = c(1,2,3,4,5 ,6). También podemos especificar los límites de los ejes: al ejecutar el código a continuación, verás que el límite del eje '*y*' ahora se extiende al valor de 50. Esto nos ayuda a mantener todos nuestros datos dentro de las etiquetas de eje que tenemos, en términos de visualización.

```
hist <- ggplot(species_counts, aes(x = plot, y = Species_number, fill = land)
) +
    geom_histogram(stat = "identity", position = "dodge") +
    scale_x_continuous(breaks = c(1,2,3,4,5,6)) +
    scale_y_continuous(limits = c(0, 50))

hist
```

1a. Agregar títulos, subtítulos, subtítulos y etiquetas de eje

Ahora es el momento de agregar más información a nuestros gráficos, por ejemplo, el título, el subtítulo, el título y las etiquetas de los ejes.

```
hist <- ggplot(species_counts, aes(x = plot, y = Species_number, fill = land)
) +
    geom_histogram(stat = "identity", position = "dodge") +
    scale_x_continuous(breaks = c(1,2,3,4,5,6)) +
    scale_y_continuous(limits = c(0, 50)) +
    labs(title = "Species richness by plot",
         x = "\n Plot number", y = "Number of species \n") +
    theme(axis.text = element_text(size = 12),
          axis.title = element_text(size = 12, face = "italic"),
          plot.title = element_text(size = 14, hjust = 0.5, face = "bold"))
```

1b. Cambiar el fondo

Todas nuestras gráficas en este momento tienen un fondo gris y, sinceramente, nadie es fanático de eso. También tienen líneas de cuadrícula mayores y menores para los ejes *y* y *x*, vamos a eliminarlos. Al agregar theme_bw() se elimina el fondo gris y lo reemplaza con uno blanco. Hay varios otros temas integrados en RStudio, pero personalmente creo que este es el más limpio.

Para eliminar las líneas de la cuadrícula, agregamos el código panel.grid = element_blank() dentro del comando theme(). Al igual que text.axis abarca tanto text.axis.x como text.axis.y, panel.grid abarca varias opciones: panel.grid.major, que a su vez gobierna panel.grid.major.x y panel.grid.major.y, lo mismo para panel.grid.minor

```
hist <- ggplot(species_counts, aes(x = plot, y = Species_number, fill = land)
) +
    geom_histogram(stat = "identity", position = "dodge") +
    scale_x_continuous(breaks = c(1,2,3,4,5,6)) +
    scale_y_continuous(limits = c(0, 50)) +
    labs(title = "Species richness by plot",
         x = "\n Plot number", y = "Number of species \n") +
    theme_bw() +
    theme(panel.grid = element_blank(),
          axis.text = element_text(size = 12),
          axis.title = element_text(size = 12),
          plot.title = element_text(size = 14, hjust = 0.5, face = "bold"))
```

1c. Arregla la leyenda y personaliza los colores.

Usaremos las funciones scale_...() para personalizar tanto el código de color como la leyenda a la vez.

La función scale_fill_manual(values = c("color-1", " color-2", ...)) permite decidir valores de color personalizados para elementos sólidos (barras, diagramas de caja, cintas, etc.), y su equivalente scale_colour_manual() funciona exactamente igual para los elementos de línea (puntos en un gráfico de dispersión, líneas de regresión, contornos de cajas o columnas, etc.). Debes asegurarte de poner tantos colores como niveles de factores haya en sus datos.

Puedes definir colores utilizando los nombres de colores integrados de R o especificando sus códigos hexadecimales.

```
hist <- ggplot(species_counts, aes(x = plot, y = Species_number, fill = land)
) +
    geom_histogram(stat = "identity", position = "dodge") +
    scale_x_continuous(breaks = c(1,2,3,4,5,6)) +
    scale_y_continuous(limits = c(0, 50)) +
    scale_fill_manual(values = c("rosybrown1", "#deebf7"),
name = "Tierra de Magia") +    labs(title = "Riqueza de especies por parcela"
,
         x = "\n Número de parcela", y = "Número de especies \n") +
```

```
    theme_bw() +
    theme(panel.grid = element_blank(),
          axis.text = element_text(size = 12),
          axis.title = element_text(size = 12),
          plot.title = element_text(size = 14, hjust = 0.5, face = "bold"),
          plot.margin = unit(c(0.5,0.5,0.5,0.5), units = , "cm"),
      legend.title = element_text(face = "bold"),
          legend.position = "bottom",
          legend.box.background = element_rect(color = "grey", size = 0.3)))
```

Otra cosa que a veces podríamos querer cambiar es la etiqueta real del grupo (es decir, los niveles de los factores). En el siguiente ejemplo, nuestra base de datos tiene especificados "Hogsmeade" y "Narnia", lo cual es una suerte, ya que se reflejarían correctamente en la leyenda creada por ggplot. Sin embargo, si simplemente se hubieran enumerado como "grupo1" y "grupo2" en el archivo de datos original, nos gustaría tener etiquetas más informativas. Podemos hacerlo manipulando labels = c("xxx", "xxx"). En el siguiente ejemplo, cambiamos las etiquetas predeterminadas de "Hogsmeade" y "Narnia" a "HOGSMEADE" y "NARNIA" solo con fines de demostración. Importante: Asegúrate de enumerar los nombres de las nuevas etiquetas en el mismo orden en que se enumeran sus factores en el conjunto de datos. Usa niveles (base-de-datos$nombre-del-factor) para ver los factores en orden (generalmente en orden alfabético).

```
hist <- ggplot(species_counts, aes(x = plot, y = Species_number, fill = land)
) +
    geom_histogram(stat = "identity", position = "dodge") +
    scale_x_continuous(breaks = c(1,2,3,4,5,6)) +
    scale_y_continuous(limits = c(0, 50)) +
    scale_fill_manual(values = c("rosybrown1", "#deebf7"),
                      labels = c("HOGSMEADE", "NARNIA"),
                      name = "Tierra de Magia") +
    labs(title = "Riqueza de especies por parcela",
         x = "\n Número de parcela", y = "Número de especies \n") +
    theme_bw() +
    theme(panel.grid = element_blank(),
          axis.text = element_text(size = 12),
          axis.title = element_text(size = 12),
          plot.title = element_text(size = 14, hjust = 0.5, face = "bold"),
          plot.margin = unit(c(0.5,0.5,0.5,0.5), units = , "cm"),
      legend.title = element_text(face = "bold"),
          legend.position = "bottom",
          legend.box.background = element_rect(color = "grey", size = 0.3)))
```

Veamos algunos de los elementos de theme() que hemos usado en los ejemplos anteriores:

- legend.title : permite cambiar el tamaño de fuente de la leyenda o su formato (por ejemplo, negrita).

- legend.position: se puede definir con posiciones aceptadas como "abajo, pero también puedes poner leyend.position= c(0.1, 0.8), lo que llevaría la leyenda a la esquina superior izquierda

- legend.box.background = element_rect() para crear un rectángulo gris claro que rodea la leyenda. Si no deseas esto, simplemente puedes eliminar esa línea de código.

3. Personaliza diagramas de caja

También podríamos graficar los datos usando diagramas de caja. Los diagramas de caja a veces se ven mejor que los diagramas de barras, ya que hacen un uso más eficiente del espacio que las barras y pueden reflejar la incertidumbre de manera más explícita.

Para hacer los diagramas de caja, remodelaremos ligeramente el conjunto de datos para tener en cuenta también el año.

```
yearly_counts <- magic_veg %>%
  group_by(land, plot, year) %>%
  summarise(Species_number = length(unique(species))) %>%
  ungroup() %>%
  mutate(plot = as.factor(plot))

boxplot <- ggplot(yearly_counts, aes(plot, Species_number, fill = land)) +
  geom_boxplot()
```

Esto hace un trabajo mucho mejor al mostrar qué parcelas son las más ricas en especies.

Ahora la podemos “enchular”

```
boxplot <- ggplot(yearly_counts, aes(x = plot, y = Species_number, fill = lan
d)) +
  geom_boxplot() +
  scale_x_discrete(breaks = 1:6) +
  scale_fill_manual(values = c("rosybrown1", "#deebf7"),
                    breaks = c("Hogsmeade","Narnia"),
                    name="Tierra de Magia",
                    labels=c("Hogsmeade", "Narnia")) +
  labs(title = "Riqueza de especies por parcela",
       x = "\n Número de parcela", y = "Número de especies \n") +
  theme_bw() +
  theme() +
  theme(panel.grid = element_blank(),
        axis.text = element_text(size = 12),
        axis.title = element_text(size = 12),
        plot.title = element_text(size = 14, hjust = 0.5, face = "bold"),
        plot.margin = unit(c(0.5,0.5,0.5,0.5), units = , "cm"),
        legend.position = "bottom",
```

¿Caja, barra, punto...?

Los diagramas de barras se usan muy comúnmente para mostrar diferencias o clasificación entre grupos. Un problema con ellos, especialmente si se usan sin una medida de incertidumbre (por ejemplo, barras de error), es que lo que muestran es un rango de valores a partir de 0. Si la variable que está trazando puede tener valores razonables de cero, entonces está bien, pero a menudo es improbable. Por ejemplo, no nos imaginaríamos que nuestras tierras de magia podrían estar completamente desprovistas de cualquier forma de vida y, por lo tanto, tener una riqueza de especies de cero. Lo mismo es cierto si estmos comparando el peso corporal, la altura de la planta y la gran mayoría de las variables ecológicas.

Una alternativa fácil es un diagrama de puntos, que podemos hacer al resumir los datos de recuentos de especies para obtener una media y una desviación estándar de los recuentos de especies para cada tierra. Luego usamos aes(x = land, y = mean) y geom_point() en lugar de geom_histogram(), y agregamos su incertidumbre con geom_errorbar(aes(x = land, ymin = mean - sd, ymax = mean + sd).

```
summary <- species_counts %>%  group_by(land) %>% summarise(mean = mean(Speci
es_number), sd = sd(Species_number))

 # Haz una grafica de puntos
```

Los diagramas de caja, al igual que los diagramas de puntos, brindan una idea más precisa del rango de valores en los datos: pero recuerda que la línea más gruesa representa la mediana, no la media.

Factores de reordenamiento

Si quisiéramos que Narnia viniera antes que Hogsmeade, primero tendríamos que reordenar los datos en el marco de datos. Desde este punto, después de reordenar los datos, ggplot siempre trazará Narnia antes que Hogsmeade. Además, ten en cuenta cómo hemos cambiado el orden de las cosas en scale_fill_manual: arriba lo teníamos como "Hogsmeade", luego "Narnia", y ahora tenemos "Narnia" antes de "Hogsmeade" para reordenar también la leyenda.

```
# Reordenar los datos
yearly_counts$land <- factor(yearly_counts$land,
                             levels = c("Narnia", "Hogsmeade"),
                             labels = c("Narnia", "Hogsmeade"))

# gráfica
boxplot <- ggplot(yearly_counts, aes(x = plot, y = Species_number, fill = lan
d)) +
    geom_boxplot() +
    scale_x_discrete(breaks = 1:6) +
    scale_fill_manual(values = c("#deebf7", "rosybrown1"),
                      breaks = c("Narnia","Hogsmeade"),
```

```
                    name = "Tierra de Magia",
                    labels = c("Narnia", "Hogsmeade")) +
  labs(title = "Riqueza de especies por parcela",
       x = "\n Número de parcela", y = "Número de especies \n") +
  theme_bw() +
  theme() +
  theme(panel.grid = element_blank(),
        axis.text = element_text(size = 12),
        axis.title = element_text(size = 12),
        plot.title = element_text(size = 14, hjust = 0.5, face = "bold"),
        plot.margin = unit(c(0.5,0.5,0.5,0.5), units = , "cm"),
        legend.position = "bottom",
        legend.box.background = element_rect(color = "grey", size = 0.3))
```

Datos cualitativos

Este tipo de datos se puede recopilar a través de respuestas a preguntas de encuestas, transcripciones de entrevistas u observaciones. Los ejemplos que usaremos provienen de un estudio sobre el comportamiento humano relacionado con acciones respetuosas con el medio ambiente.

En primer lugar, aprenderemos cómo dar formato a los datos de encuestas y entrevistas de manera efectiva para que puedan usarse fácilmente en el análisis posterior. Luego exploraremos formas de visualizar estos datos gráficamente. Finalmente, ejecutaremos algunos análisis estadísticos simples para responder algunas hipótesis.

Todos los archivos que necesitas para completar este tutorial se pueden descargar desde el repositorio. Luego, carga los paquetes, si no los tienes, instálalos.

```
library(tidyverse)

library(RColorBrewer)
library(tidytext)

library(R.utils)

library(wordcloud)

library(viridis)
```

Finalmente, cargue los archivos de datos que usaremos para el tutorial.

```
# Las respuestas de la encuesta
sust_data <- read_csv("sust_behaviour.csv")

# Una tabla que conecta cada columna en `sust_data` a las preguntas de la enc
uesta

sust_lookup <- read_csv("sust_lookup.csv")
```

Estos son datos anónimos de una encuesta en línea diseñada para investigar si el género y los diferentes arreglos de cohabitación influyen en la probabilidad de que los participantes realicen acciones respetuosas con el medio ambiente en la casa, como reciclar o comprar productos domésticos sostenibles.

Este conjunto de datos está formateado de tal forma que todavía no está listo para el análisis. Dedicaremos un tiempo a preparar los datos para el análisis.

El objeto sust_lookup es una tabla que conecta el nombre de cada columna en la base de datos con la pregunta correspondiente que se hizo en la encuesta. Reemplazar las preguntas sin procesar con nombres de columna más cortos hace que sea mucho más fácil escribir código, y con la tabla de búsqueda podemos volver a agregar el título de la pregunta real cuando estamos creando gráficos.

1. Formateo de datos cualitativos

Obtener datos cualitativos en un formato adecuado para el análisis es un requisito previo clave para el éxito. La mayoría de las herramientas analíticas se adaptan mejor a conjuntos de datos numéricos, por lo que se necesita algo de coerción para generar valores numéricos a partir de nuestras observaciones cualitativas. Si diseñas una encuesta, recuerda considerar cómo analizará los datos, dibuja cómo imaginas que se verán los gráficos, etc., esto lo hará mucho más fácil más adelante.

Algunas de las preguntas de este conjunto de datos se diseñaron en una escala de cinco puntos. Cada columna en el marco de datos contiene respuestas a una sola pregunta. Puedes Mirar los valores contenidos en la columna tituladasustainability_daily_think ingresando este código:

```
unique(sust_data$sustainability_daily_think)
```

Deberías ver que la columna contiene cinco categorías discretas que siguen un orden intuitivo de menor a mayor: Nunca, Rara vez, A veces, A menudo, Todo el tiempo. Podríamos simplemente tratar las respuestas como factores, pero esto no tiene en cuenta su naturaleza ordinal, cualquier gráfico que hagamos simplemente colocará los factores en orden alfabético, por lo que le pediremos a R que los trate como un "factor ordenado". usando este código:

```
sust_data$sustainability_daily_think <- factor(sust_data$sustainability_daily
_think,
    levels = c("Never", "Rarely", "Sometimes", "Often", "All the time"),
    ordered = TRUE)
```

 Busca otras columnas que creas que contienen datos de factores ordenados como éstos, luego cámbialos a factores ordenados de la misma manera que lo hicimos anteriormente.

Otras columnas en la base de datos, como sust_data$energy_action, contienen cadenas de letras, p. BDEFH. Esta pregunta de la encuesta presentó al usuario una lista de acciones sostenibles relacionadas con el hogar, p. “He sustituido mis bombillas por bombillas de bajo consumo” y pedía al usuario que marcara todas las que le correspondían. Cada una de las letras se refiere a una sola acción.

Imagina que queremos hacer la pregunta “¿El número de acciones relacionadas con la energía sostenible realizadas varía según el género?”. Para hacer esto, necesitamos crear una columna en la base de datos que cuente el número de letras en sust_data$energy_action. Para hacer esto podemos usar nchar(), una función simple que cuenta el número de caracteres en una cadena. Además, tenemos que usar as.character() para obligar a energy_action a una variable de carácter, ya que actualmente está codificado como un factor, que no es compatible con nchar(). :

```
sust_data$energy_action_n <- nchar(as.character(sust_data$energy_action))
```

2. Visualización de datos cualitativos

Ahora que formateamos nuestros datos para el análisis, podemos visualizar los datos para identificar patrones interesantes.

Podemos crear gráficos de barras para visualizar el número de respuestas a una pregunta que encajan en cada una de las categorías ordinales. La forma correcta del gráfico de barras dependerá del tipo de pregunta que se haya hecho y de la redacción de las distintas respuestas. Por ejemplo, si las posibles respuestas se presentaran como "Muy en desacuerdo", "En desacuerdo", "Ni de acuerdo ni en desacuerdo", "De acuerdo", "Muy de acuerdo", podríamos suponer que la respuesta neutral o cero está en el medio, con “En desacuerdo” siendo negativo y “De acuerdo” siendo positivo. Por otro lado, si las respuestas se presentaran como “Nunca”, “Rara vez”, “A veces”, “A menudo”, “Todo el tiempo”, la respuesta neutral o cero sería “Nunca”, siendo todas las demás respuestas positivas. Para el primer ejemplo, podríamos usar un "gráfico de barras apiladas divergentes", y para el último simplemente usaríamos un "gráfico de barras apiladas" estándar.

Gráfico de barras apiladas divergentes

Primero hagamos un gráfico de barras apiladas divergentes de respuestas a la pregunta: "¿Con qué frecuencia durante un día normal piensas en la sostenibilidad de tus acciones?". Para investigar cómo el género afecta la respuesta, puedes ver en la tabla de búsqueda (sust_lookup) que las respuestas a esta pregunta se almacenan en la columna llamadasustainability_daily_think.

Si miras en la columna sustain_daily_think, verás que contiene filas como “Frecuentemente”, “Nunca” y “Todo el tiempo”:

```
sust_data$sustainability_daily_think
```

Primero, necesitamos hacer una base de datos de resumen de las respuestas de esta columna, lo cual se puede hacer fácilmente usando el paquete dplyr.

```
sust_think_summ_wide <- sust_data %>%
    group_by(gender, sustainability_daily_think) %>%

    tally() %>%

    mutate(perc = n / sum(n) * 100) %>%
    dplyr::select(-n) %>%
    group_by(gender) %>%
    spread(sustainability_daily_think, perc)
```

group_by() y tally() trabajan juntos para contar el número de respuestas a la pregunta, dividiendo el conteo en cada combinación de género (p. ej., Masculino, Femenino) y tipo de respuesta (p. ej., Nunca, A menudo, etc.). mutate() crea otra columna llamada perc, que es el porcentaje del total de encuestados que respondieron en función de su género y respuesta. dplyr::select() elimina la columna n que se creó usando tally(). El segundo group_by() y spread() trabajan juntos para convertir el marco de datos de formato largo a formato ancho. Es más fácil demostrar lo que esto significa con un diagrama:

Este tipo de gráfico se denomina gráfico de barras apiladas divergentes. "Apilado" significa que cada barra se divide en subcategorías, en este caso cada barra es un género y cada subbarra es el porcentaje de ese género que da una respuesta particular. "Divergente" significa que la barra se posa sobre toda la línea cero. Formatear el gráfico de barras de esta manera nos permite hacer una distinción visual entre respuestas negativas (es decir, nunca, rara vez), respuestas positivas (es decir, a menudo, todo el tiempo) y respuestas neutrales (es decir, a veces).

Para obtener las respuestas de "A veces" a ambos lados de la línea 0, tenemos que dividir los valores de "A veces" en dos grupos (medio bajo y medio alto), un grupo se ubicará debajo de la línea cero y un grupo se ubicará arriba. Para hacer esto podemos usar dplyr nuevamente:

```
sust_think_summ_hi_lo <- sust_think_summ_wide %>%
    mutate(midlow = Sometimes / 2,
        midhigh = Sometimes / 2) %>%
    dplyr::select(gender, Never, Rarely, midlow, midhigh, Often, `All the tim
e`) %>%
    gather(key = response, value = perc, 2:7) %>%
    `colnames<-`(c("gender", "response", "perc"))
```

En el código anterior hemos creado dos nuevas columnas midhigh y midlow, que contienen valores de A veces, pero divididos por dos. La columna A veces se elimina del marco de datos usando dplyr::select(). Luego, el marco de datos se vuelve a recopilar en formato largo para que haya tres columnas, género, tipo de respuesta y porcentaje de encuestados.

A continuación, tenemos que dividir la base de datos en dos, una que contenga las respuestas negativas y la mitad de A veces (es decir, medio bajo) y otro que contenga las respuestas positivas y la otra mitad de A veces (es decir, medio alto). Necesitamos hacer esto porque en realidad hay dos conjuntos de barras en el gráfico, uno para el lado izquierdo de la línea cero y otro para el lado derecho de la línea cero.

```
sust_think_summ_hi <- sust_think_summ_hi_lo %>%
    filter(response %in% c("All the time", "Often", "midhigh")) %>%
    mutate(response = factor(response, levels = c("All the time", "Often", "m
idhigh")))

## Error in filter(., response %in% c("All the time", "Often", "midhigh")): o
bject 'sust_think_summ_hi_lo' not found

sust_think_summ_lo <- sust_think_summ_hi_lo %>%
    filter(response %in% c("midlow", "Rarely", "Never")) %>%
    mutate(response = factor(response, levels = c("Never", "Rarely", "midlow"
)))
```

A continuación, para cambiar los colores del gráfico, debemos definir un esquema de color personalizado. Para hacer esto, podemos usar una paleta de colores de RColorBrewer y modificarla un poco.

```
# Uasamos RColorBrewer para guardar el código de colores como un vector

legend_pal <- brewer.pal(name = "RdBu", n = 5)

# Duplic el valor medio, recuerda que "Sometimes" son dos grupos, "midhigh" y
"midlow"
legend_pal <- insert(legend_pal, ats = 3, legend_pal[3])

# remplazamos colores

legend_pal <- gsub("#F7F7F7", "#9C9C9C", legend_pal)

# Asignamos nombresa cada color
names(legend_pal) <- c("All the time", "Often", "midhigh", "midlow", "Rarely"
, "Never" )
```

Ahora sí, la grafica:

```
ggplot() +
    geom_bar(data = sust_think_summ_hi, aes(x = gender, y=perc, fill = respon
se), stat="identity") +
    geom_bar(data = sust_think_summ_lo, aes(x = gender, y=-perc, fill = respo
nse), stat="identity") +
    geom_hline(yintercept = 0, color =c("black")) +
    scale_fill_manual(values = legend_pal,
        breaks = c("All the time", "Often", "midhigh", "Rarely", "Never"),
        labels = c("All the time", "Often", "Sometimes", "Rarely", "Never"))
+
    coord_flip() +
    labs(x = "Género", y = "Porcentaje de encuestados (%)") +
    ggtitle(sust_lookup$survey_question[sust_lookup$column_title == "sustaina
bility_daily_think"]) +
    theme_classic()
```

Gráfico de barras apiladas básico

Para hacer un gráfico de barras apiladas convencional, usaremos la pregunta "¿Cuántas de estas acciones sostenibles relacionadas con la energía lleva a cabo?", cuyas respuestas se encuentran en sust_data$energy_action. Agruparemos las respuestas por grupo de edad.

Primero, necesitamos contar el número de acciones sostenibles realizadas, como hicimos antes:

```
sust_data$energy_action_n <- nchar(as.character(sust_data$energy_action))

barchart <- ggplot(sust_data, aes(x =energy_action_n, fill = age)) +
    geom_bar() +
    scale_fill_viridis_d() +
    scale_x_continuous(breaks = seq(1:8)) +
    theme_classic()
```

Gráfico de burbujas

Si queremos comparar las correlaciones entre dos categorías de datos, podemos usar un gráfico de burbujas. Por ejemplo, ¿hay un patrón entre la edad del encuestado y la frecuencia con la que piensa en actividades sostenibles? Los datos de esta encuesta no contienen valores de edad reales, solo rangos de edad (por ejemplo, 18-20, 21-29, etc.).

Primero, crea una tabla de resumen contando el número de respuestas de los dos grupos, la edad y la frecuencia con la que piensan en actividades sostenibles:

```
sust_bubble <- sust_data %>%
    group_by(age, sustainability_daily_think) %>%
    tally()

bubbleplot <- ggplot(sust_bubble, aes(x = age, y = sustainability_daily_think
)) +
    geom_point(aes(size = n)) +
    theme_classic()
```

Gráfico de burbujas de la edad frente a pensamientos sostenibles

Además de las preguntas de estilo de casilla de verificación, algunas preguntas de nuestra encuesta pedían comentarios de texto a mano alzada. Estos comentarios brindan información adicional y contexto a las respuestas de la encuesta y no deben ignorarse. Como ejemplo, mire los primeros 20 elementos de la columna energy_action_comment escribiendo:

```
head(sust_data$energy_action_comment, 20)
```

Podemos extraer los comentarios de palabras clave para crear una imagen más completa de lo que pensaban nuestros encuestados cuando realizaron la encuesta y si eso varía según el género. Para facilitar el trabajo con los comentarios, debemos hacer que los datos estén "ordenados" dividiendo cada comentario para que cada fila tenga una sola palabra. Comentarios de todas las preguntas.

La siguiente tubería recopila todas las columnas de comentarios junto con las columnas de género e identificación (dplyr::select()), luego reúne esas columnas de comentarios en una sola columna (reunir()), luego transforma la columna de comentarios de un factor en un clase de carácter (mutar()).

Después toma los datos recopilados y usa unnest_tokens() del paquete tidytext para dividir los comentarios de modo que solo haya una palabra por fila, luego usa la lista de palabras aburridas del objeto stop_words que cargamos anteriormente para eliminar esas palabras de nuestro conjunto de datos (filtrándolos usando filter() y los operadores ! y %in% para eliminar las palabras que aparecen en la columna stop_words$word). También estamos eliminando valores vacíos !(is.na(comment_word)) y palabras que en realidad son solo números (is.na(as.numeric(comment_word)). Luego cuenta el número de ocurrencias de cada palabra única en la columna comment_word , agrupando por género y resumiendo usando la función n() Finalmente, un poco de limpieza en la forma de eliminar palabras que ocurren menos de 10 veces (filter(n > 10)).

```
sust_comm_gather <- sust_data %>%
  dplyr::select(id, gender, energy_action_comment,
                food_action_comment, water_action_comment,
                waste_action_comment, other_action_comment) %>%
  gather(action, comment, -id, -gender) %>%
  mutate(comment = as.character(comment))

sust_comm_tidy <- sust_comm_gather %>%

  unnest_tokens(output = comment_word,

                input = comment) %>%

  filter(!(is.na(comment_word)),

         is.na(as.numeric(comment_word)),

         !(comment_word %in% stop_words$word)) %>%

  group_by(gender, comment_word) %>%

  summarise(n = n()) %>%

  ungroup() %>%

  filter(n > 10)
```

Definamos una paleta de colores personalizada para rojo y azul y nombremos los colores según nuestras categorías de género:

```
male_female_pal <- c("#0389F0", "#E30031")
names(male_female_pal) <- c("Male", "Female")
```

Ahora es fácil trazar las ocurrencias de cada palabra y colorear por género (fill = gender), usando ggplot():

```
occurrence <- ggplot(sust_comm_tidy, aes(x = comment_word, y = n, fill = gend
er)) +
    geom_bar(stat = "identity") +
    coord_flip() +
    scale_fill_manual(values = male_female_pal) +
    theme_classic()
```

Gráfico de barras de palabras más comunes

Ocurrencia de comentarios por género

Comentarios de una sola pregunta

También podríamos querer investigar los comentarios de una sola pregunta con más detalle. Por ejemplo, la columna energy_action_comment. Primero repetir la acción de convertir a formato de caracteres (mutar()), luego filtrar, resumir y agrupar siguiendo un procedimiento similar al del gráfico anterior.

```
tidy_energy_often_comment <- sust_data %>%

  mutate(energy_action_comment = as.character(energy_action_comment)) %>%

  unnest_tokens(output = energy_action_comment_word,

                input = energy_action_comment) %>%

  filter(!(is.na(energy_action_comment_word)),

         is.na(as.numeric(energy_action_comment_word)),

         !(energy_action_comment_word %in% stop_words$word)) %>%

  group_by(gender, energy_action_comment_word) %>%

  summarise(n = n()) %>%

  ungroup()
```

Luego, mantén solo las palabras más comunes y grábalas como un gráfico de barras:

```
tidy_energy_often_comment_summ <- tidy_energy_often_comment %>%

  filter(n > 10) %>%

  mutate(energy_action_comment_word = reorder(energy_action_comment_word, n ) 
)



(most_common_plot <- ggplot(tidy_energy_often_comment_summ, aes(x = energy_ac
tion_comment_word, y = n)) +

    geom_col() +

    xlab(NULL) +  # this means we don't want an axis title

    coord_flip() +

    theme_classic())
```

Nubes de palabras

Para trazar efectivamente más palabras y sus frecuencias, también puedes crear una nube de palabras:

```
tidy_energy_often_comment %>%
    with(wordcloud(words = energy_action_comment_word, freq = n, max.words = 
100))
```