Análisis de Datos Categóricos

Ayudantía 3

Felipe Olivares

Contenido

Funciones en R

Una función es un trozo de código autónomo que realiza una tarea específica.

1. Las primeras son funciones que hacen algo y devuelven un objeto Estas funciones toman algunas entradas especificadas, hacen algunas manipulaciones / operaciones, y luego devuelven un objeto. Algunos ejemplos son *mean()* (toma la media de un vector), *lm()* (ajusta un modelo lineal), o *read.csv* (carga una tabla de datos de formato excel).
2. En segundo lugar están las funciones que tienen algún efecto externo en su ordenador o entorno de trabajo. Estas funciones hacen algo pero no devuelven ningún objeto. Los ejemplos incluyen cosas como *write.csv()* (escribe un archivo excel en tu computador), *plot()* (hace un gráfico), *library()* (carga un paquete).

Funciones en R

Si bien la instalación de R base incorpora las funciones necesarias para la estimación estadística y modelamiento de nuestros datos, en ocaciones es común que necesitemos realizar tareas para las cuales el lenguaje no ha especificado una función.

Veamos un ejemplo: El agua se congela a 0º Centígrados y hierve a 100º Centígrados (diferencia de 100º). El agua se congela a 32º Fahrenheit y hierve a 212º Fahrenheit (diferencia de 180º). Por lo tanto cada grado en la escala Fahrenheit es igual a 100/180 o 5/9 grados en la escala Celsius.

```
 # Sintaxis
fahrenheit_to_celsius <- function(temp_F) {
  temp_C <- (temp_F - 32) * 5 / 9
  return(temp_C)}
```

Funciones en R

Ahora veamos los componentes de la función:

'fahrenheit_to_celsius' = nombre asignado a la función que será ejecutada para llamarla.

function = comando que define argumentos y operaciones de la función.

() = argumentos de la función.

{} = operaciones de la función

```
#Resultado
fahrenheit_to_celsius (32)
```

```
## [1] 0
```

Distribución Binomial

Dentro de las distribuciones de probabilidad discretas, una de las más populares es la distribución binomial. La simulación de variables aleatorias que sigan parámetros de distribución Bernoulli y Binomial puede ser ejecutadas a través de dos funciones incorporadas en R base.

rbinom: Esta función genera un vector de variables aleatorias distribuidas binomialmente.

ejemplo: Obtener 1 cara con el lanzamiento de una moneda "justa"

```
# Sintaxis
ej1 <- rbinom(n=1, size=1, p=0.5)
ej1
```

```
## [1] 1
```

Distribución Binomial

Donde los argumentos:

n = número de experimentos independientes.

size = número de intentos o teraciones.

p = probabilidad de éxito.

Otro ejemplo: Ahora tiramos la misma moneda "justa" 3 veces

```
set.seed(12345)
ej2 <- rbinom(n=3, size=1, p=0.5)
ej2
```

```
## [1] 1 1 1
```

Distribución Binomial

dbinom: Esta función devuelve el valor de la función de densidad de probabilidad.

tomemos el caso de una moneda justa para una probabilidad de obtener 5 caras con 10 lanzamientos

$$\mathbb{P}(Y = 5) = \frac{10!}{5!5!} \times (0.5)^5 \times (1 - 0.5)^5 = 252 \times (0.5)^{10} = 0.25$$

Distribución Binomial

Veamos ahora el código en *R*

```
#  obtener 5 "Caras" con 10 lanzamientos
ej3 <- dbinom(x=5,size=10,prob=0.5)
ej3
```

```
## [1] 0.2460938
```

Donde los argumentos:

x = número de éxitos.

size = número de intentos o teraciones.

p = probabilidad de éxito.

Distribución Binomial

Veamos otro ejemplo:

La función **dbinom** devuelve el valor de la función de densidad de probabilidad de la distribución binomial dada una determinada variable aleatoria "*x*" , número de ensayos "*size*" (tamaño de la muestra) y probabilidad de éxito en cada ensayo "*prob*"

Carlos cuando juega fútbol acerta el 80% de sus penales. Si lanza 12 penales, ¿cuál es la probabilidad de que haga exactamente 10?

```
# Sintaxis
ej4 <- dbinom(x=10,size=12,prob=0.8)
ej4
```

```
## [1] 0.2834678
```

Simulación de Monte Carlo

La simulación o método de Montecarlo es un método estadístico utilizado para resolver problemas matemáticos complejos a través de la generación de variables aleatorias. Método de Monte Carlo o una simulación de probabilidad múltiple, es una técnica matemática que se utiliza para estimar los posibles resultados de un evento incierto. Una simulación de Monte Carlo crea un modelo de posibles resultados aprovechando una distribución de probabilidades, por ejemplo, una distribución binomial.

supongamos que tenemos 2 monedas sesgadas con una probabilidad de que salga cara de un 40% y otra de 60%

```
set.seed(12345) #limpiar para randomización

#Monedas sesgadas
moneda1 <- rep(c("Cara", "Sello"), times = c(4, 6)) # %40
moneda2 <- rep(c("Cara", "Sello"), times = c(6, 4)) # %60

#simulamos un experimento donde tiramos 100000 veces la moneda
num_veces <- 100000
resultado1 <- replicate(num_veces, {
  sample(moneda1, 1) })
resultado2 <- replicate(num_veces, {
  sample(moneda2, 1)})
```

Simulación de Monte Carlo

$$\mathbb{P}(C_1 C_2) = \mathbb{P}(C_1)\mathbb{P}(C_2) = 0.4 \times 0.6 = 0.24$$

```
table(resultado1)
```

```
## resultado1
##  Cara Sello
## 40138 59862
```

```
table(resultado2)
```

```
## resultado2
##  Cara Sello
## 60002 39998
```

```
#promedio
a<-mean(moneda1=="Cara")
b<-mean(moneda2=="Cara")

#test deindependencia
a * b
```

```
## [1] 0.24
```

Gráficos

Una de las herramientas más útiles para la exploración y evaluación de nuestros datos son los análisis visuales disponibles a partir de distintas funciones de R. Con todo, uno de los repositorios más útiles y de más alta potencia gráfica es library(ggplot2) el cual forma parte de las funciones incluidas en la librería library(tidyverse).

A continuación veamos una breve introducción a su sintaxis y usos.

Gráficos

Existen 3 elementos centrales en la sintaxis del comando ggplot:

1. Una estructura visual (aesthetic) que le informa a ggplot qué variables serán mapeadas en el eje X e Y, (y frecuentemente otros atributos del gráfico, como los esquemas de color con que serán completados). Intuitivamente, aes puede ser pensada como la función que especifica qué estarás graficando.
2. Una geometría (geom) que le informa a ggplot cuál será la estructura básica para la creación del gráfico. Intuitivamente, geom puede ser pensada como la función que especifica cómo estarás graficando.
3. Otras opciones de customización (labels), como los títulos del gráfico, los tamaños de los elementos, el tipo de tema de construcción, etc. En la función esto es conocido como labs.

Entonces:

- **ggplot= aes + geom + labs**

Gráficos

Utilizando la base de datos del Observatorio de conflictos de COES. Observemos la presencia de organizaciones sociales para el conjunto de los datos 2009-2019

```
# Sintaxis gráficos de barras
ggplot(df1, aes(x = organizacion)) + # definimos eje X
  geom_bar() +  # gráficos de barras básico
  labs(title = "OCS 2009-2019",
       subtitle = "Gráficos de barras organizaciones",
       caption = "Figura 1", x = "Organizaciones", y = "conteo")
```

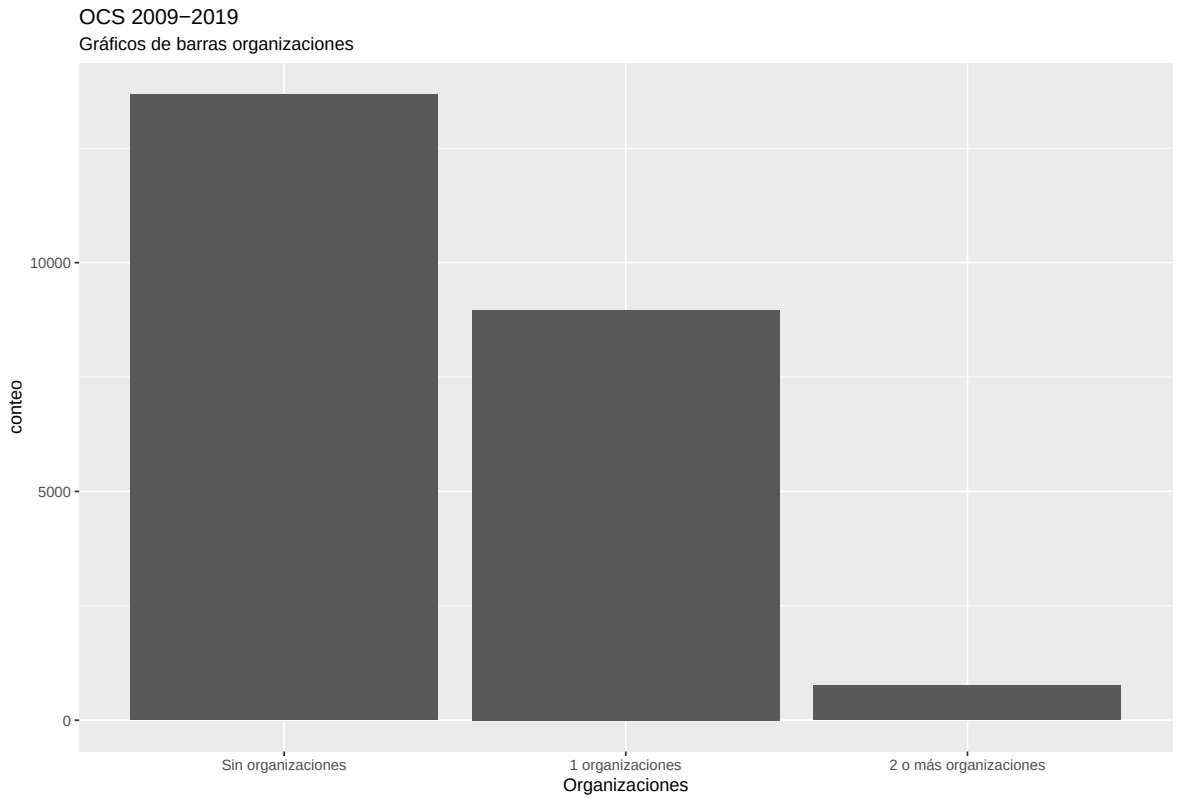


Figura 1

Gráficos

Otro ejemplo puede ser tomar las protestas laborales y ver la ocurrencia en el tiempo

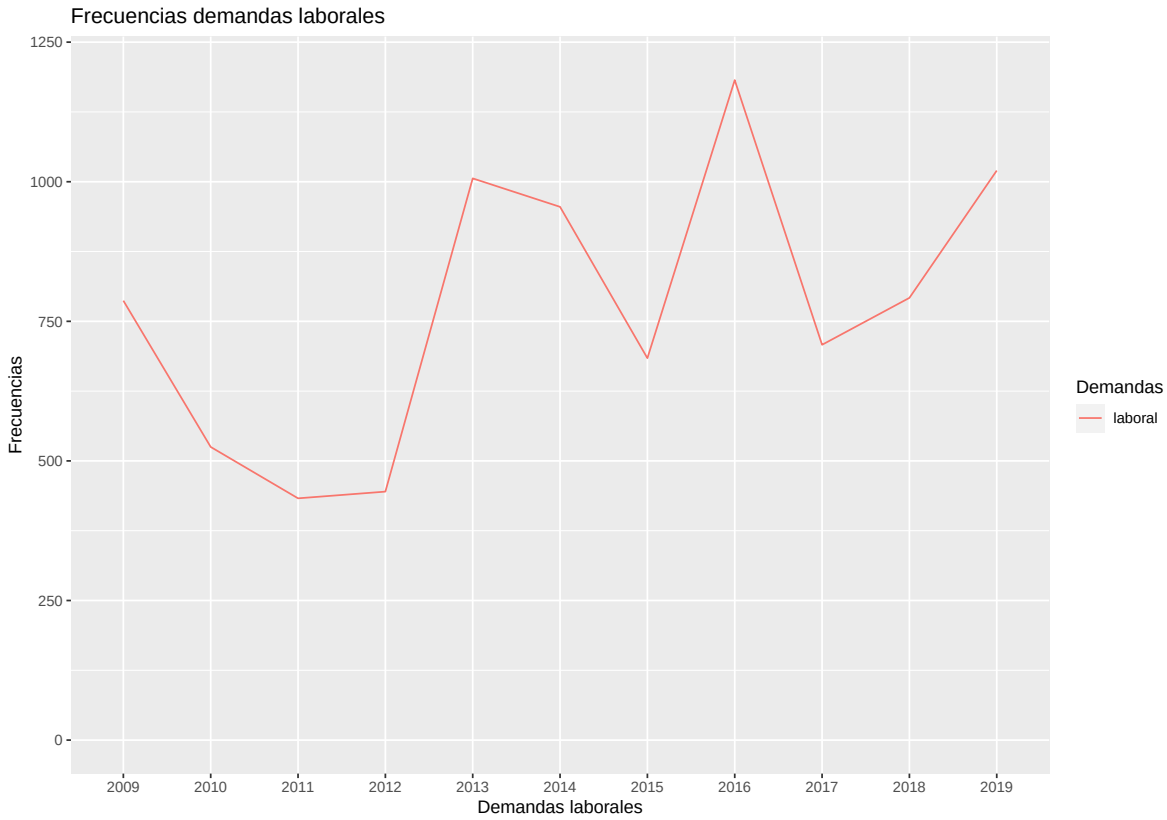
```
#trabajamos los datos
df_lab <- df1 %>% filter(laboral == "Sí") %>% # protestas laborales
  group_by(ano) %>% #agrupamos por año
   summarise(frecuencia = n()) %>% # contamos las frecuencias anuales
  mutate(demanda = "laboral") #creamos la variable

head (df_lab)
```

```
## # A tibble: 6 x 3
##   ano   frecuencia demanda
##   <fct>      <int> <chr>
## 1 2009         787 laboral
## 2 2010         525 laboral
## 3 2011         433 laboral
## 4 2012         445 laboral
## 5 2013        1006 laboral
## 6 2014         955 laboral
```

Gráficos

```
# Sintaxis gráficos de lineas
 ggplot(df_lab, aes(x = ano, y = frecuencia,group = 1, colour = demanda))+
   geom_line()+ #gráficos de lineas
   coord_cartesian(ylim=c(0,1200)) + # coordenadas del eje Y
  labs(title = 'Frecuencias demandas laborales',
       y = 'Frecuencias',
       x = 'Demandas laborales',
       colour = 'Demandas')
```

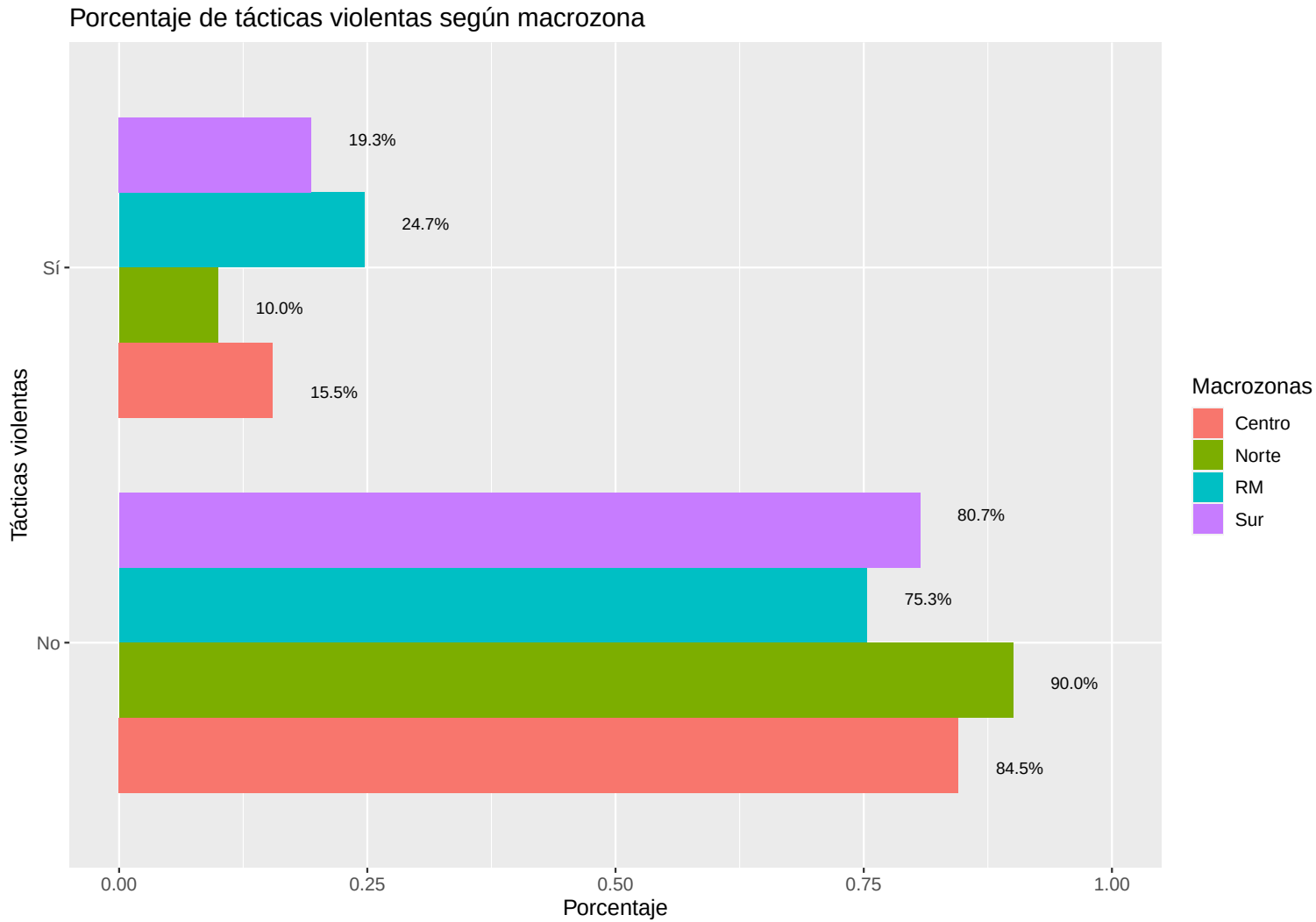


Gráficos

```
#Sintaxis gráficos de barras
g1 <-df1 %>% group_by(macrozona) %>% # guardamos el gráfico en "g1"
  count(violenta) %>%  #contamos las tácticas violentas
  mutate(prop = prop.table(n)) %>%  #sacamos el %
  ggplot(aes(x = violenta, y = prop,
             fill=macrozona, label=scales::percent(prop, accuracy =.1))) +
  geom_bar(stat="identity", width=0.8, position = "dodge")+
  coord_flip(ylim =c(0,1)) + #rotamos el gráfico
  labs(title = 'Porcentaje de tácticas violentas según macrozona',
       y = 'Porcentaje',
       x = 'Tácticas violentas',
       fill = 'Macrozonas')+
  theme(axis.title = element_text(), text = element_text(size = 12))+
   geom_text(position = position_dodge2(width = .9), # colocamos los %
             vjust = 0.35,
             hjust = -0.8,
             size = 3)
```

Gráficos

```
#llamamos el gráfico creado
g1
```



Consejos

1. El aprendizaje de R es como la práctica de un instrumento musical, entre **mayor dedicación y frecuencia de práctica mejor será la adaptación al programa, su uso y fluidez.** Recomendamos explorar RStudio y el lenguaje R todas las semanas para apoyar el aprendizaje progresivo.
2. Al ser parte de una **comunidad académica internacional**, muchas de las dudas que puedas tener en el camino son y han sido discutidas en múltiples recursos de internet como **blogs, foros, y tutoriales** que dan soluciones múltiples a un mismo problema. A modo de apoyo es recomendado consultar los recursos disponibles en internet además del material de ayudantías y consultas del curso.