Laboratorio 3: Importar datos a R

Trabajar con datos en R

Dr. Marco A. González Tagle

Semestre Agosto - Diciembre 2021

Índice

Objetivos de la práctica

- Inicair una sesión en Rstudio
- Realziar una descripción estadística de un conjunto de datos.
- Comprender el diseño del panel de RStudio.
- Conozca la sintaxis *markdown.*

Instrucciones de la práctica

Está practica consiste de tres partes. La primera (cerca de 20 minutos) se refiere a la diferentes formas de importar o introducir datos a R para su posterior análisis. La parte dos involucra realizar diferentes operaciones básicas del programa así como en el proceso de selección mediante restricciones y la generación de submuestras (60 minutos) y finalmente la parte 3 (aproximadamente 60 minutos) nos enfocaremos en formas gráficas para la representación de datos.

Parte 1: Importar datos

Importar desde archivos csv

A continuación se explicará la manera de importar un conjunto de datos que proviene de una base de datos creada en *Excel* a **R**. La importación es muy simple y se realiza mediante la función *read.csv*. El formato del archivo excel debe estar guardado en formato *.csv* (Figura 1). Los datos se anexan en el repositorio de Github para su uso.

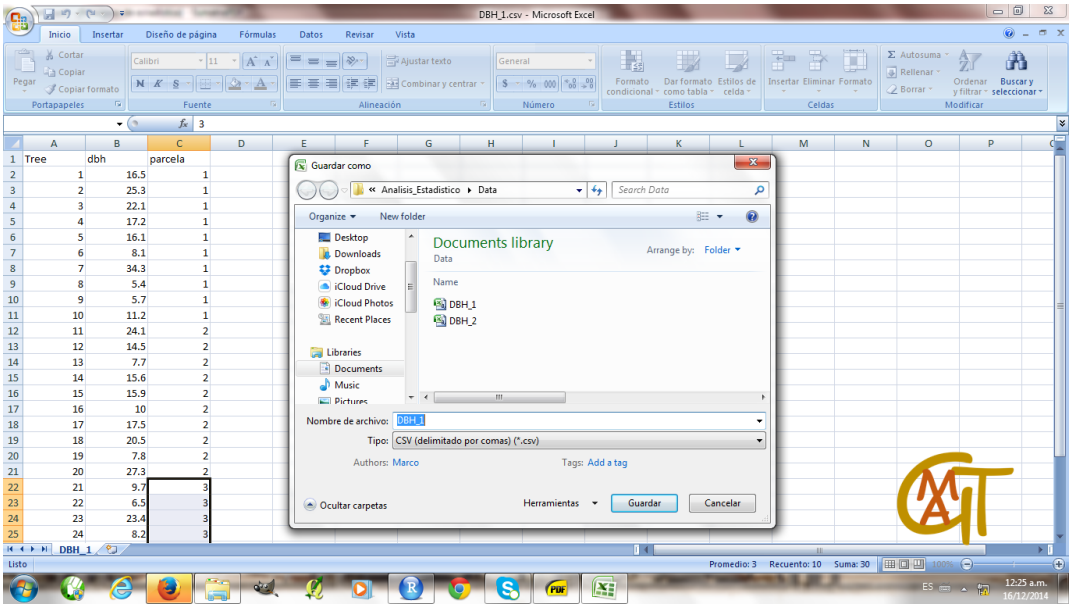

Figura 1: Guardar los datos con formato csv delimitado por comas en Excel.

Después de tener los datos guardados en el formato y nombre deseado (en este caso *DBH_1.csv*), se procede a la importación usando la función *read.csv*. Es recomendable guardar la base de datos en la misma carpeta donde se guardará el script de esta manera evitaremos tener problemas en la compilación del script. En el caso de que los datos se encuentren ubicados en otra carpeta diferente a donde se ha guardado el scprit de trabajo (suele suceder), es indispensable informar a R en que directorio se encuentra la base de datos que se desea importar. En la Figura 2 se presenta un ejemplo donde los datos se encuentran guardados en una carpeta diferente (Datos) a donde se ubica el script de trabajo (Lab_semana_4).

Para conocer en que directorio de trabajo nos encontramos podemos hacer uso de la función *getwd()* y el resultado será la ubicación actual de nuestro script de trabajo, para este ejemplo mi directorio de trabajo es /Volumes/WD Elements Marco/Cursos/GitHub/Analisis_Estadistico/Guias/Labs/Lab_semana_3. Adicionalmente podemos utilizar la función *setwd(Ubicacion_del_archivo)* para definir en que carpeta se encuentra la base de datos.

```
getwd()
```

```
## [1] "/Volumes/WD Elements Marco/Cursos/GitHub/Analisis_Estadistico/Guias/Labs/Lab_semana_3"
```

Una vez especificado el directorio donde se encuentra la base de datos, al momento de importar se debe guardar como un objeto en la memoria de R y para el siguiente ejemplo la llamaremos *trees*.

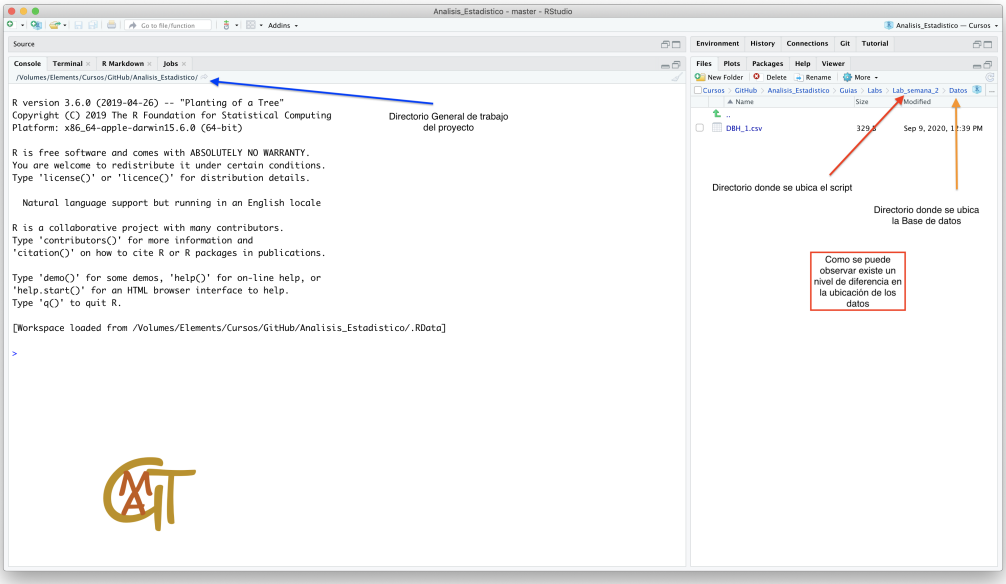

Figura 2: Ubicar directorios de trabajo de la BD y el Script.

```
trees <- read.csv("DBH_1.csv", header=TRUE)
```

finalmente para observar si los datos se encuentran en la memoria de **R** llamamos al objeto `trees`. En este ejemplo utilizamos la función `head` solo para mostrar las primeras seis filas de la base de datos.

```
head(trees)
```

```
##   Tree  dbh parcela
## 1    1 16.5       1
## 2    2 25.3       1
## 3    3 22.1       1
## 4    4 17.2       1
## 5    5 16.1       1
## 6    6  8.1       1
```

Ingresar datos directo en la consola

La manera alternativa de capturar los datos de forma manual en el programa *RStudio* de la siguiente manera: Supongamos que tenemos 30 mediciones de árboles de la especie *Pinus pseudostrobus* realizado mediante un inventario en campo, los datos son los siguientes:

Cuadro 1: Datos de *dbh* proveniente de 30 árboles inventariados de la especie *Pinus pseudostrobus* en el bosque escuela.

Trees	dbh	Trees	dbh	Trees	dbh
1	16.5	11	24.1	21	9.7
2	25.3	12	14.5	22	6.5

Trees	dbh	Trees	dbh	Trees	dbh
3	22.1	13	7.7	23	23.4
4	17.2	14	15.6	24	8.2
5	16.1	15	15.9	25	28.5
6	8.1	16	10	26	10.4
7	34.3	17	17.5	27	11.5
8	5.4	18	20.5	28	14.3
9	5.7	19	7.8	29	17.2
10	11.2	20	27.3	30	16.8

En el lenguaje R podemos ingresar este conjunto de datos de la siguiente forma:

```
dbh <- c(16.5, 25.3, 22.1, 17.2, 16.1, 8.1, 34.3, 5.4, 5.7, 11.2, 24.1,
         14.5, 7.7, 15.6, 15.9, 10, 17.5, 20.5, 7.8, 27.3,
         9.7, 6.5, 23.4, 8.2, 28.5, 10.4, 11.5, 14.3, 17.2, 16.8)
```

Almacenar y Accesar datos en la Nube

Estos servicios son Dropbox y GitHub. Aunque ambos satisfacen nuestras necesidades básicas de almacenamiento, lo hacen de diferentes maneras y requieren diferentes niveles de esfuerzo para su configuración y mantenimiento.

Todos los archivos almacenados en Dropbox tienen una dirección URL a través de la cual se puede acceder a ellos desde una computadora conectada a Internet. Los archivos de la carpeta Dropbox (Públicas o de otras carpetas no públicas) se pueden descargar en R [Gandrud].

Accesar datos de internet

Hay muchas formas de importar datos almacenados en Internet directamente en R. Tenemos que utilizar diferentes métodos dependiendo de dónde y cómo se almacenan los datos.

Datos de URL no seguras (http)

Importar datos a R que se encuentran en una URL no segura: las que comienzan con http: es sencillo siempre que:

- los datos se almacenan en un formato simple, p. ej. Texto sin formato,
- el archivo no está incrustado en un sitio web HTML más grande,

Puedes determinar si el archivo de datos está incrustado en un sitio web abriendo la URL en su navegador web. Si solo ve los datos sin formato de texto, probablemente esté listo para descargarse Para importar los datos, simplemente incluya la URL como el nombre del archivo en su comando read.csv (cuando los datos esten en formato indicado). Vamos a realziar un ejemplo con los datos disponibles en em portal de acceso de datos del gobierno de México: https://datos.gob.mx/busca/dataset/acciones-de-inspeccion-en-materia-forestal. El link de descarga de los datos es el siguiente: http://www.profepa.gob.mx/innovaportal/file/7635/1/accionesInspeccionfoanp.csv. Los datos los vamos a descargar en un objeto llamado Profepa. Los datos corresponden a las Acciones de Inspección y Vigilancia a los Recursos Forestales en ANPs.

```
prof_url <- "http://www.profepa.gob.mx/innovaportal/file/7635/1/accionesInspeccionfoanp.csv"
profepa <- read.csv(prof_url)
head(profepa)
```

```
##               Entidad Inspecciones Recorridos.de.vigilancia Operativos  X
## 1      Aguascalientes            2                        2          0 NA
## 2     Baja California            0                        5          1 NA
## 3 Baja California Sur            1                        0          1 NA
## 4            Campeche            1                        2          2 NA
## 5             Chiapas            1                        1          0 NA
## 6           Chihuahua           14                        8          1 NA
```

Muchas de las veces, los links a los archivos en internet son muy largos, alternativamente podemos seccionar este link de la siguiente manera, siempre procurando dejar la barra / al final de cada línea para indicar hasta donde llega la carpeta y después continuar con el link:

```
prof_url_2 <- paste0("http://www.profepa.gob.mx/innovaportal/",
                     "file/7635/1/accionesInspeccionfoanp.csv")
profepa2 <- read.csv(prof_url_2)
head(profepa2)
```

Datos de URL seguras (https): Dropbox y Github

El almacenamiento de datos en URL no seguras es cada vez menos común. Servicios como Dropbox y GitHub ahora almacenan sus datos en URL seguras. Puedes saber si los datos están almacenados en una dirección web segura si comienzan con *https* en lugar de *http.* Tenemos que usar diferentes comandos para descargar datos de URL seguras. Veamos el método para descargar datos en R: source_data, que se encuentra en el paquete repmis Miscellaneous Tools for Reproducible Research [Gandrud, 2016].

Los archivos almacenados en carpetas no públicas de Dropbox son un poco más complicados de descargar. Si uno va al sitio web de Dropbox y hace clic en el botón Copiar vínculo de Dropbox, se copiara al portapales el link. Esta información contiene el nombre del archivo + una llave para su acceso. Veamos el siguiente ejemplo: https://www.dropbox.com/s/hmsf07bbayxv6m3/cuadro1.csv?dl=0, Por default Dropbox añade un cero al final del link y siempre tendremos que sustituirlo por el número 1: https://www.dropbox.com/s/hmsf07bbayxv6m3/cuadro1.csv?dl=1. Puede ver que la última parte de la URL (cuadro1.csv) es el nombre del archivo de datos. La clave es la cadena de letras y números que le sigue al siguiente texto https://www.dropbox.com/s/ ... *hmsf07bbayxv6m3*. Los datos se encuentran en mi Dropbox personal y provienen de un inventario recopilado del libro de Introductory probability and Statistics [Kozak et al., 2008].

```
library(repmis) # No olvidar cargar la paquetería
conjunto <- source_data("https://www.dropbox.com/s/hmsf07bbayxv6m3/cuadro1.csv?dl=1")
```

```
## Downloading data from: https://www.dropbox.com/s/hmsf07bbayxv6m3/cuadro1.csv?dl=1
```

```
## SHA-1 hash of the downloaded data file is:
## 2bdde4663f51aa4198b04a248715d0d93498e7ba
```

```
head(conjunto) # muestra las primeras seis filas de la BD
```

```
##   Arbol Fecha Especie Clase Vecinos Diametro Altura
```

```
## 1     1    12       F     C       4     15.3  14.78
## 2     2    12       F     D       3     17.8  17.07
## 3     3     9       C     D       5     18.2  18.28
## 4     4     9       H     S       4      9.7   8.79
## 5     5     7       H     I       6     10.8  10.18
## 6     6    10       C     I       3     14.1  14.90
```

Para descargar los datos de Github podemos utilizar la paquetería *readr* y su función *read_csv*:

```
library(readr)
file <- paste0("https://raw.githubusercontent.com/mgtagle/",
               "202_Analisis_Estadistico_2020/master/cuadro1.csv")
inventario <- read_csv(file)
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   Arbol = col_double(),
##   Fecha = col_double(),
##   Especie = col_character(),
##   Clase = col_character(),
##   Vecinos = col_double(),
##   Diametro = col_double(),
##   Altura = col_double()
## )
```

```
head(inventario)
```

```
## # A tibble: 6 x 7
##   Arbol Fecha Especie Clase Vecinos Diametro Altura
##   <dbl> <dbl> <chr>   <chr>   <dbl>    <dbl>  <dbl>
## 1     1    12 F       C           4     15.3  14.8
## 2     2    12 F       D           3     17.8  17.1
## 3     3     9 C       D           5     18.2  18.3
## 4     4     9 H       S           4      9.7   8.79
## 5     5     7 H       I           6     10.8  10.2
## 6     6    10 C       I           3     14.1  14.9
```

EL link se puede obtener de la como se muestra en las siguientes figuras (3, 4).

El URl del archivo en formato raw se puede copiar de la barra de direcciones y llevar al script.

Parte 2: Operaciones con la base de datos

De la misma manera podemos trabajar los datos como los introducidos en el cuadro 1. Siempre al realizar una operación con las variables que contiene la base de datos debemos utilizar el nombre del objeto en este caso `trees`. Como primer ejemplo determinaremos la media y la desviación estándar de la variable `dbh` de la base de datos `trees`. Este conjunto cuenta con 3 columnas (Tree, dbh, parcela). Para poder tener acceso a cada columna debemos siempre especificar la base de datos seguido por $ y el nombre de la columna a trabajar.

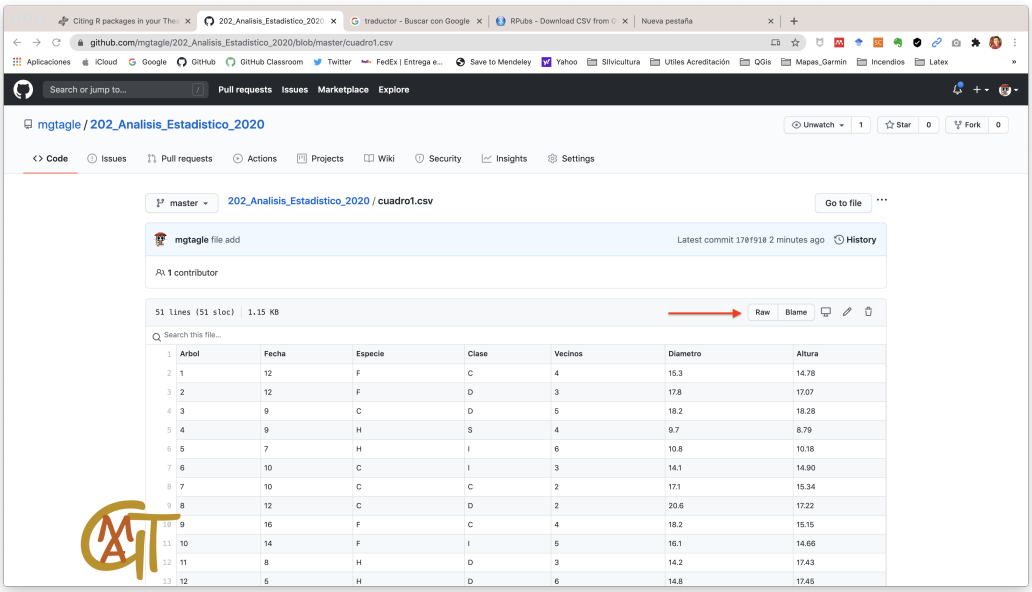

Figura 3: Ubicar archivo a descargar en Github y dar click en RAW para obtener la url.

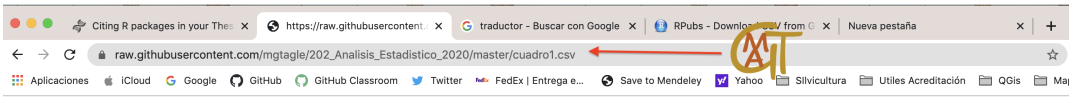

Figura 4: EL url del archivo se puede copiar y llevar al script.

```
mean(trees$dbh) # El signo de $ informa que necesitamos la columna dbh
```

```
## [1] 15.64333
```

```
sd(trees$dbh)
```

```
## [1] 7.448892
```

Selección mediante restricciones

El uso de restricciones sirve para obtener una muestra bajo ciertas condiciones restrictivas de la base de datos `trees` y es por ejemplo, una operación útil en el manejo de información de los inventarios forestales.

Los condicionantes restrictivas más empleadas son:

```
igual o mayor (>=), mayor que (>), igual que (==)
igual o menor (<=), menor que (<), no igual (!=)
```

Para mostrar como funcionan las restricciones podemos realizar las siguientes preguntas: ¿Cuántos individuos tiene un diámetro menor (<) a 10 cm?

```
# Indica la sumatoria de los individuos en el objeto tree con un dbh < a 10
sum(trees$dbh < 10)
```

```
## [1] 8
```

También es interesante saber cuales son los individuos que son inferiores al diámetro (dbh< 10 cm). Para esto, hacemos uso de la función *which* que no regresara cuales individuos son los que posen tal restricción.

```
which(trees$dbh < 10)
```

```
## [1]  6  8  9 13 19 21 22 24
```

Otro ejemplo de la utilidad de emplear las condicionantes es: Excluir los diámetros que se encuentran en la parcela 2. El objeto resultante se puede grabar como `trees.13`. El símbolo ! indica NO.

```
trees.13 <- trees[!(trees$parcela=="2"),]
trees.13
```

```
##    Tree  dbh parcela
## 1     1 16.5       1
## 2     2 25.3       1
## 3     3 22.1       1
## 4     4 17.2       1
## 5     5 16.1       1
## 6     6  8.1       1
## 7     7 34.3       1
## 8     8  5.4       1
## 9     9  5.7       1
## 10   10 11.2       1
## 21   21  9.7       3
## 22   22  6.5       3
## 23   23 23.4       3
## 24   24  8.2       3
## 25   25 28.5       3
## 26   26 10.4       3
## 27   27 11.5       3
## 28   28 14.3       3
## 29   29 17.2       3
## 30   30 16.8       3
```

Selección de una submuestra

Una submuestra se puede obtener de cualquier base de datos que este disponible en R mediante la función `subset` [Crawley, 2007]. Por ejemplo queremos obtener solo los diámetros iguales o menores a 10 cm y deseamos guardarla en un objeto que se denominará `trees.1`.

```
trees.1 <- subset(trees, dbh <= 10)
head(trees.1)
```

```
##    Tree  dbh parcela
## 6     6  8.1       1
## 8     8  5.4       1
## 9     9  5.7       1
## 13   13  7.7       2
## 16   16 10.0       2
```

```
## 19   19  7.8       2
```

El objeto *trees.1* contiene 9 observaciones obtenidas mediante la restricción (*subset(trees, dbh <= 10)*) de los diámetros iguales o menores a 10 cm de la base de datos original *trees*.

Ahora tenemos dos base de datos: `trees` y `trees.1` disponibles en R y con ambas podemos realizar las operaciones y gráficas que veremos en la siguiente clase (Figura 5).

```
mean(trees$dbh)
```

```
## [1] 15.64333
```

```
mean(trees.1$dbh)
```

```
## [1] 7.677778
```

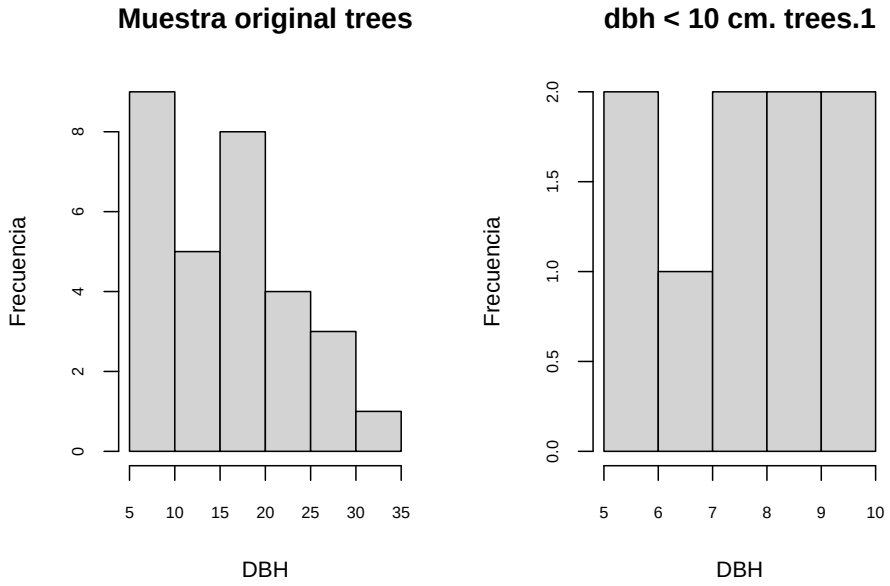

Figura 5: Histogramas de la muestra y submuestra.

Parte 3 Representación gráfica

Histogramas

Un histograma es una representación visual de la distribución de un conjunto de datos. Como tal, la forma de un histograma es su característica más evidente e informativa: le permite ver fácilmente dónde se encuentra una cantidad relativamente grande de datos y dónde hay muy pocos datos. En otras palabras, puede ver dónde está el medio en su distribución de datos, qué tan cerca están los datos alrededor de este medio y dónde se pueden encontrar posibles valores atípicos. Debido a todo esto, los histogramas son una excelente manera de conocer sus datos.

```
mamiferos <- read.csv("https://www.openintro.org/data/csv/mammals.csv")
```

Cuadro 2: Representación parcial de los datos de mamíferos y su comportamiento durante el sueño. Datos tomados de la fuente OpenIntro: t.ly/Yr5I.

	species	body_wt	brain_wt	total_sleep	life_span
2	Africangiantpouchedrat	1.000	6.6	8.3	4.5
5	Asianelephant	2547.000	4603.0	3.9	69.0
6	Baboon	10.550	179.5	9.8	27.0
7	Bigbrownbat	0.023	0.3	19.7	19.0
8	Braziliantapir	160.000	169.0	6.2	30.4
9	Cat	3.300	25.6	14.5	28.0

Por el momento trabajaremos con la variable total_sleep para generar el histograma:

```
hist(mamifero$total_sleep)
```

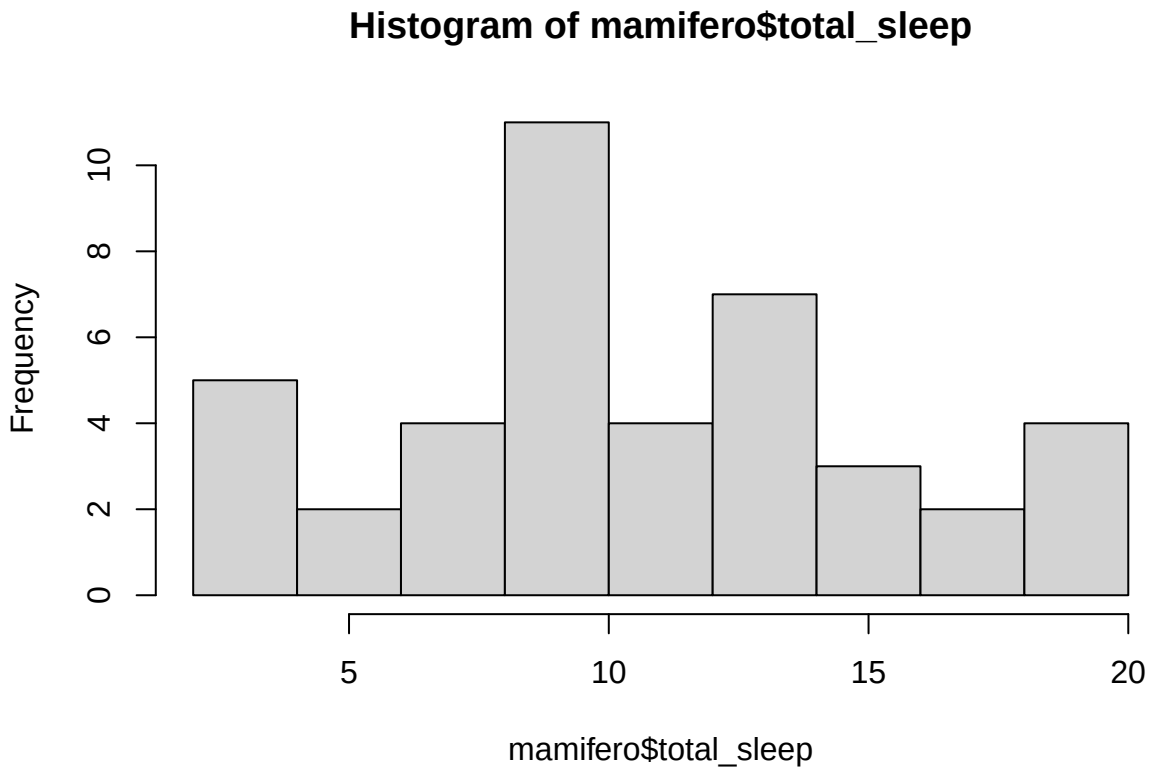

Figura 6: Histograma de la variable total_sleep.

Como puedes observar en la figura 6, el título de la figura, así como el nombre del eje de x & y se dan por default, por lo tanto tendremos que trabajar un poco más el código para obtener un histograma más presentable (Figura 7).

```
hist(mamifero$total_sleep,  # Datos
     xlim = c(0,20), ylim = c(0,14), # Cambiar los límites de x & y
     main = "Total de horas sueño de las 39 especies", # Cambiar el título
     xlab = "Horas sueño", # Cambiar eje de las x
     ylab = "Frecuencia", # Cambiar eje de las y
```

```
     las = 1, # Cambiar orientación de y
     col = "#996600") # Cambiar color de las barras
```

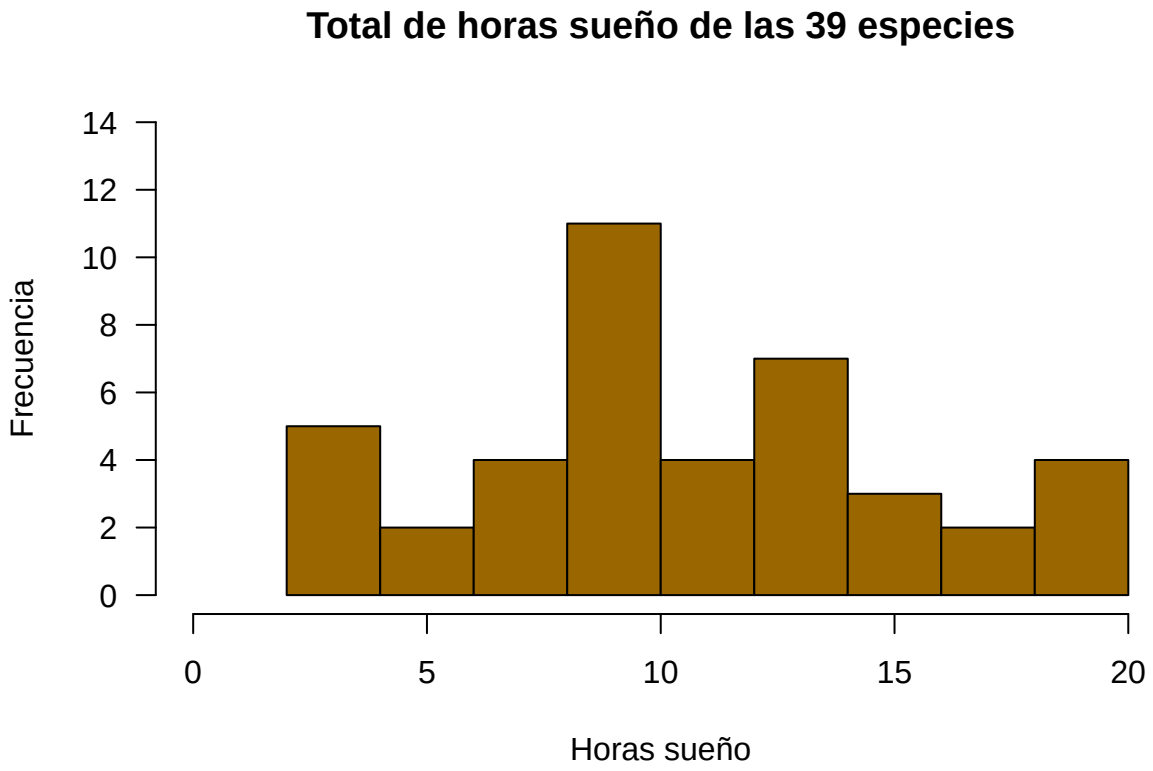

Figura 7: Histograma modificado para su mejor presentación

Barplot o gráfico de barras

Un diagrama de barras (o gráfico de barras) es uno de los tipos de gráficos más comunes. Muestra la relación entre una variable numérica y una categórica. Cada entidad de la variable categórica se representa como una barra. El tamaño de la barra representa su valor numérico.

```
data("chickwts")
head(chickwts[c(1:2,42:43, 62:64), ])
```

```
##    weight      feed
## 1     179 horsebean
## 2     160 horsebean
## 42    226 sunflower
## 43    320 sunflower
## 62    379    casein
## 63    260    casein
```

Primeramente tendremos que acomodar los datos en columnas (los datos originales estan acomodados en dos columnas (weight, feed, Peso y tipo de alimentación de los pollos.)

```
feeds <- table(chickwts$feed)
feeds
```

```
##
##    casein horsebean   linseed  meatmeal   soybean sunflower
##        12        10        12        11        14        12
```

Los datos fueron acomodados en una tabla de frecuencias (Tipo de alimentación y la cantidad de pollos alimentados por cada tipo). Como podemos observar en la Figura 8 los tipos de alimentación están ordenados alfabeticamente.

```
barplot(feeds)
```

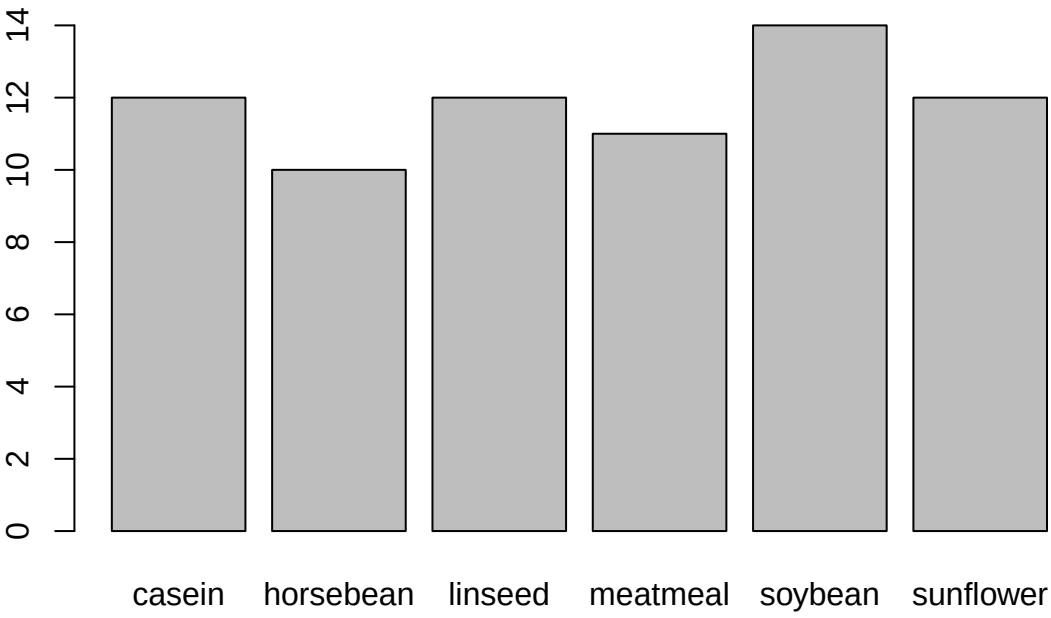

Figura 8: Barplot de los tipos de alimentación

Para ordenar de forma decreciente las barras podemos personalizar como sigue (Figura 9):

```
barplot(feeds[order(feeds, decreasing = TRUE)])
```

Para mejora la presentación de nuestra gráfica podemos realizar las operaciones anteriores y lograr personalizarla. Ahora te toca a tí, tu gráfica debería verse ahora como la Figura 10. Buena suerte.

Puedes apoyarte en los siguientes recursos:

- w3schools (15.09.2020). HTML Color Picker. [Arcivo de internet]. Retrived from t.ly/53Ac.
- Michael Butler's Elementary Statistics (15.09.2020). How To Graph in RStudio: The Basics [Archivo de video]. Retrived from: t.ly/Bn4f.
- MarinStatsLectures-R Programming & Statistics (15.09.2020). Bar Charts and Pie Charts in R | R Tutorial 2.1 | MarinStatsLectures. [Archivo de video]. Retrived from: t.ly/IA1f.

Referencias

Michael J. Crawley. *The R Book*. Wiley Publishing, 1st edition, 2007. ISBN 0470510242, 9780470510247.

Christopher Gandrud. Reproducible research with r and rstudio.

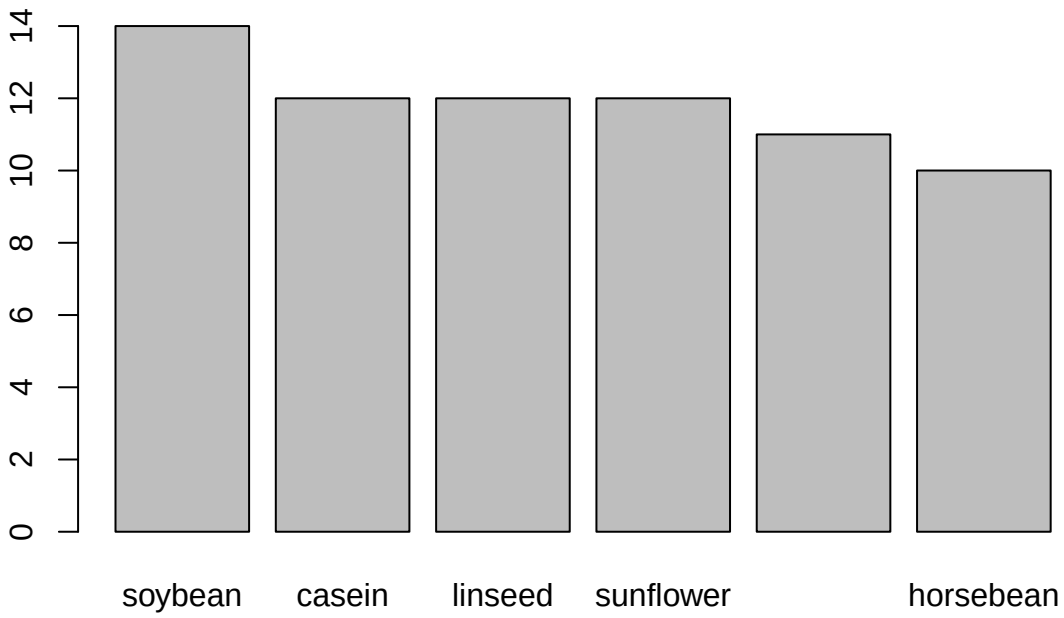

Figura 9: Barplot ordenado de forma decreciente.

Christopher Gandrud. *repmis: Miscellaneous Tools for Reproducible Research*, 2016. URL https://CRAN.R-project.org/package=repmis. R package version 0.5.

A Kozak, RA Kozak, CL Staudhammer, and SB Watte. *Introductory probability & Statistics. Applications for forestry & the natural sciences.* Cambridge: Cambridge University Press, 2008.

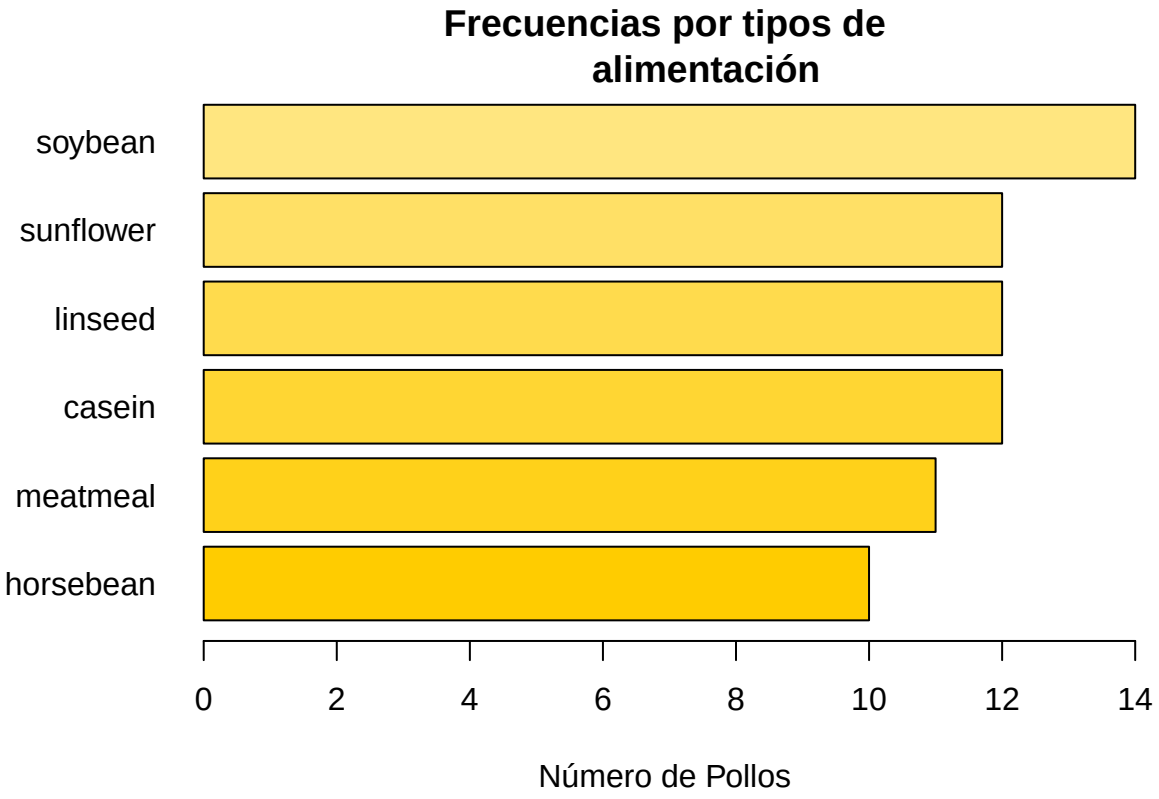

Figura 10: Barplot modificado para una mejor rpesentación.