Ejercicios

Vectores y tipos de datos básicos

1

a) Crea un vector numérico que vaya de 1 a 100 en intervalos de 0.1.

b) Usa la función "which" para saber qué elementos del vector que has creado son mayores de 90, y almacena ese resultado.

c) Crea un nuevo vector que almacene los valores del primer vector que son mayores de 90.

```
# a)
v1 <- seq(1,100,0.1)

# b)
v2 <- which(v1 > 90)

# c)
# equivalentes
v3.1 <- v1[v2]
v3.2 <- v1[which(v1 > 90)]
v3.3 <- v1[v1>90]
```

2

a) La función "runif" genera vectores de números aleatorios entre un número mínimo y un máximo. Comprueba la ayuda de R para esta función (verás que hay otras funciones en su misma página de ayuda, mira las secciones "usage" para ver su sintaxis, y "arguments" para ver sus argumentos).

b) Usa la función runif para almacenar un vector con 50 elementos aleatorios entre 0 y 10.

c) Obtén la media y la desviación típica de esos elementos.

d) Crea un nuevo vector que sólo contenga los elementos del primero que sean menores o iguales a 5 (usa la función "which" como en el primer ejercicio).

```
# a)
?runif

# b)
v4 <- runif(n = 50,min = 0,max = 10)

# c)
media.v4 <- mean(v4)
desv.v4 <- sd(v4)

# d)
v5 <- v4[which(v4 <= 5)]
```

Estructuras de datos

3

Crea un dataframe de 1000 filas en el que la primera columna sea una columna numérica que siga una distribución normal con media 0 y desviación típica 1. La segunda columna debe ser una distribución aleatoria de las letras del alfabeto. Pista: usa las funciones "rnorm" y "sample", y el vector "letters", que viene por defecto en R.

```
# número de filas
filas <- 1000

d1 <- data.frame(c1 = rnorm(n = filas,mean = 0,sd = 1),
                 c2 = sample(x = letters,size = filas,replace = TRUE))
```

4

En el dataframe del ejercicio anterior, sustituye todas las letras "c" por "k". Pista: recuerda la función "which".

```
# copiamos el original
d2 <- d1
# sustituimos
d2$c2[which(d2$c2 == "c")] <- "k"
# muestra la frecuencia de cada letra
table(d2$c2)
```

Estructuras de control

5

a) Crea una matriz de 100x3 en la que cada columna esté formada por números aleatorios, la primera entre 0-1, la segunda entre 0-10, la tercera entre 0-100. pista: `?runif`

b) Utilizando bucles o funciones de la familia apply, calcula la media y la desviación típica de cada columna.

c) Guarda los resultados en un dataframe con tres filas y, al menos, una columna "media" y una columna "desviacion.tipica".

```
# número de filas,
# así evitamos tener que repetirlo al crear la matriz
filas <- 100

# concatenamos las tres llamadas a la función runif
# fijaos en el resultado de usar el argumento "byrow" como TRUE o FALSE
m1 <- matrix(data = c(runif(filas,0,1),
                      runif(filas,0,10),
                      runif(filas,0,100)),
             nrow = filas,
             byrow = FALSE)

# b)
# bucles
# número de columnas
columnas <- ncol(m1)

# vectores de resultados
```

```r
media <- numeric(columnas)
desv <- numeric(columnas)

# bucle for iterando sobre las columnas
for(i in 1:columnas){
  media[i] <- mean(m1[,i])
  desv[i] <- sd(m1[,i])
}

# apply
media.apply <- apply(m1,2,mean)
desv.apply <- apply(m1,2,sd)

# c)
m1.df <- data.frame(media = media,desviacion.tipica = desv)
```

6

Usando el dataset "iris":

a) almacena en un vector el valor medio de cada columna

b) almacena en otro vector el número de valores únicos de cada columna pista: usa la función n_distinct del paquete tidyverse

```r
# cargamos el paquete tidyverse
library(tidyverse)

# creamos los vectores de resultados
vector.medias <- numeric(length = length(iris))
vector.unicos <- numeric(length = length(iris))

# bucle que vaya columna a columna
for(i in 1:length(iris)){

  # calculamos la media sólo si la columna i es numérica
  if(class(iris[,i]) == "numeric"){
    vector.medias[i] <- mean(iris[,i])
  }
  # la función n_distinct es válida también con caracteres
  vector.unicos[i] <- n_distinct(iris[,i])
}

# alternativa con la familia apply
# sapply aplica una función a cada columna de un dataframe
medias.2 <- sapply(iris,mean)
unicos.2 <- sapply(iris,n_distinct)
```

7

Usando el dataset "sample500tuits_starwarsandaluz.txt":

a) crea un dataframe con dos columnas: num.caracteres y num.palabras, que almacene el número de caracteres y el número de palabras de cada tuit

b) muestra por pantalla el número medio de caracteres y palabras de los datos

c) ¿cómo se harían estas operaciones sin usar bucles?

```r
# cargamos el paquete stringr
library(stringr)

# leemos el archivo
my.file <- "data/sample500tuits_starwarsandaluz.txt"
conn <- file(my.file,open = "r")
lineas <-readLines(conn)

# creamos el dataframe de resultados
# ya con la longitud adecuada (tantas filas como tuits hay en el vector lineas)
resultados <- data.frame(num.caracteres = numeric(length(lineas)),
                         num.palabras = numeric(length(lineas)))

# vamos fila a fila calculando las dos columnas
# la función para contar el número de palabras viene de stackoverflow
# https://stackoverflow.com/questions/8920145/
# count-the-number-of-all-words-in-a-string
for(i in 1:length(lineas)){
  resultados$num.caracteres[i] <- nchar(lineas[i])
  resultados$num.palabras[i] <- str_count(lineas[i], '\\w+')
}

# mostramos los valores medios en pantalla
cat("en el archivo: ",my.file, "\nhay",length(lineas),
    "tuits, con una media por tuit de\n",
    mean(resultados$num.caracteres),
    "caracteres\n",mean(resultados$num.palabras),"palabras")

# estas operaciones se pueden hacer en una sola linea,
# aprovechando la vectorización de R
caracteres2 <- nchar(lineas)
palabras2 <- str_count(lineas,'\\w+')
```

8

Crea un script que, usando la carpeta "data":

a) muestre por pantalla todos los archivos con extensión .csv que hay en esa carpeta

b) lea los archivos cuyo nombre contenga "starwars_personajes", y almacene en un dataframe tres campos:

- el nombre y ruta del archivo
- el número de filas
- el número de columnas

```r
# ruta a la carpeta que quiero evaluar
mi.carpeta <- "/home/david/Work/Projects/R_courses/CEA2020_Intro/data/"

# recupero los archivos que hay en esa carpeta
mis.archivos <- list.files(mi.carpeta)

# los muestro por pantalla
cat("en la carpeta:",mi.carpeta,
    "\nhay",length(mis.archivos),"archivos:\n")
cat(mis.archivos,sep = "\n")
```

```r
# sólo con extensión csv
mis.archivos.csv <- mis.archivos[grep(pattern = ".csv",x = mis.archivos)]

# los muestro por pantalla
cat("en la carpeta:",mi.carpeta,
    "\nhay",length(mis.archivos),"archivos con extensión .csv:\n")
cat(mis.archivos.csv,sep = "\n")

# selecciono los archivos que me interesan
archivos.sw <- mis.archivos[grep("starwars_personajes",mis.archivos)]

# creo el dataframe de resultados,
# con un número de filas igual al
# número de archivos que tengo (length(archivos.sw))
resultados.archivos <- data.frame(ruta = character(length(archivos.sw)),
                                  nombre = character(length(archivos.sw)),
                                  num.filas = numeric(length(archivos.sw)),
                                  num.columnas = numeric(length(archivos.sw)),
                                  stringsAsFactors = FALSE)

# leo archivo a archivo
# tened en cuenta que repetir la misma operación para abrir archivos
# solo funciona en todos si todos tienen la misma estructura,
# es decir, el mismo caracter para separar columnas (;) y punto decimal
# por eso este ejemplo está restringido a estos tres archivos solamente.
for(i in 1:length(archivos.sw)){
  # leo el archivo pasando al argumento "file" la ruta completa:carpeta+nombre
  mi.archivo <- read.csv2(file = paste(mi.carpeta,archivos.sw[i],sep = ""),
                          stringsAsFactors = FALSE)

  # almaceno los resultados en el dataframe directamente
  # la carpeta origen es siempre la misma
  resultados.archivos$ruta[i] <- mi.carpeta
  # nombre
  resultados.archivos$nombre[i] <- archivos.sw[i]
  # num filas
  resultados.archivos$num.filas[i] <- nrow(mi.archivo)
  # num columnas
  resultados.archivos$num.columnas[i] <- length(mi.archivo)
}
```

Tratamiento de caracteres

9

Usando el dataset "sample500tuits_starwarsandaluz.txt":

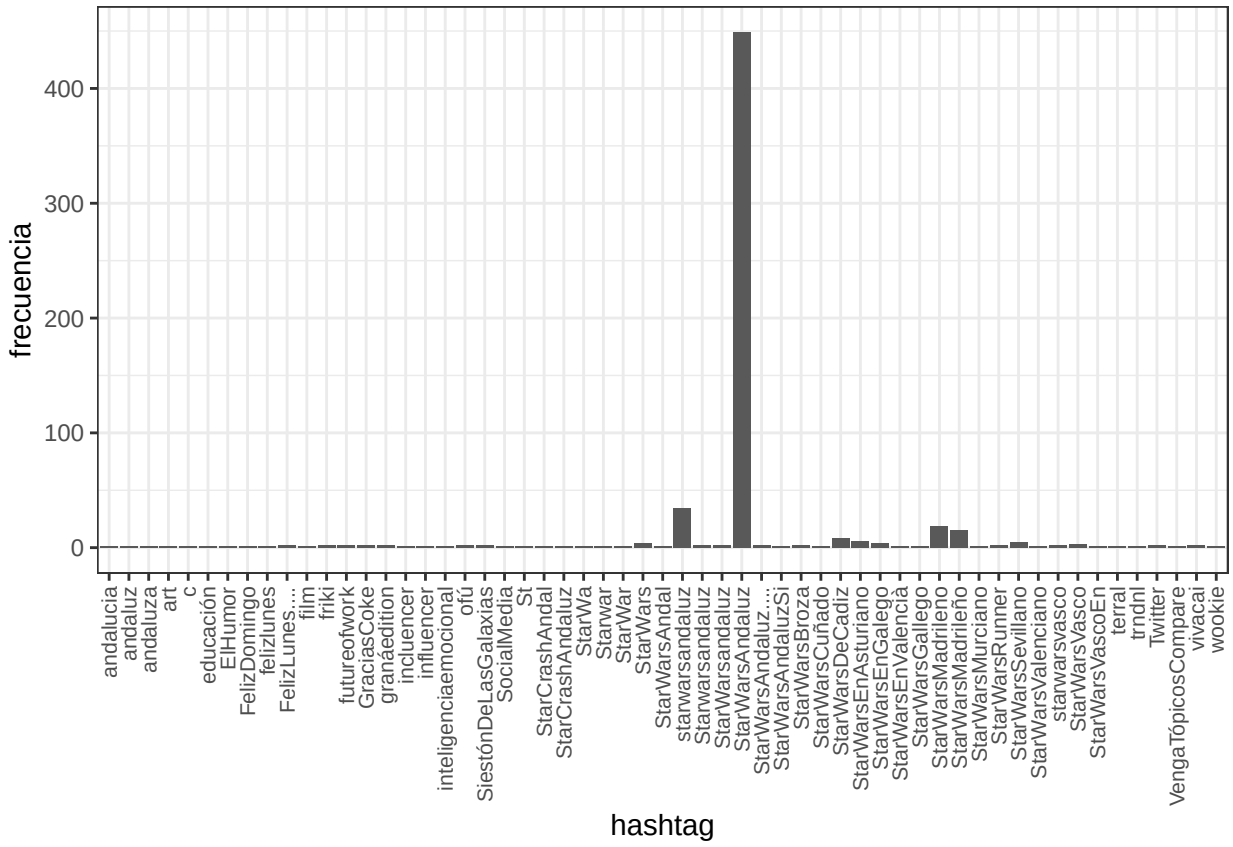
a) Extrae el número de menciones de usuarios para cada tuit, y el valor medio.

b) ¿Qué hashtags aparecen y con qué frecuencia?

```r
# para que entendáis que stackoverflow es el mejor amigo de todo
# usuario de R...
# ¿qué preguntas he usado para ayudarme con este script?
```

```r
# https://stackoverflow.com/questions/16816032/convert-named-character-vector-to-data-frame
# https://stackoverflow.com/questions/7597559/grep-using-a-character-vector-with-multiple-patterns
# https://stackoverflow.com/questions/55171086/r-regex-extract-words-beginning-with-symbol
# https://stackoverflow.com/questions/12626637/read-a-text-file-in-r-line-by-line
# https://stackoverflow.com/questions/10294284/remove-all-special-characters-from-a-string-in-r

library(tidyverse) # includes the package stringr

# leemos un archivo linea a linea
my.file <- "/home/david/Work/Projects/R_courses/CEA2020_Intro/data/sample500tuits_starwarsandaluz.txt"
conn <- file(my.file,open = "r")
lineas <- readLines(conn, encoding = "UTF-8")

# lo pasamos de vector a dataframe
df.lineas <- as.data.frame(lineas)

# buscamos los caracteres "RT" o "Retweeted" para saber qué tuits
# son retuits
is.RT <- grep("RT|Retweeted",df.lineas$lineas)

# creamos una columna "is.RT" que en principio es FALSE para todos
df.lineas$is.RT <- FALSE
# y le damos el valor TRUE a las posiciones identificadas con "grep"
df.lineas$is.RT[is.RT] <- TRUE

# ¿cuántos usuarios aparecen en cada tuit?
# primero, extraemos una lista con todos los usuarios en cada tuit
# esta orden extrae todas las palabras que empiecen por @
users.mentioned <- str_extract_all(df.lineas$lineas, "(?<=^|\\s)@[^\\s]+")
# cuántos elementos tiene cada posición de la lista
users.mentioned2 <- lapply(users.mentioned,FUN = length)
# convertimos esto a un vector
num.users <- unlist(users.mentioned2)
# y este número lo pasamos a una nueva columna del dataframe
df.lineas$users.mentioned <- num.users

cat("el número medio de menciones/usuarios por tuit es",
    mean(df.lineas$users.mentioned,na.rm = TRUE))

# ¿qué hashtags aparecen y con qué frecuencia?
# extraemos todas las palabras que comiencen por @
hashtags <- str_extract_all(df.lineas$lineas, "(?<=^|\\s)#[^\\s]+")
# limpiamos los signos de puntuación (esto elimina también la # al comienzo)
clean.hashtags <- lapply(hashtags,
                         FUN = function(x)str_replace_all(x, "[[:punct:]]", ""))
# convertimos la lista en un vector
clean.hashtags.vector <- unlist(clean.hashtags)
# extraemos las frecuencias usando la función "table" y convirtiéndolo en
# un dataframe
hashtag.freq <- as.data.frame(table(clean.hashtags.vector))
# renombramos las columnas del dataframe
names(hashtag.freq) <- c("hashtag","frecuencia")
```

```r
# creamos un gráfico de barras
freq.plot <- ggplot(hashtag.freq, aes(x = hashtag, y = frecuencia)) +
  geom_col() +
  theme_bw() +
  # para retocar el eje x he mirado en
  # http://www.cookbook-r.com/Graphs/Axes_(ggplot2)/
  theme(axis.text.x  = element_text(angle=90, vjust=0.5, size=8, hjust = 1))
freq.plot
```



Visualización

10

Usando el dataset "iris"

a) crea un gráfico de puntos donde el eje y sea la columna "sepal.length" y el eje x la columna "petal.length"

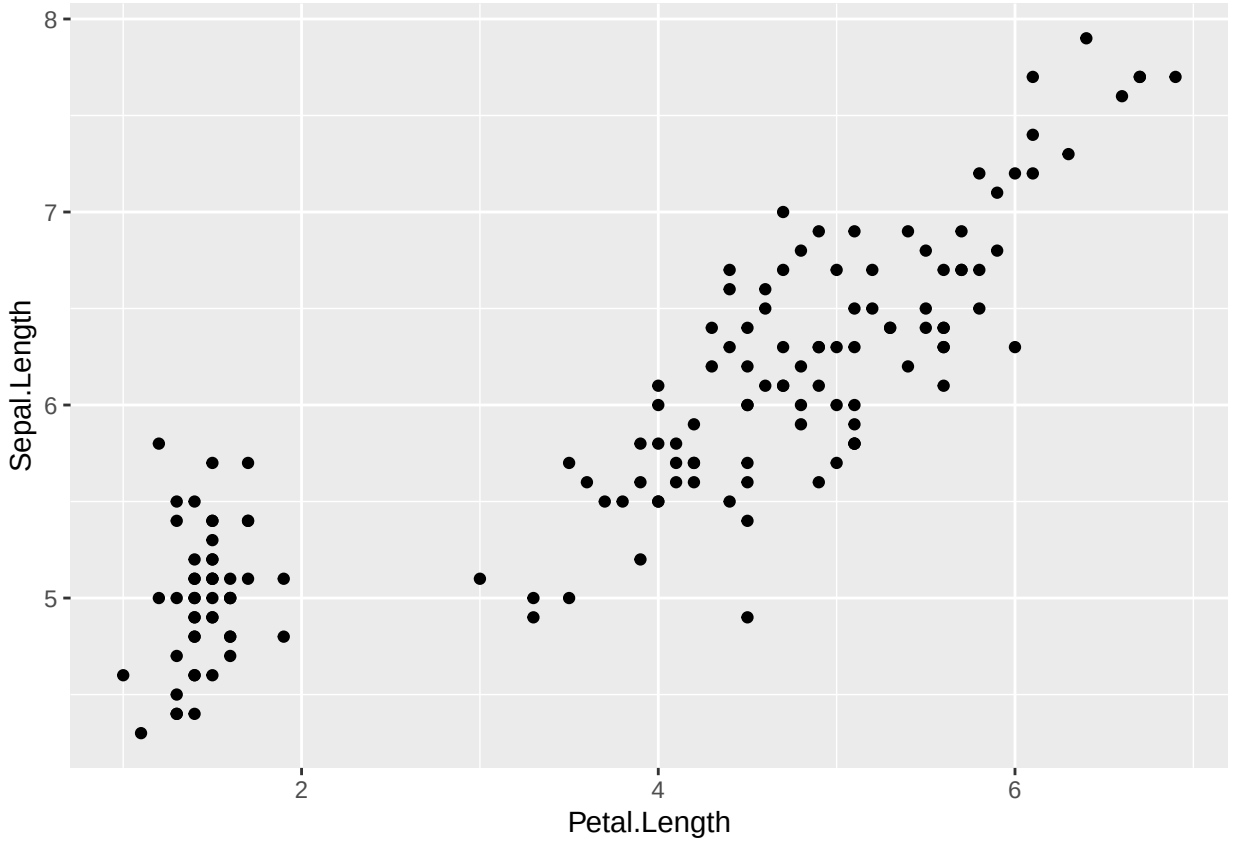
b) diferencia cada punto según la especie (por color y por forma de punto)

c) cambia el tema de la figura, los colores asociados a cada especies y los nombres de los ejes y leyenda

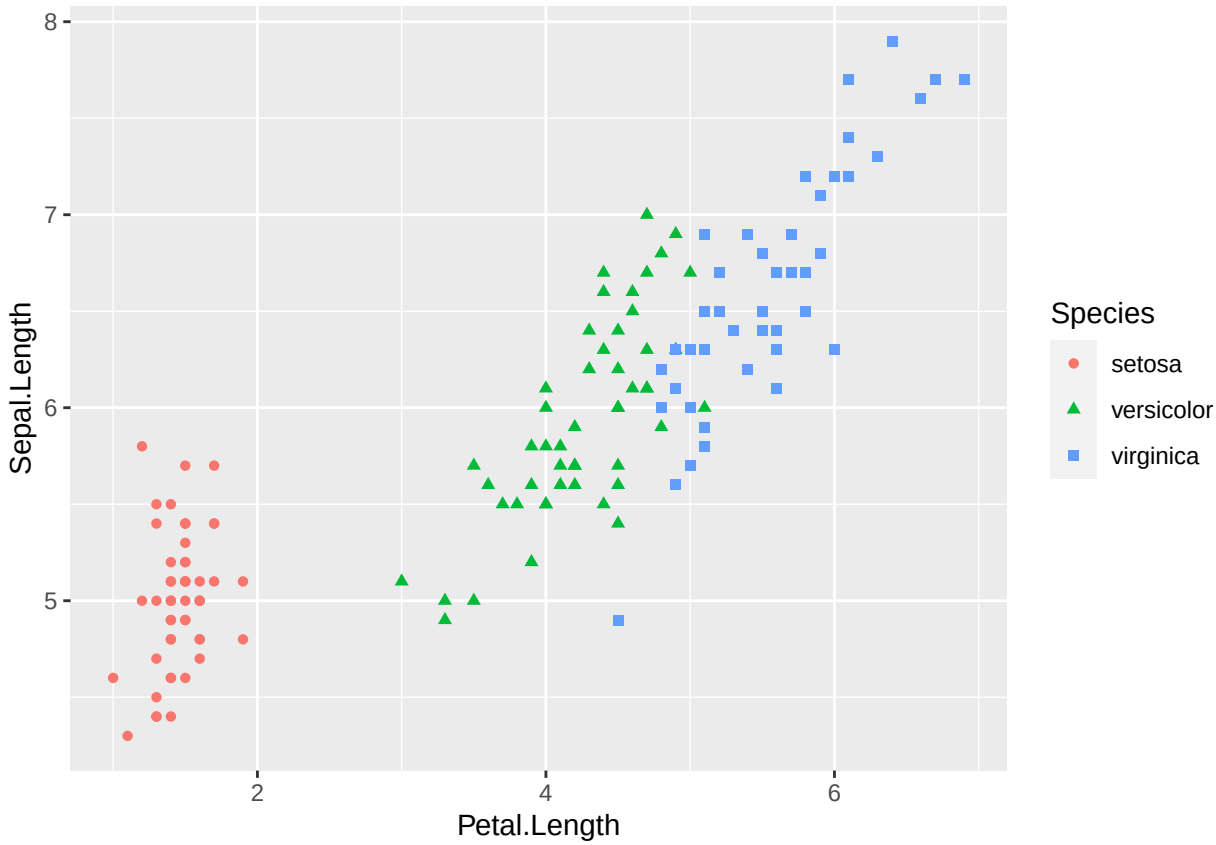
d) crea un gráfico de cajas (boxplot) que muestre la variación entre especies de todas sus medidas, separando por paneles cada medida

```r
# cargamos el paquete tidyverse, que incluye ggplot2
library(tidyverse)

# a)
```

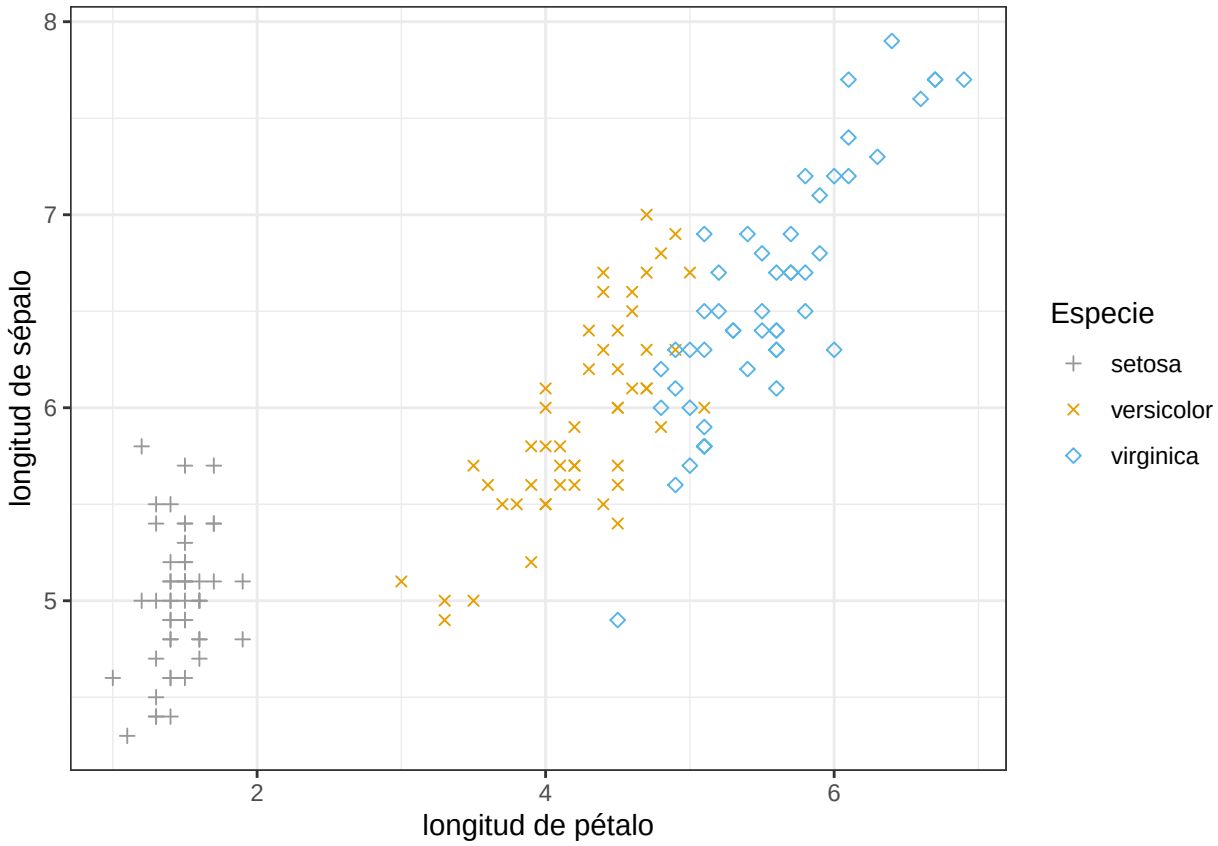
```r
fig1 <- ggplot(data = iris,aes(x = Petal.Length, y = Sepal.Length)) +
  geom_point()
fig1
```



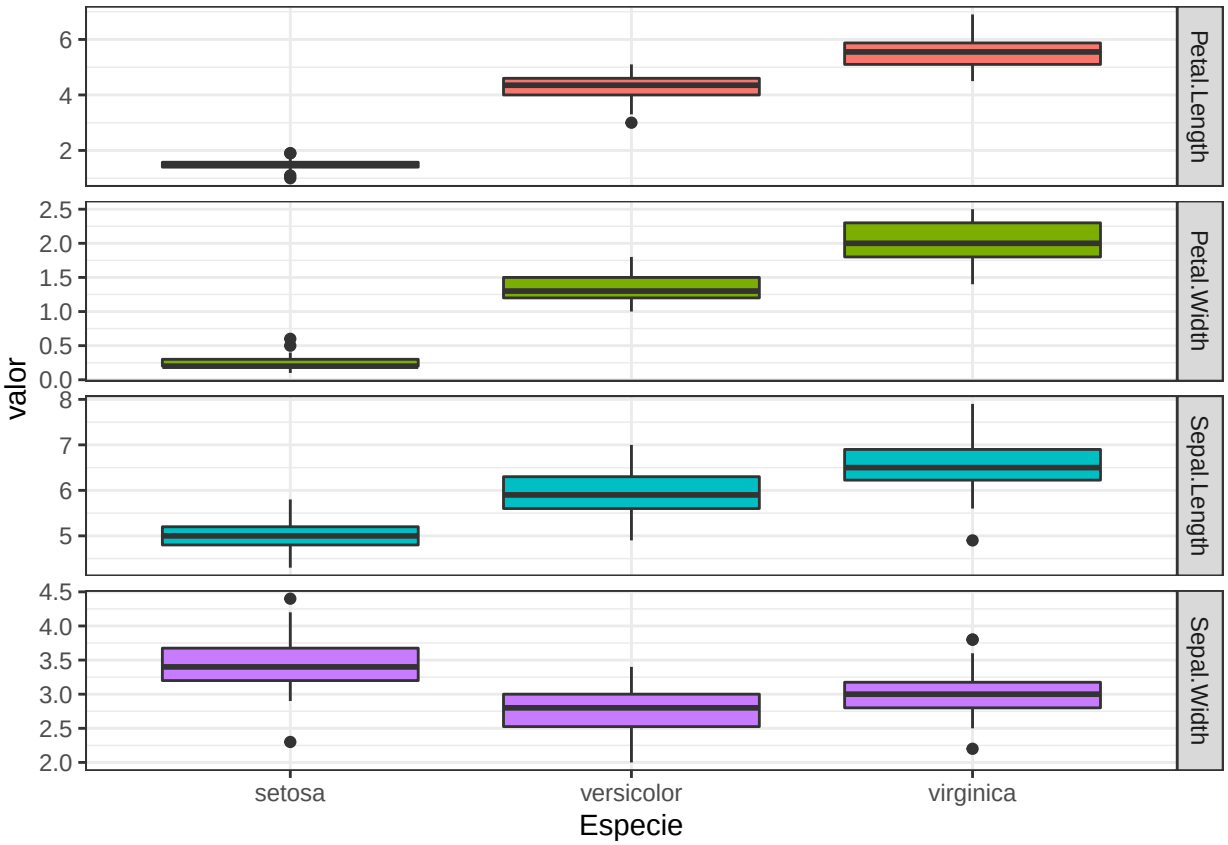

```r
# b)
fig2 <- ggplot(data = iris,aes(x = Petal.Length, y = Sepal.Length)) +
  geom_point(aes(color = Species, shape = Species))
fig2
```




```
# c)
fig3 <- ggplot(data = iris,aes(x = Petal.Length, y = Sepal.Length)) +
  geom_point(aes(color = Species, shape = Species)) +
  # nombre de ejes
  labs(x = "longitud de pétalo", y = "longitud de sépalo") +
  # escala de colores
  scale_color_manual(values=c("#999999", "#E69F00", "#56B4E9"),
                     name = "Especie") +
  # formas
  scale_shape_manual(name = "Especie", values = c(3,4,5)) +
  # tema
  theme_bw()
fig3
```




```
# d)

# para mostrar diferenets variables con distinto color/relleno,
# o en diferentes paneles,
# tienen que estar en formato long
iris.long <- pivot_longer(iris,cols = 1:4,
                          names_to = "medida",
                          values_to = "valor")

fig4 <- ggplot(data = iris.long,aes(x = Species, y = valor)) +
  # esto funciona
  # geom_boxplot() +
  # pero podemos añadir color a cada panel
  geom_boxplot(aes(fill = medida)) +
  facet_grid(medida~.,scales = "free_y")+
  labs(x = "Especie") +
  theme_bw() +
  # la leyenda es redundante, podemos eliminarla
  guides(fill = FALSE)
fig4
```



Funciones

11

Convierte el código de los ejercicios 7 y 9 en tres funciones - ¡bien documentadas! -:

a) La primera función acepta un vector de caracteres y devuelve el número de caracteres y de palabras por elemento en un dataframe.

b) La segunda función acepta un vector de caracteres y devuelve el número de menciones a usuarios (@) y el número de hashtags (#) de cada elemento.

c) La tercera función acepta un vector de caracteres y devuelve un dataframe con dos columnas: hashtag y frecuencia, el número de veces que aparece ese hashtag en todo el vector original.

```
# a)

#' número de caracteres y palabras
#'
#' Calcula el número de caracteres y de palabras en cada elemento de
#' un vector de caracteres
#'
#' @param v vector de caracteres
#'
#' @return dataframe con dos columnas, una por cada medida, y tantas
#' filas como elementos tiene el vector original
#' @export
#'
```

```r
#' @examples
#' vec <- c("este es un vector","de caracteres","con tres elementos")
#' res <- num_caracteres_palabras(vec)
num_caracteres_palabras <- function(v = NULL){

  df <- NULL
  if(!is.null(v)){
    df <- data.frame(caracteres = nchar(v),
                     palabras = stringr::str_count(v,'\\w+'))
  }
  return(df)
}

# b)

#' Menciones y hashtags
#'
#' Extrae el número de menciones y hashtagas
#' de cada elemento de un vector de caracteres.
#' Las menciones vienen definidas por el caracter 'arroba' al principio de una
#' palabra; los hashtags por almohadilla (#).
#'
#' @param v vector de caracteres
#'
#' @return dataframe con dos columnas y tantas filas como elementos del vector
#' original.
#' @export
#'
#' @examples
#' vec <- c("este es un #vector",
#' "de caracteres @con_menciones ",
#' "con #tres #elementos")
#'
#' res <- num_menciones_hashtags(vec)
num_menciones_hashtags <- function(v = NULL){
  df <- NULL
  if(!is.null(v)){

    # dataframe de resultados
    df <- data.frame(numero_menciones = numeric(length(v)),
                     numero_hashtags = numeric(length(v)))
    # extraemos el numero de menciones
    menciones <- stringr::str_extract_all(v, "(?<=^|\\s)@[^\\s]+")
    # y lo guardamos en el dataframe
    # recuerda: sapply devuelve un vector
    df$numero_menciones <- sapply(menciones,FUN = length)

    # idem con los hashtags
    hashtags <- stringr::str_extract_all(v, "(?<=^|\\s)#[^\\s]+")
    df$numero_hashtags <- sapply(hashtags,FUN = length)
  }
  return(df)
}
```

```r
# c)
#' Frecuencia de hashtags en un vector de caracteres
#'
#' Hashtags definidos por almohadilla (#) al comienzo de una palabra.
#'
#' @param v vector de caracteres
#'
#' @return dataframe con dos columnas: hashtags y frecuencia de los mismos
#' en todo el vector original.
#' @export
#'
#' @examples
#' vec <- c("este es un @vector",
#' "de caracteres @con_menciones ",
#' "con @tres @elementos")
#'
#' res <- frecuencia_hashtags(vec)
frecuencia_hashtags <- function(v = NULL){
  res <- NULL
  if(!is.null(v)){
    hashtags <- stringr::str_extract_all(v, "(?<=^|\\s)#[^\\s]+")
    # limpiamos los signos de puntuación (esto elimina también la # al comienzo)
    clean.hashtags <- lapply(hashtags,
                             FUN = function(x)str_replace_all(x,
                                                              "[[:punct:]]",
                                                              ""))
    # convertimos la lista en un vector
    clean.hashtags.vector <- unlist(clean.hashtags)
    # extraemos las frecuencias usando la función "table"
    res <- as.data.frame(table(clean.hashtags.vector))
    names(res) <- c("Hashtag","Frecuencia")
  }
  return(res)
}
```

12

Reescribe los ejercicios 7 y 9 con las nuevas funciones

```r
# cargamos el paquete stringr
library(stringr)

# leemos el archivo
my.file <- "data/sample500tuits_starwarsandaluz.txt"
conn <- file(my.file,open = "r")
lineas <-readLines(conn)

# numero de caracteres y palabras
res1 <- num_caracteres_palabras(lineas)
# numero de menciones a usuarios y de hashtags
res2 <- num_menciones_hashtags(lineas)
# hashtags y su frecuencia
res3 <- frecuencia_hashtags(lineas)
```