CUNEF R - Ejercicios

Leonardo Hansa

1. Primeros pasos con R

Ejercicio 1. Cuenta el número de flores con `Petal.Length` mayor que 5.1 en `iris`.

Ejercicio 2. Comprueba qué especies tienen flores con la anchura del sépalo mayor que 3,5 y menor que 4. *Pista.* Necesitarás `&` porque tienes más de una condición. Además, para simplificar el resultado te puede venir bien la función `unique()`. Usa `? unique` para consultar la documentación.

Ejercicio 3. ¿Cuántas flores de cada especie hay? *Pista.* La función `table()` te será útil.

Ejercicio 4. Calcula la media de la anchura del sépalo de las flores cuya anchura de sépalo está por debajo de la media general.

Ejercicio 5. Calcula la mediana de la anchura del pétalo de cada especie por separado.

Ejercicio 6. Calcula el mínimo y el máximo de la longitud del pétalo para de las flores *versicolor*. Luego cuenta cuántas flores setosa y virginica hay con la longitud del pétalo entre esos valores.

Ejercicio 7. ¿Qué pasa si usas la función `plot()` con el dataframe `iris`? ¿Cómo interpretas lo que ves?

Ejercicio 8. Si escribes `Titanic` en la consola verás un conjunto de datos con un formato un poco feo para el análisis. Usa `as.data.frame()` para convertirlo a un data frame y asígnalo a una variable (objeto) en R (elige el nombre que quieras). Ahora tienes un data frame cargado en memoria.

- Verifica que el data frame es un data frame (con `is.data.frame()`).
- Comprueba las clases de las columnas del data frame.
- Cuenta (o suma) el número de personas que sobrevivieron y el número de personas que no. Si tienes dudas sobre los datos, ejecuta `? Titanic` para acceder a la ayuda (sí, también tienes ayuda sobre conjuntos de datos en R).
- Cuenta el número de niños (`children`) que había en el barco.
- Cuenta el número de hombres que había en primera clase.

- ¿Había niños entre los miembros de la tripulación? ¿Y mujeres?

2. Lectura de ficheros

Ejercicio 1

- Read all the sheets of the Excel file `01_ ACCIDENTES POR TIPO EN  DISTRITOS.xls`. You should find some trouble with one of the years if you just copy and paste what I typed. Fix it.
- Create one data frame for each sheet. How many rows and columns each one has? *Hint.* `nrow()`, `ncol()` and `dim()` may help.
- What is the average number of accidents per type in 2009? And the standard deviation? *Hint.* You should use the `$` notation.
- What are the names of the columns? Do you consider them appropriate? *Hint.* `names()`.

Ejercicio 2 (janitor)

- Instala el paquete `janitor` y cárgalo.
- Usa `clean_names()` sobre `iris` para cambiar el formato de los nombres de columnas. *Pista.* Ten en cuenta que si solo ejecutas `clean_names(mi_data_frame_guay_recien_creado)`, R te mostrará por consola el resultado, pero los nombres del data frame seguirán iguales. Necesitas asignar el resultado al data frame de nuevo , con `<-`.

Ejercicio 3.

- Lee el fichero `iris2.csv` con la función `read_csv()` (paquete readr).
- ¿Cuántas columnas hay? ¿Qué nombres tienen? ¿Tiene sentido?

3. dplyr

Ejercicio 1. ¿Qué diferencias ves en los resultados de `glimpse(df_mtcars)` y `df_mtcars %>% glimpse()`?

Ejercicio 2. En el data frame `mtcars` decide qué columnas se pueden almacenar como `integer` sin perder información y cámbialas. Para cambiar el tipo de una columna (o de cualquier objeto) puedes usar `as.integer()`. Por ejemplo, `class(as.integer(4, 6, 1))`.

Ejercicio 3. A partir del data frame `df_inventado` construido en la sección *dplyr columns*, ahora te toca hacer unos cálculos. Necesitarás `mutate()`. Puedes optar por sobreescribir el data frame con los nuevos cálculos o simplemente mostrar los resultados por consola.

1. Calcula el área total de cada tienda (como longitud por anchura).
2. Calcula cuántos euros gasta cada cliente, por cada tienda.

3. Calcula cuántos euros gasta cada cliente en media en total. Pista._ Para hacer esto, `summarise()` es muy útil y puedes leer su documentación para ver cómo usarlo. Sin embargo, puedes emplear también el enfoque de la notación `$` y las funciones `sum()` o `mean()`.
4. Si la tienda es blanca o azul, réstale 5 metros a su longitud; en caso contrario, súmale 10 metros. *Pista.* `? if_else()`. Te vendrá bien usar `%in%`. Para aprender a usarlo, juega en la consola con códigos como `"white" %in% c("white", "blue")` o `"red" %in% c("white", "blue")` e intenta entender qué está pasando y cómo puedes usarlo dentro de `mutate()`.
5. Si todos los clientes en un día fueran a la tienda a la vez, ¿cuántos metros cuadrados por cliente habría?

Ejercicio 4. A partir del data frame `starwars`, responde estas preguntas.

- ¿Cuáles son los nombres de las columnas?
- ¿Cuáles son las clases de cada columna?
- ¿Cuáles son las dimensiones del data frame?

Ejercicio 5. Obtén las combinaciones distintas de valores que puedes encontrar usando las columnas `eye_color` y `gender` del data frame `starwars` (me refiero a combinaciones reales entre personajes que aparecen en la tabla: no intentes calcular todas las combinaciones posibles entre los valores). Crea un data frame con esas combinaciones. Exporta este data frame o tibble a un fichero csv usando una función del paquete readr. *Pista.* Después de ejecutar `library(readr)`, escribe en la consola o en un script `write_` y deja que el autocompletado te proponga opciones. Decide qué función deberías usar. *Otra pista.* Hay varias soluciones. En cualquier caso, recuerda usar `?` para leer la documentación de una función.

Ejercicio 6. Lee la ayuda de `na.omit()` para quitar todas las filas con algún `NA` del data frame `starwars`. Crea un data frame nuevo con los datos filtrados. Exporta el data frame a un fichero csv (separado por coma).

Ejercicio 7

1. Lee el fichero `volpre2019.csv` y crea un data frame con sus datos. Nómbralo como quieras. Encontrarás datos sobre el volumen y el precio de muchos productos de Merca-Madrid.
2. Llama a la librería janitor (instálala si toca) y usa la función `clean_names()` en el data frame (**recuerda sobreescribir el data frame, porque si no lo haces es como si no hubieras usado la función**).
3. Explora el data frame con las funciones que ya conoces: `nrow`, `ncol` (o `dim()`), `glimpse()`, `summary()`.

4. Cuenta cuántos `NA` hay en la columna `fecha_desde`. *Pista.* Por ahora, valdrá con usar `is.na()` para crear un vector lógico y luego usar `sum()` para sumar el número de casos `NA`.

5. Excluye casos donde `fecha_desde` sea `NA` y sobreescribe el data frame.

6. Calcula los valores únicos de origen (columna `desc_origin`) de productos `"VACUNO"` (`desc_variedad_2`).

7. Selecciona cuatro productos de la columna `desc_variedad_2` y extrae los meses en los que están disponibles (`fecha_desde`) y su origen. Hazlo de manera separada para cada uno. El data frame final de cada producto tendrá dos columnas. Ordena el data frame en base a la columna `desc_origin`. La función que necesitas es `arrange()`. *Sugerencia.* Para seleccionar productos, yo usaría `distinct()` en la columna y luego haría `sample_n(4)`, todo unido con pipes `%>%`. Lee la documentación `sample_n()`.

Ejercicio 8. A partir del data frame `storms`, generado en la sección *dplyr-aggregate*, haz los siguientes ejercicios:

1. ¿Qué observas al aplicar `summary()`?
2. ¿Cuáles son los diferentes valores en la columna `name`? Crea también un data frame que muestre los nombres distintos.
3. ¿Cuáles son los `status` diferentes? No crees data frame esta vez: solo muéstralo en consola.
4. ¿Cuáles son las combinaciones de valores disponibles entre status y presión?

Ejercicio 9 (data frame `storms`). Para storms, depressions y hurricanes que sucedieron entre 1975 y 1980, crea una nueva columna con la media y la desviación típica de `wind`. Crea una nueva columna restando la media a `wind` y dividiendo por la desviación típica. Calcula la media y la desviación típica con `summarise()` de esta nueva columna. ¿Qué ha pasado?

Ejercicio 10 (data frame `storms`).

1. Para las filas donde `hurricane_force_diameter` no sea `NA`, calcula la presión mínima, la mediana, la media y el máximo.

2. ¿Hay más filas por encima o por debajo de la media?

Ejercicio 11 (data frame `storms`). Para los huracanes, calcula la media de `tropicalstorm_force_diameter`. *Pista.* No es `NA`.

4. Visualización con ggplot2

Para los siguientes ejercicios, completa los huecos (`...`) con código.

Gráficos de dispersión

Ejercicio 1

1a. Echa un ojo al data frame `mtcars` con `glimpse()` y `? mtcars`. Luego crea un gráfico de dispersión de las millas por galón (en eje y) y el peso (en el eje x). Para crear el gráfico, sustituye los puntos en el siguiente código.

```
library(...)
ggplot(...) +
  geom_point(aes(x = wt, y = mpg))
```

1b. Repite el gráfico anterior, coloreando todos los puntos en rojo. Fíjate en que como el color no dependerá de datos del data frame, deberás indicarlo fuera de la función `aes()`.

```
ggplot(...) +
  geom_point(aes(...), colour = "red")
```

1c. Repite el mismo gráfico anterior pero coloreando los puntos según su cilindrada (columna `disp`).

```
ggplot(...) +
  geom_point(aes(x = ..., y = ..., colour = ...))
```

1d. En lugar de colorear los puntos con diferentes colores, haz que tengan un tamaño diferente en función de la variable `disp`.

```
ggplot(...) +
  geom_point(aes(...))
```

Ejercicio 2

2a. Los gráficos de dispersión también los puedes usar para datos no continuos. En R, cuando trabajas con datos continuos, normalmente usarás la clase `numeric` (peso, altura, ingresos,…). Pero hay algunas variables definidas por números que no son continuas (edad, número de cilindros, identificadores). ¿Cuál es la clase del número de cilindros en el conjunto de datos `mtcars`? Recuerda usar `? mtcars` si no sabes cuál es el nombre de columna que tienes que mirar.

2b. ¿Cuáles son sus valores únicos? Rellena los huecos:

```
mtcars %>% distinct(...)
```

2c. De acuerdo con sus valores únicos, la clase original de la columna no es la mejor. Prueba a dibujar un gráfico de dispersión de las millas por galón (eje y) frente al número de cilindros.

```
ggplot(mtcars) +
  geom_point(aes(...))
```

2d. En el ejercicio anterior, en general, no hay nada malo. Pero el eje x muestra valores que no son posibles en el conjunto de datos, como 5 y 7. Esto es porque el número de cilindros debería ser una variable categórica y no una continua. Arreglémoslo. En R, las variables categóricas pertenecen a la clase `factor`. Puedes fácilmente convertir algunas variables con la función `as.factor()`. Rellena los huecos para dibujar un gráfico de dispersión como el anterior y compara.

```
mtcars <- mtcars %>%
  mutate(cyl = as.factor(cyl))

ggplot(...) +
  ...
```

Ejercicio 3

3a. Con **ggplot2** tienes disponible el data frame `diamonds`. Echa un ojo a su información con `? diamonds` y `glimpse(diamonds)`.

3b. Crea un gráfico de dispersión con el precio de los diamentes (eje y) frente al peso (eje x). Hay más de 50.000 diamantes, por lo que puede llevar un poco más de lo esperado.

```
ggplot(diamonds) +
  ...(aes(...))
```

3c. Hay demasiados puntos, por lo que se superponen y acabamos con una nube un poco incómoda de interpretar. Para arreglarlo, primero reduce el tamaño de todos los puntos. Prueba con `size = 0.5`.

```
ggplot(...) +
  ...(aes(...), ...)
```

3d. Otra cosa que vamos a mejorar. Puedes dar algo de transparencia a los puntos. Esto facilitará la visualización en zonas donde se acumulan muchos puntos. El parámetro para esto se llama `alpha` (común no solo en R sino en otros lenguajes, como CSS). Este parámetro toma

valores entre 0 (invisible) y 1 (opaco). Fija este valor a $0,1$ en el gráfico anterior (en el que indicaste un tamaño también). Esta vez, te toca escribir el código desde cero, pero recuerda que siempre puedes copiar y pegar.

3e. Ahora toca dar algo de color al gráfico. Este color variará en función de la claridad de los diamantes, que es una columna del data frame. Fija también la transparencia en $0,4$, que ayudará a ver los puntos.

Ejercicio 4

4a. Creemos un data frame a partir de los datos `countries_of_the_world.csv`. Usa la función `clean_names()` del paquete janitor para simplificar los nombres de columnas. Elimina también todas las filas que tengan algún `NA`. Ahora, haz un gráfico de dispoersión de la columna `phones_per_1000` frente a la columna `literacy_percent`.

4b. En el mismo gráfico, colorea cada punto en función de la región.

4c. Haz otro gráfico de dispersión de la columna `service` frente a la columna `agriculture`. Cambia el tamaño de cada punto en función de su población y colorea todos ellos en `"darkblue"`.

Gráficos de líneas

Ejercicio 1

1a. Del data frame `economics` dibuja `unemploy` frente `date` con un gráfico de líneas. Necesitarás `geom_line()`.

1b. Usa dplyr para calcular una nueva columna que sea el ratio de desempleo (`umemployment` dividido por la población). Dibuja esta columna nueva como una serie temporal, tal y como hiciste antes.

```
economics_new <- ... %>%
  ...(unemployment_rate = ...)

# Code for the plot:
...
```

1c. Una cosa que mola de ggplot2 es que es capaz de hacer algunos cálculos, sin que recurras previamente a dplyr o R base. Por ejemplo, en el gráfico anterior no era necesario el paso con dplyr. Puedes calcular el ratio de desempleo al vuelo. Prueba a usar la fórmula `umemploy / pop` cuando especifiques el eje y.

```
... +
  ...(...(x = ..., y = ...))
```

Ejercicio 2

2a. Del data frame `ChickWeight`, crea un gráfico de líneas con la evolución de la columna `weight` a lo largo del tiempo. Ten en cuenta que necesitas una línea diferente para cada `Chick`. Para conseguir esto, asigna a columna `Chick` al parámetro `group=`, dentro de la función `aes()`.

```
... +
  ...(aes(x = ..., y = ..., group = ...))
```

2b. Ahora colorea cada línea en función de la columna `Diet`, que indica la dieta de cada gallina.

Gráficos de columnas

Ejercicio 1. Genera un gráfico de columnas similar al último *boxplot* que tienes en los apuntes, pero comparando la media de `gpd_per_capita`. *Pista.* Echa un ojo a la documentación de `geom_col()` para aprender a definir una posición *dodge* en las barras del gráfico. Tienes a continuación el código para el *boxplot*. El data frame `df_countries` lo tienes construido en los apuntes del módulo de ggplot2.

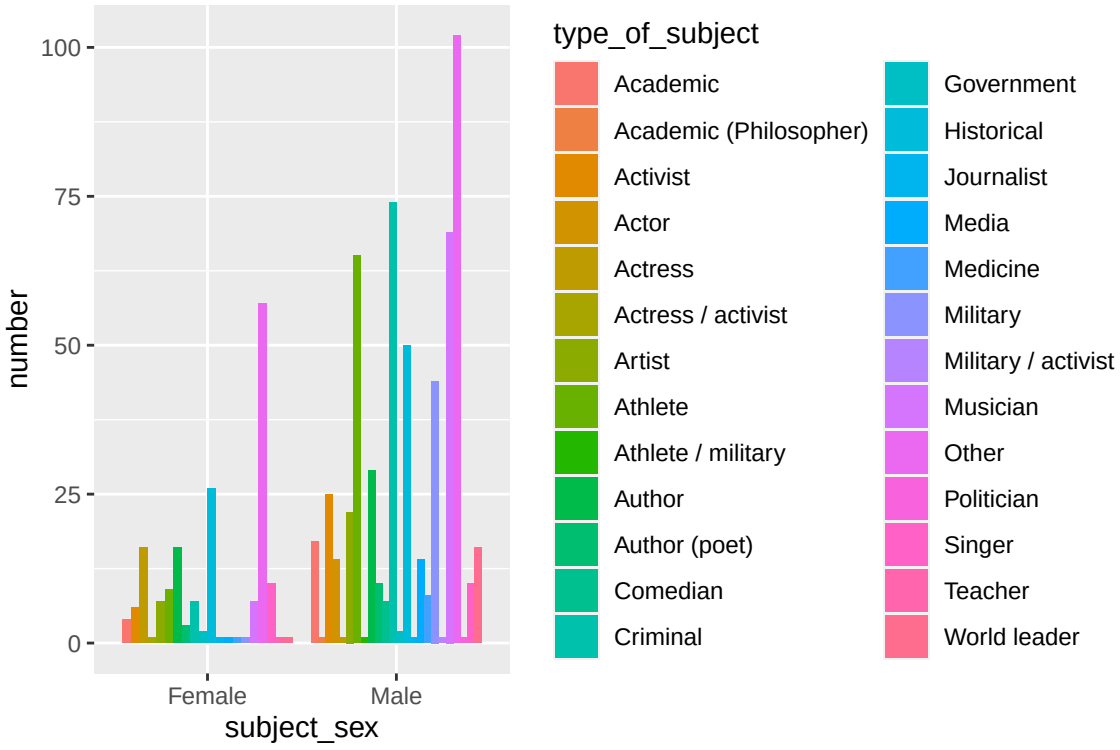
```
df_countries %>%
  mutate(literacy_percent_90 = literacy_percent > 90) %>%
  ggplot() +
  geom_boxplot(aes(x = literacy_percent_90, y = gdp_per_capita, fill = region))
```

Ejercicio 2. Usa el data frame `biopics` de la librería `fivethirtyeight`. Con dplyr calcula cuánto dinero facturó cada `type_of_subject` –hay muchas filas para las que este dato no está disponible. Pinta un gráfico de columnas para mostrar este resultado, usando `type_of_subject` como eje x. Ahora intercambia los ejes de coordenadas de forma que las columnas queden en horizontal, con una función de ggplot2 de la familia `coord_*`. También, reordena los datos de alguna manera para que el gráfico muestre los datos de más dinero a menos dinero. *Pista.* Para cualquier duda sobre el conjunto de datos, puedes acceder a la documentación con `? biopics`. También, para ordenar los datos, puedes usar la función de ggplot2 `reorder()` o trabajar con factores. Me gusta la función `as_factor()` del paquete forcats para esto.

Ejercicio 3. Repite el ejercicio anterior pero ahora también calcula el número de películas para cada `type_of_subject`. Usa esta nueva columna para colorear las columnas.

Ejercicio 4. Cuenta el número de películas por cada `year_release` y país, y luego crea un gráfico de columnas con la evolución. Colorea cada columna con la información de `country`, de forma que cada columna mostrará el número de películas de cada país en cada año.

Ejercicio 5. Replica el gráfico siguiente. Muestra el número de películas por `subject_sex` y `type_of_subject`.



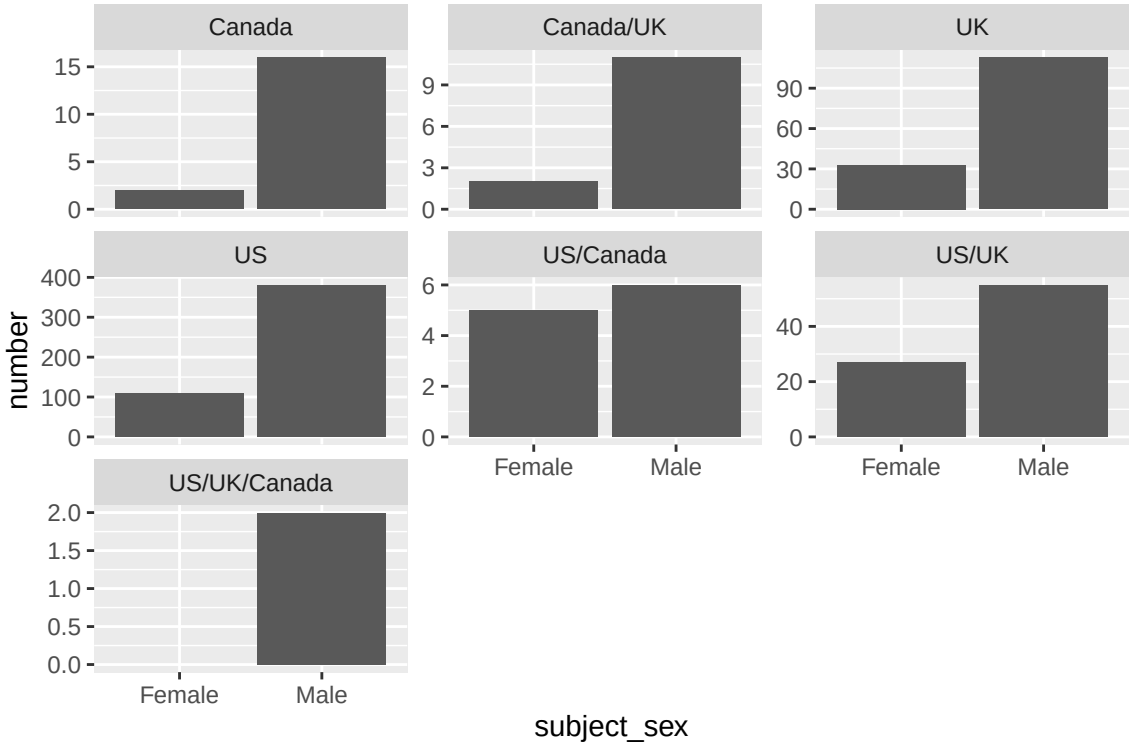
Boxplots

Ejercicio 1. Crea *boxplots* para cada `type_of_subject`. Haz que sea un único cuadro con varios boxplots, todos ellos horizontales.

Facets

Ejercicio 1. Muestra la evolución del número de películas, de manera separada para `Female` y `Male`, en un único cuadro.

Ejercicio 2. Haz un gráfico de columnas con el número de películas en función de si el protagonista es `Female` y `Male` y en función también del país `country`. Necesitarás algún cálculo con dplyr. Asegúrate de que el eje y sea diferente entre cada gráfico.



5. Elementos de programación

Funciones

Ejercicio 1. Crea una función que reciba un vector `x` de números y otro número `a` y que dibuje, en función de esos, la curva de la función matemática $f(x) = xln(ax)$, donde a es un número positivo real. Asegúrate de que `a` es igual a 1 por defecto. La función debería incluir este vector `x` como la columna de un data frame (puedes hacerlo con la función `tibble()`). Después la función creará otra columna aplicando la fórmula de la función matemática, con dplyr. Finalmente, con ggplot2, dibujará la curva. Un gráfico de líneas valdrá. Para comprobar que la función hace lo que tiene que hacer, llámala con el vector generado con el siguiente código: `seq(0.01, 1, by = 0.01)`. Prueba diferentes valores de `x` y `a` para ver cómo cambia el gráfico.

Ejercicio 2. En la función anterior, añade un código que evite casos en los que el logaritmo se evalúe con números negativos. Es decir, si el vector `a*x` tiene algún número negativo, haz que la función no devuelva nada, con `return()`, por ejemplo.

Ejercicio 3. Vas a definir una función que lea un csv sobre calidad del aire en Madrid, arregle el formato de un par de columnas que ahora te diré y devuelva la tabla. Después de eso, unificaremos toda a información de varios archivos a un único dat aframe. Los archivos son aquellos cuyo nombre empieza por `NO2` o `CO`. Primero, la función recibirá la ruta al fichero. Ahora lo leerá, (hazlo con readr; adivina la función que necesitas) y ahora toca usar dplyr para arreglar las columnas `mes` y `dia`. Queremos que sean numéricas. `mes` es fácil pero para `dia` necesitarás algo de tratamiento de caracteres. Intenta usar la función `str_remove()` del paquete stringr. Ahora asegúrate de que la función devuelve el data frame que has creado. También, excluye la columna magnitud. No es necesario porque las últimas columnas indican la magnitud. Luego falta unir todos los data frames pero eso es asunto del siguiente ejercicio. Por ahora, asegúrate de que la función funciona para los seis data frames.

Ejercicio 4. Ahora toca unir toda la información. Crea un data frame con todos los datos sobre NO2 y otro sobre el CO. Usa `bind_rows()` de dplyr para esto. Después de eso, une ambos (el de NO2 y el de CO) con `inner_join()`. ¿Cuántas filas hay? Haz lo mismo con `full_join()`, `right_join()` y `left_join()`. ¿Qué diferencias ves?

Ejercicio 5. ¿Puedes adivinar el resultado del siguiente código sin ejecutarlo? ¿Cuál es el papel de cada `c`?

```
c <- 1
c(c = c)
```

Ejercicio 6.. El vector `letters` contiene todas las letras desde la `"a"` a la `"z"`. Puedes seleccionar una muestra aleatoria con `sample()`. Crea una función que reciba un número entero n y luego cree una muestra de n letras. Después de eso, haz que ordene las letras y finalmente las una todas en una sola palabra, con la función `paste0()` y el argumento `collapse=`.

Ejercicio 7. Crea una función similar a la anterior pero, en lugar de generar una muestra aleatoria de `letters`, que genere dos muestras: una con `letters` y otra con `LETTERS`. La función debe recibir un argumento para el número de letras de `letters` y otro para `LETTERS`. Luego creará un vector con las dos muestras (usa `c()`). Ahora ordena el vector y únelo en una sola palabra.

Ejercicio 8. Crea una función que reciba dos columnas del data frame `iris` (los nombres de las columnas como un `character`). Ahora multiplicará esas dos columnas entre sí. Crea una columna nueva en el data frame con el producto que has calculado. **No uses dplyr.**

Ejercicio 9 (resuelto). Crea una función que lea el fichero `human_resources.csv`. La ruta debería ser uno de los argumentos de la función. Luego seleccionará todas las columnas menos `sales` y `salary`. Ahora genera un vector lóico aleatorio cuya longitud sea el número de filas

del data frame. Este vector debería tener más `TRUE`s que `FALSE`s: la ponderación deberá ser uno de los argumentos de la función (será un vector de dos números cuya suma sea 1, e.g., $0,7$ y $0,3$). Esto lo usarás para dividir el data frame en dos data frames. Lo puedes hacer con la función `filter()`. Luego usa la función `glm()` para ajustar un modelo predicitivo con la primera parte del data frame (la grande). Luego usa el otro data frame para hacer una predicción con `predict()`. Puedes comprobar cuántos casos has acertado con `table()`. Haz que la función devuelva esta tabla. Acabas de ajustar un modelo para predecir que empleados tendrán más propensión de dejar la compañía.

```
library(tidyverse)

genera_primer_dataframe <- function(ruta, pesos){
  df_hr <- read_csv(ruta, col_types = cols()) %>%
    select(-sales, -salary)

  filas_train <- sample(c(TRUE, FALSE),
                        size = nrow(df_hr),
                        replace = TRUE,
                        prob = pesos)

  df_hr_train <- df_hr %>%
    filter(filas_train)

  modelo <- glm(left ~ .,
                data = df_hr_train,
                family = binomial)

  df_hr_test <- df_hr %>%
    filter(!filas_train)

  dato_real <- df_hr_test$left

  prediccion <- predict(modelo,
                        newdata = df_hr_test %>% select(-left),
                        type = "response")

  prediccion2 <- if_else(prediccion > 0.3, 1, 0)

  return(table(dato_real, prediccion2))

}
```

```
genera_primer_dataframe("../data/sesiones/human_resources.csv",
                        c(0.7, 0.3))
```

Ejercicio 10. Lee estos códigos e intenta adivinar cuál será el valor de `a`, `aa` y `aaa`. No ejecutes el código hasta que tengas una idea clara de cuál será el resultado.

```
a <- 1

mi_funcion <- function(b){
  if(b > 0){
    a <- 100
  } else {
    a <- -50
  }

}

mi_funcion(10)
a
```

```
aa <- 0

mi_funcion2 <- function(b){

  if(is.character(b)){
    aa <- aa + 10
  } else if(is.numeric(b)){
    aa <- aa - 20
  }

}

mi_funcion2("hola")
aa
```

```
mi_funcion3 <- function(aaa, b){
  aaa <- aaa + b
  return(aaa)
}

b <- mi_funcion3(3, 4)
```

```
aaa
```

Ejercicio 11. Crea una función con dos argumentos. El primero se llama `case=` y recibirá una palabra. El segundo se llama `times=` y recibe un número entero. Si la palabra que recibe es `"dados"` o `"Dados"`, la función genera dos vectores de longitud `times`, con los valores aleatorios entre 1 y 6. Crea otro vector sumando esos dos vectores y cálcula cuál es el resultado más frecuente (el vector suma tendrá valores entre 2 y 12, y tendrás que ver qué valor entre 2 y 12 se repite más). Si la palabra es `"coin"` o `"Coin"`, simula el lanzamiento de una moneda con la función `bernoulli()` y calcula la probabilidad de obtener cada resultado. Lanza la moneda el número de veces indicado por `times=`. Si la palabra recibida es otra, muestra por consola `"I don't know what to do with this"`. Pruébala con valores que veas adecuados.

Ejercicio 12. Crea una función para calcular la cuota mensual de un préstamo bancario de sistema francés. La función recibirá el total prestado, el interés anual y el periodo (número de años). Pruébalo con un préstamo de 150.000 €, un interés de 2.50% y que se devuelva en 25 años. La fórmula es:

$$a = C \cdot \frac{i/12}{1 - (1 + i/12)^{-12n}},$$

donde a es la cuota mensual, C es la cantidad pedida, i es el ratio de interés anual y n el número de años.

Ejercicio 13 (interés compuesto). Crea una función para calcular la cantidad total que tendríamos después de un cierto periodo de tiempo con un interés. Los inputs son la cantidad original, el ratio de interés y el número de periodos. La fórmula es:

$$\text{final amount} = \text{init amount} \cdot (1 + r)^{periods}$$

Pruébala con diferentes ejemplos:

- Si inviertes 1000 € a un interés anual de 7%, ¿cuánto tendrás después de un año?
- Si inviertes 1000 € a un interés anual de 7%, ¿cuánto tendrás después de 10 años?
- Si tienes un depósito con tu banco que garantiza un interés de 1% trimestral, ¿cuánto tendrás en un año?

Lists and loops

Ejercicio 1. Check the class of every column in the `dslabs::brexit_polls` data frame. *Hint.* Don't write `class` 9 times.

Ejercicio 2. Try creating a data frame with `map_dfr()` from `"CO_2017.csv"` and `"NO_2017.csv"`. What's going on?

Ejercicio 3 (compound interest). Using a function similar to the one you defined for the Ejercicio 13 on the previous function sections, create a loop for comparing what happens **after a year if you invest 1 €** at these different situations:

- 100% annual interest rate.
- 50% semeter interest rate.
- 25% quarter interest rate.
- 1/12 month interest rate.
- 1/365 daily interest rate.

The new function should have one only argument: the number of periods (1 period for annual interest rate, 2 period for semester, 4 for quaterly…). Do it in two different ways: using a *for* loop and using `sapply()` or `map_dbl()`. It seems that if you kept on iterating with larger interest rates, the numbers would converged into a very special one. Do you know what this is?

Ejercicio 4. Take the `dslabs::na_example` **vector** and replace every `NA` value with the previous record. *Remark.* This is a vector, not a dataframe.

Map

Ejercicio 1. Read the file `anscombe`. **Ejercicio 2.** Get the class of all the columns. **Ejercicio 3.** Get the mean of all the columns. **Ejercicio 4.** Get the sd of all the columns. **Ejercicio 5.** Calculate again the mean and the sd of all the columns and store them in the same list (each element will be a vector). **Ejercicio 6.** Create a list of four data frames: each data frame will contain a pair of columns x and y: the first one will contain x1 and y1, the second one x2 and y2, and so on. `map()` should iterate over a vector `1:4` and you will create a function that select the columns containing the number. **Ejercicio 7.** Calculate the correlation of all the pairs x_i and y_i. You can use your previous function because maybe you just have to add something like `cor()`.

Ejercicio 8. Create scatter plots with those pairs. It may be easier changing the name of the columns inside your function using the function `set_names()`.

Ejercicio 9. Read the `babynames` dataset.

Ejercicio 10. We have selected the next set of names randomly. Create a list of summaries of the data frames you obtain filtering by those names (use `map()` and create a function that includes a filter and a summary).

```
nombres_random <- c("Mariyanna", "Lamiah", "Shandal",
                    "Rynda", "Kcee", "Camia",
                    "Isaiyah", "Barbare", "Braxon",
                    "Kailly")
```

Ejercicio 11. Create a list of size 2, each element having the 5 most used names for boys and the same for girls.

Ejercicio 12. Create a vector with those names.

Ejercicio 13. For each name, count the number of years when it appeared

Ejercicio 14. For each name, create a column plot with the evolution of babies with that name per year. Use the name as the title.

Ejercicio 15. The same as before, but now color the columns differently if it is a boys' name or girls' name.
