Introducción a la Ciencia de Datos

Guía de trabajos prácticos N°2

Les proponemos comenzar a analizar el set de datos sobre el Titanic ubicado en el Campus de la materia a partir de una serie de consignas, usando RStudio.

Recordatorio: mirar la Guía 1 para ver cómo y dónde usarlo.

Análisis sobre dataset `titanic`

En el Campus de la materia van a encontrar el dataset titanic que contiene a todos los pasajeros que viajaban en el RMS Titanic en abril del año 1912, cuando se hundió. Descárguenlo. En el mismo van a encontrar los siguientes atributos sobre cada pasajero:

`sex = Sexo del pasajero`

`age = Edad en años`

`siblings_of_the_passenger = Cantidad hermanos/conyuges a bordo`

`parents_or_children = Cantidad de padres/hijos a bordo`

`fare = Tarifa abonada`

`Port_of_Embarkation = Puerto donde embarcó: C - Cherbourg, Q - Queenstown, S - Southampton`

`class = Clase del ticket`

`who = Man/Woman/Child`

`alone = TRUE: viajaba solo, FALSE: no viajaba solo`

`survived = TRUE: Sobrevió / FALSE: No sobrevivió`

Como ya vimos en la guía anterior vamos a usar "tidyverse" para visualizar tablas ordenadas y realizar gráficos **RStudio.** Repasemos un poco como cargamos y empezamos a analizar un dataset.

1. Primero hay que cargar la librería *tidyverse* con `library(tidyverse)`
2. Abrir el archivo en RStudio usando el comando `read_csv`. Para cargar el dataset a un data frame:

```
titanic_df <- read_csv("ruta_del_archivo", locale = locale(decimal_mark = ","))
```

1. Pueden visualizar los datos como una tabla usando el comando `view(...)`. ¿Cuáles son las unidades de estudio?.
2. Aplicando los comandos `spec(...)` y `glimpse(...)` se puede obtener información sobre la tabla de datos.
   a. ¿Qué diferencias hay entre los *outputs* del comando `spec(...)` y `glimpse(...)`? ¿Qué información comparten?
   b. ¿Cuántas observaciones hay?
3. Utilizando la información que se obtiene aplicando los comandos anteriores, respondan:
   a. ¿Cuáles son las variables del dataset? ¿De qué tipos son?
   b. Muchas veces los tipos de las variables se cargan con valores por defecto como 'chr' (char). Para las variables anteriores determinar si el tipo cargado es correcto o si consideran que debería ser otro.
   c. ¿Qué variables consideran que podrían utilizarse como categóricas?

Máximos / Mínimos / Valores únicos

Vamos a empezar a conocer el dataset y los posibles valores de las variables filtrando y ordenando las unidades. Recordemos que para obtener todos los valores de un atributo (o columna de tabla) usamos el símbolo "$". Así, `titanic_df$columna_1` nos devuelve una lista con los valores de la columna_1 del *data frame*. Esto se puede usar en conjunción con otras funciones como `min(...)`, `max(..)` y `unique(...)` que extraen los valores **mínimo**, **máximo** o **elimina repetidos** de una lista de valores.

4. Seleccione todos los valores de la columna *age* (edad) ejecutando el comando `titanic_df$age`, van a ver que se los muestra todos en la consola. Si hacen `min(titanic_df$age)` están ejecutando la función mínimo sobre todos esos números. Háganlo y traten de explicar el resultado.

NA (Not Available) significa que hay un dato faltante entre los valores de la columna. Si se fijan en una de las salidas anteriores pueden ver que no es solo uno, son muchos. Esto trae problemas cuando se quiere usar el mínimo sencillamente porque el elemento NA no es un número y no se puede calcular el mínimo si no sabemos qué valor tiene ese elemento. Lo mismo pasa con muchas otras funciones. Lo que podemos hacer (por ahora) es indicarle que ignore esos valores agregando el argumento `na.rm=TRUE`. Por ejemplo: `min(titanic_df$age, na.rm=TRUE)`

Tanto `min` como `max` con funciones del paquete `base` de R, que solo funcionan sobre una columna. El paquete `dplyr`, parte de `tidyverse`, tiene una serie de funciones, la familia `slice`, que permite obtener filas completas de un dataset. Por ejemplo, `slice_min` permite obtener la fila (o filas) en las que alguna variable toma su valor mínimo. `slice_max` funciona de manera similar.

5. Prueben usar `slice_min` con el dataset, fijando el argumento `order_by` a `age`: `slice_min(titanic_df, order_by=age)`, o simplemente `slice_min(titanic_df, age)`. Comparen el resultado con el que se obtuvo en el punto antrior. ¿Pueden decir ahora si la persona sobrevivió? **Ayuda**: pueden usar `relocate` para poner delante la columna `survived`: `slice_min(titanic_df, age) %>% relocate(survived)`.

6. Hagan lo mismo con `slice_max`.
7. En alguno de los casos anteriores, agreguen el argumento `n` a la función `slice_max` o `slice_min`, y con algún valor mayor que 1. P.ej.: `slice_max(titanic_df, age, n=5)`. ¿Pueden describir qué hace ese argumento?
8. Con estas herramientas, respondan:
    a. ¿Cuál fue el boleto más barato (con *fare* menor)? ¿Y el más caro?.
    b. ¿Cuántas categorías de clase (*class*) había?.
    c. ¿Cuáles eran los puertos desde donde embarcaron (*Port_of_Embarkation*)? (TIP: si presionan "TAB" **RStudio** los ayuda a autocompletar los nombres de *data frames* y columnas)

Ordenar

El comando `arrange` permite ordenar las filas de un *data frame* en función de una o varias columnas/variables. Por ejemplo, para ordenar las filas de forma ascendente por la variable *fare* `arrange(data, fare)` y para hacerlo de forma descendente `arrange(data, desc(fare))`. Utilizando este comando responder:

9. ¿Desde qué puerto embarcaron las dos personas más jóvenes que estaban en el barco?.
10. ¿De qué clase eran las dos personas más longevas?.

Estas respuestas también pueden obtenerse con los comandos `slice` que describimos más arriba. Con el comando `arrange` también se puede ordenar por más de una variable. Para ello hay que pasarle las variables por las que queremos que ordene en el orden que queremos que lo ordene: `arrange(data, Variable1, Variable 2)`. Prueben aplicando este comando al dataset de titanic ordenando primero por `fare`, después por `age` y viceversa. Vean la tabla que resulta en cada caso.

Filtrar

Otra función útil es **filter**, que sirve para quedarnos solamente con las unidades que cumplan con ciertas condiciones. Se ejecuta de la siguiente forma: `filter(data frame, condición)`, se le pasa el data frame y la condición que deben cumplir las unidades (atributo y valores permitidos). Por ejemplo, para quedarnos solamente con aquellos pasajeros de primera clase podemos ejecutar `filter(titanic_df, class=='First')`. Esto devuelve un data frame que sólo contiene aquellos pasajeros de primera clase.

Un detalle importante es que a diferencia de las funciones que vimos más arriba, `filter` **ignora los NA**.

Para no tener que calcular todo el tiempo el mismo filtro o la misma selección de columna se pueden guardar resultados intermedios en una variable y después usarlo. Esto se hace con la asignación `<-` (o con el atajo Alt+- (alt + menos)) por ejemplo, si queremos guardar el resultado del filtro de los pasajeros de primera clase para usarlo después o la lista de todas las edades:

```
titanic_first <- filter(titanic_df, class=='First')
titanic_age <- titanic_df$age
```

Esto guarda en `titanic_first` el *data frame* filtrado y en `titanic_age` las edades, después simplemente los usamos. Por ejemplo, para calcular la edad máxima de los pasajeros de primera clase y la mínima de todos:

```
max(titanic_first$age, na.rm=TRUE)
min(titanic_age, na.rm=TRUE)
```

Se puede usar ==, !=, <, >, <=, >= (igual, distinto, menor, mayor, menor o igual, mayor o igual) o filtrar usando más de una condición[1]. Usando esto:

11. Seleccionen a los pasajeros mujeres y sobre esto encuentren la edad máxima. ¿Es el mismo resultado que calcular la edad máxima sobre todos los pasajeros?
12. Verificar: ¿qué edad máxima tienen los pasajeros considerados niños? ¿Es éste valor menor a la mínima edad de aquellos pasajeros adultos? **¡Ver cuál es la edad máxima de los niños!**

Gráficos de Barras

Después de una primera aproximación vamos a hacer algunos gráficos de barras (*bar plot*). Los gráficos de barras son útiles para mostrar **cantidades** y **proporciones**. Se generan de forma similar a los gráficos de dispersión de la guía anterior, llamando a `ggplot` primero indicando de dónde sacar los datos, luego indicando qué variable queremos agrupar para contar y por último llamando a `geom_bar()`. Por ejemplo, si queremos mostrar cuántos pasajeros hombres y mujeres había en el Titanic:

```
ggplot(data = titanic_df) +
  geom_bar(aes(x=sex))
```

Cada una de las categorías de `sex` (*male / female*) va a ser una barra y la altura va a ser la **cantidad** de unidades que pertenecen a esa categoría. Pueden ver el resultado en la primera imagen de la *Figura 1.*

13. Elaborar gráficos de barras que permitan visualizar la cantidad de pasajeros con, según
    a. la cantidad de hermanos en el barco (variable `siblings_of_the_passenger`)
    b. si son hombres / mujeres / niñx (variable `who`)

Hasta ahora estuvimos viendo cantidades de una sola variable (`sex`, `who`, etc.), pero también se puede aprovechar el diagrama de barras para mostrar los detalles de otra variable más. Por ejemplo, además de mostrar cuántos pasajeros de cada sexo habia podemos ver cuántos había **en cada clase**.

Esto se puede hacer añadiendo una estética de relleno (*fill*) a las barras de sexo indicando qué atributo usar para **desagregar** cada barra. Por ejemplo, para ver el detalle de cada **clase** en

[1] Para más detalles capítulo 5.2 del Wickham

el gráfico anterior agregamos `fill=class` al `aes`. pueden ver la diferencia en la *Figura 1*:

```
ggplot(data = titanic_df) +
  geom_bar(aes(x=sex, fill=class))
```

TIP: A veces trabajar con los nombres de columnas que tiene el dataset puede ser molesto, pueden cambiarlo usando `rename(data frame, 'nombre nuevo' = 'nombre viejo')`. Esto devuelve un nuevo *data frame* con el nuevo nombre de la columna.

Por ejemplo, cambiemos el nombre de la columna 'Port_of_Embarkation' a 'Puerto' y guardemos el nuevo data frame en *new_titanic_df*:

```
new_titanic_df <- rename(titanic_df, 'Puerto' = 'Port_of_Embarkation')
```

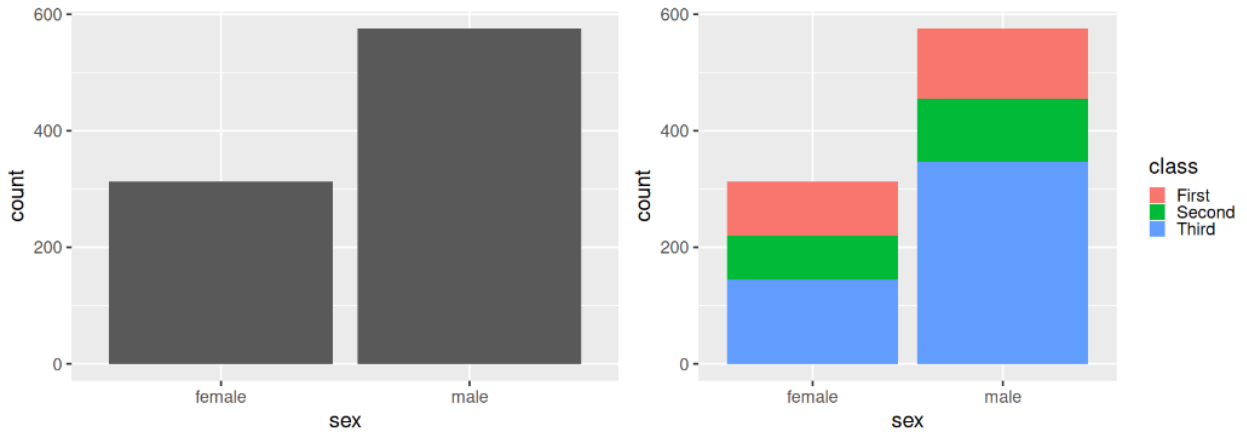

Figura 1: Cantidad de pasajeros hombres/mujeres y la misma información desagregada por clase.

14. Realizar **dos** gráficos de barras que indiquen la cantidad de pasajeros hombres, mujeres y niñxs (de forma desagregada) que viajaban en cada clase. Es decir, hagan un gráfico en función de las clases desagregando por hombre/mujer/niñx (gráfico 1) y otro gráfico en función de hombre/mujer/niñx, desagregando por clase (*Figura 2*).

15. Mostrar en dos gráficos de barras la cantidad de personas que viajaban solas y las que viajaban acompañadas, desagregando por las variables "*who*" (gráfico 1) y "*class*" (gráfico 2).

Mirando el último de los gráficos del punto anterior (el primero de la *Figura 2*), ¿podemos decir a simple vista si entre los viajeros acompañados había más personas pertenecientes a segunda clase que entre los viajeros solitarios?

Si queremos hacer este tipo de comparaciones es útil desapilar las barras, lo que se logra agregando `position = "dodge"` al geom:

```
ggplot(data = new_titanic_df) +
  geom_bar(mapping = aes(x=alone, fill = class), position = "dodge")
```

En la *figura 2* se puede ver la comparación, y ahora sí podemos responder la pregunta. Hay **más** pasajeros de segunda clase viajando solos.

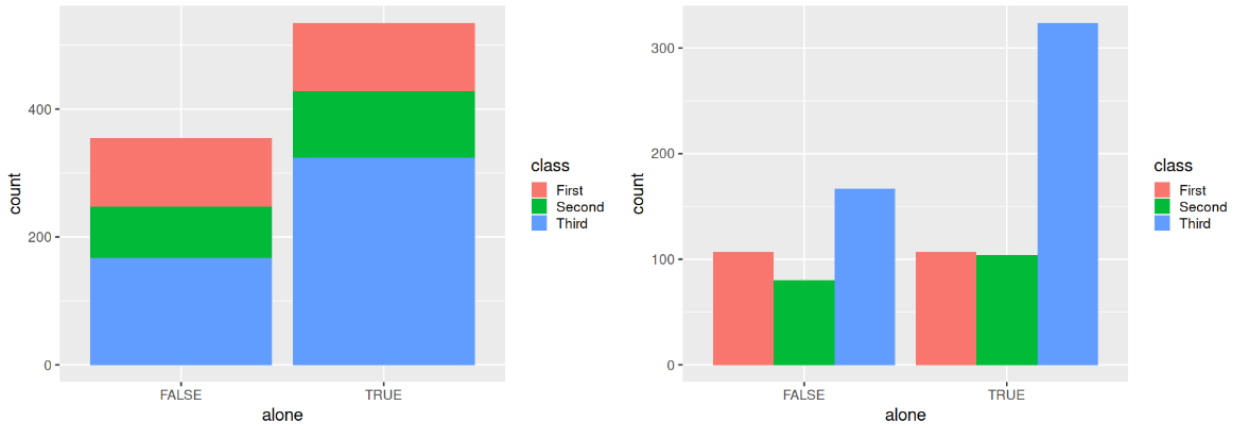

Figura 2: Desapilando para poder comparar individualmente

A veces nos resulta más útil en vez de hablar de cantidades hablar de porcentajes. Por ejemplo veamos el primer gráfico de *Figura 3,* es la cantidad de personas que viajan solas o acompañadas desagregado por `who`, que hacemos podemos producir con la línea:

```
ggplot(data=new_titanic_df) +
  geom_bar(aes(x=alone, fill=who))
```

Se puede ver que **la cantidad** de mujeres que viajan solas es casi la misma que las que no, pero a simple vista no podemos ver qué relación tiene esto con la cantidad de personas dentro de cada categoría. O sea, no podemos responder una pregunta del estilo "¿De las personas que viajan solas qué **proporción** son mujeres?"

Para poder ver esto hay que agregar `position = "fill"` al geom:

```
ggplot(data = new_titanic_df) +
  geom_bar(mapping = aes(x=alone, fill = who), position = 'fill')
```

Ahora mirando el segundo gráfico de la *Figura 3* sí podemos decir que "de las personas que viajan solas hay un 25% que son mujeres".

Hay un último detalle que quizás hayan notado, si bien cambió la escala en el segundo gráfico, sigue diciendo "count" (cantidad) esto no es del todo correcto. Para hacer gráficos más claros, recuerden que podemos cambiar el nombre a los ejes con `xlab(...)` e `ylab(...)`

```
ggplot(data = new_titanic_df) +
  geom_bar(mapping = aes(x=alone, fill = class), position = "fill")+
  xlab("Viaja Solo") +
  ylab("Proporción")
```

Si quieren repasar o explorar un poco más el tema siempre pueden recurrir el Wickham o a este link con un par de tips para agregar a las barras.

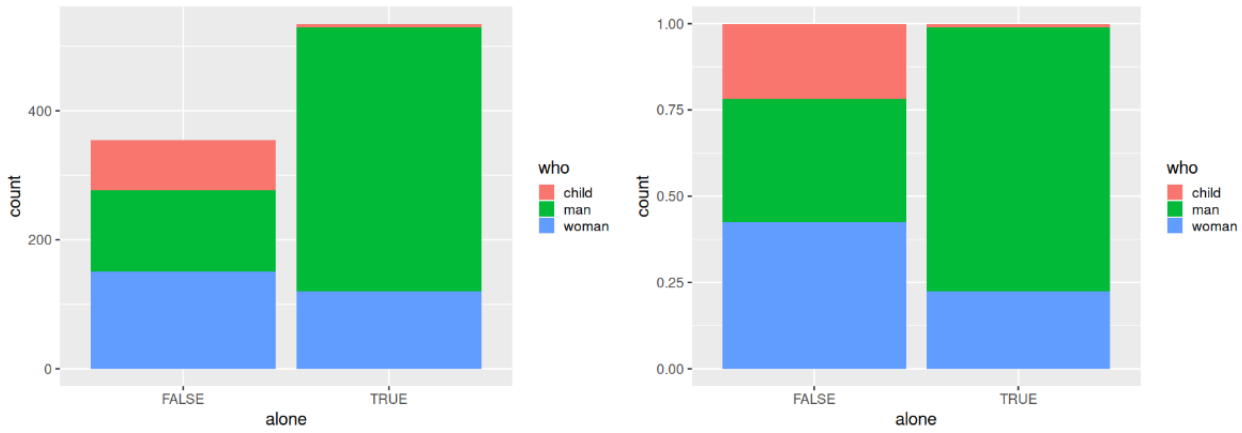

Figura 3: Cantidades y porcentajes

Figura 3: Cantidades y porcentajes

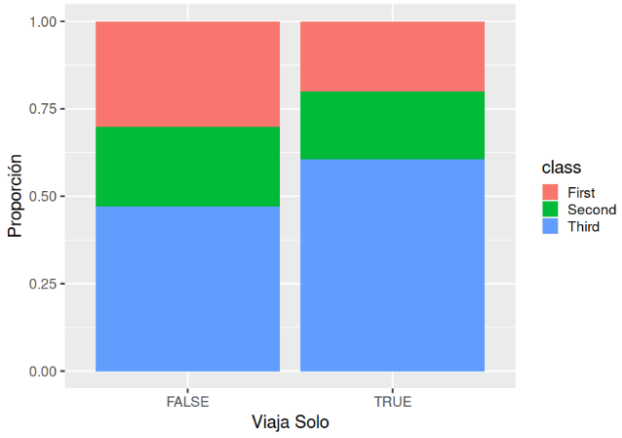

Figura 4: Proporciones de pasajeros de primera, segunda y tercera clase que viajan solos o acompañados

Agrupar

Ahora vamos a aprender a utilizar un comando que es de mucha utilidad para estudiar conjuntos de datos, ya que nos permite resumir o colapsar un grupo de valores en un sólo valor según lo que se desee. Se trata `summarise()` o `summarize()`. Este comando suele aplicarse luego de haber **agrupado** las observaciones a partir de variables categóricas, utilizando la función `group_by()`.

Entonces, por ejemplo, si corremos:

```
titanic_groupwho <- group_by(titanic_df, who)
```

`titanic_groupwho` es un nuevo data frame donde los datos están agrupados en función de la variable "who". Hagan click en esta nueva variable en el entorno de trabajo y vean cómo se ve ahora que lo agrupamos. ¿Cambió en algo su respecto al *dataframe* `titanic_df`?

Ejecutemos la línea

```
summarise(titanic_groupwho, n())
```

Verán en la consola que se imprime una tabla que muestra las variables en las cuales se agrupó al *dataset* y la cantidad de observaciones que pertenecen a ese grupo. Es decir, en este caso, la cantidad de pasajeros que pertenecían a la categoría "child", "man" y "woman".

Vean qué sucede si aplican el mismo comando, pero esta vez el data frame sin agrupar: `titanic_df`. Esta vez el resultado corresponderá a contar todas las filas del dataset pero sin agrupar de ninguna forma.

En general, todas las funciones que se apliquen a un data set agrupado previamente se aplicarán en cada grupo. Por ejemplo, si queremos calcular la edad promedio (con la función `mean`) de las personas que viajaban en el Titanic, podemos hacer:

```
summarise(titanic_df, edad.promedio = mean(age, na.rm=TRUE))
```

16. Realicen el mismo cálculo sobre el dataset agrupado creado arriba, `titanic_groupwho`. Si antes teníamos el promedio de todo el dataset, ¿qué estamos viendo acá?

17. Utilizando estos nuevos comandos, calcular la cantidad de pasajeros que había:

    a. En cada clase.
    b. Que hayan sobrevivido y que no.
    c. Con la misma cantidad de parientes en el barco.

Para estas dos últimas consignas también podrían haber utilizado el comando `count`. Investiguen cómo funciona, ejecútenlo y compárenlo con lo obtenido con `summarise()`. También pueden revisar la sección del libro de Wickham.

Resumen de lo visto en esta Guía:

- Repaso Guía anterior: Cargar un archivo .csv, . seleccionar valores de una columna.
- Funciones `glimpse`, `max`, `min`, `unique`, `slice_max`, `slice_min`, `arrange`, `filter`, `group_by`, `summarise`

- Guardar resultados intermedios en variables con usando `<-`.
- Gráficos de barras, barras apiladas y proporciones.

Para trabajar en casa

Contestá el Cuestionario de la Clase 2 antes de la clase virtual de esta semana.

Bibliografía obligatoria

Visualización de datos

Wickham: https://es.r4ds.hadley.nz/03-visualize.html

Cómo trabajar con bar plots en R

Ejemplos: https://r-graph-gallery.com/barplot.html

Etiquetas: https://www.data-to-viz.com/caveat/hard_label.html

Bibliografía para profundizar

Opción a los bar plots, los lollipop plot: https://r-graph-gallery.com/lollipop-plot

Si se puede, ordenar las barras: https://www.data-to-viz.com/caveat/order_data.html