Procesamiento de base de datos

# Contenido

Durante esta sesión veremos cómo manipular data frames, y para eso, veremos lo siguientes puntos:

* Breve resumen de la sesión anterior.
* Importación de base de datos.
* Manipulación de data frames.
* Encadenamiento de instrucciones (**uso de pipes, %>%**).
* Agrupación de observaciones.

# Breve resumen

Se revisarán los principales temas tratados en la sesión previa. Estos temas serán importantes para esta sección.

Los principales temas de la sesión anterior son:

* **¿Qué es un script?**: corresponde a un documento de texto, en el cual se escribe una serie de instrucciones para R.
* **¿Qué es un paquete?**: corresponde a una colección de funciones diseñadas para atender distintas tareas. Los paquetes complementan a las funciones de base existentes en R. Para poder ejecutar estas funciones es necesario instalar el paquete y cargar las librerías.
* **¿Qué es una función?**: corresponde a un conjunto de instrucciones que permite automatizar rutinas largas y/o complicadas (ejemplo de función).
* **¿Cómo asignar un valor a una variable?**: (ejemplo de asignación).
* **¿Qué es un vector?**: es el objeto más básico en R. Este objeto es una forma de almacenar datos. (ejemplo de vector).
* **Tipos de vectores**: los vectores son atómicos. (listado de vectores).
* **Operaciones matemáticas:** listado de operadores matemáticos y ejemplo.
* **Operaciones de relación:** listado de operadores relacionales y ejemplo.
* **Operadores lógicos (booleanos):** listado de operadores lógicos.
* **¿Qué es una data frame?:** es una estructura de datos de dos dimensiones (rectangular) que puede contener datos de diferentes tipos, por lo tanto, es heterogénea. Esta estructura de datos es la más usada para realizar análisis de datos, y es la estructura con la cual trabajaremos en esta sesión.
* **Comando $:** se utiliza para vincular una variable con un objeto.
* **Función table():** se utiliza para tabular un objeto.

# Importación de base de datos

**R** tiene distintos paquetes para importar datos.

Existen distintos paquetes para cargar archivos con distintas extensiones.

* Para importar archivos delimitados (.csv, .txt, .tab), existe el paquete **readr**. Dentro de las funciones de ese paquete, la función **read\_csv()**, permite la importación de archivos con la extensión “.csv”. (**ejemplo**)
* El paquete readxl nos permite importar archivos de Excel. Una de sus funciones es **read\_excel(). (ejemplo)**

Tanto las funciones de **readr** como **readxl** tienen argumentos que permiten setear distintas opciones para la importación.

* **readr**
  + Cambiar nombres de columnas.
  + Saltarse líneas al momento de importar.
  + Recodificar valores perdidos.
* **readxl**
  + Leer un rango específico de la hoja de cálculo.
  + Cambiar nombres de columnas.
  + Saltarse líneas al momento de importar.
  + Recodificar valores perdidos.

Para importar archivos guardados desde STATA o SPSS existe el paquete **haven**.

Al igual que los paquetes **readxl** o **readr** tiene funciones bastantes explícitas respecto de lo que hacen:

* **SPSS**: **read\_sav()** permite importar archivos “.sav”.
* **STATA: read\_dta()** permite importar archivos “.dta”.

Ejemplo: carga desde una carpeta o desde un sitio web.

Las bases de datos de R, se guardan en archivos con extensión ”.RData”.

Para importar un archivo con extensión “.RData” se utiliza la función **load()**.

(ejemplo)

**(ejercicios)**

La importación de un archivo de R no requiere una asignación de valor.

Estos archivos pueden incluir más de un objeto.

Ahora realizaremos la importación de la base de datos que utilizaremos durante esta sesión.

(cargar bases)

¿Cómo se puede revisar la estructura de la base?

(Función str).

Ahora que importamos la base de datos, trabajaremos con la manipulación del data frame.

# Manipulación de data frames

Para la manipulación de los data frames existe un paquete llamado **dplyr**.

Este paquete fue desarrollado por Hardley Wickman de RStudio.

Es una versión optimizada de un paquete anterior llamado **plyr**.

El paquete **dplyr** proporciona una “gramática” (particularmente verbos) para la manipulación y operaciones con data frames.

Los paquetes **dplyr**, **readr**, **readxl** y **haven** forman parte del universo **tidyverse**.

Durante esta sesión veremos siete funciones del paquete **dplyr** que se utilizan para la manipulación de data frames.

* **select**: selecciona y devuelve un conjunto de columnas.
* **filter:** selecciona y devuelve un conjunto de filas según una o varias condiciones lógicas.
* **arrange:** reordena filas de un data frame.
* **rename**: renombra variables en un data frame.
* **mutate**: añade nuevas variables o transforma variables existentes.
* **group\_by:** agrupa filas de un data frame.
* **summarise:** genera resúmenes estadísticos.

Estas funciones serán algunas de nuestras mejores compañeras y siempre nos acompañarán en nuestro camino de programación.

Todas estas funciones tienen en común una serie de argumentos:

* El primer argumento es el data frame a manipular.
* Los otros argumentos describen que hacer con el data frame especificado en el primer argumento.
* El valor de retorno de la función es un nuevo data frame.

Como veremos en esta sesión, una de las principales ventajas de estas funciones es que podemos referirnos a las columnas en el data frame directamente sin utilizar el operador “**$”**, es decir, solo con el nombre de la variable.

## Función select():

La primera función que veremos es la función **select()**.

Esta función fue vista en la sesión anterior, y es utilizada para seleccionar columnas dentro de un data frame.

El resultado de esta función corresponde a un data frame que solo incluye las variables seleccionadas.

El orden definido en la selección de variables, es el orden que tendrán las columnas en el resultado.

La selección de columnas puede ser realizada según nombre o según índice (indicador) de las variables.

(ejemplo).

Esta función permite utilizar varias funciones auxiliares para indicar las columnas a extraer:

* **Al incluir el signo menos (“-“) antes de la selección:** se indica que columnas **no** serán extraídas.
* **Al incluir “:” entre los indicadores de columnas:** permite seleccionar un rango de variables.
* **La función “starts\_with”:** selecciona todas las columnas que comienzan con el patrón indicado.
* **La función “ends\_with”:** selecciona todas las columnas que terminen con el patrón indicado.
* **La función “contains”:** selecciona las columnas que posean el patrón indicado.
* **La función “matches”:** funciona similar a contains, pero permite poner una **expresión regular**.

Una expresión regular corresponde a una secuencia de caracteres que forma un patrón de búsqueda.

* **La función “everything”:** sirve como “bolsa” para seleccionar todas las variables no seleccionadas explícitamente.
* **La función “one of”:** selecciona variables cuyo nombre están en un grupo de nombres.

**(ejercicios)**

## Función filter():

Esta función se utiliza para filtrar un data frame según alguna condición a definir.

Así como la función **select** es utilizada para seleccionar columnas de un data frame, la función **filter** puede ser entendida como la selección de filas de un data frame.

(ejemplo)

¿Es posible filtrar un data frame por más de una condición?.

Sí, es posible utilizando operadores lógicos.

La función seleccionará aquellas filas en donde la condición sea verdadera.

(ejemplo)

**(ejercicios)**

## Función arrange():

Como se vio en la sesión anterior, esta función se utiliza para reordenar las filas de un data frame.

* El orden se puede realizar según una o más columnas.
* El reordenamiento se realiza según el orden de las columnas seleccionadas.
* Por defecto, el orden es ascendente.
* Para ordenar de manera descendente se debe utilizar la función auxiliar **desc()**.

**(ejercicios)**

## Función rename():

Si queremos **renombrar** una variable (columna) de un data frame, existen múltiples formas y varias muy complejas.

La función **rename** permite modificar el nombre de una variable de una manera sencilla, sin editar el resto de variables.

Para obtener el nombre de las variables en un data frame se utiliza la variable **names()**.

(Ejemplo: renombrar sin rename y con rename)

**(ejercicios)**

## Función mutate():

Hasta ahora hemos visto como **renombrar** variables, como **seleccionar** columnas, como **filtrar** filas y como **reordenar** filas de un data frame. ¿Qué pasa si queremos realizar transformaciones sobre los valores de un data frame o crear nuevas variables?

La función **mutate** tiene la finalidad de realizar transformaciones sobre los valores de un data frame.

(ejemplo)

Esta función permite realizar transformaciones sobre más de una variable del data frame.

(ejemplo)

Con esta función también es posible crear nuevas variables en el data frame. Estas nuevas variables se agregan al final del data frame.

(ejemplo)

**(ejercicios)**

# Encadenamiento de instrucciones

Las variables que hemos visto, nos ayudan a manipular un data frame, ¿pero qué sucede si queremos utilizar más de una de las funciones previas sobre un data frame?

El paquete **dplyr** incluye una función para poder encadenar distintas funciones. La función **%>%** (pipe) es la que permite encadenar un conjunto de funciones.

Para utilizar un “pipe”, primero se selecciona el data frame sobre el cual trabajar, y luego se encadenan la funciones para la manipulación.

Al definir la base al principio de la sentencia, al aplicar las funciones ya no es necesario definir las variables.

(ejemplo)

**(ejercicios)**