Diseño Experimental y análisis de Datos

Table of contents

Curso MCV502 / BCM634 / BIO625

Esta es la página web del curso Diseño experimental y análisis de datos, para el año 2022.

Profesores

- Jorge Valdes (jorge.valdes@unab.cl)
- Juan Ugalde (juan.ugalde@unab.cl)

Este curso se dicta todos los días Miercoles en los modulos 7-12 (entre las 14:00 - 18:25hrs).

Esta página contiene el material práctico que iremos desarrollando en el curso, principalmente en la exploración y visualización de datos.

Bienvenidos!!!!!!!

Part I.

Introducción a R

El trabajo práctico de este curso se realizará usando los softwares R y Rstudio. Lo primero que haremos sera instalar R y RStudio para poder configurar el ambiente de trabajo.

R es un lenguaje de programación el cual permite hacer análisis estadisticos, manipular datos y generar visualizaciones. Es un lenguaje que permite ir desde una tabla cruda hasta llegar a visualizaciones complejas e incluso dashboards. **R** es un *software* de libre acceso con una comunidad que esta constantemente generando nuevas funcionalidades.

RStudio es la forma más popular (pero no es la única!) para trabajar con R, escribir *scripts*, realizar visualizacionees, entre otras tareas. La ventaja es que hace el uso de R mucho fácil e interactivo, ademas de facilitar otras tareas como escribir reportes e incluso paginas web (como la de este curso!). **RStudio** esta disponible para Linux, MacOS y Windows.

Lo primero que haremos sera instalar **R** y **RStudio** si áun no los tienen, y veremos algunos detalles simples de su configuración.

1. Configuración de R y RStudio

Lo primero que es necesario hacer es instalar R y RStudio, en ese orden!

Nota para usuarios de MacOS

Antes de instalar R, la recomendación es instalar primero Xcode y XQuartz.

1.1. Instalar R

R se puede descargar desde la pagina del Comprehensive R Archive Network o CRAN. Desde ese lugar se puede descargar la version de R e instalar según su sistema operativo.

1.2. Instalar RStudio

En la página de descarga de RStudio, elegir la versión según el sistema operativo que esten utilizando.

1.3. Resolver problemas

Tanto R como RStudio tiene paginas web en donde se pueden encontrar respuestas a algunas preguntas. En el caso de R estan *"Frequently Asked Questions"*(FAQ) de R, o para Windows R for Windows FAQ.

Otra fuente de información son paginas web y foros como Stack Overflow RStudio community. Sin embargo, lo más probable es que tus dudas ya se encuentren resueltas en la web, asi que primero haz una búsqueda en **Google** usando palabras clave o **copia y pega en el buscador el mensaje de error** que te aparezca en la consola de R. **Consejo**: haz tus búsquedas y consultas en inglés, tendrás acceso a más y mejor información.

2. Configuración del ambiente de trabajo en R

En R puedes hacer operaciones básicas como una suma o división simplemente escribiendo la instrucción en la consola, e.g.,`27 + 4`, `3 / 4`. Para hacer cálculos más complejos como calcular la mediana de un set de datos o leer una tabla de valores, usamos **funciones** que son programas predeterminados en R. Así mismo, para cálculos aún más complejos como hacer un análisis de expresión diferencial de genes o generar un gráfico, usamos **paquetes** que son un amplio conjunto de funciones preprogramadas que permiten hacer este tipo de análisis sin necesidad de tener habilidades de programación.

2.1. Instalar y cargar paquetes en R

Una de las grandes ventajas de R es que existen numerosos paquetes que permiten agregar funcionalidades adicionales al lenguaje, incluyendo análisis de secuencias, herramientas avanzadas de visualización, etc.

Los principales repositorios de paquetes son: CRAN, Bioconductor, y GitHub. Durante este curso vamos a ir utilizando diferentes paquetes que vamos a ir instalando. A modo de ejemplo estas son las instrucciones para instalar el paquete `tidyverse`, para manipular tablas y realizar gráicos.

```
install.packages("tidyverse")
```

Una vez instalado, cada vez que vayamos a utilizar este paquete, al inicio de la sesión o el script, escribimos

```
library(tidyverse)
```

2.2. Definir directorio de trabajo

Cada vez que se abre una nueva sesión de R, lo segundo que debemos hacer (después de cargar los paquetes necesarios) es definir el directorio de trabajo en nuestra computadora.

Para esto lo ideal es definir una carpeta de trabajo (por ejemplo `MCV502`), y dirigir a RStudio a utilizar esta carpeta como lugar de trabajo.

RStudio:

1. Abre RStudio y dirígete a la pestaña `Files`.
2. Navega por los directorios de tu computador hasta la ubicación de la carpeta que hayas definido (por ejemplo `MCV502`)
3. Selecciona la carpeta, una vez dentro puedes definirla como tu **directorio de trabajo**. Para ello, haz clic en `More` -> `Set As Working Directory`.

3. Intro a Tidyverse

3.1. Explorando un set de datos

Antes de empezar cualquier analisis, especialmente si estamos iniciando un script desde cero, tenemos que llamar a las librerias que vamos a utilizar. Como estuvimos viendo durante la clase, la libreria que vamos a utilizar es *tidyverse*, la cual cargamos con el siguiente comando:

```
library(tidyverse)
```

```
-- Attaching packages --------------------------------------- tidyverse 1.3.1 --

v ggplot2 3.3.6     v purrr   0.3.4
v tibble  3.1.7     v dplyr   1.0.9
v tidyr   1.2.0     v stringr 1.4.0
v readr   2.1.2     v forcats 0.5.1

-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
```

Primero vamos a explorar un set de datos que ya viene incluido en la libreria, y que tiene valores de eficiencia de combustible para diferentes modelos de automoviles

```
# Asignmamos la tabla mpg a una nueva variable que vamos a llamar
# tabla_mpg
tabla_mpg = mpg
```

Podemos ver el contenido de la tabla y el tipo de variables escribiendo.

```
tabla_mpg
```

```
# A tibble: 234 x 11
   manufacturer model      displ  year   cyl trans drv     cty   hwy fl    cl
   <chr>        <chr>      <dbl> <int> <int> <chr> <chr> <int> <int> <chr> <c
 1 audi         a4           1.8  1999     4 auto~ f        18    29 p     co
 2 audi         a4           1.8  1999     4 manu~ f        21    29 p     co
 3 audi         a4           2    2008     4 manu~ f        20    31 p     co
 4 audi         a4           2    2008     4 auto~ f        21    30 p     co
 5 audi         a4           2.8  1999     6 auto~ f        16    26 p     co
 6 audi         a4           2.8  1999     6 manu~ f        18    26 p     co
 7 audi         a4           3.1  2008     6 auto~ f        18    27 p     co
 8 audi         a4 quattro   1.8  1999     4 manu~ 4        18    26 p     co
 9 audi         a4 quattro   1.8  1999     4 auto~ 4        16    25 p     co
10 audi         a4 quattro   2    2008     4 manu~ 4        20    28 p     co
# ... with 224 more rows
```

Podemos ver que tiene varias columnas con distintas caracteristicas en donde:

- chr, indica caracteres
- dbl, números reales
- int, números enteros
- dttm, fecha y hora (en esta tabla de ejemplo no tenemos ese tipo de datos)

3.2. Funciones mas comunes para filtrar tablas

Algunas de las funciones más comunes en el paquete de análisis son:

- *select()*: extraer columnas.
- *filter()*: extraer filas según condiciones.
- *mutate()*: crear nuevas columnas usando la información de otras columnas.
- *group_by()* y *summarize()*: cálcula estadísticas en datos agrupados.
- *count()*: conteo de datos.

3.2.1. Filtrando por columnas

Una de las primeras cosas que podemos hacer es filtrar la tabla por algún valor en alguna de columna de interés. Para esto ocupamos el comando *filter*

```
# En este caso estamos filtrando la tabla_mpg, en donde la columna year sea menor al año
filter(tabla_mpg, year < 2000)
```

```
# A tibble: 117 x 11
   manufacturer model      displ  year   cyl trans drv     cty   hwy fl    class
   <chr>        <chr>      <dbl> <int> <int> <chr> <chr> <int> <int> <chr> <chr>
 1 audi         a4           1.8  1999     4 auto~ f        18    29 p     comp~
 2 audi         a4           1.8  1999     4 manu~ f        21    29 p     comp~
 3 audi         a4           2.8  1999     6 auto~ f        16    26 p     comp~
 4 audi         a4           2.8  1999     6 manu~ f        18    26 p     comp~
 5 audi         a4 quattro   1.8  1999     4 manu~ 4        18    26 p     comp~
 6 audi         a4 quattro   1.8  1999     4 auto~ 4        16    25 p     comp~
 7 audi         a4 quattro   2.8  1999     6 auto~ 4        15    25 p     comp~
 8 audi         a4 quattro   2.8  1999     6 manu~ 4        17    25 p     comp~
 9 audi         a6 quattro   2.8  1999     6 auto~ 4        15    24 p     mids~
```

```
10 chevrolet    c1500 sub~   5.7  1999     8 auto~ r        13    17 r     su
# ... with 107 more rows
```

Tambien podemos elegir un rango de años

```
filter(tabla_mpg, year < 2000 | year > 2007)
```

```
# A tibble: 234 x 11
   manufacturer model      displ  year   cyl trans drv     cty   hwy fl    cl
   <chr>        <chr>      <dbl> <int> <int> <chr> <chr> <int> <int> <chr> <c
 1 audi         a4           1.8  1999     4 auto~ f        18    29 p     co
 2 audi         a4           1.8  1999     4 manu~ f        21    29 p     co
 3 audi         a4           2    2008     4 manu~ f        20    31 p     co
 4 audi         a4           2    2008     4 auto~ f        21    30 p     co
 5 audi         a4           2.8  1999     6 auto~ f        16    26 p     co
 6 audi         a4           2.8  1999     6 manu~ f        18    26 p     co
 7 audi         a4           3.1  2008     6 auto~ f        18    27 p     co
 8 audi         a4 quattro   1.8  1999     4 manu~ 4        18    26 p     co
 9 audi         a4 quattro   1.8  1999     4 auto~ 4        16    25 p     co
10 audi         a4 quattro   2    2008     4 manu~ 4        20    28 p     co
# ... with 224 more rows
```

Ademas de estar introduciendo la función de filtro (*filter*), también estamos viendo como podemos hacer busquedas por ciertos patrones. Por ejemplo:

- year $<$ 2000: En este caso el valor de year es menor a 2000
- El simbolo | indica que son dos posibles condiciones. En el ejemplo anterior tenemos que sea menor al año 2000 **o** mayor al año 2007.
- El simbolo **&** indica que se tienen que cumplir todas las condiciones. Por ejemplo:

```
filter(tabla_mpg,
       (year < 2000 | year > 2007) & model == "a4"
)
```

```
# A tibble: 7 x 11
  manufacturer model displ  year   cyl trans      drv     cty   hwy fl    class
  <chr>        <chr> <dbl> <int> <int> <chr>      <chr> <int> <int> <chr> <chr>
1 audi         a4      1.8  1999     4 auto(l5)   f        18    29 p     compa~
2 audi         a4      1.8  1999     4 manual(m5) f        21    29 p     compa~
3 audi         a4      2    2008     4 manual(m6) f        20    31 p     compa~
4 audi         a4      2    2008     4 auto(av)   f        21    30 p     compa~
5 audi         a4      2.8  1999     6 auto(l5)   f        16    26 p     compa~
6 audi         a4      2.8  1999     6 manual(m5) f        18    26 p     compa~
7 audi         a4      3.1  2008     6 auto(av)   f        18    27 p     compa~
```

En este caso estamos aplicando dos filtros. Que el año sea menor al 2000 y mayor al 2007, y que el modelo sea igual a `a4`. El parentesis es muy importante, porque agrupa las condiciones a filtrar.

3.2.2. Seleccionando columnas

Tambien podemos seleccionar columnas de interes, luego de aplicar algún filtro (o viceversa). Por ejemplo:

```
filter(tabla_mpg,
       (year < 2000 | year > 2007) & manufacturer == "audi") %>%
  select(model, year)
```

```
# A tibble: 18 x 2
   model       year
   <chr>      <int>
```

```
 1 a4          1999
 2 a4          1999
 3 a4          2008
 4 a4          2008
 5 a4          1999
 6 a4          1999
 7 a4          2008
 8 a4 quattro  1999
 9 a4 quattro  1999
10 a4 quattro  2008
11 a4 quattro  2008
12 a4 quattro  1999
13 a4 quattro  1999
14 a4 quattro  2008
15 a4 quattro  2008
16 a6 quattro  1999
17 a6 quattro  2008
18 a6 quattro  2008
```

En este caso estamos filtrando por el año y el fabricante, y luego seleccionando dos columnas, el modelo y año.

Adicionalmente ocupamos el simbolo %>%, para hacer operaciones consecutivas utilizando **pipes**. De esta manera el output de una función (el filtro) es el input de la siguiente (select).

Esto nos permite concatenar diferentes pasos de filtrado, agrupar, selección, etc, en un solo comando.