Análisis Exploratorio de Datos y Visualización. GCID, 2022/2023.

Trabajar con ggplot2 y dplyr: CO_2 en aulas

Objetivo del trabajo

El objetivo de esta actividad es adiestrarte con las herramientas de visualización de datos que estamos trabajando en la asignatura, especialmente **ggplot2** y **dplyr**. Trabajarás con los datos de los sensores de CO_2 de las aulas del Edificio de Informática y Matemáticas. Con estos datos te pedimos que prepares visualizaciones que ayuden a explorar la evolución de las condiciones de las aulas a lo largo del tiempo, así como a formular y validar hipótesis sobre esa evolución. Como herramienta para el tratamiento de los datos, utilizarás la plataforma R a plena potencia: RStudio, RMarkdown, ggplot2 y dplyr.

Este trabajo lo has de realizar **en un grupo de hasta tres personas.**

Entrega de la tarea

Deberás subir el resultado de tu trabajo en el plazo que marque el Campus Virtual.

Tendrás que entregar un único archivo ZIP que contenga, al menos, estos dos documentos:

- Un **fichero RMarkdown**, con el código fuente en R y los textos que hagan falta (en Markdown).
- El **documento final** generado desde el RMarkdown (en HTML o PDF, como prefieras).

Recuerda que no basta con que envíes el código en R con las gráficas. En tu documento tienes que explicar qué proceso has seguido y qué decisiones has tomado. Describe las conclusiones a las que has llegado.

Fuente de datos: registros de los sensores de CO_2

Para desarrollar este trabajo contarás con un conjunto de datos que proviene de los sensores que están instalados en las aulas del Edificio de Informática y Matemáticas. Los datos están disponibles en ficheros de texto delimitado. Hay un fichero por cada aula.

En cada fichero se registran los niveles de CO_2, temperatura y humedad medidos en el aula durante un periodo de tiempo (en este caso, todo el año 2022). Se realizan observaciones sucesivas cada 5 minutos, aproximadamente. Cada línea del fichero tiene una observación, con la anotación de la fecha y hora de la observación y los datos medidos en ese momento.

Los ficheros están disponibles en la carpeta OneDrive compartida, en el directorio **«co2-dis-2022»**.

Primera fase: visualizar los datos de CO_2 de un aula

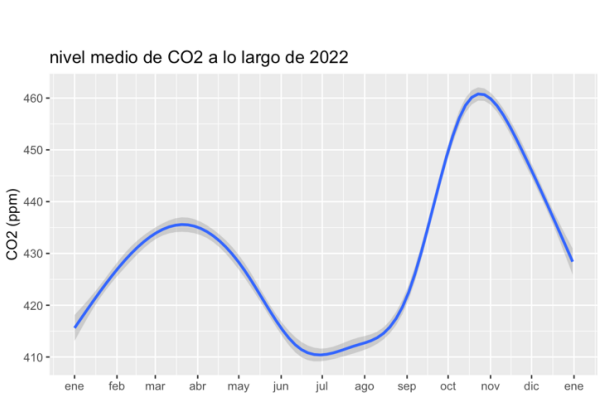
Como primera fase del trabajo, queremos que escojas uno de los ficheros de aulas, generes un *data frame* en R y explores visualmente los datos. Queremos que te centres en los niveles de CO_2. **¿Hay alguna relación entre los niveles de CO_2 y la hora del día? ¿O el día de la semana? ¿O la época del año?**

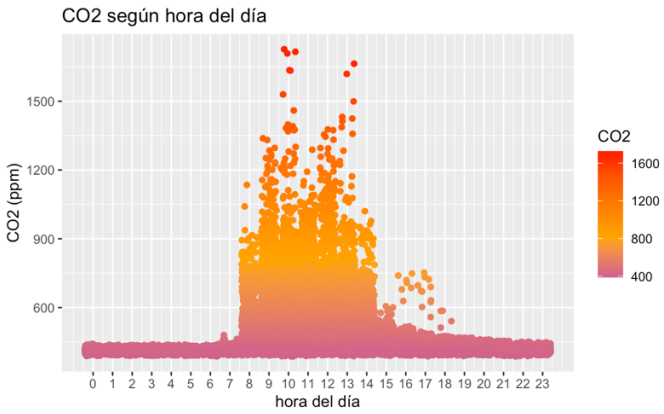
Tu misión es confeccionar unas visualizaciones que evidencien esas tres relaciones. Puedes utilizar la visualización que consideres más apropiada, siempre que sea con ggplot2. Recuerda que tienes visualizaciones básicas muy expresivas, como `geom_point`, `geom_jitter` y `geom_smooth`. Investiga las posibilidades que te da ggplot2 y encuentra las mejoras formas de demostrar las relaciones que estás investigando.

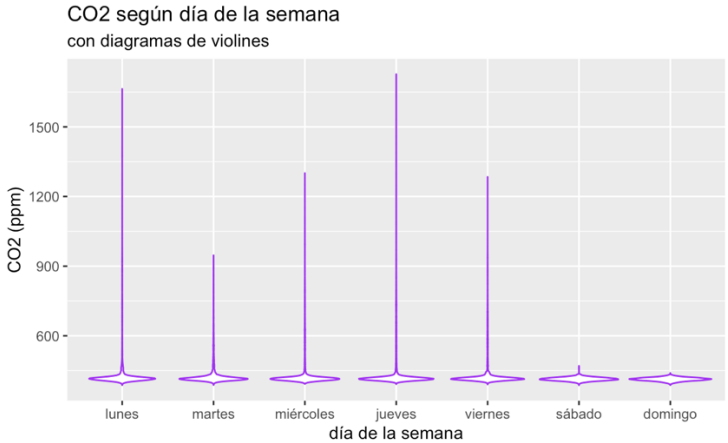
En el anexo a este documento te damos varios consejos sobre cómo implementar tu código.

Ejemplos de gráficas

Aquí tienes algunos ejemplos de gráficas que puedes obtener. Te las mostramos para que puedas hacerte una idea de lo que buscamos para esta fase del trabajo. Puedes intentar replicarlas, o puedes tratar de crear otras diferentes. ¡Échale creatividad!







Segunda fase: agrupar y resumir los datos

Es posible que hayas notado que los gráficos tardan unos segundos en generarse. Es comprensible, si tenemos en cuenta que cada aula tiene unas 100 000 observaciones anuales. Podemos reducir bastante ese tiempo si trabajamos con datos de resumen, tales como el valor medio de cada día, de cada semana o de cada hora, según nos interese.

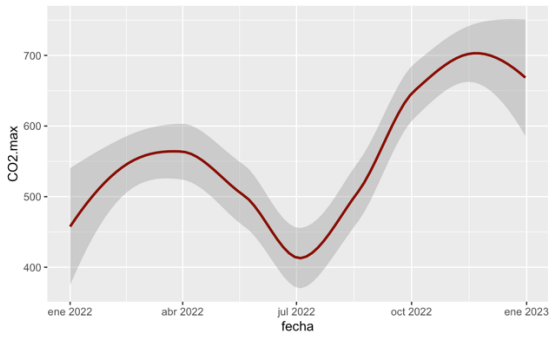
Esta reducción de datos también nos puede servir para visualizar información que no tenemos directamente con los datos en bruto, como puede ser el valor **máximo** diario del CO_2.

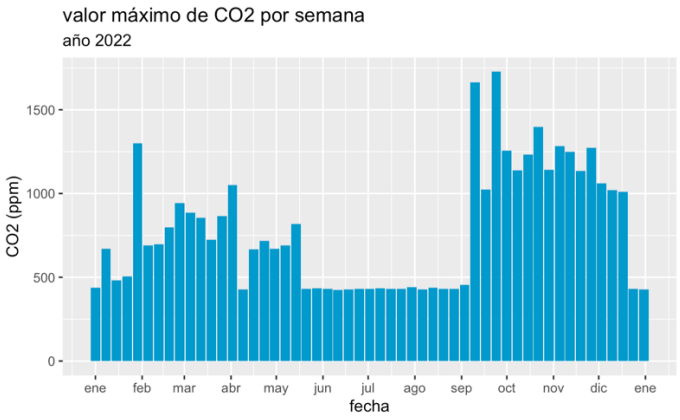
Hay muchas formas de implementar el resumen. Para este trabajo te vendrá bien la combinación de `group_by()` y `summarise()` de la biblioteca dplyr. En el anexo te explicamos algo más.

Prueba a obtener un par de gráficas con datos resumidos. ¿Notas diferencias en el tiempo de ejecución?

Ejemplos de gráficas

Aquí hay un par de ejemplos de visualización del valor máximo, usando datos resumidos:

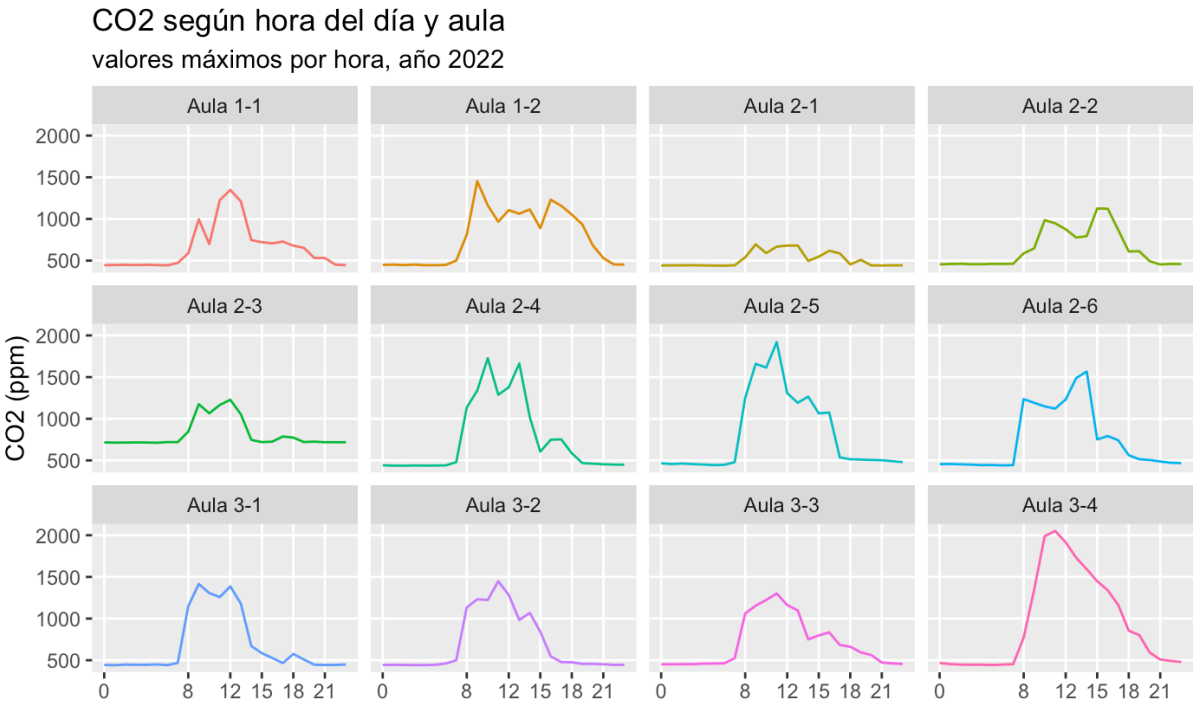




Tercera fase: combinar los datos de todas las aulas

Una vez que tengas dominada la primera fase con un aula, tendrás que integrar la totalidad de los datos de todas las aulas. Tendrás que leer todos los ficheros de texto y combinarlos en un único *data frame* en R.

Con los datos ya integrados, queremos que prepares una o varias visualizaciones que muestren las variaciones entre aulas: **¿hay diferencias entre el patrón de emisión de CO2 de las distintas aulas, en las tres dimensiones que estamos considerando?** (hora del día, día de la semana, fecha). Prepara alguna visualización que permita comparar fácilmente todas las aulas. Por ejemplo, un gráfico facetado como el de la figura adjunta.



Resumir los datos. En esta fase del trabajo, es mucho más importante tener los datos resumidos, ya que el volumen de observaciones es considerable: más de un millón de filas. Si no quieres pasarte un buen rato esperando a que ggplot2 visualice tus gráficos, tendrás que resumir los datos siguiendo alguna estrategia: por ejemplo, el valor medio/máximo de CO_2 en un determinado periodo. ¿Se nota la diferencia de tiempo de ejecución entre resumir los datos y trabajar con los datos en bruto?

Reto: mapa de calor

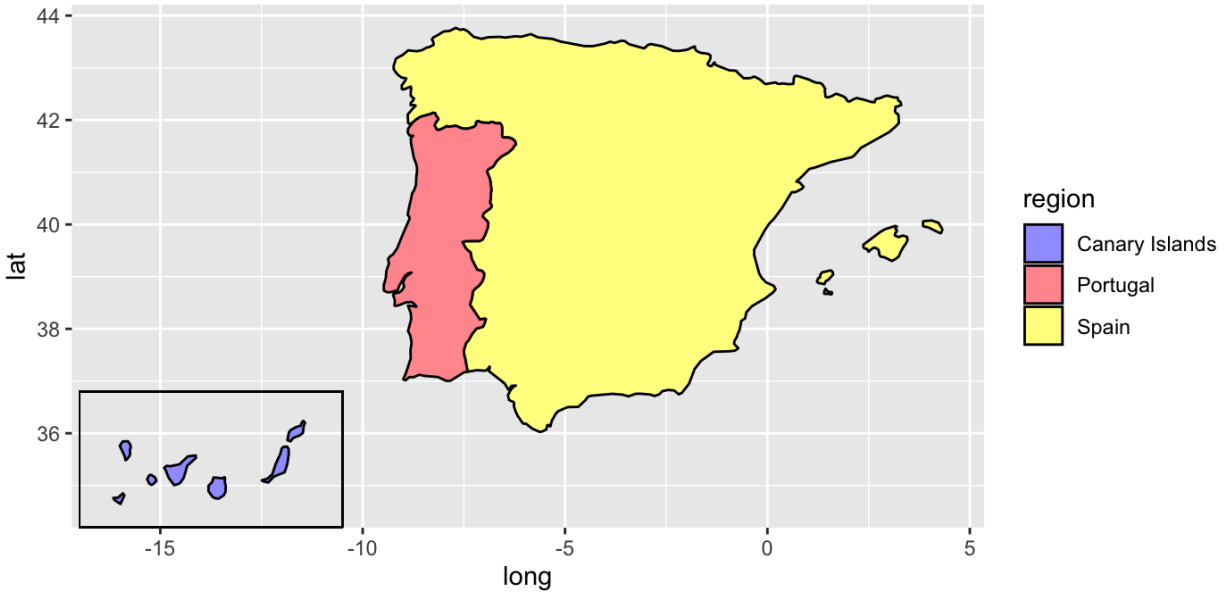
Si completas todas las fases anteriores, te planteamos una actividad final que te dará puntos adicionales. Consiste en preparar una gráfica lo más parecida a esta:

Como ves, se trata de construir un *mapa de calor* de los valores de CO_2, organizados por día de la semana y por hora del día. Observa bien el ejemplo, porque tiene bastantes elementos de personalización sobre el comportamiento por defecto de ggplot2, sobre todo en lo que se refiere a las escalas, al **tema** y a los **paneles**. Las personalizaciones más globales se resuelven con las funciones `theme()` y `labs()`. Las adaptaciones en los ejes y escalas suelen resolverse con alguna función `scale_XXX_YYY()`.

Reto extra: mapa de España

Si te animas, te proponemos un ejercicio adicional, sencillo pero vistoso.

En el laboratorio hemos utilizado el conjunto de datos geográficos que hay en `map_data("world")`. Conseguimos visualizar el mapa de España, mediante la función `filter()`. Ahora te proponemos que consigas visualizar un mapa como este:

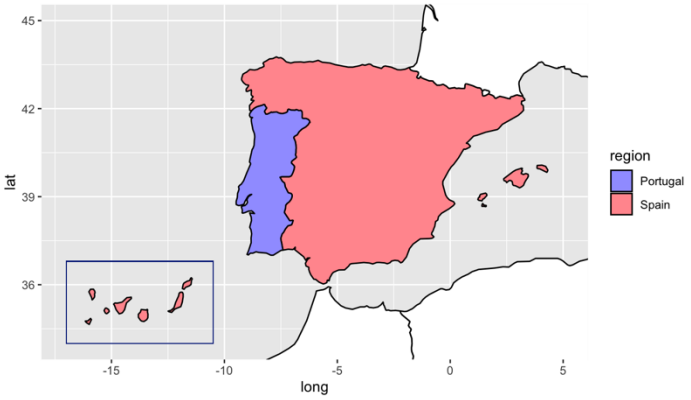


La idea es que Canarias se muestre en un recuadro cercano a la Península, más arriba de sus coordenadas reales.

Como punto de partida, usa el código de muestra que elaboramos en el laboratorio. Para transformar los datos, apóyate en las operaciones de **dplyr**.

La solución es bien simple. Solo tienes que mover a Canarias.

Si aún te quedan ganas de más, puedes tratar de representar el mismo mapa de la figura, en el que también se vean *parcialmente* los otros países de los alrededores (Francia, Andorra, Marruecos, Argelia) en color neutro. También se puede aprovechar para etiquetar a Canarias como parte de España.
Usa la figura adjunta como modelo de lo que se pretende conseguir.



Anexo: consejos de implementación

En este apartado te damos algunas guías sobre cómo implementar las partes del código en R que requieren cierto conocimiento del entorno de programación. Lee bien este anexo y aplica los consejos en tu código.

Lectura de los ficheros de datos

Los ficheros tienen los campos separados por tabuladores y los números usan la coma como separador decimal. Además, están codificados con el juego de caracteres «latin1». Usa los parámetros `sep`, `dec` y `encoding` para leer con éxito el fichero.

También es conveniente que cambies los nombres de las columnas del *data frame*, para que te resulten más manejables en el código. Esto lo puedes resolver con alguna operación de dplyr (ej. `mutate()` o `rename()`). También lo puedes resolver en el mismo momento de la lectura del fichero, con el parámetro `col.names`.

Puede ser que algún fichero tenga algunas filas con datos en un formato anómalo. Esto puede manifestarse como valores NA o como columnas que no se interpretan con el tipo adecuado. Es posible que tengas que tratar manualmente algunos de esos datos.

Fechas y horas

Necesitamos procesar las columnas de fecha y hora para poder tratarlas en *ggplot2*. Por defecto, `read.csv()` las va a interpretar como textos, lo cual nos limita bastante la producción de gráficas.

Una forma cutre, pero eficaz, de obtener una columna para la hora consiste en tomar los dos primeros caracteres de la columna de la hora: `substr(co2.dis$hora,1,2)`. Si además los convertimos en número entero con `as.integer()`, mejor aún. Esta transformación cutre te permite jugar desde el primer momento con una variable para la hora.

Como vas a necesitar un tratamiento más general de fechas y horas, te recomendamos la biblioteca **lubridate** (pertenece al paquete *tidyverse*). Tiene funciones muy sencillas para convertir textos que contienen fechas y horas en cualquier formato. Por ejemplo, puedes leer una fecha en formato día-mes-año-hora-minuto-segundo con `dmy_hms(fecha)`. La función reconoce cualquier separador (guiones, barras, etc.). Otra función interesante de *lubridate* para este trabajo es `wday(fecha)`, que te devuelve el día de la semana asociado a una fecha.

Escalas y etiquetas

Un buen gráfico, con calidad profesional, tiene unas etiquetas que explican adecuadamente los datos. No descuides ese aspecto. Observa los ejemplos gráficos que te hemos dado y verás que todos utilizan unas etiquetas y escalas bien adaptadas a la intención de cada gráfico.

Utiliza la función `labs()` y similares para darle un título a la gráfica. Dale nombre a los ejes. Utiliza las funciones para manipular las escalas, ej. `scale_x_date()`, `scale_y_continuous()`

y demás, que te permiten un control fino sobre las divisiones de la escala (`breaks`) o las etiquetas que se usan (`labels`).

Si quieres usar nombres en español para los días de la semana o los meses, juega con el parámetro `locale` que viene con las operaciones de *lubridate* y otras funciones.

Agrupar y resumir con group_by() y summarise()

Si tienes un conjunto de datos y quieres resumirlos, dplyr es tu amiga. La biblioteca ofrece operaciones muy potentes para agrupar los datos según cualquier criterio y resumirlos según las típicas operaciones (recuento, media, máximo, mínimo, percentil...). Un mecanismo muy habitual es combinar `group_by()` y `summarise()`.

- **group_by().** Esta función se utiliza para agrupar datos por una o más variables. Por ejemplo, si tienes un *data frame* con datos de clientes, puedes agrupar los datos por la variable «provincia» para ver cuántos clientes hay en cada provincia.
- **summarise().** Esta función se utiliza para resumir los datos dentro de cada grupo creado previamente con `group_by()`. Por ejemplo, puedes usar `summarise()` para calcular la edad media de tus clientes de cada provincia. `summarise()` aplica una función de resumen a los datos de cada grupo (por ejemplo, media, mediana, máximo, mínimo, etc.) y devuelve un nuevo *data frame* con los datos resumidos.

El ejemplo con los clientes podría tener este aspecto:

```
clientes <- read.csv(…)
# Supongamos que 'clientes' tiene unas columnas 'provincia' y 'edad'.
# Generamos un objeto 'resumen' con la media de edad y número de clientes,
# agrupados por provincias
resumen <- clientes %>%
  group_by(provincia) %>%
  summarise(edad_media = mean(edad), num_clientes = n())
```

Combinar todos los ficheros

Para poder trabajar con los datos de todos los sensores de manera integrada, necesitamos juntar en un único *data frame* el contenido de todos los ficheros.

Una posible estrategia para conseguirlo es leer cada fichero en un objeto distinto, y luego combinar todos esos objetos mediante la operación `dplyr::bind_rows()` (combinar por filas).

A continuación te damos un esqueleto de código que implementa esa estrategia.

```
# Esta función recibe un nombre de fichero y devuelve un data frame
lee_co2 <- function (filename) {
  df <- read.csv(filename, …otros parámetros… )
  # Añade una nueva columna con el nombre del fichero
  df$aula <- filename
```

```
  return(df)
}

# obtengo los ficheros del directorio
# "files" es un vector con los nombres de los ficheros *.txt
setwd(…el directorio donde están los ficheros…)
files <- list.files(pattern="\\.txt$")

# Aplico la función lee_co2() a la lista de ficheros.
# Devuelve una lista de 'data frames' con los datos leídos.
# La función map() es equivalente a la de Python.
aulas <- purrr::map(files,lee_co2)

# Combino los data frames en un único objeto
jaulas <- dplyr::bind_rows(aulas)
```