Autoria i Llicència

Profesorat responsable de la assignatura: Antoni Pérez Navarro
Personal docent col·laborador de l'assignatura: Xavier Balagué Boldú

Presentació

M.2957 Anàlisi de dades geoespacials

Competències:

En el context del Màster de Data Science, aquesta PAC se centra en el desenvolupament de totes les competències clau vistes al llarg de l'assignatura.

Objectius:

Els objectius específics d'aquesta pràctica són:

1. Aplicar els conceptes i competències obtingudes durant el desenvolupament de l'assignatura.
2. Aprendre a obtenir i analitzar dades geogràfiques mitjançant l'ús de *dashboards*.

Descripció de l'activitat:

Criteris de valoració:

La ponderació dels exercicis és la següent:

- Exercici 1: 20%
- Exercici 2: 20%

- Exercici 3: 30%
- Exercici 4: 30%

S'assigna una ponderació a cada exercici segons la seva rellevància. Es valorarà la precisió de les solucions, així com la claredat i pertinència de les respostes. És necessari lliurar un document que indiqui pas a pas el desenvolupament de cada exercici.

Format i data de lliurament:

5 de juny de 2025

- És necessari lliurar un únic document Word, Open Office o PDF (preferiblement aquest últim) amb les respostes a les preguntes.
- En aquesta activitat no està permès l'ús d'eines d'intel·ligència artificial ni l'ajuda de cap acadèmia, professorat particular ni cap altra persona.

  En el pla docent i en la web sobre integritat acadèmica i plagi de la UOC (`https://campus.uoc.edu/estudiant/microsites/plagi/es/index.html`) trobareu informació sobre què es considera conducta irregular en l'avaluació i les conseqüències que pot tenir.

Certificat d'autoria

S'haurà d'incloure el següent text en el document a lliurar:

"Declaro que ostento l'autoria total i plena de totes les tasques que es duen a terme en el present document. Sóc l'única persona que ha elaborat cada exercici. No he compartit els enunciats amb ningú i l'única ajuda que he rebut ha estat a través de l'aula de la UOC i el seu professorat i he citat de manera explícita els recursos externs que he usat en la preparació de la PRA."

La present pràctica té com a finalitat que apliqueu i poseu en pràctica els coneixements i habilitats adquirits al llarg del curs en l'assignatura d'Anàlisi de Dades Geoespaciales.

A través de l'anàlisi i la visualització de dades de la qualitat de l'aire de la ciutat de Madrid, tindreu l'oportunitat de treballar amb dades reals i temporals, utilitzant eines de programari com *R* i paquets específics com *Shiny*, *Leaflet* i *ggplot2*, entre altres. L'objectiu és que comprengueu millor els patrons de contaminació a la ciutat, identifiqueu àrees de major risc i analitzeu l'efecte de les polítiques públiques sobre la qualitat de l'aire. Aquesta pràctica no sols us permetrà aplicar els coneixements teòrics en un context real sinó també desenvolupar habilitats en la visualització de dades i l'anàlisi espacial.

L'objectiu d'aquesta pràctica és doble. D'una banda, busquem que visualitzeu dades de qualitat de l'aire en temps real per a entendre la situació actual de la contaminació a Madrid. Per un

altre, que realitzeu una anàlisi històrica per a identificar patrons temporals i l'impacte de diferents factors sobre la qualitat de l'aire. Es tractarà, per tant, d'una pràctica composta per dos exercicis centrals:

- Visualització de dades de qualitat de l'aire.
- Anàlisi històrica de la qualitat de l'aire.

Per a la realització d'aquests exercicis, fareu ús de dades obertes proporcionades per l'Ajuntament de Madrid, juntament amb eines i tècniques d'anàlisis i visualització de dades.

El treball a realitzar, com ja heu vist anteriorment, es divideix en tres segments clau, cadascun destinat a desenvolupar habilitats específiques dins del camp dels Sistemes d'Informació Geogràfica (SIG):

- Recollida i preparació de dades: en aquesta fase inicial, s'introduiran les tècniques per a identificar i obtenir dades espacials rellevants, utilitzant fonts públiques, com les de qualitat de l'aire de l'Ajuntament de Madrid.
- Anàlisi amb *R*: mitjançant l'ús de funcions i paquets especialitzats en *R*, *sp sf*, manipulareu les dades per a obtenir una comprensió profunda i permetre explorar com l'anàlisi i la gestió de dades espacials es pot potenciar dins del context SIG.
- Visualització tant de les dades en temps real (sense analitzar) com dels resultats de les anàlisis.

Part 1: accés a les dades

Descàrrega de dades de qualitat de l'aire

L'Ajuntament de Madrid ofereix dades de caràcter públic a través del seu Portal de Dades Obertes, els quals podeu obtenir sense cap cost. Més concretament, mitjançant el Sistema Integral de la Qualitat de l'Aire de l'Ajuntament de Madrid (Figura 1), és possible descarregar les dades dels contaminants mesurats en les estacions de monitoratge de la ciutat, disponibles des de l'any 2001 en diferents formats.

El primer pas consisteix en l'obtenció i càrrega de les dades. Per a aconseguir aquestes dades de manera automatitzada, utilitzareu l'API que ofereix la web de dades obertes. Com a punt de partida per a obtenir els enllaços de descàrrega, copieu la URL de la descàrrega des de DECAT: enllaç.

La definició i l'estructura de les dades que obtindrem la podeu trobar en l'enllaç Intèrpret d'arxius de qualitat de l'aire.

En aquesta pràctica continuareu treballant amb *R*. Creareu un nou projecte i, dins d'aquest, un arxiu R anomenat "DescargaDatos". A continuació, es detalla pas a pas com obtenir les dades.

Figura 1: Captura del portal de Dades Obertes de l'Ajuntament de Madrid.

1. Instal·leu els paquets necessaris per al processament de dades[1]: *tidyverse* per a la manipulació de dades, *xml2* per a la manipulació d'arxius XML, i *vroom* per a la lectura eficient d'arxius de text.

```
install.packages("tidyverse")
install.packages("xml2")
install.packages("vroom")

library(tidyverse)
library(xml2)
library(vroom)
```

2. Assigneu a `url` l'adreça web on es troben les dades RDF de DECAT sobre la qualitat de l'aire a Madrid. Si voleu conèixer més sobre la mena de dades RDF podeu consultar aquest enllaç: Què és RDF?.

```
url <-
↪  "https://datos.madrid.es/egob/catalogo/201410-0-calidad-aire-diario.dcat"
```

3. Utilitzeu la funció `read_xml()` per a llegir el contingut XML de la URL especificada en el punt anterior, i el converteix en una llista amb `as_list()`.

```
page <- url %>%
    read_xml() %>%
    as_list()
```

4. Navegueu a través de l'estructura de la llista per a identificar on es troben els datasets dins del XML, i filtreu per aquells que tenen la clau "distribution".

```
location <- page[["RDF"]][["Catalog"]][["dataset"]][["Dataset"]]
location <- location[names(location) == "distribution"]
```

5. Ara creareu un *tibble* (per a més informació podeu consultar aquest enllaç: Tibbles) amb l'any com a identificador i l'enllaç a les dades, utilitzant funcions d'aplicació per a iterar sobre la localització dels datasets.

```
links <-
    tibble(
        year = sapply(X = location, function(x) unname(unlist(
          x[["Distribution"]][["title"]][1]))),
        link = sapply(X = location, function(x) unname(unlist(
          x[["Distribution"]][["accessURL"]][1])))
    )
```

6. Filtreu `links` per a seleccionar només aquells enllaços que corresponen a arxius en format CSV, utilitzant l'expressió regular ".csv".

[1]Si realitzeu la pràctica en Ubuntu (unix), serà necessari instal·lar els següents paquets: suo apt-get install libfontconfig1-dev libcairo2-dev libxt-dev libssl-dev libcurl4-openssl-dev libxml2-dev libharfbuzz-dev libfribidi-dev libfreetype6-dev libpng-dev libtiff5-dev libcurl4-openssl-dev libudunits2-dev libgdal-dev libgeos-dev libproj-dev libsodium-dev

```
links <- links %>% filter(str_detect(link, pattern = ".csv"))
```

7. Afegiu al *tibble* `links` una nova columna amb el nom de l'arxiu que es descarregarà, utilitzant l'any com a part del nom.

```
links <- links %>%
    mutate(file_name = paste0("datos_aire_madrid_", year, ".csv"))
```

8. A continuació, seleccionareu les files del dataframe links on el valor d'any és igual o superior a "2013" i després extraureu la columna com a vector.

```
years <- links %>% filter(year >= "2013") %>% .$year
```

9. Utilitzeu `lapply` per a iterar sobre els anys seleccionats i descarregar els arxius CSV corresponents, comprovant primer si l'arxiu ja existeix.

10. Abans d'executar el codi haureu de crear amb anterioritat una carpeta "data" dins del projecte.

```
lapply(years, function(x) {
  file_x <- links %>%
    filter(year == x & str_detect(link, ".csv"))

  if (x == max(years)) {
    download.file(url = file_x$link,
                  destfile = paste0("data/",
                                    file_x$file_name))
  }

  if (!file.exists(paste0("data/", file_x$file_name))) {
    download.file(url = file_x$link,
                  destfile = paste0("data/",
                                    file_x$file_name))
  }
})
```

11. En el mateix script, llegiu els arxius CSV utilitzant `vroom` i realitzeu diverses transformacions de dades, incloent la conversió de columnes a numèriques i l'organització de dades. Guardeu les dades processades en un arxiu RDS.

```
data <- vroom(paste0("data/", links %>% filter(year %in% years) %>%
↪  .$file_name))

cols_to_numeric <-
  c(
    "PROVINCIA",
    "MUNICIPIO",
    "ESTACION",
    "MAGNITUD",
```

```
    "PUNTO_MUESTREO",
    "ANO",
    "MES",
    str_subset(names(data), pattern = '^D')
  )
data <- data %>%
  mutate(across(all_of(cols_to_numeric), as.numeric))

write_rds(data, "data/data_raw.RDS")
```

12. Transformeu les dades de format transversal a longitudinal.

```
air_mad <- data %>%
  gather(v, valor, D01:V31) %>%
  mutate(DIA = str_sub(v, 2, 3),
         v = str_sub(v, 1, 1))

air_mad <- air_mad %>%
  mutate(id = ESTACION,
         fecha = as.Date(paste(ANO, MES, DIA, sep = "-"))) %>%
  select(id, MAGNITUD, fecha, v, valor)

air_mad <- air_mad %>%
  unique() %>%
  pivot_wider(names_from = v, values_from = valor)
air_mad <- air_mad %>%
  mutate(valor = as.numeric(D)) %>%
  select(-D)
```

13. Ara afegireu informació addicional, com els noms d'estacions i les magnituds, i reorganitzareu les dades per a facilitar l'anàlisi. Descomprimiu l'arxiu i deixeu la carpeta *info* al mateix nivell que la carpeta *data*. Heu de descarregar els arxius CSV des del enllaç.

14. Finalment, guardeu el conjunt de dades enriquit en un nou arxiu RDS. Executeu el script.

```
estaciones <- read_csv("info/estaciones.csv")
observaciones <- read_csv("info/observaciones.csv")

air_mad <- left_join(air_mad, estaciones, by = "id")
air_mad <- left_join(air_mad, observaciones, by = "MAGNITUD")
air_mad <- air_mad %>% select(-MAGNITUD)

write_rds(air_mad, "data/air_mad.RDS")
```

Descàrrega de la capa dels districtes de Madrid

Una vegada hàgiu preparat totes les dades, obtindreu la capa dels districtes de Madrid (Figura 2). Podeu descarregar el shapefile des del següent enllaç: districtes de Madrid. Aquest arxiu

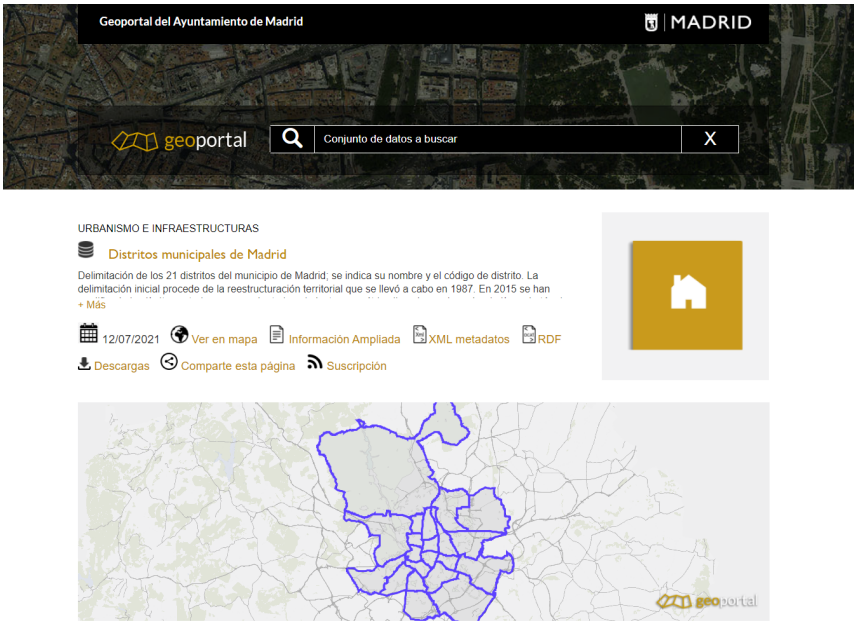

Figura 2: Geoportal de l'Ajuntament de Madrid per a la descàrrega del shapefile dels districtes. Opció polígons amb format shp.

(*Districtes.zip*) ho deixarem ara com ara, després us direm on deixar-ho.

Part 2: el paquet Shiny

Shiny és un paquet de *R* orientat a la construcció de quadres de comandament web interactius. Podeu consultar un gran nombre d'exemples de pàgines web realitzades amb *Shiny* en la següent URL: https://shiny.rstudio.com/gallery/.

Existeixen múltiples recursos en Internet que us permetran aprofundir en l'ús de *Shiny*. Per a començar, us recomanem el següent recurs de la UOC: http://datascience.recursos.uoc.edu/es/shiny/.

Com haureu pogut llegir en el recurs anterior, una aplicació *Shiny* està composta, almenys, per una carpeta on podeu trobar dos arxius:

- *ui.R*: aquest arxiu defineix la interfície de l'aplicació.
- *server.R*: aquest arxiu defineix la lògica de l'aplicació, és a dir, com l'aplicació respon a les interaccions de l'usuari.

És habitual que, en crear un projecte de *Shiny* amb *RStudio*, es creï un únic arxiu *app.R* amb dues

seccions: *ui* i *server*. Mireu, per exemple, el següent codi de l'arxiu *app.R* quan cregueu una nova aplicació Shiny.

```
#
# This is a Shiny web application. You can run the application by clicking
# the 'Run App' button above.
#
# Find out more about building applications with Shiny here:
#
#    http://shiny.rstudio.com/
#
library(shiny)
# Define UI for application that draws a histogram
ui <- fluidPage(
    # Application title
    titlePanel("Old Faithful Geyser Data"),
    # Sidebar with a slider input for number of bins
    sidebarLayout(
        sidebarPanel(
            sliderInput("bins",
                        "Number of bins:",
                        min = 1,
                        max = 50,
                        value = 30)
        ),
        # Show a plot of the generated distribution
        mainPanel(
           plotOutput("distPlot")
        )
    )
)

# Define server logic required to draw a histogram
server <- function(input, output) {

    output$distPlot <- renderPlot({
        # generate bins based on input$bins from ui.R
        x    <- faithful[, 2]
        bins <- seq(min(x), max(x), length.out = input$bins + 1)

        # draw the histogram with the specified number of bins
        hist(x, breaks = bins, col = 'darkgray', border = 'white')
    })
}
# Run the application
shinyApp(ui = ui, server = server)
```

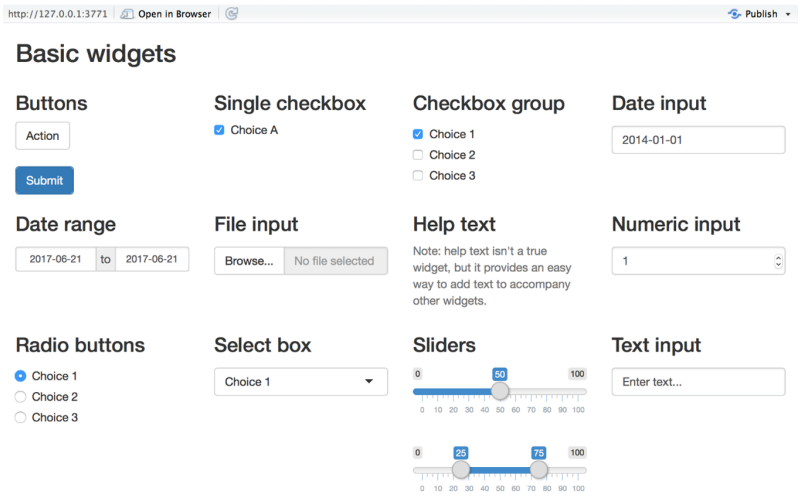

Figura 3: *Inputs* de *Shiny* (Font: https://shiny.rstudio.com/tutorial/written-tutorial/lesson3/).

Per a crear una nova aplicació de *Shiny* amb *RStudio*, accediu al menú *File* > *New Project* i, en l'assistent, seleccioneu *New Directory* > *Shiny Web App*. Genereu una carpeta anomenada *PRA* i seleccioneu-la per a crear el projecte.

Una vegada creat la Shiny Web APP, en la secció *ui*, podeu situar una sèrie de controls que us permetran definir entrades (inputs) i sortides (outputs) de dades. Alguns dels inputs bàsics de *Shiny* inclouen (Figura 3):

- Botons (*Buttons*).
- Caselles de selecció (*Checkboxes*).
- Quadres de text per a textos, números o dates (*Date input*, *Numeric input*, *Text input*).
- Combos per a selecció (*Select box*).

Quant a les sortides (*outputs*), *Shiny* us permet generar text (*textOutput*), taules (*tableOutput* o *dataTableOutput*), imatges (*imageOutput*) o gràfics (*plotOutput*), entre altres. Un dels controls que utilitzarem en aquest exercici és el widget de mapa del paquet Leaflet[2]. La funció *leafletOutput()*[3] us permetrà definir un output en la secció *ui*.

El contingut dels outputs es genera (es renderitza) en les funcions definides en la secció *server*. Per exemple, la següent taula (Taula 1) enumera les funcions de renderització que s'utilitzen per a cada tipus d'output:

A més dels inputs i outputs, en la secció *ui* és possible definir diferents dissenys utilitzant diverses plantilles i funcionalitats.

[2] `https://rstudio.github.io/leaflet/`
[3] `https://rstudio.github.io/leaflet/shiny.html`

Taula 1: Funcions de renderització (columna "server.R") associades a cada tipus d'input (columna "ui.R"). Modificat de: `http://rstudio-pubs-static.s3.amazonaws.com/21373_8afd36349b7461089c8a76659982915.html#outputs1`

server.R	ui.R
`renderPlot()`	`plotOutput()`
`renderImage()`	`imageOutput()`
`renderText()`	`textOutput()`
`renderPrint()`	`verbatimTextOutput()`
`renderTable()`	`tableOutput()`
`renderDataTable()`	`dataTableOutput()`
`renderLeaflet()`	`leafletOutput()`
`renderDygraph()`	`dygraphOutput()`

Shiny utilitza un tipus de programació que anomenem reactiva: el codi en la secció *server* escolta i reacciona a les interaccions de l'usuari amb els controls incorporats en la secció *ui*. Per a això, *Shiny* utilitza principalment tres tipus de funcions:

- *reactive*. Aquestes funcions retornen valors i es diferencien de les funcions estàndard de *R* perquè són capaces d'avaluar si les dades d'origen han canviat. La primera vegada que una funció *reactivi* és executada, emmagatzema el seu resultat en memòria. En execucions posteriors, si les dades originals no han canviat, la funció no s'executa de nou: simplement retorna les dades ja emmagatzemades.
- *eventReactive*. Les funcions *eventReactive* s'executen en resposta a determinats esdeveniments que ocorren en els controls. Per exemple, un botó tindrà associada una funció *eventReactive* anomenada *submit*. Aquestes funcions permeten capturar o calcular una dada associada al control que va llançar l'esdeveniment (per exemple, si un usuari selecciona un registre en una taula, es pot utilitzar *eventReactive* per a capturar l'ID del registre seleccionat).
- *observeEvent*. Aquesta funció permet realitzar una acció en resposta a un esdeveniment. Una funció *observeEvent* pot ser associada amb una funció *eventReactive* per a respondre a interaccions específiques.

El següent *script* mostra com s'apliquen tots aquests conceptes per a crear un quadre de comandament que visualitza les dades prèviament descarregades. **Heu d'incloure l'arxiu generat amb les dades de qualitat de l'aire (*air_mad.RDS*) de la Part 1 i el zip amb la capa dels districtes (*Districtes.zip*) en el projecte actual** (tots dos dins de la carpeta PRA, i al mateix nivell).

Prèviament serà necessari instal·lar les biblioteques: shiny, leaflet, dplyr, shinyjs, DT i sf si encara no les teniu.

```
install.packages('dplyr')
install.packages('shiny')
```

```
install.packages('leaflet')
install.packages('shinyjs')
install.packages('DT')
install.packages('sf')


library(shiny)
library(leaflet)
library(dplyr)
library(shinyjs)
library(DT)
library(sf)

air_data <- readRDS('air_mad.RDS')

temp_dir <- tempdir()
unzip("Distritos.zip", exdir = temp_dir)
distritos <- st_read(temp_dir) %>%
  st_transform(crs = 4326)

ui <- fluidPage(
  titlePanel("Calidad del Aire en Madrid"),
  sidebarLayout(
    sidebarPanel(
      selectInput("pollutant", "Selecciona Contaminante:", choices =
      ↪  unique(air_data$nom_abv)),
      dateInput("date", "Selecciona Fecha:", value = Sys.Date(), max =
      ↪  Sys.Date())
    ),
    mainPanel(
      leafletOutput(outputId = "map", height = "500px"),
      DT::dataTableOutput("data_table"),
      uiOutput("no_data_message") # User feedback on data availability
    )
  )
)

server <- function(input, output, session) {

  filtered_data <- reactive({
    req(input$pollutant, input$date) # Ensure required inputs are available
    air_data %>%
      filter(nom_abv == input$pollutant, fecha == as.Date(input$date))
  })

  output$map <- renderLeaflet({
    data_for_map <- filtered_data()
    req(nrow(data_for_map) > 0) # Ensure data is available before proceeding
```

```
    pal <- colorNumeric(palette = "YlOrRd", domain = data_for_map$valor, na.color
    ↪  = "#FFFFFF")
    leaflet() %>%
      addTiles() %>%
      addPolygons(data = distritos, fillColor = "#ffffff", weight = 1, color =
      ↪  "#000000", fillOpacity = 0.5) %>%
      addCircleMarkers(data = data_for_map, ~longitud, ~latitud, radius = 5,
      ↪  color = '#007bff', fill = TRUE, fillColor = ~pal(valor), fillOpacity =
      ↪  0.7, popup = ~paste(nom_mag, valor, ud_med))
  })

  output$data_table <- DT::renderDataTable({
    req(filtered_data()) # Ensure data is loaded before proceeding
    data_for_table <- filtered_data() %>%
      select(id_name, fecha, valor, nom_mag, ud_med) # Select specific columns
      ↪  for the table
    datatable(data_for_table, options = list(pageLength = 5, autoWidth = TRUE))
  })

}

shinyApp(ui = ui, server = server)
```

Com a resultat de l'execució del script la següent aplicació es mostrarà en el navegador embegut en RStudio (Figura 4):

Bàsicament, l'aplicació mostra una taula en el panell lateral esquerre amb el llistat de fenòmens de qualitat de l'aire mesurats i un selector per a indicar la data. Una vegada seleccionada (**assegureu-vos que la data seleccionada tingui dades**; segurament el mes actual encara no tindrà, ja que les dades són validades abans de ser publicats), s'actualitzarà el mapa amb els punts dels sensors que han reportat observacions durant els filtres seleccionats i apareixeran les observacions en la part inferior.

Exercicis

Basant-vos en el que heu practicat fins ara, haureu de resoldre els següents exercicis. És imprescindible que proporcioneu una explicació detallada del procés seguit per a cada solució, pas a pas. A més, en cas que l'exercici requereixi l'adjunció de qualsevol arxiu generat durant la seva realització, aquest haurà de ser inclòs dins d'un arxiu comprimit (*.zip*). Dins d'aquest arxiu haureu d'incloure:

- Un document (en format PDF o Word) que contingui la descripció detallada dels passos realitzats per a resoldre cada exercici;
- una carpeta específica per a cada exercici que contingui els resultats obtinguts, sempre que l'enunciat així ho especifiqui.

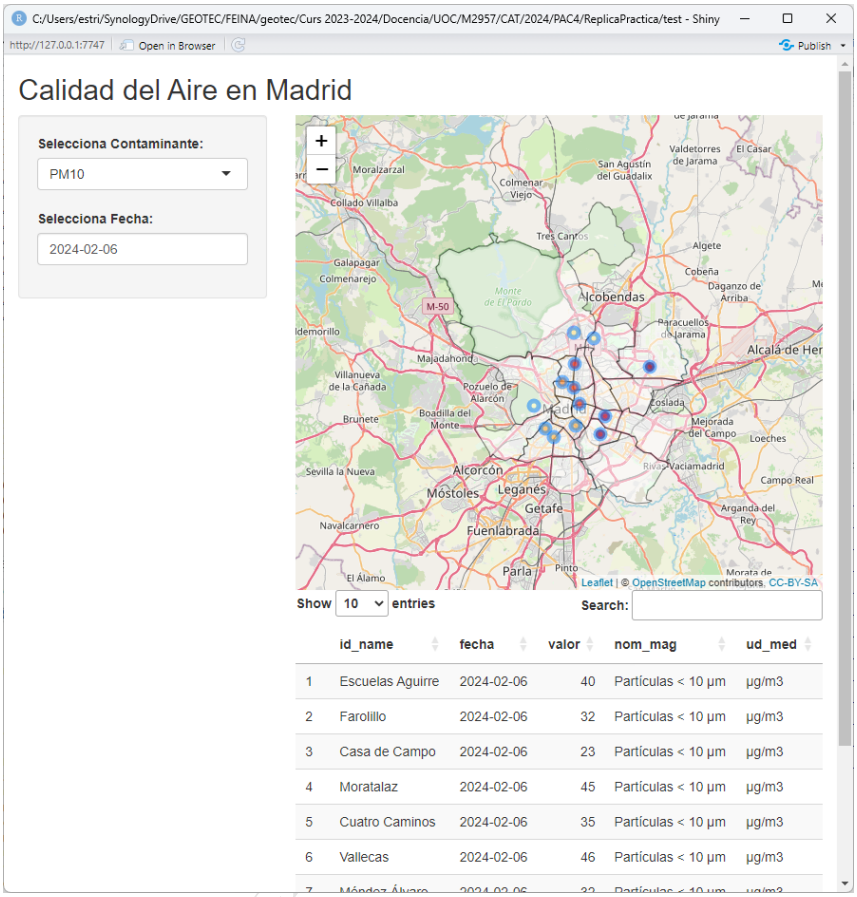

Figura 4: Aspecte de l'aplicación.

Per a realitzar els exercicis, heu de tenir certs coneixements de *Shiny* per a poder codificar les consultes que se us demanen. Podeu trobar més informació en la pàgina de *Shiny* indicada anteriorment, que segurament us serà d'ajuda http://datascience.recursos.uoc.edu/es/shiny/.

Exercici 1 [20%]: heu d'afegir un tercer blego (l'últim a seleccionar) on podreu triar l'estació a mostrar. A més, heu de mostrar un missatge indicant si no hi ha dades disponibles per a la selecció.

L'aspecte final del quadre de comandament serà similar al mostrat en les Figures 5 i 6.

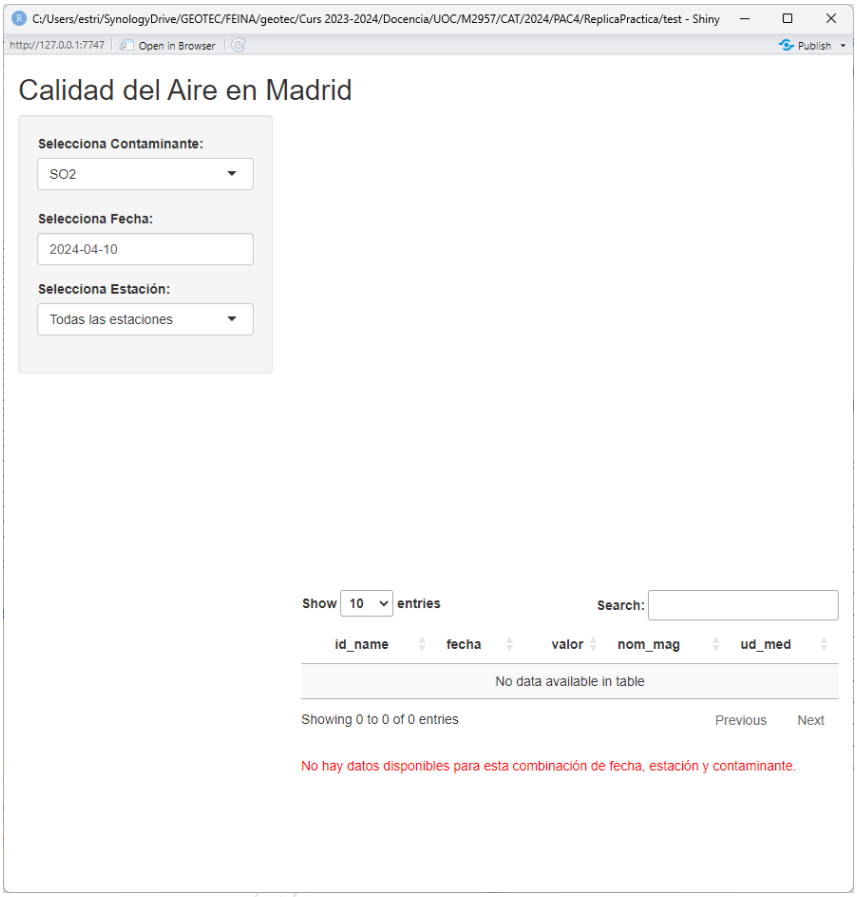

Figura 5: Aspecte de l'exercici 1 (missatge de no dades).

Exercici 2 [30%]: modifiqueu l'exemple anterior per a mostrar en un gràfic de línies les observacions d'una estació seleccionada. En aquest cas, canviarem el selector de temps per a seleccionar períodes anuals de dades. Una vegada seleccionats el contaminant, l'any i l'estació, es mostrarà el gràfic.

A més de poder seleccionar una estació mitjançant el desplegable, també s'haurà de permetre la selecció directament sobre el mapa: en fer clic sobre una estació, el gràfic ha d'adaptar-se automàticament a aquesta selecció, de manera equivalent a si s'hagués triat des del blego. Si no hi ha cap estació seleccionada en el blego, el mapa mostrarà totes les estacions disponibles per a la seva selecció directa. Aquesta funcionalitat ha de ser totalment interactiva.

L'aspecte final del quadre de comandament serà similar al mostrat en la Figura 7.

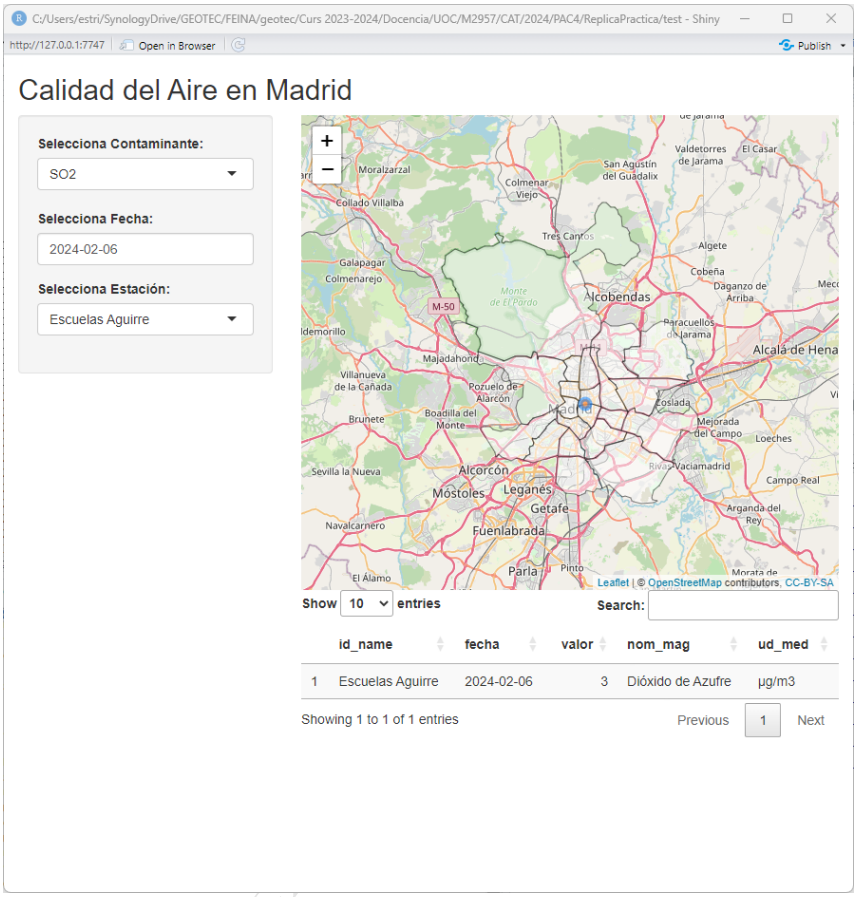

Figura 6: Aspecte de l'aplicación (general).

Exercici 3 [30%]: a partir del resultat de l'exercici 1, heu de pintar els polígons que continguin una estació amb el color basat en el valor de l'observació d'aquesta estació. Haureu de fer ús de la funció *st_intersects*

.

L'aspecte final del quadre de comandament serà similar al mostrat en la Figura 8.

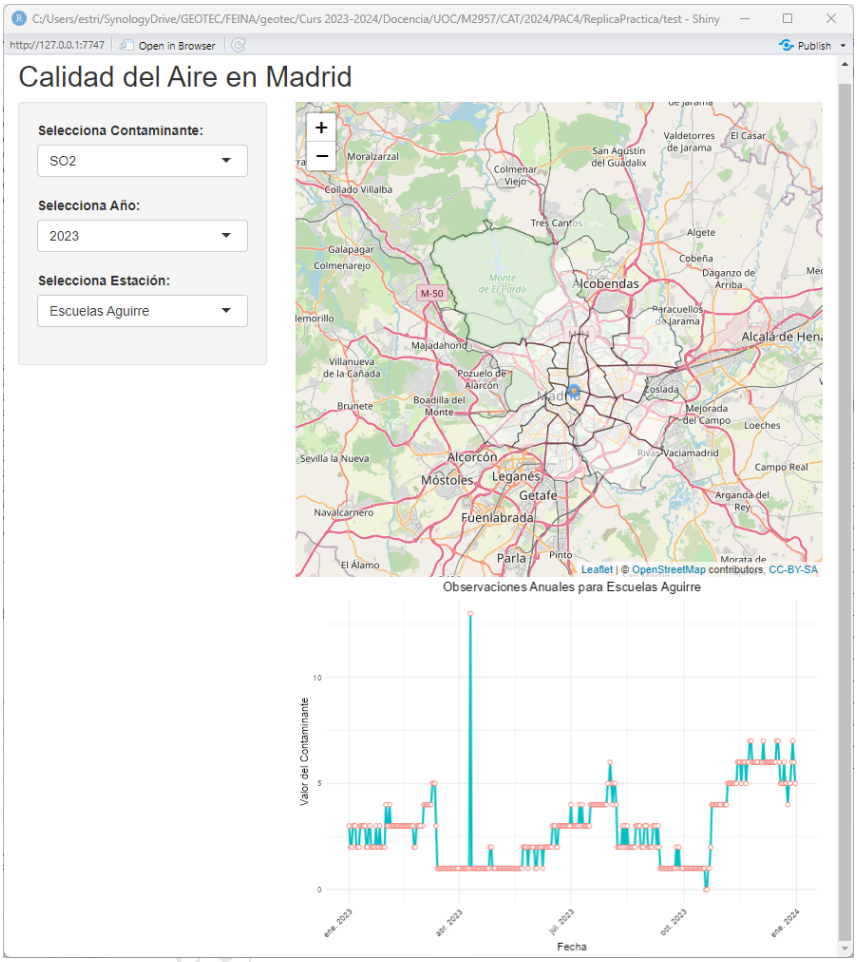

Figura 7: Aspecte de l'exercici 2.

Exercici 4 [30%]: basant-vos en l'exercici anterior, genereu interpolacions per als polígons que no disposen de valors. Haureu d'utilitzar el model d'interpolació per Distància Inversa Ponderada (IDW, per les seves sigles en anglès). Utilitzeu la biblioteca *gstat* per a això.

L'aspecte final del quadre de comandament serà similar al mostrat en la Figura 9.

Recursos

Els següents recursos són d'utilitat per a la realització de la PAC:

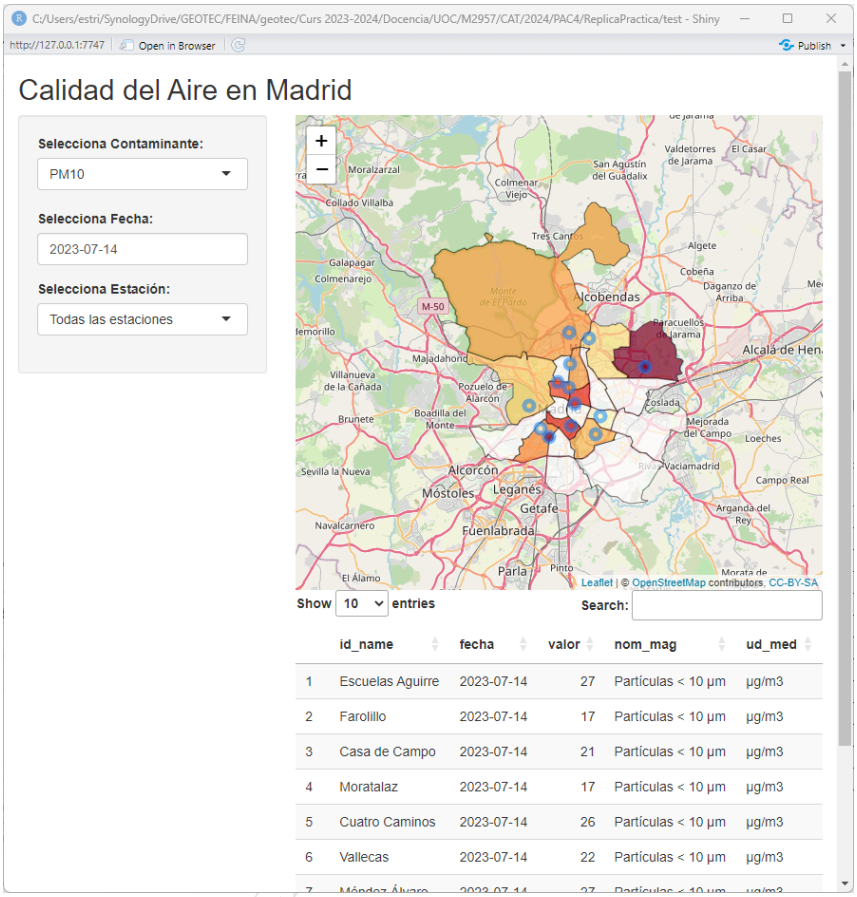

Figura 8: Aspecte de l'exercici 3.

Bàsics

- Capítol 14 - Introducció a Shiny aqui
- Capítol 15 - La meva primera aplicació ShinyR aqui
- VVAA (2018) Anàlisi estadística de dades espacials. UOC. Wiki disponible aqui
- VVAA (2017) Tutorial d'anàlisi espacial amb R. UOC. Wiki disponible aqui

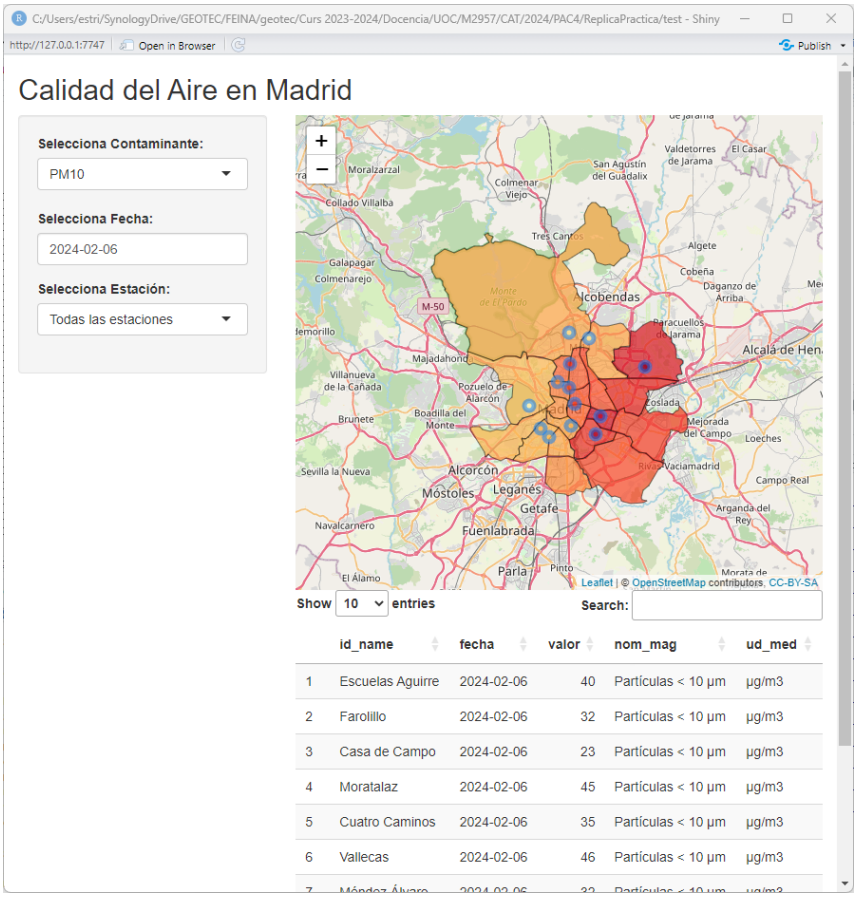

Figura 9: Aspecte de l'exercici 4.