05. - Data Collection_CU_04_15_zonas_conta_v_01

June 8, 2023

#

CU04_Optimización de vacunas

Citizenlab Data Science Methodology > II - Data Processing Domain *** > # 05.- Data Collection

Data Collection is the process to obtain and generate (if required) necessary data to model the problem.

0.0.1 15. Interpolar variables contaminación a zonas

- Dados los datos de las estaciones, interpolar por kriging a las zonas de salud del caso

Table of Contents

Settings

Data Load

ETL Processes

Import data from: CSV, Excel, Tab, JSON, SQL, and Parquet files

Synthetic Data Generation

Fake Data Generation

Open Data

Data Save

Main Conclusions

Main Actions

Acciones done

Acctions to perform

0.1 Settings

0.1.1 Packages to use

ELIMINAR O AÑADIR LO QUE TOQUE. COPIAR VERSIONES AL FINAL Y QUITAR CÓDIGO DE VERSIONES

- {tcltk} para selección interactiva de archivos locales
- {sf} para trabajar con georeferenciación

- {readr} para leer y escribir archivos csv
- {dplyr} para explorar datos
- {tidyr} para organización de datos
- {gstat} para operaciones geoestadísticas

[24]:
```
library(readr)
library(dplyr)
library(tidyr)
library(sf)
library(gstat)
library(stringr)

p <- c("tcltk", "sf", "readr", "dplyr", "tidyr", "gstat", "stringr")
```

0.1.2 Paths

[2]:
```
iPath <- "Data/Input/"
oPath <- "Data/Output/"
```

0.2 Data Load

If there are more than one input file, make as many sections as files to import.

Instrucciones - Los ficheros de entrada del proceso están siempre en Data/Input/.
- Si hay más de un fichero de entrada, se crean tantos objetos iFile_xx y file_data_xx como ficheros de entrada (xx número correlativo con dos dígitos, rellenar con ceros a la izquierda)

1. Datos de estaciones de medición contaminación

OPCION A: Seleccionar fichero en ventana para mayor comodidad

Data load using the {tcltk} package. Ucomment the line if not using this option

[3]:
```
# file_data_01 <- tcltk::tk_choose.files(multi = FALSE)
```

OPCION B: Especificar el nombre de archivo

[4]:
```
iFile_01 <- "CU_04_05_14_contaminacion.csv"
file_data_01 <- paste0(iPath, iFile_01)

if(file.exists(file_data_01)){
    cat("Se leerán datos del archivo: ", file_data_01)
} else{
    warning("Cuidado: el archivo no existe.")
}
```

```
Se leer<U+00E1>n datos del archivo:  Data/Input/CU_04_05_14_contaminacion.csv
```

Data file to dataframe Usar la función adecuada según el formato de entrada (xlsx, csv, json, ...)

[5]:
```
data_01 <- read_csv(file_data_01)
```

```
Rows: 3517 Columns: 15
-- Column specification
--------------------------------------------------------
Delimiter: ","
chr  (2): site, site_name
dbl (13): ano, semana, benzene, co, no, no2, nox, o3, pm10, pm2.5,
so2, X, Y

i Use `spec()` to retrieve the full column specification for this
data.
i Specify the column types or set `show_col_types = FALSE` to quiet
this message.
```

Estructura de los datos:

[6]:
```
data_01 |> glimpse()
```

```
Rows: 3,517
Columns: 15
$ ano       <dbl> 2021, 2021, 2021, 2021, 2021, 2021, 2021,
2021, 2021, 2021, ~
$ semana    <dbl> 35, 35, 35, 35, 35, 35, 35, 35, 35, 35, 35,
35, 35, 35, 35, ~
$ site      <chr> "es0115a", "es0118a", "es0120a", "es0124a",
"es0125a", "es01~
$ benzene   <dbl> NA, 0.2134454, 0.2889831, NA, NA,
0.1726496, 0.1705882, NA, ~
$ co        <dbl> 0.2750000, 0.2436975, NA, NA, NA, NA, NA,
0.3117647, NA, NA,~
$ no        <dbl> 8.250000, 6.050420, 5.816667, 3.375000,
10.629630, 5.428571,~
$ no2       <dbl> 20.541667, 36.630252, 35.516667, 25.866667,
43.638889, 30.06~
$ nox       <dbl> 33.175000, 45.966387, 44.533333, 31.050000,
59.935185, 38.44~
$ o3        <dbl> NA, 58.64034, NA, 67.25833, 49.82202,
51.81076, 54.88100, 61~
$ pm10      <dbl> NA, 24.983193, NA, NA, NA, 17.857143,
14.750000, NA, 14.5833~
$ pm2.5     <dbl> NA, 20.369748, NA, NA, NA, NA, 8.983333,
NA, NA, NA, 12.6083~
$ so2       <dbl> 10.116667, 6.075630, NA, NA, NA, NA, NA,
7.983193, 5.725000,~
$ site_name <chr> "PLAZA DE ESPA<U+00D1>A", "ESCUELAS
AGUIRRE", "RAM<U+00D3>N Y CAJAL", "ART~
$ X         <dbl> -3.712222, -3.682222, -3.677222, -3.639167,
```

```
-3.705000, -3.73~
$ Y         <dbl> 40.42417, 40.42167, 40.45167, 40.44000,
40.34694, 40.39472, ~
```

Muestra de datos:

[7]:
```
data_01 |> slice_head(n = 5)
```

A spec_tbl_df: 5 x 15

ano <dbl>	semana <dbl>	site <chr>	benzene <dbl>	co <dbl>	no <dbl>	no2 <dbl>	nox <dbl>	o <dbl>
2021	35	es0115a	NA	0.2750000	8.250000	20.54167	33.17500	N
2021	35	es0118a	0.2134454	0.2436975	6.050420	36.63025	45.96639	5
2021	35	es0120a	0.2889831	NA	5.816667	35.51667	44.53333	N
2021	35	es0124a	NA	NA	3.375000	25.86667	31.05000	6
2021	35	es0125a	NA	NA	10.629630	43.63889	59.93519	4

2. Datos de zonas

OPCION A: Seleccionar fichero en ventana para mayor comodidad

Data load using the {tcltk} package. Ucomment the line if not using this option

[8]:
```
# file_data_02 <- tcltk::tk_choose.files(multi = FALSE)
```

OPCION B: Especificar el nombre de archivo

[9]:
```
iFile_02 <- "CU_04_05_01_zonasgeo.json"
file_data_02 <- paste0(iPath, iFile_02)

if(file.exists(file_data_02)){
    cat("Se leerán datos del archivo: ", file_data_02)
} else{
    warning("Cuidado: el archivo no existe.")
}
```

```
Se leer<U+00E1>n datos del archivo:  Data/Input/CU_04_05_01_zonasgeo.json
```

Data file to dataframe Usar la función adecuada según el formato de entrada (xlsx, csv, json, ...)

[10]:
```
data_02 <- st_read(file_data_02)
```

```
Reading layer `CU_04_05_01_zonasgeo' from data source
  `/Users/emilio.lcano/academico/gh_repos/__transferencia/citizenlab/CitizenLab-
Research-and-Development/casos_urjc/notebooks/II_data_processing/04_vacunas/Data
/Input/CU_04_05_01_zonasgeo.json'
  using driver `GeoJSON'
Simple feature collection with 286 features and 3 fields
Geometry type: MULTIPOLYGON
Dimension:     XY
```

```
Bounding box:  xmin: -4.579396 ymin: 39.8848 xmax: -3.052977 ymax: 41.16584
Geodetic CRS:  WGS 84
```

Estructura de los datos:

[11]:
```
data_02 |> glimpse()
```

```
Rows: 286
Columns: 4
$ CODBDT    <int> 686213, 686214, 686215, 686216, 686217,
686218, 686219, 6862~
$ GEOCODIGO <chr> "001", "002", "003", "004", "005", "006",
"007", "008", "009~
$ DESBDT    <chr> "Abrantes", "Acacias", "Adelfas",
"Alameda", "Alameda de Osu~
$ geometry  <MULTIPOLYGON [arc_degree]> MULTIPOLYGON
(((-3.718306 4…, MULTIP~
```

Muestra de datos:

[14]:
```
data_02 |> tibble() |> slice_head(n = 5)
```

A tibble: 5 x 4

CODBDT <int>	GEOCODIGO <chr>	DESBDT <chr>	geometry <MULTIPOLYGON [arc_degree]>
686213	001	Abrantes	MULTIPOLYGON (((-3.718306 4...
686214	002	Acacias	MULTIPOLYGON (((-3.707966 4...
686215	003	Adelfas	MULTIPOLYGON (((-3.666363 4...
686216	004	Alameda	MULTIPOLYGON (((-3.69947 40...
686217	005	Alameda de Osuna	MULTIPOLYGON (((-3.561629 4...

0.3 ETL Processes

0.3.1 Import data from: CSV, Excel, Tab, JSON, SQL, and Parquet files

Se han importado en el apartado Data Load anterior:

- Datos de estaciones
- Geometrías de zonas

Incluir apartados si procede para: Extracción de datos (select, filter), Transformación de datos, (mutate, joins, …). Si es necesario tratar datos perdidos, indicarlo también en NB 09.2

Si no aplica: Estos datos no requieren tareas de este tipo.

Data transform Convertir estaciones a objeto sf

[16]:
```
tdata_01 <- data_01 |>
    st_as_sf(coords = c("X", "Y"), crs = 4326)
```

[17]:
```
tdata_01 |> glimpse()
```

```
Rows: 3,517
Columns: 14
$ ano       <dbl> 2021, 2021, 2021, 2021, 2021, 2021, 2021,
2021, 2021, 2021, ~
$ semana    <dbl> 35, 35, 35, 35, 35, 35, 35, 35, 35, 35, 35,
35, 35, 35, 35, ~
$ site      <chr> "es0115a", "es0118a", "es0120a", "es0124a",
"es0125a", "es01~
$ benzene   <dbl> NA, 0.2134454, 0.2889831, NA, NA,
0.1726496, 0.1705882, NA, ~
$ co        <dbl> 0.2750000, 0.2436975, NA, NA, NA, NA, NA,
0.3117647, NA, NA,~
$ no        <dbl> 8.250000, 6.050420, 5.816667, 3.375000,
10.629630, 5.428571,~
$ no2       <dbl> 20.541667, 36.630252, 35.516667, 25.866667,
43.638889, 30.06~
$ nox       <dbl> 33.175000, 45.966387, 44.533333, 31.050000,
59.935185, 38.44~
$ o3        <dbl> NA, 58.64034, NA, 67.25833, 49.82202,
51.81076, 54.88100, 61~
$ pm10      <dbl> NA, 24.983193, NA, NA, NA, 17.857143,
14.750000, NA, 14.5833~
$ pm2.5     <dbl> NA, 20.369748, NA, NA, NA, NA, 8.983333,
NA, NA, NA, 12.6083~
$ so2       <dbl> 10.116667, 6.075630, NA, NA, NA, NA, NA,
7.983193, 5.725000,~
$ site_name <chr> "PLAZA DE ESPA<U+00D1>A", "ESCUELAS
AGUIRRE", "RAM<U+00D3>N Y CAJAL", "ART~
$ geometry  <POINT [arc_degree]> POINT (-3.712222 40.42417),
POINT (-3.682222 ~
```

- Interpolar variables a las zonas

NOTA: esta operación puede tardar varios minutos.

[25]:
```r
contavars <- c("benzene", "co", "no", "no2", "nox", "o3", "pm10", "pm2.5",␣
 ↪"so2")
semanas <- paste(tdata_01$ano, tdata_01$semana, sep = "-") |> unique()
resl <- list()
data_02_centroid <- data_02 |> st_centroid()

for (i in seq_along(semanas)){
  f <- semanas[i]
  a <- str_sub(f, 1, 4)
  s <- str_sub(f, 6,7)
  contad <- tdata_01 |>
    filter(ano == a,
           semana == s) |>
```

```
    select(all_of(contavars))

    res <- data_02 |> select(GEOCODIGO) |> st_drop_geometry()
    for (j in seq_along(contavars)){
      thisvar <- contavars[j]
      contadc <- contad |>
        drop_na(all_of(thisvar))

      if(all(contadc |> pull(thisvar) == 0)){
        res <- res |>
          mutate(!!thisvar := 0)
      } else{

        fo <- as.formula(paste0(thisvar, " ~ 1"))
        v0 <- variogram(fo, contadc)
        v.m <- suppressWarnings(fit.variogram(v0,
          vgm(c("Exp", "Mat", "Sph", "Log", "Wav", "Bes", "Lin", "Leg"))))
        if(any(v.m$range < 0)){
          res <- res |>
            mutate(!!thisvar := NA)
        } else{
          invisible(
            capture.output(
              suppressWarnings(
                krg <- krige(fo,
                             locations = contadc,
                             newdata = data_02_centroid,
                             model = v.m))))
          res <- res |>
            mutate(!!thisvar := krg$var1.pred)
        }
      }
    }
    res <- res |>
      mutate(ano = a,
             semana = s)
    resl[[i]] <- res
}

data <- bind_rows(resl)
```

```
Warning message in st_centroid.sf(data_02):
"st_centroid assumes attributes are constant over geometries of x"
```

[37]:
```
data |> glimpse()
```

```
Rows: 21,164
Columns: 12
```

```
$ GEOCODIGO <chr> "001", "002", "003", "004", "005", "006",
"007", "008", "009~
$ benzene   <dbl> 0.2478896, 0.2351470, 0.2438505, 0.2311297,
0.2582252, 0.326~
$ co        <dbl> 0.2783157, 0.2720947, 0.2564127, 0.2704844,
0.3730037, 0.422~
$ no        <dbl> 4.277267, 4.277267, 4.277267, 4.277267,
4.277267, 4.277267, ~
$ no2       <dbl> 30.81135, 30.34314, 30.12568, 30.05632,
25.87842, 27.88230, ~
$ nox       <dbl> 38.70242, 38.26907, 37.77193, 37.77965,
32.30093, 35.32128, ~
$ o3        <dbl> 53.41865, 55.30295, 55.55106, 57.16719,
57.60870, 59.73277, ~
$ pm10      <dbl> 17.55392, 21.35499, 22.63530, 23.80635,
22.43009, 17.30619, ~
$ pm2.5     <dbl> 12.77851, 12.84574, 12.85695, 12.86689,
12.18365, 12.24123, ~
$ so2       <dbl> 6.747270, 7.411119, 6.654251, 7.439685,
2.376795, 1.334976, ~
$ ano       <chr> "2021", "2021", "2021", "2021", "2021",
"2021", "2021", "202~
$ semana    <chr> "35", "35", "35", "35", "35", "35", "35",
"35", "35", "35", ~
```

[38]:
```
data |> slice_head(n = 5)
```

A data.frame: 5 x 12

GEOCODIGO <chr>	benzene <dbl>	co <dbl>	no <dbl>	no2 <dbl>	nox <dbl>	o3 <dbl>	pm1 <dbl
001	0.2478896	0.2783157	4.277267	30.81135	38.70242	53.41865	17.55
002	0.2351470	0.2720947	4.277267	30.34314	38.26907	55.30295	21.35
003	0.2438505	0.2564127	4.277267	30.12568	37.77193	55.55106	22.63
004	0.2311297	0.2704844	4.277267	30.05632	37.77965	57.16719	23.80
005	0.2582252	0.3730037	4.277267	25.87842	32.30093	57.60870	22.43

0.4 Synthetic Data Generation

No aplica

0.5 Fake Data Generation

No aplica

0.6 Open Data

Los datos originales fueron descargados de fuentes abiertas

0.7 Data Save

Este proceso, puede copiarse y repetirse en aquellas partes del notebbok que necesiten guardar datos. Recuerde cambiar las cadenas añadida del fichero para diferenciarlas

Identificamos los datos a guardar

[28]:
```
data_to_save <- data
```

Estructura de nombre de archivos:

- Código del caso de uso, por ejemplo "CU_04"
- Número del proceso que lo genera, por ejemplo "_05".
- Número de la tarea que lo genera, por ejemplo "_01"
- En caso de generarse varios ficheros en la misma tarea, llevarán _01 _02 … después
- Nombre: identificativo de "properData", por ejemplo "_zonasgeo"
- Extensión del archivo

Ejemplo: "CU_04_05_01_01_zonasgeo.json, primer fichero que se genera en la tarea 01 del proceso 05 (Data Collection) para el caso de uso 04 (vacunas)

Importante mantener los guiones bajos antes de proceso, tarea, archivo y nombre

0.7.1 Proceso 05

[26]:
```
caso <- "CU_04"
proceso <- '_05'
tarea <- "_15"
archivo <- ""
proper <- "_zonas_contaminacion"
extension <- ".csv"
```

OPCION A: Uso del paquete "tcltk" para mayor comodidad

- Buscar carpeta, escribir nombre de archivo SIN extensión (se especifica en el código)
- Especificar sufijo2 si es necesario
- Cambiar datos por datos_xx si es necesario

[ ]:
```
# file_save <- paste0(caso, proceso, tarea, tcltk::tkgetSaveFile(), proper,␣
↪extension)
# path_out <- paste0(oPath, file_save)
# write_csv(data_to_save, path_out)

# cat('File saved as: ')
# path_out
```

OPCION B: Especificar el nombre de archivo

- Los ficheros de salida del proceso van siempre a Data/Output/.

[29]:
```
file_save <- paste0(caso, proceso, tarea, archivo, proper, extension)
path_out <- paste0(oPath, file_save)
```

```
write_csv(data_to_save, path_out)

cat('File saved as: ')
path_out
```

```
File saved as:
```

'Data/Output/CU_04_05_15_zonas_contaminacion.csv'

Copia del fichero a Input Si el archivo se va a usar en otros notebooks, copiar a la carpeta Input

[30]:
```
path_in <- paste0(iPath, file_save)
file.copy(path_out, path_in, overwrite = TRUE)
```

TRUE

0.8 Main Conclusions

List and describe the general conclusions of the analysis carried out.

0.8.1 Prerequisites

Para que funcione este código se necesita:

- Las rutas de archivos `Data/Input` y `Data/Output` deben existir (relativas a la ruta del *notebook*)
- El paquete tcltk instalado para seleccionar archivos interactivamente. No se necesita en producción.
- Los paquetes sf, readr, dplyr, tidyr, gstat, stringr deben estar instalados.

0.8.2 Configuration Management

This notebook has been tested with the following versions of R and packages. It cannot be assured that later versions work in the same way: * R 4.2.2 * tcltk 4.2.2 * sf 1.0.9 * readr 2.1.3 * dplyr 1.0.10 * tidyr 1.3.0 * gstat 2.1.0 * stringr 1.5.0

0.8.3 Data structures

Objeto `data`

- Hay 21164 filas
  - GEOCODIGO
  - benzene
  - co
  - no
  - no2
  - nox
  - o3
  - pm10
  - pm2.5

- so2
- ano
- semana

Observaciones generales sobre los datos

- La interpolación se ha hecho sobre los centroides de las zonas ya que la función no admite interpolar a polígonos
- En los casos en que la correlación espacial era negativa se ha asignado NA porque se producían errores

0.8.4 Consideraciones para despliegue en piloto

- Ninguna

0.8.5 Consideraciones para despliegue en producción

- Se deben crear los procesos ETL en producción necesarios para que los datos de entrada estén actualizados

0.9 Main Actions

Acciones done Indicate the actions that have been carried out in this process

- Se han calculado los centroides de las zonas
- Se han interpolado los datos de contaminación a las zonas

Acctions to perform Indicate the actions that must be carried out in subsequent processes

- Se deben unir estos datos a los datos de vacunación y sanitarios para incluir en los modelos

0.10 CODE TO DEPLOY (PILOT)

A continuación se incluirá el código que deba ser llevado a despliegue para producción, dado que se entiende efectúa operaciones necesarias sobre los datos en la ejecución del prototipo

Description

- No hay nada que desplegar en el piloto, ya que estos datos son estáticos o en todo caso cambian con muy poca frecuencia, altamente improbable durante el proyecto.

CODE

[ ]:
```
# incluir código
```