05. - Data Collection_CU_04_11_meteo_zonas_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 11. Estimar datos meteorológicos en zonas sanitarias

- Estimar variables en zonas
- Agrupar promedios semanales de las zonas en las variables meteorológicas
- Eliminando variables que no se usarán en modelos

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 Encoding

Con la siguiente expresión se evitan problemas con el encoding al ejecutar el notebook. Es posible que deba ser eliminada o adaptada a la máquina en la que se ejecute el código.

[ ]:
```
Sys.setlocale(category = "LC_ALL", locale = "es_ES.UTF-8")
```

'es_ES.UTF-8/es_ES.UTF-8/es_ES.UTF-8/C/es_ES.UTF-8/C'

0.1.2 Packages to use

- {tcltk} para selección interactiva de archivos locales
- {sf} para trabajar con georeferenciación
- {gstat} para cálculos geoestadísticos
- {readr} para leer y escribir archivos csv
- {dplyr} para explorar datos
- {stringr} para manipulación de cadenas de caracteres
- {tidyr} para quitar perdidos
- {lubridate} para manipulación de fechas

[13]:
```
library(readr)
library(dplyr)
library(sf)
library(gstat)
library(stringr)
library(tidyr)
library(lubridate)
```

```
Attaching package: 'lubridate'


The following objects are masked from 'package:base':

    date, intersect, setdiff, union
```

0.1.3 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.

1. Datos meteorológicos de estaciones

OPCION A: Seleccionar fichero en ventana para mayor comodidad

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

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

OPCION B: Especificar el nombre de archivo

[3]:
```
iFile_01 <- "CU_04_05_10_aemet.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_10_aemet.csv
```

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

[4]:
```
data_01 <- read_csv(file_data_01)
```

```
Rows: 6143 Columns: 22
-- Column specification --------------------------------------------------------
Delimiter: ","
chr   (9): indicativo, nombre, provincia, horatmin, horatmax, dir,
horaracha…
dbl  (12): altitud, tmed, prec, tmin, tmax, velmedia, racha, sol,
presMax, p…
date  (1): fecha

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:

[5]:
```
glimpse(data_01)
```

```
Rows: 6,143
Columns: 22
$ fecha       <date> 2021-09-01, 2021-09-02, 2021-09-03,
2021-09-04, 2021-09-0~
$ indicativo  <chr> "2462", "2462", "2462", "2462", "2462",
"2462", "2462", "2~
$ nombre      <chr> "PUERTO DE NAVACERRADA", "PUERTO DE
NAVACERRADA", "PUERTO ~
$ provincia   <chr> "MADRID", "MADRID", "MADRID", "MADRID",
"MADRID", "MADRID"~
$ altitud     <dbl> 1894, 1894, 1894, 1894, 1894, 1894, 1894,
1894, 1894, 1894~
```

```
$ tmed        <dbl> 13.2, 11.8, 13.8, 16.7, 18.0, 19.4, 17.1,
13.8, 12.6, 12.6~
$ prec        <dbl> 2.8, 3.2, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.3, 0.0, 0.0, 0.0~
$ tmin        <dbl> 8.2, 8.5, 9.4, 11.4, 13.1, 16.3, 13.9,
10.3, 9.5, 8.8, 9.9~
$ horatmin    <chr> "01:00", "02:00", "04:50", "05:20",
"03:30", "02:30", "23:~
$ tmax        <dbl> 18.3, 15.2, 18.3, 22.0, 22.9, 22.5, 20.3,
17.3, 15.6, 16.5~
$ horatmax    <chr> "11:50", "14:40", "16:10", "13:20",
"12:00", "13:20", "14:~
$ dir         <chr> "16", "24", "21", "19", "17", "15", "14",
"21", "27", "32"~
$ velmedia    <dbl> 3.1, 2.2, 2.2, 3.3, 3.6, 2.8, 6.4, 2.5,
1.9, 1.7, 4.2, 4.4~
$ racha       <dbl> 15.6, 8.3, 6.7, 8.3, 11.1, 19.4, 21.4,
12.8, 10.3, 9.2, 15~
$ horaracha   <chr> "00:30", "19:50", "06:00", "09:10",
"23:40", "23:40", "06:~
$ sol         <dbl> 5.1, 2.9, 7.5, 12.1, 11.7, 2.9, 7.6, 7.5,
2.5, 6.8, 5.9, 1~
$ presMax     <dbl> 816.6, 816.4, 816.4, 816.3, 819.5, 820.1,
817.8, 814.8, 81~
$ horaPresMax <chr> "01", "23", "11", "Varias", "24", "12",
"00", "00", "23", ~
$ presMin     <dbl> 813.4, 813.5, 815.1, 814.7, 815.9, 817.8,
814.7, 813.2, 81~
$ horaPresMin <chr> "04", "03", "06", "05", "04", "24", "06",
"06", "Varias", ~
$ X           <dbl> -4.010556, -4.010556, -4.010556,
-4.010556, -4.010556, -4.~
$ Y           <dbl> 40.79306, 40.79306, 40.79306, 40.79306,
40.79306, 40.79306~
```

Muestra de datos:

[6]:
```
slice_head(data_01, n = 5)
```

A spec_tbl_df: 5 x 22

fecha <date>	indicativo <chr>	nombre <chr>	provincia <chr>	altitud <dbl>	tmed <dbl>
2021-09-01	2462	PUERTO DE NAVACERRADA	MADRID	1894	13.2
2021-09-02	2462	PUERTO DE NAVACERRADA	MADRID	1894	11.8
2021-09-03	2462	PUERTO DE NAVACERRADA	MADRID	1894	13.8
2021-09-04	2462	PUERTO DE NAVACERRADA	MADRID	1894	16.7
2021-09-05	2462	PUERTO DE NAVACERRADA	MADRID	1894	18.0

2. Datos de contornos zonas

OPCION A: Seleccionar fichero en ventana para mayor comodidad

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

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

OPCION B: Especificar el nombre de archivo

[8]:
```
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, ...)

[9]:
```
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:

[10]:
```
glimpse(data_02)
```

```
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:

[11]:
```
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 diarios meteorológicos por estación
- Contornos de zonas sanitarias

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 transformation

- Convertir zonas a centroide para poder interpolar

[27]:
```
tdata_02 <- data_02 |> st_centroid()
```

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

[36]:
```
glimpse(tdata_02)
```

```
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  <POINT [arc_degree]> POINT (-3.72614 40.37898),
POINT (-3.708702 4~
```

[37]:
```
tdata_02 |> tibble() |> slice_head(n = 5)
```

	CODBDT <int>	GEOCODIGO <chr>	DESBDT <chr>	geometry <POINT [arc_degree]>
A tibble: 5 x 4	686213	001	Abrantes	POINT (-3.72614 40.37898)
	686214	002	Acacias	POINT (-3.708702 40.40045)
	686215	003	Adelfas	POINT (-3.672841 40.40145)
	686216	004	Alameda	POINT (-3.697291 40.41036)
	686217	005	Alameda de Osuna	POINT (-3.569774 40.47515)

Data extract

- Seleccionar variables temperatura, precipitaciones y presión
- Agrupar por semana

[29]:
```
edata_01 <- data_01 |>
    mutate(ano = year(fecha),
        semana = isoweek(fecha)) |>
    select(ano, semana, nombre, tmed, prec, velmedia, presMax, X, Y) |>
    group_by(ano, semana, nombre, X, Y) |>
    summarise(across(everything(), ~mean(.x, na.rm = TRUE)))
```

```
`summarise()` has grouped output by 'ano', 'semana', 'nombre', 'X'. You
can
override using the `.groups` argument.
```

Data transformation

- Convertir datos a objeto espacial para poder interpolar

[58]:
```
tdata_01 <- edata_01 |>
    st_as_sf(coords = c("X", "Y"), crs = 4326)
```

[59]:
```
glimpse(tdata_01)
```

```
Rows: 909
Columns: 8
Groups: ano, semana, nombre [909]
$ ano      <dbl> 2021, 2021, 2021, 2021, 2021, 2021, 2021,
2021, 2021, 2021, 2~
$ semana   <dbl> 35, 35, 35, 35, 35, 35, 35, 35, 35, 35, 35,
35, 36, 36, 36, 3~
$ nombre   <chr> "ARANJUEZ", "BUITRAGO DEL LOZOYA", "COLMENAR
VIEJO", "GETAFE"~
$ tmed     <dbl> 23.08000, 19.00000, 20.40000, 23.04000,
21.84000, 22.40000, 2~
$ prec     <dbl> 2.96000000, NaN, 1.60000000, 0.22000000,
1.24000000, 1.680000~
$ velmedia <dbl> 1.600000, 1.560000, 1.840000, 2.380000,
2.500000, 2.160000, N~
$ presMax  <dbl> NaN, 906.1400, 906.1400, 947.2600, 951.1800,
```

```
939.5800, 941.20~
$ geometry <POINT [arc_degree]> POINT (-3.546111 40.06722),
POINT (-3.613611 4~
```

[35]:
```
tdata_01 |> tibble() |> slice_head(n = 5)
```

A tibble: 5 x 8

ano <dbl>	semana <dbl>	nombre <chr>	tmed <dbl>	prec <dbl>	velmedia <dbl>	presMax <dbl>	geomet <POI
2021	35	ARANJUEZ	23.08	2.96	1.60	NaN	POINT
2021	35	BUITRAGO DEL LOZOYA	19.00	NaN	1.56	906.14	POINT
2021	35	COLMENAR VIEJO	20.40	1.60	1.84	906.14	POINT
2021	35	GETAFE	23.04	0.22	2.38	947.26	POINT
2021	35	MADRID AEROPUERTO	21.84	1.24	2.50	951.18	POINT

- Interpolar temperatura media, precipitación, velocidad media del viento y presión máxima en las coordenadas de cada infraestructura.

NOTA: esta operación puede tardar varios minutos.

[65]:
```
meteovars <- c("tmed", "prec", "velmedia", "presMax")
semanas <- paste(tdata_01$ano, tdata_01$semana, sep = "-") |> unique()
resl <- list()

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

  res <- data_02 |> select(GEOCODIGO) |> st_drop_geometry()
  for (j in seq_along(meteovars)){
    thisvar <- meteovars[j]
    meteodc <- meteod |>
      drop_na(all_of(thisvar))

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

      fo <- as.formula(paste0(thisvar, " ~ 1"))
      v0 <- variogram(fo, meteodc)
      v.m <- suppressWarnings(fit.variogram(v0,
        vgm(c("Exp", "Mat", "Sph", "Log", "Wav", "Bes", "Lin", "Leg"))))
      invisible(capture.output(suppressWarnings(krg <- krige(fo,
                  locations = meteodc,
```

```
                    newdata = tdata_02,
                    model = v.m))))
      res <- res |>
        mutate(!!thisvar := krg$var1.pred)
    }
  }
    res <- res |>
      mutate(ano = a,
             semana = s)
  resl[[i]] <- res
}

data <- bind_rows(resl)
```

[66]:
```
glimpse(data)
```

```
Rows: 21,736
Columns: 7
$ GEOCODIGO <chr> "001", "002", "003", "004", "005", "006",
"007", "008", "009~
$ tmed      <dbl> 22.53414, 22.42739, 22.44443, 22.37906,
21.83238, 22.57096, ~
$ prec      <dbl> 1.2086417, 1.2461203, 1.2778878, 1.2615862,
1.2790072, 1.019~
$ velmedia  <dbl> 2.299920, 2.310469, 2.382023, 2.321377,
2.461297, 2.029950, ~
$ presMax   <dbl> 940.7107, 938.2132, 941.6089, 939.4657,
951.1882, 944.2638, ~
$ ano       <chr> "2021", "2021", "2021", "2021", "2021",
"2021", "2021", "202~
$ semana    <chr> "35", "35", "35", "35", "35", "35", "35",
"35", "35", "35", ~
```

0.4 Synthetic Data Generation

No aplica

0.5 Fake Data Generation

No aplica

0.6 Open Data

Los datos se obtuvieron de fuentes abiertas en la tarea anterior

0.7 Data Save

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

Identificamos los datos a guardar

[70]:
```
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

[67]:
```
caso <- "CU_04"
proceso <- '_05'
tarea <- "_11"
archivo <- ""
proper <- "_meteo_zonas"
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

[68]:
```
# 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/.

[71]:
```
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_11_meteo_zonas.csv'

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

[72]:
```
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 y lubridate 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 * lubridate 1.9.1

0.8.3 Data structures

Objeto `data`

- Hay 21736 filas
  - GEOCODIGO
  - tmed
  - prec
  - velmedia
  - presMax
  - ano
  - semana

Observaciones generales sobre los datos

- Los datos se han promediado por zona sanitaria y semana
- Se incluyen cuatro variables meteorológicas
- En vez de la humedad que estaba prevista en el MVP, se incluye la precipitación

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 meteorología a las zonas

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

- Se debe comprobar por qué se crea una seman 2023-52

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
```