05. - Data Collection_CU_25_06_indicadores_areas_v_01

June 10, 2023

#

CU25_Modelo de gestión de Lista de Espera Quirúrgica

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 06. Obtener indicadores por área sanitaria

- Agrupar indicadores del INE por área sanitaria para introducir en los modelos

1. Primero se asignan zonas básicas de salud a las áreas sanitarias (vienen de CU_04).
2. Se obtienen los indicadores del INE para zonas básicas de salud (vienen de CU_04).
3. Se agregan los indicadores por área sanitaria

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.

[1]:
```
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

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
- {stringr} para manipulación de cadenas de caracteres
- {tidyr} para organización de datos

[40]:
```
library(sf)
library(readr)
library(dplyr)

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

0.1.3 Paths

[3]:
```
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. Zonas básicas de salud

OPCION A: Seleccionar fichero en ventana para mayor comodidad

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

[4]:
```
# file_data_01 <- tcltk::tk_choose.files(multi = FALSE)
```

OPCION B: Especificar el nombre de archivo

[5]:
```
iFile_01 <- "CU_04_05_01_zonasgeo.json"
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 leerán 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, …)

[6]:
```
data_01 <- st_read(file_data_01)
```

```
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/25_listas_esper
a/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:

[7]:
```
data_01 |> 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 [°]> MULTIPOLYGON (((-3.718306
4…, MULTIPOLYGON ((…
```

Muestra de datos:

[8]:
```
data_01 |> tibble() |> slice_head(n = 5)
```

	CODBDT <int>	GEOCODIGO <chr>	DESBDT <chr>	geometry <MULTIPOLYGON [°]>
A tibble: 5 x 4	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...

2. Áreas de salud

OPCION A: Seleccionar fichero en ventana para mayor comodidad

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

[9]:
```
# file_data_02 <- tcltk::tk_choose.files(multi = FALSE)
```

OPCION B: Especificar el nombre de archivo

[10]:
```
iFile_02 <- "CU_25_05_03_areasgeo.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 leerán datos del archivo:  Data/Input/CU_25_05_03_areasgeo.json
```

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

[11]:
```
data_02 <- st_read(file_data_02)
```

```
Reading layer `CU_25_05_03_areasgeo' from data source
  `/Users/emilio.lcano/academico/gh_repos/__transferencia/citizenlab/CitizenLab-
Research-and-Development/casos_urjc/notebooks/II_data_processing/25_listas_esper
a/Data/Input/CU_25_05_03_areasgeo.json'
  using driver `GeoJSON'
Simple feature collection with 11 features and 3 fields
Geometry type: MULTIPOLYGON
Dimension:     XY
Bounding box:  xmin: -4.580315 ymin: 39.88284 xmax: -3.054265 ymax: 41.16397
Geodetic CRS:  WGS 84
```

Estructura de los datos:

[12]:
```
data_02 |> glimpse()
```

```
Rows: 11
Columns: 4
$ CODBDT    <int> 958285, 958286, 958287, 958288, 958289,
```

```
958290, 958291, 9582…
$ GEOCODIGO <chr> "01", "02", "03", "04", "05", "06", "07",
"08", "09", "10", …
$ DESBDT    <chr> "Sur-Este", "Centro-Norte", "Este",
"Noreste", "Norte", "Oes…
$ geometry  <MULTIPOLYGON [°]> MULTIPOLYGON (((-3.423161
4…, MULTIPOLYGON ((…
```

Muestra de datos:

[13]:
```
data_02 |> tibble() |> slice_head(n = 5)
```

A tibble: 5 x 4

CODBDT <int>	GEOCODIGO <chr>	DESBDT <chr>	geometry <MULTIPOLYGON [°]>
958285	01	Sur-Este	MULTIPOLYGON (((-3.423161 4...
958286	02	Centro-Norte	MULTIPOLYGON (((-3.499207 4...
958287	03	Este	MULTIPOLYGON (((-3.323953 4...
958288	04	Noreste	MULTIPOLYGON (((-3.653662 4...
958289	05	Norte	MULTIPOLYGON (((-3.537981 4...

3. Indicadores por zona básica de salud

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_03 <- tcltk::tk_choose.files(multi = FALSE)
```

OPCION B: Especificar el nombre de archivo

[14]:
```
iFile_03 <- "CU_04_05_05_02_indicadores_zonas.csv"
file_data_03 <- paste0(iPath, iFile_03)

if(file.exists(file_data_03)){
    cat("Se leerán datos del archivo: ", file_data_03)
} else{
    warning("Cuidado: el archivo no existe.")
}
```

```
Se leerán datos del archivo:  Data/Input/CU_04_05_05_02_indicadores_zonas.csv
```

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

[15]:
```
data_03 <- read_csv(file_data_03)
```

```
Rows: 286 Columns: 20
  Column specification

Delimiter: ","
chr  (2): id_zona, nombre_zona
dbl (18): nsec, t3_1, t1_1, t2_1, t2_2, t4_1, t4_2, t4_3, t5_1, t6_1,
```

```
t7_1, …

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

Estructura de los datos:

[16]:
```
data_03 |> glimpse()
```

```
Rows: 286
Columns: 20
$ id_zona          <chr> "001", "002", "003", "004", "005",
"006", "007", "008"…
$ nombre_zona      <chr> "Abrantes", "Acacias", "Adelfas",
"Alameda", "Alameda …
$ nsec             <dbl> 23, 11, 19, 18, 18, 15, 10, 17, 11,
12, 14, 17, 8, 20,…
$ t3_1             <dbl> 42.31249, 46.15717, 45.30472,
43.70575, 43.31968, 44.2…
$ t1_1             <dbl> 29872, 17961, 28933, 21393, 27259,
21186, 12367, 27448…
$ t2_1             <dbl> 0.5345094, 0.5328775, 0.5383134,
0.4919805, 0.5153616,…
$ t2_2             <dbl> 0.4654906, 0.4671225, 0.4616866,
0.5080195, 0.4846384,…
$ t4_1             <dbl> 0.15619346, 0.10929873, 0.12408817,
0.07409008, 0.1742…
$ t4_2             <dbl> 0.6656459, 0.6516215, 0.6541986,
0.7685363, 0.6063697,…
$ t4_3             <dbl> 0.17816190, 0.23909411, 0.22173279,
0.15740288, 0.2193…
$ t5_1             <dbl> 0.21839930, 0.04313645, 0.07236948,
0.26145728, 0.0705…
$ t6_1             <dbl> 0.34508551, 0.07700825, 0.12420792,
0.35309699, 0.1280…
$ t7_1             <dbl> 0.04090796, 0.08494593, 0.07195008,
0.04949630, 0.0715…
$ t8_1             <dbl> 0.02880326, 0.07576304, 0.06532400,
0.04323113, 0.0626…
$ t9_1             <dbl> 0.2685716, 0.6368488, 0.6031800,
0.5672350, 0.5862647,…
$ t10_1            <dbl> 0.16656066, 0.08377987, 0.08766432,
0.12302998, 0.0851…
$ t11_1            <dbl> 0.4723181, 0.5083022, 0.5262385,
0.5410470, 0.5028547,…
$ t12_1            <dbl> 0.5637381, 0.5548573, 0.5770966,
0.6171510, 0.5490471,…
```

```
$ area            <dbl> 1571618.8, 771569.7, 854805.6,
547596.1, 35137989.9, 6…
$ densidad_hab_km <dbl> 19007.1534, 23278.5190, 33847.4619,
39067.1135, 775.77…
```

Muestra de datos:

[17]:
```
data_03 |> tibble() |> slice_head(n = 5)
```

A tibble: 5 x 20

id_zona <chr>	nombre_zona <chr>	nsec <dbl>	t3_1 <dbl>	t1_1 <dbl>	t2_1 <dbl>	t2_2 <dbl>	t4_1 <dbl>
001	Abrantes	23	42.31249	29872	0.5345094	0.4654906	0.1561934
002	Acacias	11	46.15717	17961	0.5328775	0.4671225	0.1092987
003	Adelfas	19	45.30472	28933	0.5383134	0.4616866	0.1240881
004	Alameda	18	43.70575	21393	0.4919805	0.5080195	0.0740900
005	Alameda de Osuna	18	43.31968	27259	0.5153616	0.4846384	0.1742765

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:

- Zonas básicas de salud
- Áreas sanitarias
- Indicadores del INE por zona básica

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

- Asignar área sanitaria a cada una de las zonas básicas de salud

[18]:
```
tdata_01 <- data_01 |>
  select(id_zona = GEOCODIGO,
         nombre_zona = DESBDT) |>
  st_join(data_02 |>
            select(id_area = GEOCODIGO,
                   nombre_area = DESBDT),
          largest = TRUE) |>
  st_drop_geometry()
```

```
Warning message:
"attribute variables are assumed to be spatially constant throughout all
geometries"
```

[19]:
```
glimpse(tdata_01)
```

```
Rows: 286
Columns: 4
$ id_zona     <chr> "001", "002", "003", "004", "005", "006",
"007", "008", "0…
$ nombre_zona <chr> "Abrantes", "Acacias", "Adelfas",
"Alameda", "Alameda de O…
$ id_area     <chr> "11", "11", "01", "07", "04", "08", "01",
"05", "11", "05"…
$ nombre_area <chr> "Sur Ii", "Sur Ii", "Sur-Este", "Centro-
Oeste", "Noreste",…
```

[20]:
```
tdata_01 |> slice_head(n = 5)
```

A data.frame: 5 x 4

id_zona <chr>	nombre_zona <chr>	id_area <chr>	nombre_area <chr>
001	Abrantes	11	Sur Ii
002	Acacias	11	Sur Ii
003	Adelfas	01	Sur-Este
004	Alameda	07	Centro-Oeste
005	Alameda de Osuna	04	Noreste

- Añadir información del área sanitaria a la tabla de indicadores y agregar por área

Los datos están en porcentajes y tenemos población total. Calculamos porcentajes globales multiplicando por la población de cada zona sanitaria, sumando población de la variable y dividiendo por la suma de la población total en el área.

[31]:
```
tdata_03 <- data_03 |>
    inner_join(tdata_01, by = c("id_zona", "nombre_zona")) |>
    mutate(across(c(t2_1:t2_2, t4_1:t12_1), ~ .x*t1_1)) |>
    group_by(id_area, nombre_area) |>
    summarise(t3_1 = weighted.mean(x = t3_1, w = t1_1, na.rm = TRUE),
            t1_1 = sum(t1_1, na.rm = TRUE),
            across(c(t2_1:t2_2, t4_1:t12_1), ~sum(.x, na.rm = TRUE)/sum(t1_1,␣
 ↪na.rm = TRUE)),
            .groups = "keep")
```

[33]:
```
glimpse(tdata_03)
```

```
Rows: 11
Columns: 17
Groups: id_area, nombre_area [11]
$ id_area     <chr> "01", "02", "03", "04", "05", "06", "07",
"08", "09", "10"…
$ nombre_area <chr> "Sur-Este", "Centro-Norte", "Este",
"Noreste", "Norte", "O…
$ t3_1        <dbl> 41.89330, 44.85910, 41.16384, 43.50412,
41.38840, 40.89333…
$ t1_1        <dbl> 862340, 404085, 383079, 615157, 912485,
763506, 511605, 53…
```

```
$ t2_1        <dbl> 0.5207432, 0.5419660, 0.5091281,
0.5321335, 0.5202002, 0.5…
$ t2_2        <dbl> 0.4792568, 0.4580340, 0.4908719,
0.4678665, 0.4791324, 0.4…
$ t4_1        <dbl> 0.1585425, 0.1293202, 0.1671827,
0.1495244, 0.1700457, 0.1…
$ t4_2        <dbl> 0.6732152, 0.6456867, 0.6697046,
0.6530478, 0.6590844, 0.6…
$ t4_3        <dbl> 0.1682410, 0.2249913, 0.1631133,
0.1974297, 0.1702033, 0.1…
$ t5_1        <dbl> 0.1399664, 0.1279791, 0.1520691,
0.1266269, 0.1248427, 0.1…
$ t6_1        <dbl> 0.2042778, 0.1854376, 0.1903965,
0.2132414, 0.1950003, 0.1…
$ t7_1        <dbl> 0.05120722, 0.05930819, 0.05762767,
0.05909755, 0.06406634…
$ t8_1        <dbl> 0.03895576, 0.05200647, 0.04557209,
0.04845935, 0.05318346…
$ t9_1        <dbl> 0.3600324, 0.5572380, 0.3196460,
0.4600102, 0.4872258, 0.5…
$ t10_1       <dbl> 0.14408131, 0.09486110, 0.14520694,
0.11547830, 0.10328147…
$ t11_1       <dbl> 0.5193163, 0.5069128, 0.5250289,
0.5117620, 0.5420084, 0.5…
$ t12_1       <dbl> 0.6047756, 0.5605799, 0.6109222,
0.5773072, 0.6035461, 0.5…
```

[35]:
```
tdata_03 |> slice_head(n = 5)
```

A grouped_df: 11 x 17

id_area <chr>	nombre_area <chr>	t3_1 <dbl>	t1_1 <dbl>	t2_1 <dbl>	t2_2 <dbl>	t4_1 <dbl>	t4_2 <dbl
01	Sur-Este	41.89330	862340	0.5207432	0.4792568	0.1585425	0.673
02	Centro-Norte	44.85910	404085	0.5419660	0.4580340	0.1293202	0.645
03	Este	41.16384	383079	0.5091281	0.4908719	0.1671827	0.669
04	Noreste	43.50412	615157	0.5321335	0.4678665	0.1495244	0.653
05	Norte	41.38840	912485	0.5202002	0.4791324	0.1700457	0.659
06	Oeste	40.89333	763506	0.5178944	0.4821056	0.1769738	0.663
07	Centro-Oeste	45.37878	511605	0.5296804	0.4703198	0.1054260	0.674
08	Sur-Oeste I	42.34715	532487	0.5122493	0.4877507	0.1659665	0.637
09	Sur-Oeste Ii	42.05168	406320	0.5116119	0.4883883	0.1603501	0.660
10	Sur I	40.02239	409470	0.5067508	0.4932492	0.1863513	0.657
11	Sur Ii	42.06149	899702	0.5240445	0.4759555	0.1540793	0.675

0.4 Synthetic Data Generation

No aplica

0.5 Fake Data Generation

No aplica

0.6 Open Data

No aplica

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

[36]:
```
data_to_save <- tdata_03
```

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

[37]:
```
caso <- "CU_25"
proceso <- '_05'
tarea <- "_06"
archivo <- ""
proper <- "_indicadores_area"
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/.

[38]:
```
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_25_05_06_indicadores_area.csv'

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

[39]:
```
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 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

0.8.3 Data structures

Objeto `tdata_03`

- Hay filas con información de las siguientes variables:
  - id_area
  - nombre_area

  - t3_1
  - t1_1
  - t2_1
  - t2_2
  - t4_1
  - t4_2
  - t4_3
  - t5_1
  - t6_1
  - t7_1
  - t8_1
  - t9_1
  - t10_1
  - t11_1
  - t12_1

Observaciones generales sobre los datos

- Los datos iniciales estaban por sección censal. En el cu_04 se asignaron a zona básica de salud. Ahora se han agrupado por área sanitaria, que es donde hemos clasificado a los hospitales.

0.8.4 Consideraciones para despliegue en piloto

- No aplica

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 asignado las zonas básicas a las zonas de salud por intersección de las geometrías
- Se han agregado los indicadores por área de salud

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

- Se deben combinar con el resto de datos para aplicar 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
```