05. - Data Collection_CU_04_02_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 02. Zonas básicas de salud

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

- {readr} from *tidyverse* for reading and writing csv files
- {tcltk} for selecting files and paths (if needed)
- {sf} for reading and writing files with spatial information
- {dplyr} for data exploration

[82]:
```
library(readr)
library(tcltk)
library(sf)
library(dplyr)
```

0.1.3 Paths

[83]:
```
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.

OPCION A: Seleccionar fichero en ventana para mayor comodidad

Data load using the {tcltk} package (uncomment for using this option)

[84]:
```
# file_data_01 <- tcltk::tk_choose.files(multi = FALSE)
```

OPCION B: Especificar el nombre de archivo

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) - Comentar si se usa la opción A

[85]:
```
iFile <- "CU_04_05_01_zonasgeo.json"
file_data <- paste0(iPath, iFile)

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

```
Se leerán datos del archivo:  Data/Input/CU_04_05_01_zonasgeo.json
```

Data file to dataframe Utilizamos en todos los notebooks SIEMPRE data como nombre de la matriz (dataframe) principal donde cargamos los datos. Si se importan más ficheros, se le pone un sufijo con número correlativo en dos cifras y una palabra informativa. Por ejemplo, data_01_poblacion. Se repiten las celdas de estructura y muestra de datos para dataframe.

[86]:
```
data <- st_read(file_data)
```

```
Reading layer `CU_04_05_01_zonasgeo' from data source
  `/Users/emilio.lcano/academico/gh_repos/__transferencia/citizenlab/notebooks/I
I_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
```

NOTE: This object is a special type of dataframe that includes the geometry of each observation.

Estructura de los datos:

[87]:
```
glimpse(data)
```

```
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:

[88]:
```
slice_head(data, n = 5)
```

A sf: 5 × 4

CODBDT <int>	GEOCODIGO <chr>	DESBDT <chr>	geometry <MULTIPOLYGON [°]>
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 Transform data

- Para asegurar la trazabilidad, se guardan las transformaciones en un objeto diferente con el prefijo t
- Convertir los datos erróneos a válidos
- Calcular coordenadas del centroide de la zona sanitaria y guardar en columnas
- Calcular superficie de la zona para posteriormente calcular la densidad de población
- Eliminar información de área (polígonos)

Calcular área de la zona

[89]:
```
sf_use_s2(FALSE)
tdata <- data |>
    st_centroid()
tdata <- tdata |>
    mutate(area = st_area(data)) |>
    bind_cols(st_coordinates(tdata)) |>
    st_drop_geometry()
```

```
Warning message in st_centroid.sf(data):
"st_centroid assumes attributes are constant over geometries of x"
Warning message in st_centroid.sfc(st_geometry(x), of_largest_polygon =
of_largest_polygon):
"st_centroid does not give correct centroids for longitude/latitude data"
```

Muestra datos transformados:

[90]:
```
tdata |> slice_head(n = 5)
```

A data.frame: 5 × 6

CODBDT <int>	GEOCODIGO <chr>	DESBDT <chr>	area <[m^2]>	X <dbl>	Y <dbl>
686213	001	Abrantes	1571618.8 [m^2]	-3.726140	40.37898
686214	002	Acacias	771569.7 [m^2]	-3.708702	40.40045
686215	003	Adelfas	854805.6 [m^2]	-3.672841	40.40145
686216	004	Alameda	547596.1 [m^2]	-3.697291	40.41036
686217	005	Alameda de Osuna	35137989.9 [m^2]	-3.569774	40.47516

0.4 Synthetic Data Generation

Estos datos no requieren tareas de este tipo.

0.5 Fake Data Generation

Estos datos no requieren tareas de este tipo.

0.6 Open Data

Los datos de origen fueron descargados de una fuente abierta, pero ya están preparados como fichero de entrada

0.7 Data Save

Identificamos los datos a guardar

[91]:
```
data_to_save <- tdata
```

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

[92]:
```
caso <- "CU_04"
proceso <- '_05'
tarea <- "_02"
archivo <- ""
proper <- "_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
- Descomentar si se usa esta opción

[93]:
```
# file_save <- paste0(caso, proceso, tarea, tcltk::tkgetSaveFile(), proper,␣
 ↪extension)
# path_out <- paste0(oPath, file_save)
# write_csv(datos, 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/.

[94]:
```
file_save <- paste0(caso, proceso, tarea, archivo, proper, extension)
path_out <- paste0(oPath, file_save)
write_csv(tdata, path_out)

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

```
File saved as:
```

’Data/Output/CU_04_05_02_zonas.csv’

Copia del fichero a Input Será usado en otros notebooks

[95]:
```
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

This working code needs the following conditions:

- For using the interactive selection of file, the {tcltk} package must be installed. It is not needed in production.
- The {readr} and {dplyr} packages must be installed. They are part of the *tidyverse*.
- The {sf} package must be installed. It needs some system requierements, check the package documentation at https://r-spatial.github.io/sf/
- The data paths `Data/Input` and `Data/Output` must exist (relative to the notebook path), as well as the input file

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 * readr 2.1.3 * sf 1.0.9 * dplyr 1.0.10

0.8.3 Data structures

- Los datos de origen fueron obtenidos de https://gestiona.comunidad.madrid/nomecalles/DescargaBDTCorte. y transformados previamente a un archivo .json, formato GeoJSON. Si la información geográfica cambia, se debería actualizar el fichero y volver a ejecutar todos los procesos.
- Los datos geográficos en el fichero JSON vienen con la proyección CRS WGS 84 (código 4326)
- Hay 286 zonas básicas de salud de las que se dispone de:
  - GEOCODIGO, string de tamaño tres con el código de la zona básica sanitaria que después se puede unir a otras tablas.
  - DESBDT, string de tamaño variable que contiene el nombre de la zona básica sanitaria y se usará para describir la zona.
  - CODBDT, entero que no utilizamos y no está claro para qué puede servir

0.9 Main Actions

Acciones done

- Se han calculado las coordenadas de los centroides de las zonas de salud por si son útiles en futuros procesos

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

- Se deben unir a estos datos los de indicadores del INE
- Se debe calcular la densidad de población de cada zona

- Los datos estarán relacionados con los de vacunación, meteorológicos y contaminación, de los que habrá varios registros por cada zona (uno por semana)
- También se relacionarán a través del municipio con los de tendencias en internet

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

[96]:
```
# incluir código
```

0.11 Consideraciones para despliegue en piloto

- El archivo de entrada se utilizará para la representación de mapas.

0.12 Consideraciones para despliegue en producción

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