05. - Data Collection_CU_25_03_areasgeo_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 03. Áreas de salud de shp a json

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

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

[1]:
```
library(tcltk)
library(sf)
library(dplyr)
```

```
Linking to GEOS 3.10.2, GDAL 3.4.2, PROJ 8.2.1; sf_use_s2() is TRUE


Attaching package: 'dplyr'


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

    filter, lag


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

    intersect, setdiff, setequal, union
```

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.

OPCION A: Seleccionar fichero en ventana para mayor comodidad

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

[3]:
```
# file_data <- 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

[4]:
```
iFile <- "Areas de Salud/200001906.shp"
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 leer<U+00E1>n datos del archivo:  Data/Input/Areas de Salud/200001906.shp
```

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.

[5]:
```
data <- st_read(file_data,
                options = "ENCODING=WINDOWS-1252")
```

```
options:        ENCODING=WINDOWS-1252
Reading layer `200001906' 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/Areas de Salud/200001906.shp'
  using driver `ESRI Shapefile'
Simple feature collection with 11 features and 3 fields
Geometry type: MULTIPOLYGON
Dimension:     XY
Bounding box:  xmin: 365635.7 ymin: 4415272 xmax: 495484 ymax: 4557310
CRS:           NA
```

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

Estructura de los datos:

[6]:
```
glimpse(data)
```

```
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 (((464211.8 44…,
MULTIPOLYGON (((457~
```

Muestra de datos:

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

	CODBDT <int>	GEOCODIGO <chr>	DESBDT <chr>	geometry <MULTIPOLYGON>
A tibble: 5 x 4	958285	01	Sur-Este	MULTIPOLYGON (((464211.8 44...
	958286	02	Centro-Norte	MULTIPOLYGON (((457789.2 44...
	958287	03	Este	MULTIPOLYGON (((472715.8 44...
	958288	04	Noreste	MULTIPOLYGON (((444729.3 44...
	958289	05	Norte	MULTIPOLYGON (((454972 4556...

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 a crs 4326 que es el estándar del caso de uso (proyecto?)

[8]:
```
tdata <- data |>
  st_set_crs(23030) |>
  st_transform(4326)
```

Muestra datos transformados:

[9]:
```
tdata |> tibble() |> slice_head(n = 5)
```

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

Estructura de la proyección:

[10]:
```
st_crs(tdata)
```

```
Coordinate Reference System:
  User input: EPSG:4326
  wkt:
GEOGCRS["WGS 84",
    ENSEMBLE["World Geodetic System 1984 ensemble",
        MEMBER["World Geodetic System 1984 (Transit)"],
        MEMBER["World Geodetic System 1984 (G730)"],
        MEMBER["World Geodetic System 1984 (G873)"],
        MEMBER["World Geodetic System 1984 (G1150)"],
        MEMBER["World Geodetic System 1984 (G1674)"],
        MEMBER["World Geodetic System 1984 (G1762)"],
        MEMBER["World Geodetic System 1984 (G2139)"],
        ELLIPSOID["WGS 84",6378137,298.257223563,
            LENGTHUNIT["metre",1]],
        ENSEMBLEACCURACY[2.0]],
```

```
PRIMEM["Greenwich",0,
    ANGLEUNIT["degree",0.0174532925199433]],
CS[ellipsoidal,2],
    AXIS["geodetic latitude (Lat)",north,
        ORDER[1],
        ANGLEUNIT["degree",0.0174532925199433]],
    AXIS["geodetic longitude (Lon)",east,
        ORDER[2],
        ANGLEUNIT["degree",0.0174532925199433]],
USAGE[
    SCOPE["Horizontal component of 3D system."],
    AREA["World."],
    BBOX[-90,-180,90,180]],
ID["EPSG",4326]]
```

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 en fichero zip que se ha descomprimido dentro de la carpeta Data/Input.

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

[11]:
```
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

[12]:
```
caso <- "CU_25"
proceso <- '_05'
tarea <- "_03"
archivo <- ""
proper <- "_areasgeo"
extension <- ".json"
```

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 líneas si se usa esta opción

[13]:
```
# file_save <- paste0(caso, proceso, tarea, archivo, tcltk::tkgetSaveFile(),␣
 ↪extension)
# path_out <- paste0(oPath, file_save)
# st_write(obj = data_to_save,
        #  dsn = path_out,
        #  driver = "GeoJSON",
        #  delete_dsn = TRUE)

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

OPCION B: Especificar el nombre de archivo

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

[14]:
```
file_save <- paste0(caso, proceso, tarea, archivo, proper, extension)
path_out <- paste0(oPath, file_save)
st_write(obj = data_to_save,
         dsn = path_out,
         driver = "GeoJSON",
         delete_dsn = TRUE)


cat('\nFile saved as: ', path_out)
```

```
Deleting source `Data/Output/CU_25_05_03_areasgeo.json' using driver `GeoJSON'
Writing layer `CU_25_05_03_areasgeo' to data source
  `Data/Output/CU_25_05_03_areasgeo.json' using driver `GeoJSON'
Writing 11 features with 3 fields and geometry type Multi Polygon.

File saved as:  Data/Output/CU_25_05_03_areasgeo.json
```

Copia del fichero a Input Si fichero es usado en otras tareas, copiar a carpeta Input

[15]:
```
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 Prerequisitos

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 {dplyr} package must be installed. It is 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)

0.8.2 Gestión de la configuración

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 * dplyr 1.0.10

0.8.3 Estructuras de datos

- Los datos geográficos en el fichero JSON vienen con la proyección CRS 23030
- Hay 11 áreas de salud de las que se dispone de:
  - GEOCODIGO, string de tamaño dos con el código del área de salud que después se puede unir a otras tablas.
  - DESBDT, string de tamaño variable que contiene el nombre del área sanitaria y se usará para describir el área.
  - CODBDT, entero que no utilizamos y no está claro para qué puede servir

0.8.4 Consideraciones para despliegue en piloto

- Los datos de origen fueron obtenidos de https://gestiona.comunidad.madrid/nomecalles/DescargaBDTCorte. en fichero zip con formato .shp y descomprimido en la carpeta Data/Input
- Si la información geográfica cambia, se debería actualizar el fichero y volver a ejecutar todos los procesos.

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 ha usado el encoding de Windows para la importación
- Se ha convertido la proyección a EPSG:4326 (WGS84)
- Se ha exportado a formato GeoJson

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

- Se deben asignar los hospitales a las zonas según sus coordenadas

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

[16]:
```
# incluir código
```