05. - Data Collection_CU_18_03_distritos_geo_v_01

June 13, 2023

#

CU18_Infraestructuras_eventos

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. Obtener datos de geometrías secciones censales en formato json

- Dados los ficheros shp del INE, obtener un solo archivo json con la información georeferenciada
- Los datos están por sección censal, hay que unir los polígonos en distritos censales

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} para seleccionar archivos locales de forma interactiva
- {sf} para trabajar con georeferenciación
- {dplyr} para explorar datos

[18]:
```
library(sf)
library(dplyr)
```

0.1.2 Paths

[19]:
```
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)

Archivo de geometrías OPCION A: Seleccionar fichero en ventana para mayor comodidad

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

[20]:
```
# file_data <- tcltk::tk_choose.files(multi = FALSE)
```

OPCION B: Especificar el nombre de archivo

[21]:
```
iFile <- "Espana_Seccionado2022_ETRS89H30/SECC_CE_20220101.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/Espana_Seccionado2022_ETRS89H30/SECC_CE_20220101.shp
```

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

[22]:
```
data <- st_read(file_data)
```

```
Reading layer `SECC_CE_20220101' from data source
  `/Users/emilio.lcano/academico/gh_repos/__transferencia/citizenlab/CitizenLab-
Research-and-Development/casos_urjc/notebooks/II_data_processing/18_infraestruct
uras/Data/Input/Espana_Seccionado2022_ETRS89H30/SECC_CE_20220101.shp'
  using driver `ESRI Shapefile'
Simple feature collection with 36382 features and 16 fields
Geometry type: MULTIPOLYGON
Dimension:     XY
```

```
Bounding box:  xmin: -1004502 ymin: 3132130 xmax: 1126932 ymax: 4859240
Projected CRS: ETRS89 / UTM zone 30N
```

Estructura de los datos:

[23]:
```
glimpse(data)
```

```
Rows: 36,382
Columns: 17
$ CUSEC    <chr> "0100101001", "0100101002", "0100201001",
"0100201002", "0100~
$ CUMUN    <chr> "01001", "01001", "01002", "01002", "01002",
"01002", "01002"~
$ CSEC     <chr> "001", "002", "001", "002", "003", "004",
"005", "006", "007"~
$ CDIS     <chr> "01", "01", "01", "01", "01", "01", "01",
"01", "01", "01", "~
$ CMUN     <chr> "001", "001", "002", "002", "002", "002",
"002", "002", "002"~
$ CPRO     <chr> "01", "01", "01", "01", "01", "01", "01",
"01", "01", "01", "~
$ CCA      <chr> "16", "16", "16", "16", "16", "16", "16",
"16", "16", "16", "~
$ CUDIS    <chr> "0100101", "0100101", "0100201", "0100201",
"0100201", "01002~
$ CLAU2    <chr> "01001", "01001", "01002", "01002", "01002",
"01002", "01002"~
$ NPRO     <chr> "Araba/<U+00C1>lava", "Araba/<U+00C1>lava",
"Araba/<U+00C1>lava", "Araba/<U+00C1>lava", "~
$ NCA      <chr> "Pa<U+00ED>s Vasco", "Pa<U+00ED>s Vasco",
"Pa<U+00ED>s Vasco", "Pa<U+00ED>s Vasco", "Pa<U+00ED>s~
$ CNUT0    <chr> "ES", "ES", "ES", "ES", "ES", "ES", "ES",
"ES", "ES", "ES", "~
$ CNUT1    <chr> "2", "2", "2", "2", "2", "2", "2", "2", "2",
"2", "2", "2", "~
$ CNUT2    <chr> "1", "1", "1", "1", "1", "1", "1", "1", "1",
"1", "1", "1", "~
$ CNUT3    <chr> "1", "1", "1", "1", "1", "1", "1", "1", "1",
"1", "1", "1", "~
$ NMUN     <chr> "Alegr<U+00ED>a-Dulantzi",
"Alegr<U+00ED>a-Dulantzi", "Amurrio", "Amurrio",~
$ geometry <MULTIPOLYGON [m]> MULTIPOLYGON (((539753 4743…,
MULTIPOLYGON (((~
```

Muestra de datos:

[24]:
```
slice_head(data.frame(data), n = 5)
```

A data.frame: 5 × 17

CUSEC <chr>	CUMUN <chr>	CSEC <chr>	CDIS <chr>	CMUN <chr>	CPRO <chr>	CCA <chr>	CUDIS <chr>	CLAU2 <chr>
0100101001	01001	001	01	001	01	16	0100101	01001
0100101002	01001	002	01	001	01	16	0100101	01001
0100201001	01002	001	01	002	01	16	0100201	01002
0100201002	01002	002	01	002	01	16	0100201	01002
0100201003	01002	003	01	002	01	16	0100201	01002

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:

- Geometrías de secciones censales del INE

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

Extracción de datos

- Filtrar solo comunidad de Madrid

[25]:
```
tdata <- data |>
    filter(CPRO == "28")
```

Transformación de datos

- Transformar sistema de referencia CRS a 4326 (WGS 84)
- Agrupar datos por distrito
- Unir geometrías

[26]:
```
tdata <- tdata |>
    st_transform(4326) |>
    group_by(CMUN, CDIS) |>
    summarise(geometry = st_union(geometry))
```

```
`summarise()` has grouped output by 'CMUN', 'CDIS'. You can override
using the
`.groups` argument.
```

[36]:
```
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]]
```

Muestra de primeros datos:

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

A tibble: 5 × 3

CMUN <chr>	CDIS <chr>	geometry <POLYGON [arc_degree]>
001	01	POLYGON ((-3.64502 41.12129...
002	01	POLYGON ((-3.503032 40.526,...
003	01	POLYGON ((-3.808664 40.8921...
004	01	POLYGON ((-4.00197 40.25642...
005	01	POLYGON ((-3.361691 40.4762...

0.4 Synthetic Data Generation

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

0.5 Fake Data Generation

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

0.6 Open Data

Los archivos shp se descargaron de la fuente abierta como ficheros comprimidos .zip que se descomprimieron para tener la estructura de carpetas necesaria

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

[37]:
```
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

[38]:
```
caso <- "CU_18"
proceso <- '_05'
tarea <- "_03"
archivo <- ""
proper <- "_distritos_geo"
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

[30]:
```
# 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/.

[41]:
```
file_save <- paste0(caso, proceso, tarea, archivo, proper, extension)
path_out <- paste0(oPath, file_save)
```

```
st_write(obj = data_to_save,
         dsn = path_out,
         crs = 4326,
         driver = "GeoJSON",
         delete_dsn = TRUE)

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

```
Deleting source `Data/Output/CU_18_05_03_distritos_geo.json' using driver
`GeoJSON'
Writing layer `CU_18_05_03_distritos_geo' to data source
  `Data/Output/CU_18_05_03_distritos_geo.json' using driver `GeoJSON'
Writing 247 features with 2 fields and geometry type Unknown (any).
File saved as:
```

'Data/Output/CU_18_05_03_distritos_geo.json'

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

[42]:
```
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 {sf} and {dplyr} packages must be installed.
- The data paths `Data/Input` and `Data/Output` must exist (relative to the notebook path)

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 * dplyr 1.0.10 * sf 1.0.9

0.8.3 Data structures

Objeto `data`

- Hay 247 filas correspondientes a los distritos de la comunidad de madrid, con la información:
  - CMUN
  - CDIS

- geometría

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

```
Rows: 247
Columns: 3
Groups: CMUN, CDIS [246]
$ CMUN     <chr> "001", "002", "003", "004", "005", "005",
"005", "005", "005"~
$ CDIS     <chr> "01", "01", "01", "01", "01", "02", "03",
"04", "05", "01", "~
$ geometry <POLYGON [arc_degree]> POLYGON ((-3.64502
41.12129…, POLYGON ((-3~
```

Observaciones generales sobre los datos

- Los datos están por sección censal, y se agrupan por distrito
- Si se cambia la proyección por defecto del proyecto, por ejemplo al estándar WGS84, hay que cambiar en las transformaciones y al guardar el archivo.

0.8.4 Consideraciones para despliegue en piloto

- Los datos de origen fueron obtenidos de https://www.ine.es/ss/Satellite?L=es_ES&c=Page&cid=125995202
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 tenido que renombrar la carpeta descargada del INE de Espa§a_Seccionado2022_ETRS89H30 a Espana_Seccionado2022_ETRS89H30
- Se ha transformado al sistema de referencia CRS a 4326 (WGS 84)
- Se han agrupado datos por distrito
- Se han unido geometrías

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

- Se deben asignar distritos censales a cada infraestructura

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

[34]:
```
# incluir código
```