05. - Data Collection_CU_34_01_secciones_geo_v_01

June 16, 2023

#

CU34_Predicción de demanda de servicios

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 01. Obtener datos de geometrías distritos censales

- Obtener como conjunto de archivos shp, contornos de secciones censales, unir en distritos y guardar en json.
- Se descargan del siguiente enlace:

https://www.ine.es/ss/Satellite?L=es_ES&c=Page&cid=1259952026632&p=1259952026632&pagename=Produc

NOTA: no se sincroniza en el repositorio al superar los 100Mb

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

[2]:
```
library(sf)
library(dplyr)

p <- c("tcltk", "sf", "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
```

0.1.3 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)

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

OPCION B: Especificar el nombre de archivo

[5]:
```
iFile <- "Espa§a_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 leerán datos del archivo:
Data/Input/Espa§a_Seccionado2022_ETRS89H30/SECC_CE_20220101.shp
```

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

[6]:
```
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/34_servicios/Da
ta/Input/Espa§a_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:

[7]:
```
data |> tibble() |> glimpse()
```

```
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/Álava", "Araba/Álava", "Araba/Álava",
"Araba/Álava", "…
$ NCA      <chr> "País Vasco", "País Vasco", "País Vasco",
"País Vasco", "País…
$ 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> "Alegría-Dulantzi", "Alegría-Dulantzi",
"Amurrio", "Amurrio",…
$ geometry <MULTIPOLYGON [m]> MULTIPOLYGON (((539753 4743…,
MULTIPOLYGON (((…
```

Muestra de datos:

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

	CUSEC <chr>	CUMUN <chr>	CSEC <chr>	CDIS <chr>	CMUN <chr>	CPRO <chr>	CCA <chr>	CUDIS <chr>	CLAU2 <chr>	NPI <ch
A tibble: 5 x 17	0100101001	01001	001	01	001	01	16	0100101	01001	Ara
	0100101002	01001	002	01	001	01	16	0100101	01001	Ara
	0100201001	01002	001	01	002	01	16	0100201	01002	Ara
	0100201002	01002	002	01	002	01	16	0100201	01002	Ara
	0100201003	01002	003	01	002	01	16	0100201	01002	Ara

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 secciones censales España 2022

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 extract

- Fitrar Comunidad de Madrid

[9]:
```
edata <- data |> filter(CPRO == "28")
```

[10]:
```
edata |> glimpse()
```

```
Rows: 4,432
Columns: 17
$ CUSEC    <chr> "2800101001", "2800201001", "2800201002",
"2800301001", "2800…
$ CUMUN    <chr> "28001", "28002", "28002", "28003", "28004",
"28004", "28004"…
$ CSEC     <chr> "001", "001", "002", "001", "001", "002",
"003", "004", "001"…
$ CDIS     <chr> "01", "01", "01", "01", "01", "01", "01",
"01", "01", "01", "…
$ CMUN     <chr> "001", "002", "002", "003", "004", "004",
"004", "004", "005"…
$ CPRO     <chr> "28", "28", "28", "28", "28", "28", "28",
"28", "28", "28", "…
$ CCA      <chr> "13", "13", "13", "13", "13", "13", "13",
"13", "13", "13", "…
$ CUDIS    <chr> "2800101", "2800201", "2800201", "2800301",
"2800401", "28004…
$ CLAU2    <chr> "28001", "28002", "28002", "28003", "28004",
"28004", "28004"…
$ NPRO     <chr> "Madrid", "Madrid", "Madrid", "Madrid",
"Madrid", "Madrid", "…
```

```
$ NCA      <chr> "Comunidad de Madrid", "Comunidad de
Madrid", "Comunidad de M…
$ CNUT0    <chr> "ES", "ES", "ES", "ES", "ES", "ES", "ES",
"ES", "ES", "ES", "…
$ CNUT1    <chr> "3", "3", "3", "3", "3", "3", "3", "3", "3",
"3", "3", "3", "…
$ CNUT2    <chr> "0", "0", "0", "0", "0", "0", "0", "0", "0",
"0", "0", "0", "…
$ CNUT3    <chr> "0", "0", "0", "0", "0", "0", "0", "0", "0",
"0", "0", "0", "…
$ NMUN     <chr> "Acebeda, La", "Ajalvir", "Ajalvir",
"Alameda del Valle", "Ál…
$ geometry <MULTIPOLYGON [m]> MULTIPOLYGON (((446007.9 45…,
MULTIPOLYGON (((…
```

Data transform

- Transformar crs a 4258

[11]:
```
tdata <- edata |> st_transform(4326)
```

[12]:
```
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]]
```

[13]:
```
tdata |> glimpse()
```

```
Rows: 4,432
Columns: 17
$ CUSEC    <chr> "2800101001", "2800201001", "2800201002",
"2800301001", "2800…
$ CUMUN    <chr> "28001", "28002", "28002", "28003", "28004",
"28004", "28004"…
$ CSEC     <chr> "001", "001", "002", "001", "001", "002",
"003", "004", "001"…
$ CDIS     <chr> "01", "01", "01", "01", "01", "01", "01",
"01", "01", "01", "…
$ CMUN     <chr> "001", "002", "002", "003", "004", "004",
"004", "004", "005"…
$ CPRO     <chr> "28", "28", "28", "28", "28", "28", "28",
"28", "28", "28", "…
$ CCA      <chr> "13", "13", "13", "13", "13", "13", "13",
"13", "13", "13", "…
$ CUDIS    <chr> "2800101", "2800201", "2800201", "2800301",
"2800401", "28004…
$ CLAU2    <chr> "28001", "28002", "28002", "28003", "28004",
"28004", "28004"…
$ NPRO     <chr> "Madrid", "Madrid", "Madrid", "Madrid",
"Madrid", "Madrid", "…
$ NCA      <chr> "Comunidad de Madrid", "Comunidad de
Madrid", "Comunidad de M…
$ CNUT0    <chr> "ES", "ES", "ES", "ES", "ES", "ES", "ES",
"ES", "ES", "ES", "…
$ CNUT1    <chr> "3", "3", "3", "3", "3", "3", "3", "3", "3",
"3", "3", "3", "…
$ CNUT2    <chr> "0", "0", "0", "0", "0", "0", "0", "0", "0",
"0", "0", "0", "…
$ CNUT3    <chr> "0", "0", "0", "0", "0", "0", "0", "0", "0",
"0", "0", "0", "…
$ NMUN     <chr> "Acebeda, La", "Ajalvir", "Ajalvir",
"Alameda del Valle", "Ál…
$ geometry <MULTIPOLYGON [°]> MULTIPOLYGON (((-3.64316 41…,
MULTIPOLYGON (((…
```

[14]:
```
tdata |> tibble() |> slice_head(n = 5)
```

	CUSEC <chr>	CUMUN <chr>	CSEC <chr>	CDIS <chr>	CMUN <chr>	CPRO <chr>	CCA <chr>	CUDIS <chr>	CLAU2 <chr>	NPI <ch
A tibble: 5 x 17	2800101001	28001	001	01	001	28	13	2800101	28001	Mad
	2800201001	28002	001	01	002	28	13	2800201	28002	Mad
	2800201002	28002	002	01	002	28	13	2800201	28002	Mad
	2800301001	28003	001	01	003	28	13	2800301	28003	Mad
	2800401001	28004	001	01	004	28	13	2800401	28004	Mad

0.4 Synthetic Data Generation

No aplica

0.5 Fake Data Generation

No aplica

0.6 Open Data

Los datos se han descargado de una fuente abierta (INE)

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

[15]:
```
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

[16]:
```
caso <- "CU_34"
proceso <- '_05'
tarea <- "_01"
archivo <- ""
proper <- "_secciones_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

[17]:
```
# 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/.

[18]:
```
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('File saved as: ')
path_out
```

```
Deleting source `Data/Output/CU_34_05_01_secciones_geo.json' using driver
`GeoJSON'
Writing layer `CU_34_05_01_secciones_geo' to data source
  `Data/Output/CU_34_05_01_secciones_geo.json' using driver `GeoJSON'
Writing 4432 features with 16 fields and geometry type Multi Polygon.
File saved as:
```

’Data/Output/CU_34_05_01_secciones_geo.json’

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

[19]:
```
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, 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 * dplyr 1.0.10

0.8.3 Data structures

Objeto `tdata`

- Hay 4432 filas con información de las siguientes variables:
  - CUSEC
  - CUMUN
  - CSEC
  - CDIS
  - CMUN
  - CPRO
  - CCA
  - CUDIS
  - CLAU2
  - NPRO
  - NCA
  - CNUT0
  - CNUT1
  - CNUT2
  - CNUT3
  - NMUN
  - geometry

Observaciones generales sobre los datos

- Los contornos se han descargado para 2022. Si se usan datos históricos puede que cambien las secciones censales

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 ha transformado el sistema de referencia a EPSG 4258
- Se han guardado los datos en un único archivo JSON

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

- Se debe calcular la superficie de los polígonos

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

[20]:
```
# incluir código
```