05. - Data Collection_CU_45_01_municipios_geo_v_01

June 11, 2023

#

CU45_Planificación y promoción del destino en base a los patrones en origen de los turistas

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 municipios

- Obtener como sf de IGN con paquete mapSpain. Contornos de municipios, solo C. Madrid

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
- {mapSpain} para obtener los contornos de municipios
- {dplyr} para explorar datos

[3]:
```
library(sf)
library(mapSpain)
library(dplyr)

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

0.1.3 Paths

[4]:
```
iPath <- "Data/Input/"
oPath <- "Data/Output/"
```

0.2 Data Load

No aplica al leer los datos en línea de fuentes abiertas.

0.3 ETL Processes

0.3.1 Import data from: CSV, Excel, Tab, JSON, SQL, and Parquet files

No aplica

0.4 Synthetic Data Generation

No aplica

0.5 Fake Data Generation

No aplica

0.6 Open Data

- Obtención de los contornos con el paquete `mapSpain` de `rOpenSpain`

[5]:
```
data <- esp_get_munic() |>
  filter(cpro == "28")
```

Estructura de los datos:

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

```
Rows: 179
Columns: 8
$ codauto       <chr> "13", "13", "13", "13", "13", "13",
"13", "13", "13", "1…
$ ine.ccaa.name <chr> "Madrid, Comunidad de", "Madrid,
Comunidad de", "Madrid,…
$ cpro          <chr> "28", "28", "28", "28", "28", "28",
"28", "28", "28", "2…
$ ine.prov.name <chr> "Madrid", "Madrid", "Madrid", "Madrid",
"Madrid", "Madri…
$ cmun          <chr> "001", "002", "003", "004", "005",
"006", "007", "008", …
$ name          <chr> "Acebeda, La", "Ajalvir", "Alameda del
Valle", "Álamo, E…
$ LAU_CODE      <chr> "28001", "28002", "28003", "28004",
"28005", "28006", "2…
$ geometry      <GEOMETRY [°]> POLYGON ((-3.61592
41.06982…, POLYGON ((-3.51…
```

Muestra de datos:

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

A tibble: 5 x 8

codauto <chr>	ine.ccaa.name <chr>	cpro <chr>	ine.prov.name <chr>	cmun <chr>	name <chr>	LAU_ <chr>
13	Madrid, Comunidad de	28	Madrid	001	Acebeda, La	28001
13	Madrid, Comunidad de	28	Madrid	002	Ajalvir	28002
13	Madrid, Comunidad de	28	Madrid	003	Alameda del Valle	28003
13	Madrid, Comunidad de	28	Madrid	004	Álamo, El	28004
13	Madrid, Comunidad de	28	Madrid	005	Alcalá de Henares	28005

0.7 ETL Processes

0.7.1 Import data from: CSV, Excel, Tab, JSON, SQL, and Parquet files

- Se han importado datos abiertos

Data transformation

[8]:
```
tdata <- data |> st_transform(crs = 4326)
```

0.8 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

[9]:
```
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

0.8.1 Proceso 05

[10]:
```
caso <- "CU_45"
proceso <- '_05'
tarea <- "_01"
archivo <- ""
proper <- "_municipios_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

[ ]:
```
# 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/.

[11]:
```
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_45_05_01_municipios_geo.json' failed
```

```
Writing layer `CU_45_05_01_municipios_geo' to data source
  `Data/Output/CU_45_05_01_municipios_geo.json' using driver `GeoJSON'
Writing 179 features with 7 fields and geometry type Unknown (any).
File saved as:
```

'Data/Output/CU_45_05_01_municipios_geo.json'

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

[12]:
```
path_in <- paste0(iPath, file_save)
file.copy(path_out, path_in, overwrite = TRUE)
```

TRUE

0.9 Main Conclusions

List and describe the general conclusions of the analysis carried out.

0.9.1 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, mapSpain, dplyr deben estar instalados.

0.9.2 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 * mapSpain 0.7.0 * dplyr 1.1.0

NOTA: En los resultados preliminares, problemas al instalar paquete geojsonio por dependencia con GDAL. Resuelta instalando todo el sistema gdal con homebrew. En todo caso para el piloto estamos solo trabajando con GEOJSON y no hace falta ese paquete.

0.9.3 Data structures

Objeto `data`

- Hay 179 filas con información de las siguientes variables:
  - codauto
  - ine.ccaa.name
  - cpro
  - ine.prov.name
  - cmun
  - name
  - LAU_CODE

- geometry

Observaciones generales sobre los datos

- Son datos oficiales

0.9.4 Consideraciones para despliegue en piloto

- No se espera que esos datos cambien, se integrarán en el archivo a utilizar

0.9.5 Consideraciones para despliegue en producción

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

0.10 Main Actions

Acciones done Indicate the actions that have been carried out in this process

- Se han obtenido los contornos de municipios
- Se ha guardado en archivo json

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

- Se deben unir estos datos al resto para el análisis

0.11 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
```