05. - Data Collection_CU_34_04_meteo_secciones_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 04. Estimar datos meteorológicos en secciones censales

- Estimar variables en secciones
- Eliminando variables que no se usarán en modelos

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

- {tcltk} para selección interactiva de archivos locales
- {sf} para trabajar con georeferenciación
- {gstat} para cálculos geoestadísticos
- {readr} para leer y escribir archivos csv
- {dplyr} para explorar datos
- {stringr} para manipulación de cadenas de caracteres
- {tidyr} para quitar perdidos
- {lubridate} para manipulación de fechas

[2]:
```
library(readr)
library(dplyr)
library(sf)
library(gstat)
library(stringr)
library(tidyr)
library(lubridate)
```

```
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


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


Attaching package: 'lubridate'


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

    date, intersect, setdiff, 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.

1. Datos meteorológicos de estaciones

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_01 <- tcltk::tk_choose.files(multi = FALSE)
```

OPCION B: Especificar el nombre de archivo

[5]:
```
iFile_01 <- "CU_34_05_03_aemet.csv"
file_data_01 <- paste0(iPath, iFile_01)

if(file.exists(file_data_01)){
    cat("Se leerán datos del archivo: ", file_data_01)
} else{
    warning("Cuidado: el archivo no existe.")
}
```

```
Se leerán datos del archivo:  Data/Input/CU_34_05_03_aemet.csv
```

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

[6]:
```
data_01 <- read_csv(file_data_01)
```

```
Rows: 372 Columns: 22
  Column specification

Delimiter: ","
chr   (9): indicativo, nombre, provincia, horatmin, horatmax, dir,
horaracha…
dbl  (12): altitud, tmed, prec, tmin, tmax, velmedia, racha, sol,
presMax, p…
date  (1): fecha

  Use `spec()` to retrieve the full column specification for this
data.
  Specify the column types or set `show_col_types = FALSE` to quiet
this message.
```

Estructura de los datos:

[7]:
```
glimpse(data_01)
```

```
Rows: 372
Columns: 22
$ fecha       <date> 2022-01-01, 2022-01-02, 2022-01-03,
2022-01-04, 2022-01-0…
$ indicativo  <chr> "2462", "2462", "2462", "2462", "2462",
"2462", "2462", "2…
$ nombre      <chr> "PUERTO DE NAVACERRADA", "PUERTO DE
NAVACERRADA", "PUERTO …
$ provincia   <chr> "MADRID", "MADRID", "MADRID", "MADRID",
"MADRID", "MADRID"…
$ altitud     <dbl> 1894, 1894, 1894, 1894, 1894, 1894, 1894,
1894, 1894, 1894…
$ tmed        <dbl> 10.6, 9.8, 7.8, 1.6, -3.0, -5.6, -4.0,
-2.8, 0.0, 1.6, 1.0…
$ prec        <dbl> 0.0, 0.0, 0.0, 24.2, 13.9, 0.0, 0.2, 0.0,
6.6, 0.8, 0.0, 0…
$ tmin        <dbl> 7.4, 6.0, 4.2, -3.9, -4.2, -8.3, -6.6,
-4.6, -3.0, 0.3, -0…
$ horatmin    <chr> "00:10", "20:40", "06:20", "23:59",
"03:10", "06:40", "03:…
$ tmax        <dbl> 13.7, 13.5, 11.3, 7.0, -1.8, -2.8, -1.3,
-0.9, 3.0, 3.0, 2…
$ horatmax    <chr> "Varias", "13:30", "Varias", "01:40",
"Varias", "00:00", "…
$ dir         <chr> "22", "22", "27", "33", "32", "35", "36",
"33", "31", "33"…
$ velmedia    <dbl> 3.9, 2.5, 3.1, 7.8, 4.2, 5.6, 4.2, 5.6,
6.1, 6.1, 2.5, 1.7…
$ racha       <dbl> 12.2, 11.1, 15.3, 23.6, 19.2, 10.8, 10.3,
16.1, 23.3, 16.7…
$ horaracha   <chr> "22:30", "00:00", "22:50", "18:30",
"00:30", "20:00", "08:…
$ sol         <dbl> 8.1, 6.0, 7.6, 3.3, 0.0, 6.3, 1.7, 3.9,
0.0, 0.0, 3.4, 8.1…
$ presMax     <dbl> 823.3, 823.5, 821.5, 814.7, 809.5, 813.5,
819.3, 819.2, 81…
$ horaPresMax <chr> "11", "Varias", "00", "00", "11", "24",
"24", "Varias", "2…
$ presMin     <dbl> 821.8, 821.5, 814.7, 803.6, 806.4, 807.6,
813.4, 814.7, 81…
$ horaPresMin <chr> "06", "Varias", "24", "17", "Varias",
"01", "01", "21", "1…
$ X           <dbl> -4.010556, -4.010556, -4.010556,
-4.010556, -4.010556, -4.…
```

```
$ Y           <dbl> 40.79306, 40.79306, 40.79306, 40.79306,
40.79306, 40.79306…
```

Muestra de datos:

[8]:
```
slice_head(data_01, n = 5)
```

A spec_tbl_df: 5 x 22

fecha <date>	indicativo <chr>	nombre <chr>	provincia <chr>	altitud <dbl>	tmed <dbl>
2022-01-01	2462	PUERTO DE NAVACERRADA	MADRID	1894	10.6
2022-01-02	2462	PUERTO DE NAVACERRADA	MADRID	1894	9.8
2022-01-03	2462	PUERTO DE NAVACERRADA	MADRID	1894	7.8
2022-01-04	2462	PUERTO DE NAVACERRADA	MADRID	1894	1.6
2022-01-05	2462	PUERTO DE NAVACERRADA	MADRID	1894	-3.0

2. Datos de contornos secciones

OPCION A: Seleccionar fichero en ventana para mayor comodidad

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

[9]:
```
# file_data_02 <- tcltk::tk_choose.files(multi = FALSE)
```

OPCION B: Especificar el nombre de archivo

[10]:
```
iFile_02 <- "CU_34_05_01_secciones_geo.json"
file_data_02 <- paste0(iPath, iFile_02)

if(file.exists(file_data_02)){
    cat("Se leerán datos del archivo: ", file_data_02)
} else{
    warning("Cuidado: el archivo no existe.")
}
```

```
Se leerán datos del archivo:  Data/Input/CU_34_05_01_secciones_geo.json
```

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

[11]:
```
data_02 <- st_read(file_data_02)
```

```
Reading layer `CU_34_05_01_secciones_geo' 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/CU_34_05_01_secciones_geo.json'
  using driver `GeoJSON'
Simple feature collection with 4432 features and 16 fields
Geometry type: MULTIPOLYGON
Dimension:     XY
Bounding box:  xmin: -4.579006 ymin: 39.8848 xmax: -3.052983 ymax: 41.16584
Geodetic CRS:  WGS 84
```

Estructura de los datos:

[12]:
```
glimpse(data_02)
```

```
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 (((…
```

Muestra de datos:

[13]:
```
data_02 |> 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

0.3 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:

- Datos diarios meteorológicos por estación
- Contornos de secciones censales

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 transformation

- Convertir zonas a centroide para poder interpolar

[14]:
```
tdata_02 <- data_02 |> st_centroid()
```

```
Warning message in st_centroid.sf(data_02):
"st_centroid assumes attributes are constant over geometries of x"
```

[15]:
```
glimpse(tdata_02)
```

```
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 <POINT [°]> POINT (-3.63571 41.09315), POINT
(-3.475764 40.5202), P…
```

[16]:
```
tdata_02 |> tibble() |> slice_head(n = 5)
```

A tibble: 5 x 17

CUSEC <chr>	CUMUN <chr>	CSEC <chr>	CDIS <chr>	CMUN <chr>	CPRO <chr>	CCA <chr>	CUDIS <chr>	CLAU2 <chr>	NPI <ch
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

Data extract

- Seleccionar variables temperatura, precipitaciones y presión

[17]:
```
edata_01 <- data_01 |>
   select(fecha, nombre, tmed, prec, velmedia, presMax, X, Y)
```

Data transformation

- Convertir datos a objeto espacial para poder interpolar

[18]:
```
tdata_01 <- edata_01 |>
    st_as_sf(coords = c("X", "Y"), crs = 4326)
```

[19]:
```
glimpse(tdata_01)
```

```
Rows: 372
Columns: 7
$ fecha    <date> 2022-01-01, 2022-01-02, 2022-01-03,
2022-01-04, 2022-01-05, …
```

```
$ nombre   <chr> "PUERTO DE NAVACERRADA", "PUERTO DE
NAVACERRADA", "PUERTO DE …
$ tmed     <dbl> 10.6, 9.8, 7.8, 1.6, -3.0, -5.6, -4.0, -2.8,
0.0, 1.6, 1.0, 2…
$ prec     <dbl> 0.0, 0.0, 0.0, 24.2, 13.9, 0.0, 0.2, 0.0,
6.6, 0.8, 0.0, 0.0,…
$ velmedia <dbl> 3.9, 2.5, 3.1, 7.8, 4.2, 5.6, 4.2, 5.6, 6.1,
6.1, 2.5, 1.7, 5…
$ presMax  <dbl> 823.3, 823.5, 821.5, 814.7, 809.5, 813.5,
819.3, 819.2, 815.1…
$ geometry <POINT [°]> POINT (-4.010556 40.79306), POINT
(-4.010556 40.79306),…
```

[20]:
```
tdata_01 |> tibble() |> slice_head(n = 5)
```

A tibble: 5 x 7

fecha <date>	nombre <chr>	tmed <dbl>	prec <dbl>	velmedia <dbl>	presMax <dbl>	geometry <POINT
2022-01-01	PUERTO DE NAVACERRADA	10.6	0.0	3.9	823.3	POINT
2022-01-02	PUERTO DE NAVACERRADA	9.8	0.0	2.5	823.5	POINT
2022-01-03	PUERTO DE NAVACERRADA	7.8	0.0	3.1	821.5	POINT
2022-01-04	PUERTO DE NAVACERRADA	1.6	24.2	7.8	814.7	POINT
2022-01-05	PUERTO DE NAVACERRADA	-3.0	13.9	4.2	809.5	POINT

- Interpolar temperatura media, precipitación, velocidad media del viento y presión máxima en las coordenadas de cada sección censal.

NOTA: esta operación puede tardar varios minutos.

[21]:
```
meteovars <- c("tmed", "prec", "velmedia", "presMax")
dias <- tdata_01$fecha |> unique()
resl <- list()

for (i in seq_along(dias)){
  f <- dias[i]
  meteod <- tdata_01 |>
    filter(fecha == f) |>
    select(all_of(meteovars))

  res <- data_02 |> select(CMUN, CDIS, CSEC) |> st_drop_geometry()
  for (j in seq_along(meteovars)){
    thisvar <- meteovars[j]
    meteodc <- meteod |>
      drop_na(all_of(thisvar))

    if(all(meteodc |> pull(thisvar) == 0)){
      res <- res |>
        mutate(!!thisvar := 0)
    } else{
```

```
      fo <- as.formula(paste0(thisvar, " ~ 1"))
      v0 <- variogram(fo, meteodc)
      v.m <- suppressWarnings(fit.variogram(v0,
        vgm(c("Exp", "Mat", "Sph", "Log", "Wav", "Bes", "Lin", "Leg"))))
      invisible(capture.output(suppressWarnings(krg <- krige(fo,
                  locations = meteodc,
                  newdata = tdata_02,
                  model = v.m))))
      res <- res |>
        mutate(fecha = f) |>
        mutate(!!thisvar := krg$var1.pred)
    }
  }
    # res <- res |>
    #   mutate(ano = a,
    #          semana = s)
  resl[[i]] <- res
}

data <- bind_rows(resl)
```

[22]:
```
glimpse(data)
```

```
Rows: 137,392
Columns: 8
$ CMUN     <chr> "001", "002", "002", "003", "004", "004",
"004", "004", "005"…
$ CDIS     <chr> "01", "01", "01", "01", "01", "01", "01",
"01", "01", "01", "…
$ CSEC     <chr> "001", "001", "002", "001", "001", "002",
"003", "004", "001"…
$ fecha    <date> 2022-01-01, 2022-01-01, 2022-01-01,
2022-01-01, 2022-01-01, …
$ tmed     <dbl> 9.472965, 8.170448, 8.526402, 10.012282,
10.574368, 10.508450…
$ prec     <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0…
$ velmedia <dbl> 0.8229741, 0.9750191, 0.8874318, 2.3162929,
0.6970652, 0.6875…
$ presMax  <dbl> 901.6717, 970.1249, 967.2948, 849.5611,
964.2144, 965.4248, 9…
```

0.4 Synthetic Data Generation

No aplica

0.5 Fake Data Generation

No aplica

0.6 Open Data

Los datos se obtuvieron de fuentes abiertas en la tarea anterior

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

[23]:
```
data_to_save <- data
```

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

[24]:
```
caso <- "CU_34"
proceso <- '_05'
tarea <- "_04"
archivo <- ""
proper <- "_meteo_secciones"
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

[25]:
```
# 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/.

[26]:
```
file_save <- paste0(caso, proceso, tarea, archivo, proper, extension)
path_out <- paste0(oPath, file_save)
write_csv(data_to_save, path_out)

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

```
File saved as:
```

'Data/Output/CU_34_05_04_meteo_secciones.csv'

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

[27]:
```
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, readr, dplyr, tidyr, gstat, stringr y lubridate 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 * readr 2.1.3 * dplyr 1.0.10 * tidyr 1.3.0 * gstat 2.1.0 * stringr 1.5.0 * lubridate 1.9.1

0.8.3 Data structures

Objeto `data`

- Hay 137392 filas
  - CMUN

  - CDIS
  - CSEC
  - tmed
  - prec
  - velmedia
  - presMax
  - fecha

Observaciones generales sobre los datos

- Se incluyen cuatro variables meteorológicas

0.8.4 Consideraciones para despliegue en piloto

- Ninguna

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 han calculado los centroides de las zonas
- Se han interpolado los datos de meteorología a las zonas

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

- Se debe comprobar por qué se crea una seman 2023-52

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

[28]:
```
# incluir código
```