05. - Data Collection_CU_04_14_contaminacion_v_01

June 8, 2023

#

CUxx_xxxxx

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 14. Obtener datos de contaminación

- Importar datos de contaminiación de las estaciones (datos horarios)
- Resumir por día

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

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
- {tidyr} para organización de datos
- {lubridate} para manipulación de fecha
- {saqgter} para obtener datos de contaminación
- {mapSpain} para obtener el contorno de la CM

[1]:
```
library(sf)
library(readr)
library(dplyr)
library(stringr)
library(tidyr)
library(lubridate)
library(saqgetr)
library(mapSpain)

p <- c("tcltk", "sf", "readr", "dplyr", "tidyr", "lubridate", "saqgetr",␣
  ↪"mapSpain")
```

```
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



Attaching package: 'lubridate'


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

    date, intersect, setdiff, union
```

0.1.2 Paths

[2]:
```
iPath <- "Data/Input/"
oPath <- "Data/Output/"
```

0.2 Data Load

No aplica

0.3 Open Data

- Obtener contornos de la comunidad de Madrid

[4]:
```
data_01 <- esp_get_ccaa(13, epsg = 4326)
```

- Obtener estaciones de medición de contaminación de España, sin coordenadas nulas y transformados a objetos sf

[5]:
```
data_02 <- get_saq_sites() |>
  filter(country_iso_code == "ES") |>
  drop_na(longitude, latitude) |>
  st_as_sf(coords = c("longitude", "latitude"),
           crs = 4326)
```

0.4 ETL Processes

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

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

- Filtrar sites que están dentro del contorno de la CM y tienen datos necesarios para el caso (septiembre 2021)
- Extraer solo columnas relevantes

[38]:
```
edata_02 <- st_join(data_02, cm, left = FALSE) |>
  filter(date_end >= "2021-09-01") |>
  select(site, site_name)
```

[40]:
```
edata_02 |> glimpse()
```

```
Rows: 48
Columns: 3
$ site      <chr> "es0115a", "es0118a", "es0120a", "es0124a",
"es0125a", "es01~
$ site_name <chr> "PLAZA DE ESPA<U+00D1>A", "ESCUELAS
AGUIRRE", "RAM<U+00D3>N Y CAJAL", "ART~
```

```
$ geometry  <POINT [arc_degree]> POINT (-3.712222 40.42417),
POINT (-3.682222 ~
```

[41]:
```
edata_02 |> tibble() |> slice_head(n = 5)
```

A tibble: 5 x 3

site	site_name	geometry
<chr>	<chr>	<POINT [arc_degree]>
es0115a	PLAZA DE ESPA<U+00D1>A	POINT (-3.712222 40.42417)
es0118a	ESCUELAS AGUIRRE	POINT (-3.682222 40.42167)
es0120a	RAM<U+00D3>N Y CAJAL	POINT (-3.677222 40.45167)
es0124a	ARTURO SORIA	POINT (-3.639167 40.44)
es0125a	VILLAVERDE	POINT (-3.705 40.34694)

0.5 Open Data

- Obtener datos de las estaciones

[42]:
```
data <- get_saq_observations(site = tdata_02 |> pull(site),
                             start = "2021-09-01")
```

[34]:
```
data |> glimpse()
```

```
Rows: 3,118,601
Columns: 9
$ date     <dttm> 2021-09-01 00:00:00, 2021-09-01 01:00:00,
2021-09-01 02:00:0~
$ date_end <dttm> 2021-09-01 01:00:00, 2021-09-01 02:00:00,
2021-09-01 03:00:0~
$ site     <chr> "es0115a", "es0115a", "es0115a", "es0115a",
"es0115a", "es011~
$ variable <chr> "so2", "so2", "so2", "so2", "so2", "so2",
"so2", "so2", "so2"~
$ process  <int> 305559, 305559, 305559, 305559, 305559,
305559, 305559, 30555~
$ summary  <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1~
$ validity <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1~
$ unit     <chr> "ug.m-3", "ug.m-3", "ug.m-3", "ug.m-3",
"ug.m-3", "ug.m-3", "~
$ value    <dbl> 10, 10, 10, 11, 11, 11, 11, 11, 12, 11, 10,
10, 9, 9, 9, 10, ~
```

Verificación unidades (una variable no debería tener más de una unidad porque después vamos a agregar)

[37]:
```
data |> count(variable, unit)
```

variable	unit	n
<chr>	<chr>	<int>
benzene	ug.m-3	118776
co	mg.m-3	115890
no	ug.m-3	566414
no2	ug.m-3	566415
nox	ug.m-3	566405
o3	ug.m-3	439971
pm10	ug.m-3	381684
pm2.5	ug.m-3	247298
so2	ug.m-3	115748

A tibble: 9 x 3

Data transform

- Agrupar por semana y extender datos en columnas

[19]:
```
tdata <-  data |>
  saq_clean_observations(summary = "hour",
                         spread = TRUE) |>
  mutate(semana = isoweek(date),
            ano = year(date)) |>
  select(-date, -date_end) |>
  group_by(ano, semana, site) |>
  summarise(across(everything(), ~mean(.x, na.rm = TRUE))) |>
  ungroup()
```

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

[20]:
```
tdata |> glimpse()
```

```
Rows: 3,517
Columns: 12
$ ano     <dbl> 2021, 2021, 2021, 2021, 2021, 2021, 2021,
2021, 2021, 2021, 20~
$ semana  <dbl> 35, 35, 35, 35, 35, 35, 35, 35, 35, 35, 35,
35, 35, 35, 35, 35~
$ site    <chr> "es0115a", "es0118a", "es0120a", "es0124a",
"es0125a", "es0126~
$ benzene <dbl> NaN, 0.2134454, 0.2889831, NaN, NaN,
0.1726496, 0.1705882, NaN~
$ co      <dbl> 0.2750000, 0.2436975, NaN, NaN, NaN, NaN,
NaN, 0.3117647, NaN,~
$ no      <dbl> 8.250000, 6.050420, 5.816667, 3.375000,
10.629630, 5.428571, 2~
$ no2     <dbl> 20.541667, 36.630252, 35.516667, 25.866667,
43.638889, 30.0672~
$ nox     <dbl> 33.175000, 45.966387, 44.533333, 31.050000,
```

```
59.935185, 38.4453~
$ o3      <dbl> NaN, 58.64034, NaN, 67.25833, 49.82202,
51.81076, 54.88100, 61~
$ pm10    <dbl> NaN, 24.983193, NaN, NaN, NaN, 17.857143,
14.750000, NaN, 14.5~
$ pm2.5   <dbl> NaN, 20.369748, NaN, NaN, NaN, NaN, 8.983333,
NaN, NaN, NaN, 1~
$ so2     <dbl> 10.116667, 6.075630, NaN, NaN, NaN, NaN, NaN,
7.983193, 5.7250~
```

[21]:
```
tdata |> tibble() |> slice_head(n = 10)
```

A tibble: 10 x 12

ano <dbl>	semana <dbl>	site <chr>	benzene <dbl>	co <dbl>	no <dbl>	no2 <dbl>	nox <dbl>	o3 <dbl>
2021	35	es0115a	NaN	0.2750000	8.250000	20.54167	33.17500	NaN
2021	35	es0118a	0.2134454	0.2436975	6.050420	36.63025	45.96639	58.640
2021	35	es0120a	0.2889831	NaN	5.816667	35.51667	44.53333	NaN
2021	35	es0124a	NaN	NaN	3.375000	25.86667	31.05000	67.258
2021	35	es0125a	NaN	NaN	10.629630	43.63889	59.93519	49.822
2021	35	es0126a	0.1726496	NaN	5.428571	30.06723	38.44538	51.810
2021	35	es1193a	0.1705882	NaN	2.858333	19.26667	23.70833	54.881
2021	35	es1422a	NaN	0.3117647	5.638655	30.89916	39.47059	61.859
2021	35	es1426a	NaN	NaN	3.625000	30.29167	35.83333	NaN
2021	35	es1521a	NaN	NaN	8.508333	32.55000	45.52500	58.436

- Quitar geometría a estaciones y poner coordenadas como columnas numéricas

[45]:
```
tdata_02 <- edata_02 |>
    bind_cols(st_coordinates(edata_02)) |>
    st_drop_geometry()
```

[47]:
```
tdata_02 |> slice_head(n = 5)
```

A tibble: 5 x 4

site <chr>	site_name <chr>	X <dbl>	Y <dbl>
es0115a	PLAZA DE ESPA<U+00D1>A	-3.712222	40.42417
es0118a	ESCUELAS AGUIRRE	-3.682222	40.42167
es0120a	RAM<U+00D3>N Y CAJAL	-3.677222	40.45167
es0124a	ARTURO SORIA	-3.639167	40.44000
es0125a	VILLAVERDE	-3.705000	40.34694

- Agregar coordenadas y nombre de la estación

[48]:
```
tdata_out <- tdata |>
    inner_join(tdata_02)
```

```
Joining, by = "site"
```

[49]:
```
tdata_out |> glimpse()
```

```
Rows: 3,517
Columns: 15
$ ano       <dbl> 2021, 2021, 2021, 2021, 2021, 2021, 2021,
2021, 2021, 2021, ~
$ semana    <dbl> 35, 35, 35, 35, 35, 35, 35, 35, 35, 35, 35,
35, 35, 35, 35, ~
$ site      <chr> "es0115a", "es0118a", "es0120a", "es0124a",
"es0125a", "es01~
$ benzene   <dbl> NaN, 0.2134454, 0.2889831, NaN, NaN,
0.1726496, 0.1705882, N~
$ co        <dbl> 0.2750000, 0.2436975, NaN, NaN, NaN, NaN,
NaN, 0.3117647, Na~
$ no        <dbl> 8.250000, 6.050420, 5.816667, 3.375000,
10.629630, 5.428571,~
$ no2       <dbl> 20.541667, 36.630252, 35.516667, 25.866667,
43.638889, 30.06~
$ nox       <dbl> 33.175000, 45.966387, 44.533333, 31.050000,
59.935185, 38.44~
$ o3        <dbl> NaN, 58.64034, NaN, 67.25833, 49.82202,
51.81076, 54.88100, ~
$ pm10      <dbl> NaN, 24.983193, NaN, NaN, NaN, 17.857143,
14.750000, NaN, 14~
$ pm2.5     <dbl> NaN, 20.369748, NaN, NaN, NaN, NaN,
8.983333, NaN, NaN, NaN,~
$ so2       <dbl> 10.116667, 6.075630, NaN, NaN, NaN, NaN,
NaN, 7.983193, 5.72~
$ site_name <chr> "PLAZA DE ESPA<U+00D1>A", "ESCUELAS
AGUIRRE", "RAM<U+00D3>N Y CAJAL", "ART~
$ X         <dbl> -3.712222, -3.682222, -3.677222, -3.639167,
-3.705000, -3.73~
$ Y         <dbl> 40.42417, 40.42167, 40.45167, 40.44000,
40.34694, 40.39472, ~
```

[50]:
```
tdata_out |> slice_head(n = 5)
```

A tibble: 5 x 15

ano <dbl>	semana <dbl>	site <chr>	benzene <dbl>	co <dbl>	no <dbl>	no2 <dbl>	nox <dbl>	o3 <dbl>
2021	35	es0115a	NaN	0.2750000	8.250000	20.54167	33.17500	NaN
2021	35	es0118a	0.2134454	0.2436975	6.050420	36.63025	45.96639	58.6403
2021	35	es0120a	0.2889831	NaN	5.816667	35.51667	44.53333	NaN
2021	35	es0124a	NaN	NaN	3.375000	25.86667	31.05000	67.2583
2021	35	es0125a	NaN	NaN	10.629630	43.63889	59.93519	49.8220

0.6 Synthetic Data Generation

No aplica

0.7 Fake Data Generation

No aplica

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

[51]:
```
data_to_save <- tdata_out
```

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

[52]:
```
caso <- "CU_04"
proceso <- '_05'
tarea <- "_14"
archivo <- ""
proper <- "_contaminacion"
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

[ ]:
```
# 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/.

[53]:
```
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_04_05_14_contaminacion.csv'

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

[54]:
```
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 tcltk, sf, readr, dplyr, tidyr, lubridate, saqgetr, mapSpain 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 * readr 2.1.3 * dplyr 1.0.10 * tidyr 1.3.0 * lubridate 1.9.1 * saqgetr 0.2.21 * mapSpain 0.7.0

0.9.3 Data structures

Objeto `data`

- Los datos de origen se han obtenido de distintas bases de datos que maneja el paquete `saqgetr`
- Hay 3517 filas con las sigientes variables:
  - ano
  - semana
  - site
  - benzene

  - co
  - no
  - no2
  - nox
  - o3
  - pm10
  - pm2.5
  - so2

Observaciones generales sobre los datos

- No todas las estaciones tienen información de todas las variables
- Sin embargo, todas tienen de no, no2 y nox
- nox es una medida conjunta de no y no2

0.9.4 Consideraciones para despliegue en piloto

- Como es un periodo estático el del caso de uso, no hay que actualizar nada. Si hubiera datos en tiempo real de años anteriores, se podría también automatizar esta importación.

0.9.5 Consideraciones para despliegue en producción

- Se deben elegir las fechas de acuerdo al momento en que se ejecute el caso de uso

0.10 Main Actions

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

- Se han comprobado las unidades de las variables
- Se han filtrado las estaciones de la CM
- Se han agrupado los datos por semana

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

- Se debe interpolar el dato de nox a las zonas sanitarias

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
```