05. - Data Collection_CU_18_18_infra_estaciones_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 18. Agrupar estaciones de metro

- Los datos de metro están geolocalizados por "boca" de metro
- La capacidad es previsible que esté por estación (o como mucho estación-línea)
- Así que hay que agrupar, en esta tarea se hace por estación.

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
- {stringr} para manipulación de cadenas de caracteres
- {tidyr} para organización de datos

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

p <- c("tcltk", "sf", "readr", "dplyr", "stringr", "tidyr")
```

```
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
```

0.1.2 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 <- "CU_18_05_17_infraestructuras.csv"
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/CU_18_05_17_infraestructuras.csv
```

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

[6]:
```
data <- read_csv(file_data)
```

```
Rows: 1633 Columns: 18
-- Column specification --------------------------------------------------------
Delimiter: ","
chr  (8): grupo, tipo, nombre, CODMUN, DIRECCION, info, CMUN, CDIS
dbl (10): dist_aeropuerto, dist_intercambiador, dist_cercanias,
dist_hospita…

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

Estructura de los datos:

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

```
Rows: 1,633
Columns: 18
$ grupo               <chr> "Transporte", "Transporte",
"Transporte", "Transpo~
$ tipo                <chr> "Intercambiadores",
"Intercambiadores", "Intercamb~
$ nombre              <chr> "Grandes Intercambiadores Plaza
El<U+00ED>ptica", "Grande~
$ CODMUN              <chr> "079", "079", "079", "079",
"079", "079", "079", "~
$ DIRECCION           <chr> "Plaza El<U+00ED>ptica s/n",
"Calle Princesa, 89", "Estac~
$ info                <chr> NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA~
$ CMUN                <chr> "079", "079", "079", "079",
```

```
"079", "079", "079", "~
$ CDIS                <chr> "12", "09", "09", "05", "05",
"05", "06", "02", "0~
$ dist_aeropuerto     <dbl> 5.2031430, 8.5507475, 7.3002463,
9.9339069, 9.1988~
$ dist_intercambiador <dbl> 0.00000000, 0.00000000,
0.00000000, 0.00000000, 0.~
$ dist_cercanias      <dbl> 1.648973148, 1.468638183,
0.088338692, 0.736877403~
$ dist_hospital       <dbl> 1.8154156, 0.4221115, 1.3855767,
1.0733484, 0.3488~
$ dist_universidad    <dbl> 1.29960755, 0.30460781,
0.92076145, 0.08835165, 0.~
$ dist_gob            <dbl> 0.61772353, 0.62638474,
1.17005116, 0.14904962, 1.~
$ dist_embajada       <dbl> 3.921324e-01, 1.457397e+00,
5.624560e+00, 8.316773~
$ dist_ccomercial     <dbl> 0.73115706, 1.61056736,
0.90608573, 0.77583440, 0.~
$ X                   <dbl> -3.716577, -3.719508, -3.719560,
-3.689044, -3.676~
$ Y                   <dbl> 40.38540, 40.43474, 40.42080,
40.46719, 40.43797, ~
```

Muestra de datos:

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

A spec_tbl_df: 5 x 18

grupo <chr>	tipo <chr>	nombre <chr>
Transporte	Intercambiadores	Grandes Intercambiadores Plaza El<U+00ED>ptica
Transporte	Intercambiadores	Grandes Intercambiadores Moncloa
Transporte	Intercambiadores	Grandes Intercambiadores Pr<U+00ED>ncipe P<U+00
Transporte	Intercambiadores	Grandes Intercambiadores Plaza de Castilla
Transporte	Intercambiadores	Grandes Intercambiadores Avenida de Am<U+00E9>ric

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:

- Infraestructuras completas con distancias

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

0.4 Data transform

- Conversión de coordenadas numéricas a objeto espacial

[9]:
```
tdata_sf <- data |>
    st_as_sf(coords = c("X", "Y"))
```

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

```
Rows: 1,633
Columns: 17
$ grupo               <chr> "Transporte", "Transporte",
"Transporte", "Transpo~
$ tipo                <chr> "Intercambiadores",
"Intercambiadores", "Intercamb~
$ nombre              <chr> "Grandes Intercambiadores Plaza
El<U+00ED>ptica", "Grande~
$ CODMUN              <chr> "079", "079", "079", "079",
"079", "079", "079", "~
$ DIRECCION           <chr> "Plaza El<U+00ED>ptica s/n",
"Calle Princesa, 89", "Estac~
$ info                <chr> NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA~
$ CMUN                <chr> "079", "079", "079", "079",
"079", "079", "079", "~
$ CDIS                <chr> "12", "09", "09", "05", "05",
"05", "06", "02", "0~
$ dist_aeropuerto     <dbl> 5.2031430, 8.5507475, 7.3002463,
9.9339069, 9.1988~
$ dist_intercambiador <dbl> 0.00000000, 0.00000000,
0.00000000, 0.00000000, 0.~
$ dist_cercanias      <dbl> 1.648973148, 1.468638183,
0.088338692, 0.736877403~
$ dist_hospital       <dbl> 1.8154156, 0.4221115, 1.3855767,
1.0733484, 0.3488~
$ dist_universidad    <dbl> 1.29960755, 0.30460781,
0.92076145, 0.08835165, 0.~
$ dist_gob            <dbl> 0.61772353, 0.62638474,
1.17005116, 0.14904962, 1.~
$ dist_embajada       <dbl> 3.921324e-01, 1.457397e+00,
5.624560e+00, 8.316773~
$ dist_ccomercial     <dbl> 0.73115706, 1.61056736,
0.90608573, 0.77583440, 0.~
$ geometry            <POINT> POINT (-3.716577 40.3854),
POINT (-3.719508 40.4~
```

[11]:
```
tdata_sf |> slice_head(n = 5) |> tibble()
```

	grupo <chr>	tipo <chr>	nombre <chr>
A tibble: 5 x 17	Transporte	Intercambiadores	Grandes Intercambiadores Plaza El<U+00ED>ptica
	Transporte	Intercambiadores	Grandes Intercambiadores Moncloa
	Transporte	Intercambiadores	Grandes Intercambiadores Pr<U+00ED>ncipe P<U+00ED>o
	Transporte	Intercambiadores	Grandes Intercambiadores Plaza de Castilla
	Transporte	Intercambiadores	Grandes Intercambiadores Avenida de Am<U+00E9>rica

Data extract

- Extraer solo tipo bocas de metro

[12]:
```
edata <- tdata_sf |>
    filter(tipo == "Bocas de metro")
```

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

```
Rows: 771
Columns: 17
$ grupo               <chr> "Transporte", "Transporte",
"Transporte", "Transpo~
$ tipo                <chr> "Bocas de metro", "Bocas de
metro", "Bocas de metr~
$ nombre              <chr> "Plaza de Castilla - Castellana",
"Plaza de Castil~
$ CODMUN              <chr> "079", "079", "079", "079",
"079", "079", "079", "~
$ DIRECCION           <chr> "Frente dep<U+00F3>sito C. Isabel
II", "Paseo de la Caste~
$ info                <chr> "L<U+00ED>neas: 1, 10, 9",
"L<U+00ED>neas: 1, 10, 9", "L<U+00ED>neas: 1~
$ CMUN                <chr> "079", "079", "079", "079",
"079", "079", "079", "~
$ CDIS                <chr> "05", "06", "06", "06", "06",
"06", "06", "06", "0~
$ dist_aeropuerto     <dbl> 9.905774, 9.942879, 9.974996,
10.482757, 10.338293~
$ dist_intercambiador <dbl> 0.190340745, 0.041478689,
0.041290988, 0.666464368~
$ dist_cercanias      <dbl> 0.8811005, 0.7749713, 0.7651209,
1.3860106, 1.2245~
$ dist_hospital       <dbl> 1.0921155, 1.0997205, 1.0512395,
1.2298147, 1.1677~
$ dist_universidad    <dbl> 1.11680447, 1.31909995,
1.28279115, 0.55139515, 0.~
$ dist_gob            <dbl> 0.7804341, 0.6398650, 0.6801415,
0.1105947, 1.1016~
$ dist_embajada       <dbl> 0.40036461, 0.44549955,
0.45305634, 0.05910184, 0.~
```

```
$ dist_ccomercial     <dbl> 0.8043085, 0.7790027, 0.8031386,
0.2866241, 0.4305~
$ geometry            <POINT> POINT (-3.688866 40.46549),
POINT (-3.689186 40.~
```

Data transform

- Obtener estación de metro
  - En el nombre, está estación - salida

[14]:
```
tdata_estacion <- edata |>
  mutate(tipo = "Estaciones de metro") |>
  separate(nombre, c("nombre", "salida"), sep = " - ")
```

```
Warning message:
"Expected 2 pieces. Additional pieces discarded in 41 rows [75, 82,
133, 159,
160, 189, 190, 304, 305, 312, 313, 359, 367, 368, 370, 371, 372, 373, 374, 375,
…]."
```

[15]:
```
tdata_estacion |> glimpse()
```

```
Rows: 771
Columns: 18
$ grupo               <chr> "Transporte", "Transporte",
"Transporte", "Transpo~
$ tipo                <chr> "Estaciones de metro",
"Estaciones de metro", "Est~
$ nombre              <chr> "Plaza de Castilla", "Plaza de
Castilla", "Plaza d~
$ salida              <chr> "Castellana", "Plaza de
Castilla", "P<U+00BA> Castellana,~
$ CODMUN              <chr> "079", "079", "079", "079",
"079", "079", "079", "~
$ DIRECCION           <chr> "Frente dep<U+00F3>sito C. Isabel
II", "Paseo de la Caste~
$ info                <chr> "L<U+00ED>neas: 1, 10, 9",
"L<U+00ED>neas: 1, 10, 9", "L<U+00ED>neas: 1~
$ CMUN                <chr> "079", "079", "079", "079",
"079", "079", "079", "~
$ CDIS                <chr> "05", "06", "06", "06", "06",
"06", "06", "06", "0~
$ dist_aeropuerto     <dbl> 9.905774, 9.942879, 9.974996,
10.482757, 10.338293~
$ dist_intercambiador <dbl> 0.190340745, 0.041478689,
0.041290988, 0.666464368~
$ dist_cercanias      <dbl> 0.8811005, 0.7749713, 0.7651209,
1.3860106, 1.2245~
$ dist_hospital       <dbl> 1.0921155, 1.0997205, 1.0512395,
```

```
1.2298147, 1.1677~
$ dist_universidad    <dbl> 1.11680447, 1.31909995,
1.28279115, 0.55139515, 0.~
$ dist_gob            <dbl> 0.7804341, 0.6398650, 0.6801415,
0.1105947, 1.1016~
$ dist_embajada       <dbl> 0.40036461, 0.44549955,
0.45305634, 0.05910184, 0.~
$ dist_ccomercial     <dbl> 0.8043085, 0.7790027, 0.8031386,
0.2866241, 0.4305~
$ geometry            <POINT> POINT (-3.688866 40.46549),
POINT (-3.689186 40.~
```

- Agrupar datos de bocas de metro en su estación
  - Las distancias nos quedamos con la mínima de todas las bocas
  - El resto, el primer valor (se supone que serían todos iguales)
  - La geometría, el centroide de todas las bocas

[16]:
```
tdata_agrupado <- tdata_estacion |>
  mutate(tipo = "Estaciones de metro") |>
  separate(nombre, c("nombre", "salida"), sep = " - ") |>
  group_by(grupo, tipo, nombre) |>
  summarise(across(CODMUN:CDIS, first),
            across(dist_aeropuerto:dist_ccomercial, min)) |>
  st_centroid()
```

```
Warning message:
"Expected 2 pieces. Missing pieces filled with `NA` in 771 rows [1, 2,
3, 4, 5,
6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, …]."
`summarise()` has grouped output by 'grupo', 'tipo'. You can override
using the
`.groups` argument.
Warning message in
st_centroid.sf(summarise(group_by(separate(mutate(tdata_estacion, :
"st_centroid assumes attributes are constant over geometries of x"
```

- Geometrías a coordenadas numéricas para exportar a csv

[17]:
```
tdata_metro <- tdata_agrupado |>
    bind_cols(st_coordinates(tdata_agrupado)) |>
    st_drop_geometry()
```

- Sustituir datos de metro

[18]:
```
tdata_out <-
    data |>
    filter(tipo != "Bocas de metro") |>
    bind_rows(tdata_metro)
```

[19]:
```
tdata_out |> glimpse()
```

```
Rows: 1,138
Columns: 18
$ grupo               <chr> "Transporte", "Transporte",
"Transporte", "Transpo~
$ tipo                <chr> "Intercambiadores",
"Intercambiadores", "Intercamb~
$ nombre              <chr> "Grandes Intercambiadores Plaza
El<U+00ED>ptica", "Grande~
$ CODMUN              <chr> "079", "079", "079", "079",
"079", "079", "079", "~
$ DIRECCION           <chr> "Plaza El<U+00ED>ptica s/n",
"Calle Princesa, 89", "Estac~
$ info                <chr> NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA~
$ CMUN                <chr> "079", "079", "079", "079",
"079", "079", "079", "~
$ CDIS                <chr> "12", "09", "09", "05", "05",
"05", "06", "02", "0~
$ dist_aeropuerto     <dbl> 5.2031430, 8.5507475, 7.3002463,
9.9339069, 9.1988~
$ dist_intercambiador <dbl> 0.0000000, 0.0000000, 0.0000000,
0.0000000, 0.0000~
$ dist_cercanias      <dbl> 1.648973148, 1.468638183,
0.088338692, 0.736877403~
$ dist_hospital       <dbl> 1.8154156, 0.4221115, 1.3855767,
1.0733484, 0.3488~
$ dist_universidad    <dbl> 1.29960755, 0.30460781,
0.92076145, 0.08835165, 0.~
$ dist_gob            <dbl> 0.61772353, 0.62638474,
1.17005116, 0.14904962, 1.~
$ dist_embajada       <dbl> 3.921324e-01, 1.457397e+00,
5.624560e+00, 8.316773~
$ dist_ccomercial     <dbl> 0.73115706, 1.61056736,
0.90608573, 0.77583440, 0.~
$ X                   <dbl> -3.716577, -3.719508, -3.719560,
-3.689044, -3.676~
$ Y                   <dbl> 40.38540, 40.43474, 40.42080,
40.46719, 40.43797, ~
```

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

A spec_tbl_df: 5 x 18

grupo <chr>	tipo <chr>	nombre <chr>
Transporte	Intercambiadores	Grandes Intercambiadores Plaza El<U+00ED>ptica
Transporte	Intercambiadores	Grandes Intercambiadores Moncloa
Transporte	Intercambiadores	Grandes Intercambiadores Pr<U+00ED>ncipe P<U+00
Transporte	Intercambiadores	Grandes Intercambiadores Plaza de Castilla
Transporte	Intercambiadores	Grandes Intercambiadores Avenida de Am<U+00E9>ric

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

0.5 Synthetic Data Generation

No aplica

0.6 Fake Data Generation

No aplica

0.7 Open Data

Los datos originales fueron descargados de fuentes abiertas

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

[21]:
```
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

[22]:
```
caso <- "CU_18"
proceso <- '_05'
tarea <- "_18"
archivo <- ""
proper <- "_infraestructuras"
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

[23]:
```
# 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/.

[24]:
```
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_18_05_18_infraestructuras.csv'

Copia del fichero a Input Si el archivo se va a usar en otros notebooks, copiar a la carpeta Input (descomentar si hace falta)

[25]:
```
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 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

0.9.3 Data structures

Objeto `data`

- Hay 1138 filas con las variables:
  - grupo
  - tipo
  - nombre
  - CODMUN
  - DIRECCION
  - info
  - CMUN
  - CDIS
  - dist_aeropuerto
  - dist_intercambiador
  - dist_cercanias
  - dist_hospital
  - dist_universidad
  - dist_gob
  - dist_embajada
  - dist_ccomercial
  - X
  - Y

Observaciones generales sobre los datos

- Se ha reducido al número de puntos considerablemente

0.9.4 Consideraciones para despliegue en piloto

- Ninguna

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 calculado los centroides de las estaciones de metro con más de una boca
- Se han calculad las distancias mínimas a puntos relevantes desde todas las bocas de metro

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

- Se deben obtener datos de demanda y capacidad diaria de todas las infraestructuras resultantes

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

[28]:
```
# incluir código
```