05. - Data Collection_CU_18_05_infraestructuras_distrito_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 05. Agrupar datos de infraestructuras por distrito

- A partir de los datos puntuales, agregar por distrito censal y contar infraestructruras para hacer mapas de regiones.
- Adicionalmente crear metadatos con descripción y agrupamiento de variables.

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

- {tcltk} para selección interactiva de archivos locales
- {readr} para leer y escribir archivos csv

- {dplyr} para explorar datos
- {dityr} para tranformar datos
- {janitor} para limpiar datos

[31]:
```
library(readr)
library(dplyr)
library(tidyr)
library(janitor)
```

0.1.2 Paths

[32]:
```
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

[33]:
```
# file_data <- tcltk::tk_choose.files(multi = FALSE)
```

OPCION B: Especificar el nombre de archivo

[34]:
```
iFile <- "CU_18_05_04_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 leerán datos del archivo:  Data/Input/CU_18_05_04_infraestructuras.csv
```

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

[35]:
```
data <- read_csv(file_data)
```

```
Rows: 1633 Columns: 10
  Column specification
```

```
Delimiter: ","
chr (8): grupo, tipo, nombre, CODMUN, DIRECCION, info, CMUN, CDIS
dbl (2): X, Y

 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:

[36]:
```
glimpse(data)
```

```
Rows: 1,633
Columns: 10
$ grupo     <chr> "Transporte", "Transporte", "Transporte",
"Transporte", "Tra…
$ tipo      <chr> "Intercambiadores", "Intercambiadores",
"Intercambiadores", …
$ nombre    <chr> "Grandes Intercambiadores Plaza Elíptica",
"Grandes Intercam…
$ CODMUN    <chr> "079", "079", "079", "079", "079", "079",
"079", "079", "079…
$ DIRECCION <chr> "Plaza Elíptica s/n", "Calle Princesa, 89",
"Estación Prínci…
$ X         <dbl> -3.716577, -3.719508, -3.719560, -3.689044,
-3.676731, -3.68…
$ Y         <dbl> 40.38540, 40.43474, 40.42080, 40.46719,
40.43797, 40.47198, …
$ info      <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, …
$ CMUN      <chr> "079", "079", "079", "079", "079", "079",
"079", "079", "079…
$ CDIS      <chr> "12", "09", "09", "05", "05", "05", "06",
"02", "01", "10", …
```

Muestra de datos:

[37]:
```
slice_head(data, n = 5)
```

A spec_tbl_df: 5 × 10

grupo <chr>	tipo <chr>	nombre <chr>	CODM <chr>
Transporte	Intercambiadores	Grandes Intercambiadores Plaza Elíptica	079
Transporte	Intercambiadores	Grandes Intercambiadores Moncloa	079
Transporte	Intercambiadores	Grandes Intercambiadores Príncipe Pío	079
Transporte	Intercambiadores	Grandes Intercambiadores Plaza de Castilla	079
Transporte	Intercambiadores	Grandes Intercambiadores Avenida de América	079

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

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

Data Transform

- Contar número de infraestructuras de cada tipo por distrito
- Extender en columnas para caracterizar distritos

[38]:
```
tdata_01 <- data |>
  count(CMUN, CDIS, tipo) |>
  pivot_wider(names_from = tipo,
              values_from = n,
              values_fill = 0)
```

[39]:
```
glimpse(tdata_01)
```

```
Rows: 239
Columns: 14
$ CMUN                                      <chr> "001",
"002", "003", "004", …
$ CDIS                                      <chr> "01", "01",
"01", "01", "01"…
$ `Consultorios de Salud`                   <int> 1, 1, 1, 1,
0, 0, 0, 0, 0, 0…
$ Helisuperficies                           <int> 1, 0, 1, 0,
0, 0, 0, 0, 0, 0…
$ `Centros de Atención a Drogodependientes` <int> 0, 0, 0, 0,
1, 0, 0, 0, 0, 1…
$ `Centros de Salud`                        <int> 0, 0, 0, 0,
2, 3, 2, 1, 2, 5…
$ `Estaciones de Cercanías`                 <int> 0, 0, 0, 0,
1, 0, 1, 1, 0, 2…
$ Hospitales                                <int> 0, 0, 0, 0,
1, 0, 0, 1, 0, 0…
$ `Otros Centros de Salud`                  <int> 0, 0, 0, 0,
1, 0, 0, 0, 0, 0…
$ `Centros de Salud Mental`                 <int> 0, 0, 0, 0,
0, 1, 0, 1, 0, 1…
$ `Centros de Especialidades`               <int> 0, 0, 0, 0,
0, 0, 0, 1, 0, 1…
$ `Bocas de metro`                          <int> 0, 0, 0, 0,
0, 0, 0, 0, 0, 9…
```

```
$ Intercambiadores                    <int> 0, 0, 0, 0,
0, 0, 0, 0, 0, 0…
$ Aeropuertos                         <int> 0, 0, 0, 0,
0, 0, 0, 0, 0, 0…
```

[40]:
```
tdata_01 |> slice_head(n = 5)
```

A tibble: 5 × 14

CMUN <chr>	CDIS <chr>	Consultorios de Salud <int>	Helisuperficies <int>	Centros de Atención a Drogodepend <int>
001	01	1	1	0
002	01	1	0	0
003	01	1	1	0
004	01	1	0	0
005	01	0	0	1

Data Extract

- Extraer nombres de columnas

[41]:
```
tdata_02 <- data.frame(desc_var = colnames(tdata_01)[3:ncol(tdata_01)])
```

[42]:
```
tdata_02
```

A data.frame: 12 × 1

desc_var <chr>
Consultorios de Salud
Helisuperficies
Centros de Atención a Drogodependientes
Centros de Salud
Estaciones de Cercanías
Hospitales
Otros Centros de Salud
Centros de Salud Mental
Centros de Especialidades
Bocas de metro
Intercambiadores
Aeropuertos

Data Transform

- Limpiar nombres de columnas

[43]:
```
tdata_01 <- tdata_01 |>
  clean_names()
```

[44]:
```
glimpse(tdata_01)
```

```
Rows: 239
Columns: 14
$ cmun                                <chr> "001", "002",
```

```
"003", "004", "0…
$ cdis                                    <chr> "01", "01",
"01", "01", "01", …
$ consultorios_de_salud                   <int> 1, 1, 1, 1,
0, 0, 0, 0, 0, 0, …
$ helisuperficies                         <int> 1, 0, 1, 0,
0, 0, 0, 0, 0, 0, …
$ centros_de_atencion_a_drogodependientes <int> 0, 0, 0, 0,
1, 0, 0, 0, 0, 1, …
$ centros_de_salud                        <int> 0, 0, 0, 0,
2, 3, 2, 1, 2, 5, …
$ estaciones_de_cercanias                 <int> 0, 0, 0, 0,
1, 0, 1, 1, 0, 2, …
$ hospitales                              <int> 0, 0, 0, 0,
1, 0, 0, 1, 0, 0, …
$ otros_centros_de_salud                  <int> 0, 0, 0, 0,
1, 0, 0, 0, 0, 0, …
$ centros_de_salud_mental                 <int> 0, 0, 0, 0,
0, 1, 0, 1, 0, 1, …
$ centros_de_especialidades               <int> 0, 0, 0, 0,
0, 0, 0, 1, 0, 1, …
$ bocas_de_metro                          <int> 0, 0, 0, 0,
0, 0, 0, 0, 0, 9, …
$ intercambiadores                        <int> 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, …
$ aeropuertos                             <int> 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, …
```

- Completar metadatos

[45]:
```
tdata_02 <- tdata_02 |>
    mutate(nombre_var = colnames(tdata_01)[3:ncol(tdata_01)]) |>
    left_join(data |> count(grupo, tipo),
              by = c("desc_var" = "tipo"))
```

[46]:
```
glimpse(tdata_02)
```

```
Rows: 12
Columns: 4
$ desc_var   <chr> "Consultorios de Salud",
"Helisuperficies", "Centros de Ate…
$ nombre_var <chr> "consultorios_de_salud",
"helisuperficies", "centros_de_ate…
$ grupo      <chr> "Salud", "Transporte", "Salud", "Salud",
"Transporte", "Sal…
$ n          <int> 156, 87, 35, 267, 93, 87, 29, 53, 28, 771,
24, 3
```

[47]:
```
tdata_02
```

A data.frame: 12 × 4

desc_var <chr>	nombre_var <chr>
Consultorios de Salud	consultorios_de_salud
Helisuperficies	helisuperficies
Centros de Atención a Drogodependientes	centros_de_atencion_a_drogodependientes
Centros de Salud	centros_de_salud
Estaciones de Cercanías	estaciones_de_cercanias
Hospitales	hospitales
Otros Centros de Salud	otros_centros_de_salud
Centros de Salud Mental	centros_de_salud_mental
Centros de Especialidades	centros_de_especialidades
Bocas de metro	bocas_de_metro
Intercambiadores	intercambiadores
Aeropuertos	aeropuertos

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

0.4 Synthetic Data Generation

Estos datos no requieren tareas de este tipo.

0.5 Fake Data Generation

Estos datos no requieren tareas de este tipo.

0.6 Open Data

Estos datos no requieren tareas de este tipo.

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

1. Infraestructuras agregadas por distrito

[48]:
```
data_to_save_01 <- tdata_01
```

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

[49]:
```
caso <- "CU_18"
proceso <- '_05'
tarea <- "_05"
archivo <- "_01"
proper <- "_infraestructuras_distrito"
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

[50]:
```
# file_save_01 <- paste0(caso, proceso, tarea, tcltk::tkgetSaveFile(), proper,␣
 ↪extension)
# path_out_01 <- paste0(oPath, file_save_01)
# write_csv(data_to_save_01, path_out_01)

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

OPCION B: Especificar el nombre de archivo

- Los ficheros de salida del proceso van siempre a Data/Output/.

[51]:
```
file_save_01 <- paste0(caso, proceso, tarea, archivo, proper, extension)
path_out_01 <- paste0(oPath, file_save_01)
write_csv(data_to_save, path_out_01)

cat('File saved as: ')
path_out_01
```

```
File saved as:
```

’Data/Output/CU_18_05_05_01_infraestructuras_distrito.csv’

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

[52]:
```
path_in_01 <- paste0(iPath, file_save_01)
file.copy(path_out_01, path_in_01, overwrite = TRUE)
```

TRUE

2. Metadatos de infraestructuras

[53]:
```
data_to_save_02 <- tdata_02
```

[54]:
```
archivo <- "_02"
proper <- "_infraestructuras_meta"
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

[55]:
```
# file_save_02 <- paste0(caso, proceso, tarea, tcltk::tkgetSaveFile(), proper,␣
↪extension)
# path_out_02 <- paste0(oPath, file_save_02)
# write_csv(data_to_save_02, path_out_02)

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

OPCION B: Especificar el nombre de archivo

- Los ficheros de salida del proceso van siempre a Data/Output/.

[56]:
```
file_save_02 <- paste0(caso, proceso, tarea, archivo, proper, extension)
path_out_02 <- paste0(oPath, file_save_02)
write_csv(data_to_save_02, path_out_02)

cat('File saved as: ')
path_out_02
```

```
File saved as:
```

’Data/Output/CU_18_05_05_02_infraestructuras_meta.csv’

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

[57]:
```
path_in_02 <- paste0(iPath, file_save_02)
file.copy(path_out_02, path_in_02, overwrite = TRUE)
```

TRUE

0.8 Main Conclusions

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

0.8.1 Prerequisites

This working code needs the following conditions:

- For using the interactive selection of file, the {tcltk} package must be installed. It is not needed in production.
- The {readr}, {dplyr}, {tidyr} and {janitor} packages must be installed.
- The data paths `Data/Input` and `Data/Output` must exist (relative to the notebook path)

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 * tidyr 1.3.0 * dplyr 1.0.10 * janitor 2.1.0 * readr 2.1.3

0.8.3 Data structures

Objeto `tdata_01`

- Tenemos 239 filas, una por distrito, con los recuentos de cada uno de los tipos de infraestructuras (12)

[58]:
```
glimpse(tdata_01)
```

```
Rows: 239
Columns: 14
$ cmun                                    <chr> "001", "002",
"003", "004", "0…
$ cdis                                    <chr> "01", "01",
"01", "01", "01", …
$ consultorios_de_salud                   <int> 1, 1, 1, 1,
0, 0, 0, 0, 0, 0, …
$ helisuperficies                         <int> 1, 0, 1, 0,
0, 0, 0, 0, 0, 0, …
$ centros_de_atencion_a_drogodependientes <int> 0, 0, 0, 0,
1, 0, 0, 0, 0, 1, …
$ centros_de_salud                        <int> 0, 0, 0, 0,
2, 3, 2, 1, 2, 5, …
$ estaciones_de_cercanias                 <int> 0, 0, 0, 0,
1, 0, 1, 1, 0, 2, …
$ hospitales                              <int> 0, 0, 0, 0,
1, 0, 0, 1, 0, 0, …
$ otros_centros_de_salud                  <int> 0, 0, 0, 0,
1, 0, 0, 0, 0, 0, …
$ centros_de_salud_mental                 <int> 0, 0, 0, 0,
0, 1, 0, 1, 0, 1, …
$ centros_de_especialidades               <int> 0, 0, 0, 0,
0, 0, 0, 1, 0, 1, …
$ bocas_de_metro                          <int> 0, 0, 0, 0,
0, 0, 0, 0, 0, 9, …
```

```
$ intercambiadores                          <int> 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, …
$ aeropuertos                               <int> 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, …
```

0.9 Objeto tdata_02

- Tenemos 12 filas, una por tipo de infraestructura, con sus metadatos

[59]:
```
glimpse(tdata_02)
```

```
Rows: 12
Columns: 4
$ desc_var   <chr> "Consultorios de Salud",
"Helisuperficies", "Centros de Ate…
$ nombre_var <chr> "consultorios_de_salud",
"helisuperficies", "centros_de_ate…
$ grupo      <chr> "Salud", "Transporte", "Salud", "Salud",
"Transporte", "Sal…
$ n          <int> 156, 87, 35, 267, 93, 87, 29, 53, 28, 771,
24, 3
```

Observaciones generales sobre los datos

- No aplica

0.9.1 Consideraciones para despliegue en piloto

- Utilizar los metadatos para etiquetar gráficos y otras salidas.

0.9.2 Consideraciones para despliegue en producción

- No aplica

0.10 Main Actions

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

- Se han agrupado las infraesctructuras por distrito
- Se han calculado el número de infraestructuras por tipo en cada distrito

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

- Se deben unir los datos de indicadores del INE por distrito

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

[60]:
```
# incluir código
```