05. - Data Collection_CU_18_10_indicadores_distritos_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 10. Indicadores por distritos censales

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

- {readr} from *tidyverse* for reading and writing csv files
- {tcltk} for selecting files and paths (if needed)
- {dplyr} for data exploration
- {sf} for spatial objects and files

[29]:
```
library(readr)
library(tcltk)
library(dplyr)
library(sf)
```

0.1.2 Paths

[30]:
```
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. Indicadores por sección censal OPCION A: Seleccionar fichero en ventana para mayor comodidad

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

[31]:
```
# file_data_01 <- tcltk::tk_choose.files(multi = FALSE)
```

OPCION B: Especificar el nombre de archivo

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)

[32]:
```
iFile_01 <- "CU_04_05_03_01_indicadores_secciones.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 leer<U+00E1>n datos del archivo:
Data/Input/CU_04_05_03_01_indicadores_secciones.csv
```

Data file to dataframe

[33]:
```
data_01 <- read_csv(file_data_01)
```

```
Rows: 4417 Columns: 20
-- Column specification --------------------------------------------------------
Delimiter: ","
chr  (3): CMUN, dist, secc
dbl (17): ccaa, CPRO, t1_1, t2_1, t2_2, t3_1, t4_1, t4_2, t4_3, t5_1,
```

```
t6_1, …

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:

[34]:
```
glimpse(data_01)
```

```
Rows: 4,417
Columns: 20
$ ccaa  <dbl> 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13,
13, 13, 13, 13, ~
$ CPRO  <dbl> 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28,
28, 28, 28, 28, ~
$ CMUN  <chr> "001", "002", "002", "003", "004", "004",
"004", "004", "005", "~
$ dist  <chr> "01", "01", "01", "01", "01", "01", "01", "01",
"01", "01", "01"~
$ secc  <chr> "001", "001", "002", "001", "001", "002",
"003", "004", "001", "~
$ t1_1  <dbl> 55, 2358, 2435, 248, 2886, 3376, 2161, 1523,
1648, 1015, 1728, 1~
$ t2_1  <dbl> NA, 0.4724, 0.4830, 0.3952, 0.5135, 0.4793,
0.5016, 0.5279, 0.54~
$ t2_2  <dbl> NA, 0.5276, 0.5170, 0.6048, 0.4865, 0.5207,
0.4984, 0.4721, 0.45~
$ t3_1  <dbl> NA, 40.5161, 38.3339, 47.4597, 44.4099,
40.6810, 38.0088, 42.537~
$ t4_1  <dbl> NA, 0.1425, 0.1959, 0.1371, 0.1611, 0.1739,
0.2230, 0.1753, 0.11~
$ t4_2  <dbl> NA, 0.7443, 0.7002, 0.6573, 0.6067, 0.6727,
0.6395, 0.6448, 0.69~
$ t4_3  <dbl> NA, 0.1132, 0.1039, 0.2056, 0.2322, 0.1534,
0.1374, 0.1799, 0.18~
$ t5_1  <dbl> NA, 0.1569, 0.1544, 0.1129, 0.1639, 0.1440,
0.1749, 0.1326, 0.16~
$ t6_1  <dbl> NA, 0.1968, 0.1951, 0.1411, 0.2128, 0.1730,
0.2138, 0.1589, 0.22~
$ t7_1  <dbl> NA, 0.4016, 0.4244, 0.1542, 0.1470, 0.1596,
0.1668, 0.1584, 0.09~
$ t8_1  <dbl> NA, 0.3971, 0.4224, 0.1449, 0.1409, 0.1506,
0.1602, 0.1545, 0.09~
$ t9_1  <dbl> NA, 0.4377, 0.4438, 0.5280, 0.2602, 0.3234,
0.2859, 0.3057, 0.51~
$ t10_1 <dbl> NA, 0.0912, 0.1226, 0.0932, 0.1814, 0.1478,
```

```
0.1658, 0.1824, 0.10~
$ t11_1 <dbl> NA, 0.6063, 0.5955, 0.5000, 0.4213, 0.5170,
0.4943, 0.4745, 0.52~
$ t12_1 <dbl> NA, 0.6672, 0.6788, 0.5514, 0.5147, 0.6067,
0.5926, 0.5804, 0.58~
```

Muestra de primeros datos:

[35]:
```
slice_head(data_01, n = 5)
```

A spec_tbl_df: 5 x 20

ccaa <dbl>	CPRO <dbl>	CMUN <chr>	dist <chr>	secc <chr>	t1_1 <dbl>	t2_1 <dbl>	t2_2 <dbl>	t3_1 <dbl>	t4_1 <dbl
13	28	001	01	001	55	NA	NA	NA	NA
13	28	002	01	001	2358	0.4724	0.5276	40.5161	0.142
13	28	002	01	002	2435	0.4830	0.5170	38.3339	0.195
13	28	003	01	001	248	0.3952	0.6048	47.4597	0.137
13	28	004	01	001	2886	0.5135	0.4865	44.4099	0.161

2. Distritos para extraer el área OPCION A: Seleccionar fichero en ventana para mayor comodidad

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

[36]:
```
# file_data_02 <- tcltk::tk_choose.files(multi = FALSE)
```

OPCION B: Especificar el nombre de archivo

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) - Comentar si se usa la opción anterior

[37]:
```
iFile_02 <- "CU_18_05_03_distritos_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 leer<U+00E1>n datos del archivo:  Data/Input/CU_18_05_03_distritos_geo.json
```

Data file to dataframe:

[38]:
```
data_02 <- st_read(file_data_02)
```

```
Reading layer `CU_18_05_03_distritos_geo' from data source
  `/Users/emilio.lcano/academico/gh_repos/__transferencia/citizenlab/CitizenLab-
Research-and-Development/casos_urjc/notebooks/dominios_II_y_III/18_infraestructu
ras/Data/Input/CU_18_05_03_distritos_geo.json'
```

```
  using driver `GeoJSON'
Simple feature collection with 247 features and 2 fields
Geometry type: GEOMETRY
Dimension:     XY
Bounding box:  xmin: -4.579006 ymin: 39.8848 xmax: -3.052983 ymax: 41.16584
Geodetic CRS:  WGS 84
```

Estructura de los datos:

[39]:
```
glimpse(data_02)
```

```
Rows: 247
Columns: 3
$ CMUN     <chr> "001", "002", "003", "004", "005", "005",
"005", "005", "005"~
$ CDIS     <chr> "01", "01", "01", "01", "01", "02", "03",
"04", "05", "01", "~
$ geometry <POLYGON [arc_degree]> POLYGON ((-3.64502
41.12129…, POLYGON ((-3~
```

Muestra de datos

[40]:
```
data_02 |> tibble() |> slice_head(n = 5)
```

A tibble: 5 x 3

CMUN	CDIS	geometry
<chr>	<chr>	<POLYGON [arc_degree]>
001	01	POLYGON ((-3.64502 41.12129...
002	01	POLYGON ((-3.503032 40.526,...
003	01	POLYGON ((-3.808664 40.8921...
004	01	POLYGON ((-4.00197 40.25642...
005	01	POLYGON ((-3.361691 40.4762...

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:

- Indicadores del INE por sección censal
- Georreferenciaciónd de distritos

0.3.2 Transform data

Para asegurar la trazabilidad, guardar las transformaciones en un objeto diferente con el prefijo 't'.

Calcular indicadores por distrito

- Un distrito se corresponde con varias secciones censales
- Para calcular los datos agregados, hay que transformar los porcentajes a números absolutos, sumar y después volver a calcular el porcentaje
- Al agregar, si todos los datos son NA queda cero, y hay que volver a asignar NA

[41]:
```
tdata_01 <- data_01 |>
  mutate(across(c(t2_1:t2_2, t4_1:t12_1), ~ .x*t1_1)) |>
  group_by(CMUN, dist) |>
  summarise(nsec = n(),
            t3_1 = weighted.mean(x = t3_1, w = t1_1, na.rm = TRUE),
            t1_1 = sum(t1_1, na.rm = TRUE),
            across(c(t2_1:t2_2, t4_1:t12_1), ~sum(.x, na.rm = TRUE)/sum(t1_1,
 ↪na.rm = TRUE)),
            .groups = "keep") |>
  mutate(across(t3_1:t12_1, ~ if_else(.x == 0, NA_real_, .x))) |>
  arrange(CMUN, dist) |>
  ungroup()
```

[42]:
```
glimpse(tdata_01)
```

```
Rows: 246
Columns: 18
$ CMUN  <chr> "001", "002", "003", "004", "005", "005",
"005", "005", "005", "~
$ dist  <chr> "01", "01", "01", "01", "01", "02", "03", "04",
"05", "01", "01"~
$ nsec  <int> 1, 2, 1, 4, 23, 36, 16, 18, 32, 67, 19, 37, 29,
35, 1, 9, 8, 1, ~
$ t3_1  <dbl> NA, 39.40747, 47.45970, 41.46662, 45.99877,
42.75131, 41.42790, ~
$ t1_1  <dbl> 55, 4793, 248, 9946, 31006, 54049, 28117,
38778, 43620, 116895, ~
$ t2_1  <dbl> NA, 0.4777851, 0.3952000, 0.5015109, 0.5298303,
0.5115035, 0.508~
$ t2_2  <dbl> NA, 0.5222149, 0.6048000, 0.4984891, 0.4701697,
0.4884965, 0.491~
$ t4_1  <dbl> NA, 0.1696289, 0.1371000, 0.1810684, 0.1141978,
0.1468103, 0.169~
$ t4_2  <dbl> NA, 0.7218958, 0.6573000, 0.6420633, 0.6427127,
0.6644351, 0.668~
$ t4_3  <dbl> NA, 0.10847530, 0.20560000, 0.17684664,
0.24307467, 0.18876028, ~
$ t5_1  <dbl> NA, 0.15562992, 0.11290000, 0.15474242,
0.19647849, 0.18272495, ~
$ t6_1  <dbl> NA, 0.1959363, 0.1411000, 0.1912543, 0.2400069,
0.2147111, 0.202~
$ t7_1  <dbl> NA, 0.41318314, 0.15420000, 0.15732451,
0.07438075, 0.05696805, ~
$ t8_1  <dbl> NA, 0.40995322, 0.14490000, 0.15046840,
0.06545227, 0.04598075, ~
$ t9_1  <dbl> NA, 0.4407990, 0.5280000, 0.2942034, 0.3850913,
0.2235184, 0.345~
```

```
$ t10_1 <dbl> NA, 0.10715222, 0.09320000, 0.16675872,
0.14129129, 0.18975353, ~
$ t11_1 <dbl> NA, 0.6008132, 0.5000000, 0.4777910, 0.4565679,
0.4588558, 0.526~
$ t12_1 <dbl> NA, 0.6730932, 0.5514000, 0.5729139, 0.5316057,
0.5641214, 0.610~
```

[43]:
```
tdata_01 |> slice_head(n = 5)
```

A tibble: 5 x 18

CMUN <chr>	dist <chr>	nsec <int>	t3_1 <dbl>	t1_1 <dbl>	t2_1 <dbl>	t2_2 <dbl>	t4_1 <dbl>	t4_2 <dbl>
001	01	1	NA	55	NA	NA	NA	NA
002	01	2	39.40747	4793	0.4777851	0.5222149	0.1696289	0.7218958
003	01	1	47.45970	248	0.3952000	0.6048000	0.1371000	0.6573000
004	01	4	41.46662	9946	0.5015109	0.4984891	0.1810684	0.6420633
005	01	23	45.99877	31006	0.5298303	0.4701697	0.1141978	0.6427127

Comprobar que no se repiten distritos

[44]:
```
length(unique(tdata_01$dist, tdata_01$dist))
nrow(tdata_01)
```

246

246

Cálculo de la superficie y centroide del distrito

[45]:
```
sf_use_s2(FALSE)
tdata_02 <- data_02 |>
    st_centroid()
tdata_02 <- tdata_02 |>
    mutate(area = st_area(data_02)) |>
    bind_cols(st_coordinates(tdata_02)) |>
    st_drop_geometry()
```

```
Warning message in st_centroid.sf(data_02):
"st_centroid assumes attributes are constant over geometries of x"
Warning message in st_centroid.sfc(st_geometry(x), of_largest_polygon =
of_largest_polygon):
"st_centroid does not give correct centroids for longitude/latitude data"
```

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

```
Rows: 247
Columns: 5
$ CMUN <chr> "001", "002", "003", "004", "005", "005", "005",
"005", "005", "0~
$ CDIS <chr> "01", "01", "01", "01", "01", "02", "03", "04",
"05", "01", "01",~
```

```
$ area [m^2] 21877471 [m^2], 19756620 [m^2], 25707163 [m^2], 22023015
[m^2], 1~
$ X    <dbl> -3.635710, -3.480171, -3.849961, -3.989259,
-3.364638, -3.392389,~
$ Y    <dbl> 41.09315, 40.52396, 40.92033, 40.23240,
40.48268, 40.46546, 40.50~
```

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

A data.frame: 5 x 5

CMUN <chr>	CDIS <chr>	area <[m^2]>	X <dbl>	Y <dbl>
001	01	21877471 [m^2]	-3.635710	41.09315
002	01	19756620 [m^2]	-3.480171	40.52396
003	01	25707163 [m^2]	-3.849961	40.92033
004	01	22023015 [m^2]	-3.989259	40.23240
005	01	1766786 [m^2]	-3.364638	40.48268

Hay un distrito con dos regiones, una de ellas con área cero lo que provocaba también valores infinitos en densidad, se elimina

[48]:
```
tdata_01 |> slice(170)
tdata_02 |> filter(CMUN == 115, CDIS == "01")
```

A tibble: 1 x 18

CMUN <chr>	dist <chr>	nsec <int>	t3_1 <dbl>	t1_1 <dbl>	t2_1 <dbl>	t2_2 <dbl>	t4_1 <dbl>	t4_2 <dbl>	t4 <c
115	01	30	41.691	50471	0.5193472	0.4806528	0.167183	0.6541547	0.1

A data.frame: 2 x 5

CMUN <chr>	CDIS <chr>	area <[m^2]>	X <dbl>	Y <dbl>
115	01	0 [m^2]	-3.831702	40.40128
115	01	33864451 [m^2]	-3.819654	40.42554

[49]:
```
tdata_02 <- tdata_02 |>
    filter(as.numeric(area) != 0)
```

Asignación de la superficie a los distritos de los indicadores y cálculo de densidad de población

[50]:
```
tdata_01 <- tdata_01 |>
    inner_join(tdata_02,
               by = c("CMUN" = "CMUN",
               "dist" = "CDIS")) |>
    mutate(densidad_hab_km = as.numeric(t1_1/(area*10^(-6))),
        area_km2 = as.numeric(area*10^(-6))) |>
    select(-area)
```

Estructura de los datos:

[51]:
```
glimpse(tdata_01)
```

```
Rows: 246
Columns: 22
$ CMUN            <chr> "001", "002", "003", "004", "005",
"005", "005", "005"~
$ dist            <chr> "01", "01", "01", "01", "01", "02",
"03", "04", "05", ~
$ nsec            <int> 1, 2, 1, 4, 23, 36, 16, 18, 32, 67,
19, 37, 29, 35, 1,~
$ t3_1            <dbl> NA, 39.40747, 47.45970, 41.46662,
45.99877, 42.75131, ~
$ t1_1            <dbl> 55, 4793, 248, 9946, 31006, 54049,
28117, 38778, 43620~
$ t2_1            <dbl> NA, 0.4777851, 0.3952000, 0.5015109,
0.5298303, 0.5115~
$ t2_2            <dbl> NA, 0.5222149, 0.6048000, 0.4984891,
0.4701697, 0.4884~
$ t4_1            <dbl> NA, 0.1696289, 0.1371000, 0.1810684,
0.1141978, 0.1468~
$ t4_2            <dbl> NA, 0.7218958, 0.6573000, 0.6420633,
0.6427127, 0.6644~
$ t4_3            <dbl> NA, 0.10847530, 0.20560000,
0.17684664, 0.24307467, 0.~
$ t5_1            <dbl> NA, 0.15562992, 0.11290000,
0.15474242, 0.19647849, 0.~
$ t6_1            <dbl> NA, 0.1959363, 0.1411000, 0.1912543,
0.2400069, 0.2147~
$ t7_1            <dbl> NA, 0.41318314, 0.15420000,
0.15732451, 0.07438075, 0.~
$ t8_1            <dbl> NA, 0.40995322, 0.14490000,
0.15046840, 0.06545227, 0.~
$ t9_1            <dbl> NA, 0.4407990, 0.5280000, 0.2942034,
0.3850913, 0.2235~
$ t10_1           <dbl> NA, 0.10715222, 0.09320000,
0.16675872, 0.14129129, 0.~
$ t11_1           <dbl> NA, 0.6008132, 0.5000000, 0.4777910,
0.4565679, 0.4588~
$ t12_1           <dbl> NA, 0.6730932, 0.5514000, 0.5729139,
0.5316057, 0.5641~
$ X               <dbl> -3.635710, -3.480171, -3.849961,
-3.989259, -3.364638,~
$ Y               <dbl> 41.09315, 40.52396, 40.92033,
40.23240, 40.48268, 40.4~
$ densidad_hab_km <dbl> 2.514002, 242.602224, 9.647117,
451.618447, 17549.3832~
$ area_km2        <dbl> 21.877471, 19.756620, 25.707163,
22.023015, 1.766786, ~
```

Muestra primeros datos:

[52]:
```
tdata_01 |>
    slice_head(n = 10)
```

A tibble: 10 x 22

CMUN <chr>	dist <chr>	nsec <int>	t3_1 <dbl>	t1_1 <dbl>	t2_1 <dbl>	t2_2 <dbl>	t4_1 <dbl>	t4_2 <dbl>
001	01	1	NA	55	NA	NA	NA	NA
002	01	2	39.40747	4793	0.4777851	0.5222149	0.1696289	0.7218958
003	01	1	47.45970	248	0.3952000	0.6048000	0.1371000	0.6573000
004	01	4	41.46662	9946	0.5015109	0.4984891	0.1810684	0.6420633
005	01	23	45.99877	31006	0.5298303	0.4701697	0.1141978	0.6427127
005	02	36	42.75131	54049	0.5115035	0.4884965	0.1468103	0.6644351
005	03	16	41.42790	28117	0.5083306	0.4916694	0.1698581	0.6688123
005	04	18	36.86569	38778	0.5056366	0.4943634	0.2129622	0.7023030
005	05	32	45.96147	43620	0.5196190	0.4803810	0.1207394	0.6332726
006	01	67	40.89541	116895	0.5220080	0.4779920	0.1733979	0.6579805

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 la extensión añadida del fichero para diferenciarlas

Identificamos los datos a guardar

[53]:
```
data_to_save <- 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

Archivo de indicadores por distrito

[54]:
```
caso <- "CU_18"
proceso <- '_05'
tarea <- "_10"
archivo <- ""
proper <- "_indicadores_distritos"
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 <- paste0(caso, proceso, tarea, tcltk::tkgetSaveFile(), proper,␣
 ↪extension)
# path_out <- paste0(oPath, file_save_01)
# 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/.

[56]:
```
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_10_indicadores_distritos.csv'

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

[57]:
```
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

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 {sf}, {readr} and {dplyr} 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 * readr 2.1.3 * dplyr 1.0.10

0.8.3 Data structures

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

```
Rows: 246
Columns: 22
$ CMUN            <chr> "001", "002", "003", "004", "005",
"005", "005", "005"~
$ dist            <chr> "01", "01", "01", "01", "01", "02",
"03", "04", "05", ~
$ nsec            <int> 1, 2, 1, 4, 23, 36, 16, 18, 32, 67,
19, 37, 29, 35, 1,~
$ t3_1            <dbl> NA, 39.40747, 47.45970, 41.46662,
45.99877, 42.75131, ~
$ t1_1            <dbl> 55, 4793, 248, 9946, 31006, 54049,
28117, 38778, 43620~
$ t2_1            <dbl> NA, 0.4777851, 0.3952000, 0.5015109,
0.5298303, 0.5115~
$ t2_2            <dbl> NA, 0.5222149, 0.6048000, 0.4984891,
0.4701697, 0.4884~
$ t4_1            <dbl> NA, 0.1696289, 0.1371000, 0.1810684,
0.1141978, 0.1468~
$ t4_2            <dbl> NA, 0.7218958, 0.6573000, 0.6420633,
0.6427127, 0.6644~
$ t4_3            <dbl> NA, 0.10847530, 0.20560000,
0.17684664, 0.24307467, 0.~
$ t5_1            <dbl> NA, 0.15562992, 0.11290000,
0.15474242, 0.19647849, 0.~
$ t6_1            <dbl> NA, 0.1959363, 0.1411000, 0.1912543,
0.2400069, 0.2147~
$ t7_1            <dbl> NA, 0.41318314, 0.15420000,
0.15732451, 0.07438075, 0.~
$ t8_1            <dbl> NA, 0.40995322, 0.14490000,
0.15046840, 0.06545227, 0.~
$ t9_1            <dbl> NA, 0.4407990, 0.5280000, 0.2942034,
```

```
0.3850913, 0.2235~
$ t10_1           <dbl> NA, 0.10715222, 0.09320000,
0.16675872, 0.14129129, 0.~
$ t11_1           <dbl> NA, 0.6008132, 0.5000000, 0.4777910,
0.4565679, 0.4588~
$ t12_1           <dbl> NA, 0.6730932, 0.5514000, 0.5729139,
0.5316057, 0.5641~
$ X               <dbl> -3.635710, -3.480171, -3.849961,
-3.989259, -3.364638,~
$ Y               <dbl> 41.09315, 40.52396, 40.92033,
40.23240, 40.48268, 40.4~
$ densidad_hab_km <dbl> 2.514002, 242.602224, 9.647117,
451.618447, 17549.3832~
$ area_km2        <dbl> 21.877471, 19.756620, 25.707163,
22.023015, 1.766786, ~
```

Objeto `tdata_01` (indicadores por distrito censal)

- Los mismos datos de entrada por sección censal, a los que se le añade la superficie del distrito y su superficie, y se recalculan los indicadores
- Indicadores de los 247 distritos de la Comunidad de Madrid y número de secciones que los componen

Observaciones generales sobre los datos

- Ninguna

0.9 Consideraciones para despliegue en piloto

- Ninguna

0.10 Consideraciones para despliegue en producción

- Se deben crear los procesos ETL en producción necesarios para que los datos de entrada estén actualizados y tengan su correspondencia en las dos fuentes (INE, CM)

0.11 Main Actions

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

- Se han calculado los indicadores por distrito
- Se ha calculado la densidad de población de cada distrito

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

- Se deben unir los datos a los de infraestructuras por distrito para los modelos

0.12 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

[59]:
```
# incluir código
```