05. - Data Collection_CU_04_05_indicadores_zonas_v_01

June 8, 2023

#

CU04_Optimización de vacunas

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. Indicadores por zona de salud

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.

[ ]:
```
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

- {readr} from *tidyverse* for reading and writing csv files
- {tcltk} for selecting files and paths (if needed)
- {dplyr} for data exploration
- {tidyr} for missing data imputation
- {readxl} for reading excel files
- {stringr} for manipulating strings

[2]:
```
library(readr)
library(tcltk)
library(dplyr)
library(tidyr)
```

```
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
```

0.1.3 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. 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

[4]:
```
# 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)

[5]:
```
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

[6]:
```
data_01_indicadores <- 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:

[7]:
```
glimpse(data_01_indicadores)
```

```
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:

[8]:
```
slice_head(data_01_indicadores, 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. Correspondencia secciones censales y zonas sanitarias OPCION A: Seleccionar fichero en ventana para mayor comodidad

Data load using the {tcltk} package

[9]:
```
# 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)

[10]:
```
iFile_02 <- "CU_04_05_04_zonas_secciones.csv"
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_04_05_04_zonas_secciones.csv
```

Data file to dataframe:

[11]:
```
data_02_zonasec <- read_csv(file_data_02)
```

```
Rows: 4194 Columns: 5
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (5): id_zona, nombre_zona, CMUN, dist, secc

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:

[12]:
```
glimpse(data_02_zonasec)
```

```
Rows: 4,194
Columns: 5
$ id_zona     <chr> "001", "001", "001", "001", "001", "001",
"001", "001", "0~
$ nombre_zona <chr> "Abrantes", "Abrantes", "Abrantes",
"Abrantes", "Abrantes"~
$ CMUN        <chr> "079", "079", "079", "079", "079", "079",
"079", "079", "0~
$ dist        <chr> "11", "11", "11", "11", "11", "11", "11",
"11", "11", "11"~
$ secc        <chr> "157", "158", "159", "160", "161", "162",
"163", "165", "1~
```

Muestra de datos:

[13]:
```
slice_head(data_02_zonasec, n = 5)
```

	id_zona <chr>	nombre_zona <chr>	CMUN <chr>	dist <chr>	secc <chr>
A spec_tbl_df: 5 x 5	001	Abrantes	079	11	157
	001	Abrantes	079	11	158
	001	Abrantes	079	11	159
	001	Abrantes	079	11	160
	001	Abrantes	079	11	161

3. Zonas sanitarias 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

[14]:
```
# file_data_03 <- 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

[15]:
```
iFile_03 <- "CU_04_05_02_zonas.csv"
file_data_03 <- paste0(iPath, iFile_03)

if(file.exists(file_data_03)){
    cat("Se leerán datos del archivo: ", file_data_03)
} else{
    warning("Cuidado: el archivo no existe.")
}
```

```
Se leer<U+00E1>n datos del archivo:  Data/Input/CU_04_05_02_zonas.csv
```

Data file to dataframe:

[16]:
```
data_03_zonas <- read_csv(file_data_03)
```

```
Rows: 286 Columns: 6
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (2): GEOCODIGO, DESBDT
dbl (4): CODBDT, area, X, Y

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:

[17]:
```
glimpse(data_03_zonas)
```

```
Rows: 286
Columns: 6
$ CODBDT    <dbl> 686213, 686214, 686215, 686216, 686217,
686218, 686219, 6862~
$ GEOCODIGO <chr> "001", "002", "003", "004", "005", "006",
"007", "008", "009~
$ DESBDT    <chr> "Abrantes", "Acacias", "Adelfas",
"Alameda", "Alameda de Osu~
$ area      <dbl> 1571618.8, 771569.7, 854805.6, 547596.1,
35137989.9, 661765.~
$ X         <dbl> -3.726140, -3.708702, -3.672841, -3.697291,
-3.569774, -3.86~
$ Y         <dbl> 40.37898, 40.40045, 40.40145, 40.41036,
40.47516, 40.32250, ~
```

Muestra de datos

[18]:
```
data_03_zonas |> slice_head(n = 5)
```

A spec_tbl_df: 5 x 6

CODBDT <dbl>	GEOCODIGO <chr>	DESBDT <chr>	area <dbl>	X <dbl>	Y <dbl>
686213	001	Abrantes	1571618.8	-3.726140	40.37898
686214	002	Acacias	771569.7	-3.708702	40.40045
686215	003	Adelfas	854805.6	-3.672841	40.40145
686216	004	Alameda	547596.1	-3.697291	40.41036
686217	005	Alameda de Osuna	35137989.9	-3.569774	40.47516

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
- Correspondencia de zonas básicas de salud y secciones censales de la Comunidad de Madrid
- Localizaciones y áreas de zonas básicas de salud

0.3.2 Transform data

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

Añadir a la tabla de indicadores la zona sanitaria a la que pertenece

- Comprobar discrepancias

Municipios de los indicadores que no están en las zonas sanitarias

[19]:
```
m1 <- data_01_indicadores |>
    count(CMUN) |> pull(CMUN)
```

```
m2 <- data_02_zonasec |>
    count(CMUN) |> pull(CMUN)
all(m1 %in% m2)
all(m2 %in% m1)
```

TRUE

TRUE

Todos los municipios están en ambos archivos

Secciones censales en fichero del INE que no tienen asignada una zona sanitaria en la Comunidad de Madrid

[20]:
```
data_01_indicadores |>
    anti_join(data_02_zonasec,
    by = c("CMUN", "dist", "secc")) |>
    count()
```

A spec_tbl_df: 1 x 1

n
<int>
345

Zonas básicas de Salud que tienen secciones censales que no aparecen en los indicadores del INE

[21]:
```
data_02_zonasec |>
    anti_join(data_01_indicadores,
    by = c("CMUN", "dist", "secc")) |>
    count()
```

A spec_tbl_df: 1 x 1

n
<int>
122

Unión de las tablas. Se asignan a las secciones censales las zonas de salud, dejando como NA las secciones que no tienen zona de salud. Se hace full join para no perder ningún dato.

[22]:
```
tdata_01_indicadores_join <- data_01_indicadores |>
    full_join(data_02_zonasec,
                by = c("CMUN", "dist", "secc")) |>
    arrange(CMUN, dist, secc)
slice_head(tdata_01_indicadores_join, n = 12)
```

A spec_tbl_df: 12 x 22

ccaa <dbl>	CPRO <dbl>	CMUN <chr>	dist <chr>	secc <chr>	t1_1 <dbl>	t2_1 <dbl>	t2_2 <dbl>	t3_1 <dbl>	t4_1 <db
13	28	001	01	001	55	NA	NA	NA	NA
13	28	002	01	001	2358	0.4724	0.5276	40.5161	0.14
13	28	002	01	002	2435	0.4830	0.5170	38.3339	0.19
13	28	003	01	001	248	0.3952	0.6048	47.4597	0.13
13	28	004	01	001	2886	0.5135	0.4865	44.4099	0.16
13	28	004	01	002	3376	0.4793	0.5207	40.6810	0.17
13	28	004	01	003	2161	0.5016	0.4984	38.0088	0.22
13	28	004	01	004	1523	0.5279	0.4721	42.5371	0.17
13	28	005	01	001	1648	0.5461	0.4539	43.7360	0.11
13	28	005	01	002	1015	0.5399	0.4601	42.0837	0.13
13	28	005	01	003	1728	0.5046	0.4954	42.9693	0.11
13	28	005	01	004	1349	0.5441	0.4559	48.5871	0.10

Imputación zona de salud a secciones censales que no la tienen Para no perder los indicadores de las secciones censales que no tienen zona básica, se le asigna la inmediatamente anterior (están ordenadas por municipio y zona)

[23]:
```
tdata_01_indicadores_fill <- tdata_01_indicadores_join |>
    fill(id_zona, nombre_zona)
```

Muestra primeros datos

[24]:
```
slice_head(tdata_01_indicadores_fill, n = 10)
```

A tibble: 10 x 22

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	0.1425	...
13	28	002	01	002	2435	0.4830	0.5170	38.3339	0.1959	...
13	28	003	01	001	248	0.3952	0.6048	47.4597	0.1371	...
13	28	004	01	001	2886	0.5135	0.4865	44.4099	0.1611	...
13	28	004	01	002	3376	0.4793	0.5207	40.6810	0.1739	...
13	28	004	01	003	2161	0.5016	0.4984	38.0088	0.2230	...
13	28	004	01	004	1523	0.5279	0.4721	42.5371	0.1753	...
13	28	005	01	001	1648	0.5461	0.4539	43.7360	0.1195	...
13	28	005	01	002	1015	0.5399	0.4601	42.0837	0.1379	...

Calcular indicadores por zona sanitaria

- Una zona sanitaria 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

[25]:
```
data_04_zonas_ind <- tdata_01_indicadores_fill |>
  mutate(across(c(t2_1:t2_2, t4_1:t12_1), ~ .x*t1_1)) |>
  group_by(id_zona, nombre_zona) |>
```

```
  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(id_zona) |>
  ungroup()
data_04_zonas_ind |> slice_head(n = 10)
```

A tibble: 10 x 18

id_zona <chr>	nombre_zona <chr>	nsec <int>	t3_1 <dbl>	t1_1 <dbl>	t2_ <d
001	Abrantes	23	42.31249	29872	0.5
002	Acacias	11	46.15717	17961	0.5
003	Adelfas	19	45.30472	28933	0.5
004	Alameda	18	43.70575	21393	0.4
005	Alameda de Osuna	18	43.31968	27259	0.5
006	Alcalde Bartolom<U+00E9> Gonz<U+00E1>lez	15	44.26034	21186	0.5
007	Alcal<U+00E1> de Guadaira	10	44.97266	12367	0.5
008	Alcobendas - Chopera	17	42.93862	27448	0.5
009	Alcocer	11	42.05643	15096	0.5
010	Algete	12	39.13500	28128	0.5

Comprobar que no se repiten zonas

[26]:
```
length(unique(data_04_zonas_ind$id_zona))
nrow(data_04_zonas_ind)
```

286

286

Análisis de los valores perdidos

[27]:
```
data_02_zonasec |> filter(id_zona == "123")
data_01_indicadores |> filter(CMUN == "005", dist == "08")
```

A spec_tbl_df: 3 x 5

id_zona <chr>	nombre_zona <chr>	CMUN <chr>	dist <chr>	secc <chr>
123	La Garena	005	08	013
123	La Garena	005	08	024
123	La Garena	005	08	029

A spec_tbl_df: 0 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>

Hay distritos en la correspondencia de zonas de salud que no existen en los indicadores del INE. Algunos forman una zona completa, por lo que se quedará esa zona sin los indicadores.

Asignación de la superficie a las zonas de salud y cálculo de densidad de población

[28]:
```
tdata_04_zonas_dens <- data_04_zonas_ind |>
    inner_join(data_03_zonas |> select(GEOCODIGO, area),
               by = c("id_zona" = "GEOCODIGO")) |>
    mutate(densidad_hab_km = t1_1/(area*10^(-6)))
```

Estructura de los datos:

[29]:
```
glimpse(tdata_04_zonas_dens)
```

```
Rows: 286
Columns: 20
$ id_zona         <chr> "001", "002", "003", "004", "005",
"006", "007", "008"~
$ nombre_zona     <chr> "Abrantes", "Acacias", "Adelfas",
"Alameda", "Alameda ~
$ nsec            <int> 23, 11, 19, 18, 18, 15, 10, 17, 11,
12, 14, 17, 8, 20,~
$ t3_1            <dbl> 42.31249, 46.15717, 45.30472,
43.70575, 43.31968, 44.2~
$ t1_1            <dbl> 29872, 17961, 28933, 21393, 27259,
21186, 12367, 27448~
$ t2_1            <dbl> 0.5345094, 0.5328775, 0.5383134,
0.4919805, 0.5153616,~
$ t2_2            <dbl> 0.4654906, 0.4671225, 0.4616866,
0.5080195, 0.4846384,~
$ t4_1            <dbl> 0.15619346, 0.10929873, 0.12408817,
0.07409008, 0.1742~
$ t4_2            <dbl> 0.6656459, 0.6516215, 0.6541986,
0.7685363, 0.6063697,~
$ t4_3            <dbl> 0.17816190, 0.23909411, 0.22173279,
0.15740288, 0.2193~
$ t5_1            <dbl> 0.21839930, 0.04313645, 0.07236948,
0.26145728, 0.0705~
$ t6_1            <dbl> 0.34508551, 0.07700825, 0.12420792,
0.35309699, 0.1280~
$ t7_1            <dbl> 0.04090796, 0.08494593, 0.07195008,
0.04949630, 0.0715~
$ t8_1            <dbl> 0.02880326, 0.07576304, 0.06532400,
0.04323113, 0.0626~
$ t9_1            <dbl> 0.2685716, 0.6368488, 0.6031800,
0.5672350, 0.5862647,~
$ t10_1           <dbl> 0.16656066, 0.08377987, 0.08766432,
0.12302998, 0.0851~
$ t11_1           <dbl> 0.4723181, 0.5083022, 0.5262385,
0.5410470, 0.5028547,~
$ t12_1           <dbl> 0.5637381, 0.5548573, 0.5770966,
0.6171510, 0.5490471,~
$ area            <dbl> 1571618.8, 771569.7, 854805.6,
```

```
547596.1, 35137989.9, 6~
$ densidad_hab_km <dbl> 19007.1534, 23278.5190, 33847.4619,
39067.1135, 775.77~
```

Muestra primeros datos:

[30]:
```
tdata_04_zonas_dens |>
    slice_head(n = 10)
```

A tibble: 10 x 20

id_zona <chr>	nombre_zona <chr>	nsec <int>	t3_1 <dbl>	t1_1 <dbl>	t2_ <c
001	Abrantes	23	42.31249	29872	0.5
002	Acacias	11	46.15717	17961	0.5
003	Adelfas	19	45.30472	28933	0.5
004	Alameda	18	43.70575	21393	0.4
005	Alameda de Osuna	18	43.31968	27259	0.5
006	Alcalde Bartolom<U+00E9> Gonz<U+00E1>lez	15	44.26034	21186	0.5
007	Alcal<U+00E1> de Guadaira	10	44.97266	12367	0.5
008	Alcobendas - Chopera	17	42.93862	27448	0.5
009	Alcocer	11	42.05643	15096	0.5
010	Algete	12	39.13500	28128	0.5

Zonas "sin población"

[31]:
```
tdata_04_zonas_dens |>
    filter(is.na(t1_1))
```

A tibble: 4 x 20

id_zona <chr>	nombre_zona <chr>	nsec <int>	t3_1 <dbl>	t1_1 <dbl>	t2_1 <dbl>	t2_2 <dbl
123	La Garena	3	NA	NA	NA	NA
165	Mar<U+00ED>a de Guzm<U+00E1>n	16	NA	NA	NA	NA
171	Miguel de Cervantes	10	NA	NA	NA	NA
226	Reyes Magos	17	NA	NA	NA	NA

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

[30]:
```
data_to_save_01 <- tdata_01_indicadores_fill
data_to_save_02 <- tdata_04_zonas_dens
```

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 con la zona sanitaria asignada

[31]:
```
caso <- "CU_04"
proceso <- '_05'
tarea <- "_05"
archivo <- "_01"
proper <- "_indicadores_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

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

[32]:
```
file_save <- paste0(caso, proceso, tarea, archivo, proper, extension)
path_out <- paste0(oPath, file_save)
write_csv(data_to_save_01, path_out)

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

```
path_out
```

```
File saved as:
```

'Data/Output/CU_04_05_05_01_indicadores_secciones.csv'

Archivo de indicadores por zona sanitaria

[33]:
```
archivo <- "_02"
proper <- "_indicadores_zonas"
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_02 <- paste0(caso, proceso, tarea, tcltk::tkgetSaveFile(), proper,␣
↪extension)
# path_out <- paste0(oPath, file_save_02)
# write_csv(data_to_save_02, path_out)

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

OPCION B: Especificar el nombre de archivo

[34]:
```
file_save <- paste0(caso, proceso, tarea, archivo, proper, extension)
path_out <- paste0(oPath, file_save)
write_csv(data_to_save_02, path_out)

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

```
File saved as:
```

'Data/Output/CU_04_05_05_02_indicadores_zonas.csv'

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

[ ]:
```
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 {readr}, {tidyr} and {dplyr} packages must be installed. They is part of the *tidyverse*.
- 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 * tidyr 1.2.1

0.8.3 Data structures

[ ]:
```
glimpse(tdata_04_zonas_ind)
```

Objeto `tdata_01_indicadores_fill` (indicadores por sección censal)

- Los mismos datos de entrada a los que se le añade el código y nombre de zona básica de salud que le corresponde a la sección censal

Objeto `tdata_04_zonas_ind` (indicadores por zona básica de salud)

- Indicadores de las 286 zonas básicas de salud
- Número de secciones censales que la componen
- Área en m^2 y Densidad de población de cada área en hab/km^2
- Hay valores perdidos por haber secciones censales en el archivo de zonas que no existen en el archivo del INE

Observaciones generales sobre los datos

- Una zona básica de saludo puede agrupar varios municipios, por lo que hay que decidir si agrupar por zona de salud o por zona de salud+municipio, optando por la primera
- Hay 345 secciones censales en el fichero de indicadores del INE que no se corresponden con ninguna zona sanitaria de la Comunidad de Madrid
- Hay 122 secciones censales en el fichero de zonas básicas de salud que no aparecen en el fichero de indicadores del INE
- Como los indicadores están en porcentajes, para agrupar datos hay que transformar primero a valores absolutos y después volver a calcular el porcentaje
- Los valores que se calculan como cero deben tratarse como NA
- El código de municipio en la tabla de correspondencias no se ha correspondido con el código de municipio en los datos de geolocalización en algunas zonas

0.9 Consideraciones para despliegue en piloto

- Los datos del INE son posteriores a los datos de las zonas sanitarias, lo que puede afectar a los modelos

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 unido las tablas de zonas sanitarias con secciones censales
- Se han calculado los indicadores del INE por zona de salud
- Se ha calculado la densidad de población de cada zona sanitaria
- Se ha asignado la zona sanitaria inmediatamente anterior a las secciones censales que no tenían

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

- Se debe comprobar por qué hay secciones censales sin correspondencia con zona sanitaria.
- Se debe comprobar por qué no coincide el código de municipio

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

[ ]:
```
# incluir código
```