05. - Data Collection_CU_34_05_servicios_completo_v_01

June 16, 2023

#

CU34_Predicción de demanda de servicios

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. Unir todos los datos en un data.frame con repeticiones

- Unión de todas las tablas donde se reptirán los datos que no tengan la máxima resolución

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.

[32]:
```
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

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

[33]:
```
library(readr)
library(dplyr)
```

0.1.3 Paths

[34]:
```
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)

1. Datos por día, sección y servicio

OPCION A: Seleccionar fichero en ventana para mayor comodidad

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

[35]:
```
# file_data_01 <- tcltk::tk_choose.files(multi = FALSE)
```

OPCION B: Especificar el nombre de archivo

[36]:
```
iFile_01 <- "CU_34_05_02_02_demanda_sim.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 leerán datos del archivo:  Data/Input/CU_34_05_02_02_demanda_sim.csv
```

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

[37]:
```
data_01 <- read_csv(file_data_01)
```

```
Rows: 274784 Columns: 7
  Column specification

Delimiter: ","
chr  (4): Servicio, CMUN, CDIS, CSEC
dbl  (2): Futbol, nservicios
date (1): Fecha

 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:

[38]:
```
data_01 |> glimpse()
```

```
Rows: 274,784
Columns: 7
$ Fecha      <date> 2022-01-01, 2022-01-01, 2022-01-01,
2022-01-01, 2022-01-01…
$ Servicio   <chr> "Taxi", "Taxi", "Taxi", "Taxi", "Taxi",
"Taxi", "Taxi", "Ta…
$ CMUN       <chr> "001", "002", "002", "003", "004", "004",
"004", "004", "00…
$ CDIS       <chr> "01", "01", "01", "01", "01", "01", "01",
"01", "01", "01",…
$ CSEC       <chr> "001", "001", "002", "001", "001", "002",
"003", "004", "00…
$ Futbol     <dbl> 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 1, 0, 0,…
$ nservicios <dbl> 0, 0, 0, 50, 0, 50, 50, 0, 2, 0, 0, 7, 0,
1, 4, 7, 10, 50, …
```

Muestra de datos:

[39]:
```
data_01 |> slice_head(n = 5)
```

A spec_tbl_df: 5 x 7

Fecha <date>	Servicio <chr>	CMUN <chr>	CDIS <chr>	CSEC <chr>	Futbol <dbl>	nservicios <dbl>
2022-01-01	Taxi	001	01	001	0	0
2022-01-01	Taxi	002	01	001	0	0
2022-01-01	Taxi	002	01	002	0	0
2022-01-01	Taxi	003	01	001	1	50
2022-01-01	Taxi	004	01	001	0	0

2. Datos por municipio y servicio

OPCION A: Seleccionar fichero en ventana para mayor comodidad

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

[40]:
```
# file_data <- tcltk::tk_choose.files(multi = FALSE)
```

OPCION B: Especificar el nombre de archivo

[41]:
```
iFile_02 <- "CU_34_05_02_03_capacidad_sim.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 leerán datos del archivo:  Data/Input/CU_34_05_02_03_capacidad_sim.csv
```

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

[42]:
```
data_02 <- read_csv(file_data_02)
```

```
Rows: 358 Columns: 3
  Column specification

Delimiter: ","
chr (2): CMUN, Servicio
dbl (1): capacidad

 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:

[43]:
```
data_02 |> glimpse()
```

```
Rows: 358
Columns: 3
$ CMUN      <chr> "001", "001", "002", "002", "003", "003",
"004", "004", "005…
$ Servicio  <chr> "Taxi", "Delivery", "Taxi", "Delivery",
"Taxi", "Delivery", …
$ capacidad <dbl> 50, 50, 50, 59, 50, 61, 50, 56, 65, 70, 66,
75, 64, 70, 50, …
```

Muestra de datos:

[44]:
```
data_02 |> slice_head(n = 5)
```

A spec_tbl_df: 5 x 3

CMUN <chr>	Servicio <chr>	capacidad <dbl>
001	Taxi	50
001	Delivery	50
002	Taxi	50
002	Delivery	59
003	Taxi	50

3. Datos por fecha y sección

OPCION A: Seleccionar fichero en ventana para mayor comodidad

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

[45]:
```
# file_data_03 <- tcltk::tk_choose.files(multi = FALSE)
```

OPCION B: Especificar el nombre de archivo

[46]:
```
iFile_03 <- "CU_34_05_04_meteo_secciones.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 leerán datos del archivo:  Data/Input/CU_34_05_04_meteo_secciones.csv
```

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

[47]:
```
data_03 <- read_csv(file_data_03)
```

```
Rows: 137392 Columns: 8
  Column specification

Delimiter: ","
chr  (3): CMUN, CDIS, CSEC
dbl  (4): tmed, prec, velmedia, presMax
date (1): fecha

 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:

[48]:
```
data_03 |> glimpse()
```

```
Rows: 137,392
Columns: 8
$ CMUN     <chr> "001", "002", "002", "003", "004", "004",
"004", "004", "005"…
$ CDIS     <chr> "01", "01", "01", "01", "01", "01", "01",
"01", "01", "01", "…
$ CSEC     <chr> "001", "001", "002", "001", "001", "002",
"003", "004", "001"…
$ fecha    <date> 2022-01-01, 2022-01-01, 2022-01-01,
2022-01-01, 2022-01-01, …
$ tmed     <dbl> 9.472965, 8.170448, 8.526402, 10.012282,
10.574368, 10.508450…
$ prec     <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0…
$ velmedia <dbl> 0.8229741, 0.9750191, 0.8874318, 2.3162929,
0.6970652, 0.6875…
$ presMax  <dbl> 901.6717, 970.1249, 967.2948, 849.5611,
964.2144, 965.4248, 9…
```

Muestra de datos:

[49]:
```
data_03 |> slice_head(n = 5)
```

A spec_tbl_df: 5 x 8

CMUN <chr>	CDIS <chr>	CSEC <chr>	fecha <date>	tmed <dbl>	prec <dbl>	velmedia <dbl>	presMax <dbl>
001	01	001	2022-01-01	9.472965	0	0.8229741	901.6717
002	01	001	2022-01-01	8.170448	0	0.9750191	970.1249
002	01	002	2022-01-01	8.526402	0	0.8874318	967.2948
003	01	001	2022-01-01	10.012282	0	2.3162929	849.5611
004	01	001	2022-01-01	10.574368	0	0.6970652	964.2144

4. Datos por sección

OPCION A: Seleccionar fichero en ventana para mayor comodidad

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

[50]:
```
# file_data_04 <- tcltk::tk_choose.files(multi = FALSE)
```

OPCION B: Especificar el nombre de archivo

[51]:
```
iFile_04 <- "CU_34_05_02_01_indicadores_secc_extendido.csv"
file_data_04 <- paste0(iPath, iFile_04)

if(file.exists(file_data_04)){
    cat("Se leerán datos del archivo: ", file_data_04)
} else{
    warning("Cuidado: el archivo no existe.")
}
```

```
Se leerán datos del archivo:
Data/Input/CU_34_05_02_01_indicadores_secc_extendido.csv
```

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

[52]:
```
data_04 <- read_csv(file_data_04)
```

```
Rows: 4440 Columns: 24
  Column specification

Delimiter: ","
chr  (4): CMUN, dist, secc, NSEC
dbl (20): ccaa, CPRO, t1_1, t2_1, t2_2, t3_1, t4_1, t4_2, t4_3, t5_1,
t6_1, …

 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:

[53]:
```
data_04 |> glimpse()
```

```
Rows: 4,440
Columns: 24
$ ccaa             <dbl> 13, 13, 13, 13, 13, 13, 13, 13, 13,
13, 13, 13, 13, 1…
$ CPRO             <dbl> 28, 28, 28, 28, 28, 28, 28, 28, 28,
28, 28, 28, 28, 2…
$ CMUN             <chr> "001", "002", "002", "003", "004",
"004", "004", "004…
$ dist             <chr> "01", "01", "01", "01", "01", "01",
"01", "01", "01",…
$ secc             <chr> "001", "001", "002", "001", "001",
"002", "003", "004…
$ t1_1             <dbl> 55, 2358, 2435, 248, 2886, 3376,
2161, 1523, 1648, 10…
$ t2_1             <dbl> NA, 0.4724, 0.4830, 0.3952, 0.5135,
0.4793, 0.5016, 0…
$ t2_2             <dbl> NA, 0.5276, 0.5170, 0.6048, 0.4865,
0.5207, 0.4984, 0…
$ t3_1             <dbl> NA, 40.5161, 38.3339, 47.4597,
44.4099, 40.6810, 38.0…
$ t4_1             <dbl> NA, 0.1425, 0.1959, 0.1371, 0.1611,
0.1739, 0.2230, 0…
$ t4_2             <dbl> NA, 0.7443, 0.7002, 0.6573, 0.6067,
0.6727, 0.6395, 0…
```

```
$ t4_3             <dbl> NA, 0.1132, 0.1039, 0.2056, 0.2322,
0.1534, 0.1374, 0…
$ t5_1             <dbl> NA, 0.1569, 0.1544, 0.1129, 0.1639,
0.1440, 0.1749, 0…
$ t6_1             <dbl> NA, 0.1968, 0.1951, 0.1411, 0.2128,
0.1730, 0.2138, 0…
$ t7_1             <dbl> NA, 0.4016, 0.4244, 0.1542, 0.1470,
0.1596, 0.1668, 0…
$ t8_1             <dbl> NA, 0.3971, 0.4224, 0.1449, 0.1409,
0.1506, 0.1602, 0…
$ t9_1             <dbl> NA, 0.4377, 0.4438, 0.5280, 0.2602,
0.3234, 0.2859, 0…
$ t10_1            <dbl> NA, 0.0912, 0.1226, 0.0932, 0.1814,
0.1478, 0.1658, 0…
$ t11_1            <dbl> NA, 0.6063, 0.5955, 0.5000, 0.4213,
0.5170, 0.4943, 0…
$ t12_1            <dbl> NA, 0.6672, 0.6788, 0.5514, 0.5147,
0.6067, 0.5926, 0…
$ NSEC             <chr> "Acebeda, La - 01.001", "Ajalvir -
01.001", "Ajalvir …
$ area             <dbl> 21848790.60, 15585050.39,
4148273.41, 25674490.85, 50…
$ elevation        <dbl> 1401, 653, 715, 1150, 593, 604, 587,
570, 594, 594, 5…
$ densidad_hab_km2 <dbl> 2.517302, 151.298837, 586.991204,
9.659393, 571.86106…
```

Muestra de datos:

[54]:
```
data_04 |> slice_head(n = 5)
```

A spec_tbl_df: 5 x 24

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

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:

- Datos por fecha, servicio y sección (demanda)
- Datos por munipio (capacidad)
- Datos por fecha y sección (meteorológicos)
- Datos por sección (indicadores)

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 transformation

[55]:
```
colnames(data_01)
colnames(data_02)
colnames(data_03)
colnames(data_04)
```

1. 'Fecha' 2. 'Servicio' 3. 'CMUN' 4. 'CDIS' 5. 'CSEC' 6. 'Futbol' 7. 'nservicios'

1. 'CMUN' 2. 'Servicio' 3. 'capacidad'

1. 'CMUN' 2. 'CDIS' 3. 'CSEC' 4. 'fecha' 5. 'tmed' 6. 'prec' 7. 'velmedia' 8. 'presMax'

1. 'ccaa' 2. 'CPRO' 3. 'CMUN' 4. 'dist' 5. 'secc' 6. 't1_1' 7. 't2_1' 8. 't2_2' 9. 't3_1' 10. 't4_1' 11. 't4_2' 12. 't4_3' 13. 't5_1' 14. 't6_1' 15. 't7_1' 16. 't8_1' 17. 't9_1' 18. 't10_1' 19. 't11_1' 20. 't12_1' 21. 'NSEC' 22. 'area' 23. 'elevation' 24. 'densidad_hab_km2'

[56]:
```
data <- data_01 |>
    full_join(data_02, by = c("CMUN", "Servicio")) |>
    full_join(data_03, by = c("Fecha" = "fecha", "CMUN", "CDIS", "CSEC")) |>
    full_join(data_04, by = c("CMUN", "CDIS" = "dist", "CSEC" = "secc"))
```

[57]:
```
data |> glimpse()
```

```
Rows: 274,792
Columns: 33
$ Fecha            <date> 2022-01-01, 2022-01-01, 2022-01-01,
2022-01-01, 2022…
$ Servicio         <chr> "Taxi", "Taxi", "Taxi", "Taxi",
"Taxi", "Taxi", "Taxi…
$ CMUN             <chr> "001", "002", "002", "003", "004",
"004", "004", "004…
$ CDIS             <chr> "01", "01", "01", "01", "01", "01",
"01", "01", "01",…
$ CSEC             <chr> "001", "001", "002", "001", "001",
"002", "003", "004…
$ Futbol           <dbl> 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0,
0, 0, 0, 0, 1,…
$ nservicios       <dbl> 0, 0, 0, 50, 0, 50, 50, 0, 2, 0, 0,
7, 0, 1, 4, 7, 10…
$ capacidad        <dbl> 50, 50, 50, 50, 50, 50, 50, 50, 65,
65, 65, 65, 65, 6…
$ tmed             <dbl> 9.472965, 8.170448, 8.526402,
10.012282, 10.574368, 1…
$ prec             <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0,…
```

```
$ velmedia         <dbl> 0.8229741, 0.9750191, 0.8874318,
2.3162929, 0.6970652…
$ presMax          <dbl> 901.6717, 970.1249, 967.2948,
849.5611, 964.2144, 965…
$ ccaa             <dbl> 13, 13, 13, 13, 13, 13, 13, 13, 13,
13, 13, 13, 13, 1…
$ CPRO             <dbl> 28, 28, 28, 28, 28, 28, 28, 28, 28,
28, 28, 28, 28, 2…
$ t1_1             <dbl> 55, 2358, 2435, 248, 2886, 3376,
2161, 1523, 1648, 10…
$ t2_1             <dbl> NA, 0.4724, 0.4830, 0.3952, 0.5135,
0.4793, 0.5016, 0…
$ t2_2             <dbl> NA, 0.5276, 0.5170, 0.6048, 0.4865,
0.5207, 0.4984, 0…
$ t3_1             <dbl> NA, 40.5161, 38.3339, 47.4597,
44.4099, 40.6810, 38.0…
$ t4_1             <dbl> NA, 0.1425, 0.1959, 0.1371, 0.1611,
0.1739, 0.2230, 0…
$ t4_2             <dbl> NA, 0.7443, 0.7002, 0.6573, 0.6067,
0.6727, 0.6395, 0…
$ t4_3             <dbl> NA, 0.1132, 0.1039, 0.2056, 0.2322,
0.1534, 0.1374, 0…
$ t5_1             <dbl> NA, 0.1569, 0.1544, 0.1129, 0.1639,
0.1440, 0.1749, 0…
$ t6_1             <dbl> NA, 0.1968, 0.1951, 0.1411, 0.2128,
0.1730, 0.2138, 0…
$ t7_1             <dbl> NA, 0.4016, 0.4244, 0.1542, 0.1470,
0.1596, 0.1668, 0…
$ t8_1             <dbl> NA, 0.3971, 0.4224, 0.1449, 0.1409,
0.1506, 0.1602, 0…
$ t9_1             <dbl> NA, 0.4377, 0.4438, 0.5280, 0.2602,
0.3234, 0.2859, 0…
$ t10_1            <dbl> NA, 0.0912, 0.1226, 0.0932, 0.1814,
0.1478, 0.1658, 0…
$ t11_1            <dbl> NA, 0.6063, 0.5955, 0.5000, 0.4213,
0.5170, 0.4943, 0…
$ t12_1            <dbl> NA, 0.6672, 0.6788, 0.5514, 0.5147,
0.6067, 0.5926, 0…
$ NSEC             <chr> "Acebeda, La - 01.001", "Ajalvir -
01.001", "Ajalvir …
$ area             <dbl> 21848790.60, 15585050.39,
4148273.41, 25674490.85, 50…
$ elevation        <dbl> 1401, 653, 715, 1150, 593, 604, 587,
570, 594, 594, 5…
$ densidad_hab_km2 <dbl> 2.517302, 151.298837, 586.991204,
9.659393, 571.86106…
```

[58]:
```
data |> slice_head(n = 5)
```

A spec_tbl_df: 5 x 33

Fecha <date>	Servicio <chr>	CMUN <chr>	CDIS <chr>	CSEC <chr>	Futbol <dbl>	nservicios <dbl>	capacidad <dbl>	tmed <db
2022-01-01	Taxi	001	01	001	0	0	50	9.47
2022-01-01	Taxi	002	01	001	0	0	50	8.17
2022-01-01	Taxi	002	01	002	0	0	50	8.52
2022-01-01	Taxi	003	01	001	1	50	50	10.0
2022-01-01	Taxi	004	01	001	0	0	50	10.5

0.4 Synthetic Data Generation

No aplica

0.5 Fake Data Generation

No aplica

0.6 Open Data

No aplica

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

[59]:
```
data_to_save <- data
```

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

[60]:
```
caso <- "CU_34"
proceso <- '_05'
tarea <- "_05"
archivo <- ""
```

```
proper <- "_servicios_completo"
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/.

[61]:
```
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_34_05_05_servicios_completo.csv'

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

[62]:
```
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

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 readr, dplyr deben estar instalados.

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

Objeto `data`

- Hay 274792 filas con información de las siguientes variables:
  - Fecha
  - Servicio
  - CMUN
  - CDIS
  - CSEC
  - Futbol
  - nservicios
  - capacidad
  - tmed
  - prec
  - velmedia
  - presMax
  - ccaa
  - CPRO
  - t1_1
  - t2_1
  - t2_2
  - t3_1
  - t4_1
  - t4_2
  - t4_3
  - t5_1
  - t6_1
  - t7_1
  - t8_1
  - t9_1
  - t10_1
  - t11_1
  - t12_1
  - NSEC
  - area
  - elevation
  - densidad_hab_km2

Observaciones generales sobre los datos

- Los datos se repiten en aquellas columnas que no tienen la misma resolución que el conjunto más amplio, es decir:
  - Los datos de capacidad son los mismos para todos los municipios todos los días
  - Los datos de indicadores son los mismos para todos los días en la misma sección
- El fichero incluye algunos datos reales y otros sintéticos

0.8.4 Consideraciones para despliegue en piloto

- Este es el fichero principal que contiene todos los datos del caso
- Si llegaran datos reales solicitados a la empresa, habría que sustituir los sintéticos por los reales
- Para la representación en mapas, se deben combinar los datos del csv con los datos del json en la implementación que se haga

0.8.5 Consideraciones para despliegue en producción

- Se deben crear los procesos ETL en producción necesarios para que los datos de entrada estén actualizados
- Se deben conseguir datos reales para sustituir los sintéticos
- Para la representación en mapas, se deben combinar los datos del csv con los datos del json de la primera trea en la implementación que se haga

0.9 Main Actions

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

- Se han unido las tablas de datos del caso de uso en un único archivo

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

- Se deben realizar el resto de procesos de la dimensión Data Processing

0.10 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 código 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.
- El fichero deberá estar incluido en la ruta de datos de la implementación

CODE

[ ]:
```
# incluir código
```