05. - Data Collection_CU_34_02_demanda_sim_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 01. Simulación de demanda de servicios

- Simulación de demanda de servicios durante un periodo de tiempo por sección censal
- Se toman unas fechas de inicio y fin y un conjunto de servicios
- Se expanden a secciones censales y se simulan demandas de servicios

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.

[1]:
```
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

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(lubridate)
library(elevatr)
```

```
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



Attaching package: ‘lubridate’


The following objects are masked from ‘package:base’:

    date, intersect, setdiff, 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.

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. Geometrías de las secciones censales

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

[5]:
```
iFile_01 <- "CU_34_05_01_secciones_geo.json"
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_01_secciones_geo.json
```

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

[6]:
```
data_01 <- st_read(file_data_01)
```

```
Reading layer `CU_34_05_01_secciones_geo' from data source
  `/Users/emilio.lcano/academico/gh_repos/__transferencia/citizenlab/CitizenLab-
Research-and-Development/casos_urjc/notebooks/II_data_processing/34_servicios/Da
ta/Input/CU_34_05_01_secciones_geo.json'
  using driver `GeoJSON'
Simple feature collection with 4432 features and 16 fields
Geometry type: MULTIPOLYGON
Dimension:     XY
Bounding box:  xmin: -4.579006 ymin: 39.8848 xmax: -3.052983 ymax: 41.16584
Geodetic CRS:  WGS 84
```

Estructura de los datos:

[7]:
```
data_01 |> glimpse()
```

```
Rows: 4,432
Columns: 17
$ CUSEC    <chr> "2800101001", "2800201001", "2800201002",
"2800301001", "2800…
$ CUMUN    <chr> "28001", "28002", "28002", "28003", "28004",
"28004", "28004"…
$ CSEC     <chr> "001", "001", "002", "001", "001", "002",
"003", "004", "001"…
$ CDIS     <chr> "01", "01", "01", "01", "01", "01", "01",
"01", "01", "01", "…
$ CMUN     <chr> "001", "002", "002", "003", "004", "004",
"004", "004", "005"…
$ CPRO     <chr> "28", "28", "28", "28", "28", "28", "28",
"28", "28", "28", "…
$ CCA      <chr> "13", "13", "13", "13", "13", "13", "13",
"13", "13", "13", "…
$ CUDIS    <chr> "2800101", "2800201", "2800201", "2800301",
"2800401", "28004…
$ CLAU2    <chr> "28001", "28002", "28002", "28003", "28004",
"28004", "28004"…
$ NPRO     <chr> "Madrid", "Madrid", "Madrid", "Madrid",
"Madrid", "Madrid", "…
$ NCA      <chr> "Comunidad de Madrid", "Comunidad de
Madrid", "Comunidad de M…
$ CNUT0    <chr> "ES", "ES", "ES", "ES", "ES", "ES", "ES",
"ES", "ES", "ES", "…
$ CNUT1    <chr> "3", "3", "3", "3", "3", "3", "3", "3", "3",
"3", "3", "3", "…
$ CNUT2    <chr> "0", "0", "0", "0", "0", "0", "0", "0", "0",
"0", "0", "0", "…
$ CNUT3    <chr> "0", "0", "0", "0", "0", "0", "0", "0", "0",
"0", "0", "0", "…
$ NMUN     <chr> "Acebeda, La", "Ajalvir", "Ajalvir",
"Alameda del Valle", "Ál…
$ geometry <MULTIPOLYGON [°]> MULTIPOLYGON (((-3.64316 41…,
MULTIPOLYGON (((…
```

Muestra de datos:

[8]:
```
data_01 |> tibble() |> slice_head(n = 5)
```

	CUSEC <chr>	CUMUN <chr>	CSEC <chr>	CDIS <chr>	CMUN <chr>	CPRO <chr>	CCA <chr>	CUDIS <chr>	CLAU2 <chr>	NPI <ch
A tibble: 5 x 17	2800101001	28001	001	01	001	28	13	2800101	28001	Mad
	2800201001	28002	001	01	002	28	13	2800201	28002	Mad
	2800201002	28002	002	01	002	28	13	2800201	28002	Mad
	2800301001	28003	001	01	003	28	13	2800301	28003	Mad
	2800401001	28004	001	01	004	28	13	2800401	28004	Mad

2. Indicadores de las secciones censales

OPCION A: Seleccionar fichero en ventana para mayor comodidad

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

[9]:
```
# file_data_02 <- tcltk::tk_choose.files(multi = FALSE)
```

OPCION B: Especificar el nombre de archivo

[10]:
```
iFile_02 <- "CU_04_05_03_01_indicadores_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 leerán datos del archivo:
Data/Input/CU_04_05_03_01_indicadores_secciones.csv
```

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

[11]:
```
data_02 <- read_csv(file_data_02)
```

```
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, …

 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:

[12]:
```
data_02|> glimpse()
```

```
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 datos:

[13]:
```
data_02 |> tibble() |> slice_head(n = 5)
```

	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>	t4_ <d
	13	28	001	01	001	55	NA	NA	NA	NA	NA
A tibble: 5 x 20	13	28	002	01	001	2358	0.4724	0.5276	40.5161	0.1425	0.7
	13	28	002	01	002	2435	0.4830	0.5170	38.3339	0.1959	0.7
	13	28	003	01	001	248	0.3952	0.6048	47.4597	0.1371	0.6
	13	28	004	01	001	2886	0.5135	0.4865	44.4099	0.1611	0.6

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:

- Secciones censales de la Comunidad de Madrid
- Indicadores del INE por sección censal

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

- Asignar nombre de sección
- Calcular área
- Obtener elevación
- Eliminar geometrías

[14]:
```
tdata_01 <- data_01 |>
    mutate(NSEC = paste0(NMUN, " - ", CDIS, ".", CSEC),
        area = st_area(data_01)) |>
    st_centroid() |>
    get_elev_point(src = "aws")  |>
    select(-elev_units) |>
    st_drop_geometry()
```

```
Warning message in st_centroid.sf(mutate(data_01, NSEC = paste0(NMUN, " - ",
CDIS, :
"st_centroid assumes attributes are constant over geometries of x"
Mosaicing & Projecting

Note: Elevation units are in meters
```

[15]:
```
tdata_01 |> glimpse()
```

```
Rows: 4,432
Columns: 19
$ CUSEC     <chr> "2800101001", "2800201001", "2800201002",
"2800301001", "280…
```

```
$ CUMUN     <chr> "28001", "28002", "28002", "28003",
"28004", "28004", "28004…
$ CSEC      <chr> "001", "001", "002", "001", "001", "002",
"003", "004", "001…
$ CDIS      <chr> "01", "01", "01", "01", "01", "01", "01",
"01", "01", "01", …
$ CMUN      <chr> "001", "002", "002", "003", "004", "004",
"004", "004", "005…
$ CPRO      <chr> "28", "28", "28", "28", "28", "28", "28",
"28", "28", "28", …
$ CCA       <chr> "13", "13", "13", "13", "13", "13", "13",
"13", "13", "13", …
$ CUDIS     <chr> "2800101", "2800201", "2800201", "2800301",
"2800401", "2800…
$ CLAU2     <chr> "28001", "28002", "28002", "28003",
"28004", "28004", "28004…
$ NPRO      <chr> "Madrid", "Madrid", "Madrid", "Madrid",
"Madrid", "Madrid", …
$ NCA       <chr> "Comunidad de Madrid", "Comunidad de
Madrid", "Comunidad de …
$ CNUT0     <chr> "ES", "ES", "ES", "ES", "ES", "ES", "ES",
"ES", "ES", "ES", …
$ CNUT1     <chr> "3", "3", "3", "3", "3", "3", "3", "3",
"3", "3", "3", "3", …
$ CNUT2     <chr> "0", "0", "0", "0", "0", "0", "0", "0",
"0", "0", "0", "0", …
$ CNUT3     <chr> "0", "0", "0", "0", "0", "0", "0", "0",
"0", "0", "0", "0", …
$ NMUN      <chr> "Acebeda, La", "Ajalvir", "Ajalvir",
"Alameda del Valle", "Á…
$ NSEC      <chr> "Acebeda, La - 01.001", "Ajalvir - 01.001",
"Ajalvir - 01.00…
$ area      [m^2] 21848790.60 [m^2], 15585050.39 [m^2], 4148273.41
[m^2], 2567…
$ elevation <dbl> 1401, 653, 715, 1150, 593, 604, 587, 570,
594, 594, 594, 594…
```

- Unir área y elevación a la tabla de indicadores, y eliminar geometría

[16]:
```
tdata_02 <- data_02 |>
    full_join(tdata_01 |> select(NSEC, CMUN, CSEC, CDIS, area, elevation),
        by = c("CMUN" = "CMUN", "dist" = "CDIS", "secc" = "CSEC")) |>
    mutate(densidad_hab_km2 = as.numeric(t1_1/(area*1e-6))) |>
    st_drop_geometry()
```

[17]:
```
tdata_02 |> 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             [m^2] 21848790.60 [m^2], 15585050.39 [m^2],
4148273.41 [m^2…
$ 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…

0.4 Fake Data Generation

- Se crea una función para simular el número de servicios de cada uno de los dos tipos propuestos:
  - Taxi y delivery
  - Dependen de si hay evento importante (fútbol) del número de titulados y de la densidad de población
- Después se simula a lo largo del periodo parametrizado
- Parámetros:

[18]:
```
fechas <- seq(from = as.Date("2022-01-01"),
                              to = as.Date("2022-01-31"),
              by = 1)

servicios <- c("Taxi", "Delivery")
```

- Funciones:

[19]:
```
futbol <- Vectorize(
  function(fecha, servicio){
    p <- 0.8
    if(wday(fecha) %in% 6:7){
      p <- p*(1+0.1*(servicio == "Taxi"))
    }
    sample(0:1, 1, prob = c(p, 1-p))
  })

n_servicios <- Vectorize(
  function(futbol, servicio, titulados, densidad){
    titulados <- ifelse(is.na(titulados), 0, titulados)
    densidad <- ifelse(is.na(densidad), 1e-6, densidad)
    l <- rpois(1, round(densidad*1e-6*100)) + 50*futbol
    if(servicio == "Delivery" & !is.na(titulados)){
      l <- l + round(20*titulados)
    }
    return(l)
  }
)
```

- Simulación de datos fake demanda de servicio

[20]:
```
set.seed(1)
fdata_01 <- expand.grid(NSEC = tdata_01$NSEC,
                        Fecha = fechas,
                        Servicio = servicios) |>
  inner_join(tdata_01 |>
```

```
              select(NSEC, CMUN, CDIS, CSEC, area, elevation)) |>
  inner_join(tdata_02 |>
              select(personas = t1_1,
                     edad = t3_1,
                     mayores = t4_3,
                     titulados = t9_1,
                     CMUN, dist, secc, densidad_hab_km2),
             by = c("CMUN" = "CMUN",
                    "CDIS" = "dist",
                    "CSEC" = "secc")) |>
  mutate(Futbol = futbol(Fecha, Servicio)) |>
  mutate(nservicios = n_servicios(Futbol, Servicio, titulados,␣
↪densidad_hab_km2)) |>
  select(-NSEC, -area, -elevation, -personas, -edad, -mayores, -titulados,␣
↪-densidad_hab_km2)
```

```
Joining with `by = join_by(NSEC)`
```

[21]:
```
fdata_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   <fct> Taxi, Taxi, Taxi, Taxi, Taxi, Taxi, Taxi,
Taxi, Taxi, Taxi,…
$ 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     <int> 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, …
```

[22]:
```
fdata_01 |> slice_head(n = 5)
```

A data.frame: 5 x 7

Fecha <date>	Servicio <fct>	CMUN <chr>	CDIS <chr>	CSEC <chr>	Futbol <int>	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

- Simulación datos fake capacidad de servicios. Asumimos capacidad por municipio, el máximo

de todos los días

[23]:
```
fdata_02 <- fdata_01 |>
    group_by(CMUN, Servicio) |>
    summarise(capacidad = max(nservicios), .groups = "drop")
```

[24]:
```
glimpse(fdata_02)
```

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

[25]:
```
fdata_02 |> slice_head(n = 5)
```

A tibble: 5 x 3

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

0.5 Synthetic 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

1. Indicadores INE con identificador sección, área, elevación y densidad

Identificamos los datos a guardar

[26]:
```
data_to_save_01 <- tdata_02
```

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

[27]:
```
caso <- "CU_34"
proceso <- '_05'
tarea <- "_02"
archivo <- "_01"
proper <- "_indicadores_secc_extendido"
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

[28]:
```
# 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_01
```

OPCION B: Especificar el nombre de archivo

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

[29]:
```
file_save_01 <- paste0(caso, proceso, tarea, archivo, 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_01
```

```
File saved as:
```

’Data/Output/CU_34_05_02_01_indicadores_secc_extendido.csv’

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

[30]:
```
path_in_01 <- paste0(iPath, file_save_01)
file.copy(path_out_01, path_in_01, overwrite = TRUE)
```

TRUE

2. Simulación de demanda de servicios

Identificamos los datos a guardar

[31]:
```
data_to_save_02 <- fdata_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

0.7.2 Proceso 05

[32]:
```
archivo <- "_02"
proper <- "_demanda_sim"
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

[33]:
```
# 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_02
```

OPCION B: Especificar el nombre de archivo

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

[34]:
```
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_34_05_02_02_demanda_sim.csv'

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

[35]:
```
path_in_02 <- paste0(iPath, file_save_02)
file.copy(path_out_02, path_in_02, overwrite = TRUE)
```

TRUE

3. Simulación de capacidad de servicios

Identificamos los datos a guardar

[36]:
```
data_to_save_03 <- fdata_02
```

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

0.7.3 Proceso 05

[37]:
```
archivo <- "_03"
proper <- "_capacidad_sim"
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

[38]:
```
# file_save_03 <- paste0(caso, proceso, tarea, tcltk::tkgetSaveFile(), proper,␣
 ↪extension)
# path_out_03 <- paste0(oPath, file_save_03)
# write_csv(data_to_save_02, path_out_02)
```

```
# cat('File saved as: ')
# path_out_03
```

OPCION B: Especificar el nombre de archivo

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

[39]:
```
file_save_03 <- paste0(caso, proceso, tarea, archivo, proper, extension)
path_out_03 <- paste0(oPath, file_save_03)
write_csv(data_to_save_03, path_out_03)

cat('File saved as: ')
path_out_03
```

```
File saved as:
```

'Data/Output/CU_34_05_02_03_capacidad_sim.csv'

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

[40]:
```
path_in_03 <- paste0(iPath, file_save_03)
file.copy(path_out_03, path_in_03, 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 sf, readr, dplyr, lubridate, elevatr 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 * sf 1.0.9 * readr 2.1.3 * dplyr 1.1.0 * lubridate 1.9.1 * elevatr 0.4.2

Objeto `tdata_02`

- Datos de indicadores ampliados
- Hay 4440 filas con información de las siguientes variables:

  - ccaa
  - CPRO
  - CMUN
  - dist
  - secc
  - 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

Objeto `fdata_01`

- Datos de demanda de servicios
- Hay 274784 filas con información de las siguientes variables:
  - Fecha
  - Servicio
  - CMUN
  - CDIS
  - CSEC
  - area
  - elevation
  - personas
  - edad
  - mayores
  - titulados
  - densidad_hab_km2
  - Futbol
  - nservicios

Objeto `fdata_02`

- Datos de capacidad de servicios
- Hay 358 filas con información de las siguientes variables:

  - CMUN
  - Servicio
  - capacidad

Observaciones generales sobre los datos

- No aplica

0.8.3 Consideraciones para despliegue en piloto

- Los datos son simulados

0.8.4 Consideraciones para despliegue en producción

- Se deben crear los procesos ETL en producción necesarios para obtener datos reales

0.9 Main Actions

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

- Se ha asignado una etiqueta a cada sección censal
- Se ha calculado la densidad de población de cada sección
- Se ha calculado la elevación de las secciones censales
- Se han simulado datos de servicios

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

- Se deben añadir los datos meteorológicos

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 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

[41]:
```
# incluir código
```