05. - Data Collection_CU_18_19_diario_infraestructuras_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 19. Simular datos diarios de infraestructuras

- Datos diarios para 2019
- Datos de ocupación y demanda de cada infraestructura cada día
- Datos de dos tipos de eventos: evento infraestructura y evento zona
- Hace falta identificar a las infraestructuras

Table of Contents

Settings

Data Load

ETL Processes

Import data from: CSV, Excel, Tab, JSON, SQL, and Parquet files

Synthetic Data Generation

Fake Data Generation

Open Data

Data Save

Main Conclusions

Main Actions

Acciones done

Acctions to perform

0.1 Settings

0.1.1 Packages to use

- {tcltk} para selección interactiva de archivos locales

- {tibble} para operaciones con data.frames
- {readr} para leer y escribir archivos csv
- {dplyr} para explorar datos

[1]:
```
library(tibble)
library(readr)
library(dplyr)
```

```
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
```

0.1.2 Paths

[2]:
```
iPath <- "Data/Input/"
oPath <- "Data/Output/"
```

0.2 Data Load

If there are more than one input file, make as many sections as files to import.

Instrucciones - Los ficheros de entrada del proceso están siempre en Data/Input/.
- Si hay más de un fichero de entrada, se crean tantos objetos iFile_xx y file_data_xx como ficheros de entrada (xx número correlativo con dos dígitos, rellenar con ceros a la izquierda)

OPCION A: Seleccionar fichero en ventana para mayor comodidad

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

[3]:
```
# file_data <- tcltk::tk_choose.files(multi = FALSE)
```

OPCION B: Especificar el nombre de archivo

[4]:
```
iFile <- "CU_18_05_18_infraestructuras.csv"
file_data <- paste0(iPath, iFile)

if(file.exists(file_data)){
    cat("Se leerán datos del archivo: ", file_data)
} else{
```

```
    warning("Cuidado: el archivo no existe.")
}
```

```
Se leer<U+00E1>n datos del archivo:  Data/Input/CU_18_05_18_infraestructuras.csv
```

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

[5]:
```
data_01 <- read_csv(file_data)
```

```
Rows: 1138 Columns: 18
-- Column specification --------------------------------------------------------
Delimiter: ","
chr  (8): grupo, tipo, nombre, CODMUN, DIRECCION, info, CMUN, CDIS
dbl (10): dist_aeropuerto, dist_intercambiador, dist_cercanias,
dist_hospita…

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:

[6]:
```
glimpse(data_01)
```

```
Rows: 1,138
Columns: 18
$ grupo               <chr> "Transporte", "Transporte",
"Transporte", "Transpo~
$ tipo                <chr> "Intercambiadores",
"Intercambiadores", "Intercamb~
$ nombre              <chr> "Grandes Intercambiadores Plaza
El<U+00ED>ptica", "Grande~
$ CODMUN              <chr> "079", "079", "079", "079",
"079", "079", "079", "~
$ DIRECCION           <chr> "Plaza El<U+00ED>ptica s/n",
"Calle Princesa, 89", "Estac~
$ info                <chr> NA, NA, NA, NA, NA, NA, NA, NA,
NA, NA, NA, NA, NA~
$ CMUN                <chr> "079", "079", "079", "079",
"079", "079", "079", "~
$ CDIS                <chr> "12", "09", "09", "05", "05",
"05", "06", "02", "0~
$ dist_aeropuerto     <dbl> 5.2031430, 8.5507475, 7.3002463,
9.9339069, 9.1988~
$ dist_intercambiador <dbl> 0.0000000, 0.0000000, 0.0000000,
0.0000000, 0.0000~
```

```
$ dist_cercanias      <dbl> 1.648973148, 1.468638183,
0.088338692, 0.736877403~
$ dist_hospital       <dbl> 1.8154156, 0.4221115, 1.3855767,
1.0733484, 0.3488~
$ dist_universidad    <dbl> 1.29960755, 0.30460781,
0.92076145, 0.08835165, 0.~
$ dist_gob            <dbl> 0.61772353, 0.62638474,
1.17005116, 0.14904962, 1.~
$ dist_embajada       <dbl> 3.921324e-01, 1.457397e+00,
5.624560e+00, 8.316773~
$ dist_ccomercial     <dbl> 0.73115706, 1.61056736,
0.90608573, 0.77583440, 0.~
$ X                   <dbl> -3.716577, -3.719508, -3.719560,
-3.689044, -3.676~
$ Y                   <dbl> 40.38540, 40.43474, 40.42080,
40.46719, 40.43797, ~
```

Muestra de datos:

[7]:
```
slice_head(data_01, n = 5)
```

A spec_tbl_df: 5 x 18

grupo <chr>	tipo <chr>	nombre <chr>
Transporte	Intercambiadores	Grandes Intercambiadores Plaza El<U+00ED>ptica
Transporte	Intercambiadores	Grandes Intercambiadores Moncloa
Transporte	Intercambiadores	Grandes Intercambiadores Pr<U+00ED>ncipe P<U+00
Transporte	Intercambiadores	Grandes Intercambiadores Plaza de Castilla
Transporte	Intercambiadores	Grandes Intercambiadores Avenida de Am<U+00E9>ric

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 de infraestructuras

Incluir apartados si procede para: Extracción de datos (select, filter), Transformación de datos, (mutate, joins, …). Si es necesario tratar datos perdidos, indicarlo también en NB 09.2

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

Data transform

- Crear identificador único para cada infraestructura

[8]:
```
tdata_01 <- data_01 |>
    rowid_to_column("id_inf")
```

0.4 Synthetic Data Generation

No aplica

0.5 Fake Data Generation

Se generan datos de un año para cada una de las infraestructuras con las variables:

- capacidad
- demanda
- evento_infra
- evento_zona

- Generar fechas

[9]:
```
inicio <- as.Date("2019-01-01")
fin <- as.Date("2019-12-31")
dias <- seq(inicio, fin, by = 1)
```

- Generar las filas suficientes

[10]:
```
data_02 <- expand.grid(id_inf = tdata_01$id_inf,
            fecha = dias)
```

- Generar variables aleatoriamente

[11]:
```
tdata_02 <- data_02 |>
    mutate(capacidad = rpois(n(), 1000)) |>
    mutate(demanda = capacidad + sample(c(-1, 1), n(), TRUE)*round(capacidad*0.
↪1 + rnorm(n(), 10, 2))) |>
    rowwise() |>
    mutate(evento_infra = sample(c(0, 1), n(), TRUE, c((demanda > capacidad)*0.
↪2 + (demanda < capacidad)*0.8,
                                                       (demanda > capacidad)*0.
↪8 + (demanda < capacidad)*0.2))) |>
    mutate(evento_zona = sample(c(0, 1), n(), TRUE, c((demanda > capacidad)*0.4␣
↪+ (demanda < capacidad)*0.6,
                                                       (demanda > capacidad)*0.
↪6 + (demanda < capacidad)*0.4)))
```

[12]:
```
tdata_02 |> glimpse()
```

```
Rows: 415,370
Columns: 6
Rowwise:
$ id_inf       <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12,
13, 14, 15, 16, 17~
$ fecha        <date> 2019-01-01, 2019-01-01, 2019-01-01,
2019-01-01, 2019-01-~
$ capacidad    <int> 993, 996, 1036, 1020, 992, 1026, 1007,
976, 1037, 972, 94~
$ demanda      <dbl> 883, 888, 922, 1134, 1103, 1139, 897,
1086, 1150, 861, 83~
$ evento_infra <dbl> 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0,
```

```
0, 0, 1, 0, 1, 1, ~
$ evento_zona  <dbl> 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0,
1, 1, 1, 1, 0, 1, ~
```

[13]:
```
tdata_02 |> slice_head(n = 5)
```

	id_inf <int>	fecha <date>	capacidad <int>	demanda <dbl>	evento_infra <dbl>	evento_zona <dbl>
	1	2019-01-01	993	883	1	1
	2	2019-01-01	996	888	0	0
	3	2019-01-01	1036	922	0	0
	4	2019-01-01	1020	1134	1	0
	5	2019-01-01	992	1103	1	1
	6	2019-01-01	1026	1139	0	1
	7	2019-01-01	1007	897	0	0
	8	2019-01-01	976	1086	1	0
	9	2019-01-01	1037	1150	1	1
	10	2019-01-01	972	861	0	1
	11	2019-01-01	940	837	1	0
	12	2019-01-01	1000	1109	1	0
	13	2019-01-01	993	881	0	0
	14	2019-01-01	1014	901	0	1
	15	2019-01-01	990	884	0	1
	16	2019-01-01	989	1094	1	1
	17	2019-01-01	993	1103	0	1
	18	2019-01-01	980	1091	1	0
	19	2019-01-01	985	1093	1	1
	20	2019-01-01	974	1084	1	0
	21	2019-01-01	1017	910	0	0
	22	2019-01-01	953	848	1	1
	23	2019-01-01	982	871	0	0
	24	2019-01-01	1020	1132	1	0
	25	2019-01-01	1042	1155	1	1
	26	2019-01-01	1054	1169	1	1
	27	2019-01-01	1004	1116	1	0
	28	2019-01-01	973	868	0	1
	29	2019-01-01	1041	929	1	1
A rowwise_df: 415370 x 6	30	2019-01-01	978	872	0	0
	...	...	...	...	...	...
	1109	2019-12-31	1026	1139	1	1
	1110	2019-12-31	1002	1111	1	0
	1111	2019-12-31	989	879	0	0
	1112	2019-12-31	973	865	0	1
	1113	2019-12-31	939	836	0	0
	1114	2019-12-31	1056	941	1	1
	1115	2019-12-31	939	1041	1	0
	1116	2019-12-31	1034	1145	0	1
	1117	2019-12-31	997	889	1	1
	1118	2019-12-31	1041	1154	1	1
	1119	2019-12-31	1013	1122	1	1
	1120	2019-12-31	999	890	0	0
	1121	2019-12-31	1060	940	0	0
	1122	2019-12-31	959	1064	1	1
	1123	2019-12-31	974	1080	1	1
	1124	2019-12-31	971	1080	1	1
	1125	2019-12-31	1037	1151	1	1
	1126	2019-12-31	963	1069	1	0
	1127	2019-12-31	1037	1152	1	1
	1128	2019-12-31	966	1071	1	0

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. Infraestructuras con su identificador único

Identificamos los datos a guardar

[14]:
```
data_to_save_01 <- tdata_01
```

Estructura de nombre de archivos:

- Código del caso de uso, por ejemplo "CU_04"
- Número del proceso que lo genera, por ejemplo "_05".
- Número de la tarea que lo genera, por ejemplo "_01"
- En caso de generarse varios ficheros en la misma tarea, llevarán _01 _02 … después
- Nombre: identificativo de "properData", por ejemplo "_zonasgeo"
- Extensión del archivo

Ejemplo: "CU_04_05_01_01_zonasgeo.json, primer fichero que se genera en la tarea 01 del proceso 05 (Data Collection) para el caso de uso 04 (vacunas)

Importante mantener los guiones bajos antes de proceso, tarea, archivo y nombre

0.7.1 Proceso 05

[15]:
```
caso <- "CU_18"
proceso <- '_05'
tarea <- "_19"
archivo <- "_01"
proper <- "_infraestructuras"
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

[16]:
```
# file_save_01 <- paste0(caso, proceso, tarea, tcltk::tkgetSaveFile(), proper,␣
↪extension)
# path_out_01 <- paste0(oPath, file_save_01)
# write_csv(data_to_save_01, path_out_01)

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

OPCION B: Especificar el nombre de archivo

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

[17]:
```
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_18_05_19_01_infraestructuras.csv'

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

[21]:
```
path_in_01 <- paste0(iPath, file_save_01)
file.copy(path_out_01, path_in_01, overwrite = TRUE)
```

TRUE

2. Diario de infraestructuras

Identificamos los datos a guardar

[22]:
```
data_to_save_02 <- 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

0.7.2 Proceso 05

[23]:
```
archivo <- "_02"
proper <- "_diario_infra"
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

[24]:
```
# file_save_02 <- paste0(caso, proceso, tarea, tcltk::tkgetSaveFile(), proper,␣
↪extension)
# path_out_02 <- paste0(oPath, file_save_02)
# write_csv(data_to_save_01, path_out_02)

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

OPCION B: Especificar el nombre de archivo

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

[25]:
```
file_save_02 <- paste0(caso, proceso, tarea, archivo, proper, extension)
path_out_02 <- paste0(oPath, file_save_02)
write_csv(data_to_save_02, path_out_02)

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

```
File saved as:
```

'Data/Output/CU_18_05_19_02_diario_infra.csv'

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

[26]:
```
path_in_02 <- paste0(iPath, file_save_02)
file.copy(path_out_02, path_in_02, overwrite = TRUE)
```

TRUE

0.8 Main Conclusions

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

0.8.1 Prerequisites

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 tcltk, 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 `tdata_01`

- Los datos de origen son los de infraestructuras pero sin indizar
- Los datos de salida son los mismos pero con el índice de fila para unir al diario

Objeto `tdata_02`

- Los datos de salida son 415370 filas de las columnas:
  - id_inf
  - fecha
  - capacidad
  - demanda
  - evento_infra
  - evento_zona

Observaciones generales sobre los datos

- Los datos han sido generados de forma aleatoria y arbitraria

0.8.4 Consideraciones para despliegue en piloto

- Es posible que los modelos no arrojen resultados esclarecedores al ser datos *fake*

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 deberían tener datos reales para la puesta en producción

0.9 Main Actions

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

- Se han simulado datos de ocupación, capacidad y vivienda

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

- Se debe añadir la información meteorológica de cada día en cada infraestructura

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.

- El fichero de infraestructuras se debe incluir en la ruta de datos de la implementación ya que será utilizado en los modelos

CODE

[27]:
```
# incluir código
```