06.- Data Adequacy_CU_55_01_gasto_comunidad_v_01

June 15, 2023

#

CU55_Modelo agregado de estimación del gasto medio por turista

Citizenlab Data Science Methodology > II - Data Processing Domain *** > # 06.- Data Adequacy

Data Adequacy is the process to adapt basic and fundamental aspects of the raw data (File Format, Data Separator, Feature Names, Tidy Data, etcetera).

0.1 Tasks

File format
- Verify/Obtain Tabular CSV (row x column) if is appropriate - Change Data Separator Character in CSV files to ","
- Verify decimal point in numeric data is "."

Feature Names
- Verify names are Simple, Usable and Recognizable
- Rename Target column name="Target" and always will be the last column - Verify column names are the first row of csv file - Remove spaces others characters in the column names

Data Types - Verify data types - Change data types - Verify that object type is changed

Tidy Data
- Verify each variable forms a column - Verify each observation forms a row
- Verify each type of observational unit forms a table

0.2 Consideraciones casos CitizenLab programados en R

- La mayoría de las tareas de este proceso se han realizado en los notebooks del proceso 05 Data Collection porque eran necesarias para las tareas ETL. En esos casos, en este notebook se referencia al notebook del proceso 05 correspondiente
- Por tanto en los notebooks de este proceso de manera general se incluyen las comprobaciones necesarias, y comentarios si procede
- Las tareas del proceso se van a aplicar solo a los archivos que forman parte del despliegue, ya que hay muchos archivos intermedios que no procede pasar por este proceso
- El nombre de archivo del notebook hace referencia al nombre de archivo del proceso 05 al que se aplica este proceso, por eso pueden no ser correlativa la numeración
- Las comprobaciones se van a realizar teniendo en cuenta que el lenguaje utilizado en el despliegue de este caso es R

0.3 File

- Input File: CU_55_05_02_gasto_municipio

- Output File: CU_55_06_03_gasto_municipio.csv

0.4 Settings

0.4.1 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.

```
In [76]: Sys.setlocale(category = "LC_ALL", locale = "es_ES.UTF-8")
```

'LC_COLLATE=es_ES.UTF-8;LC_CTYPE=es_ES.UTF-8;LC_MONETARY=es_ES.UTF-8;LC_NUMERIC=C;LC_TIME=es_ES.UTF-8'

0.4.2 Libraries to use

```
In [77]: install.packages("sf")
         library(readr)
         library(dplyr)
         library(sf)
         library(tidyr)
         library(stringr)
```

```
Warning message:
"package 'sf' is in use and will not be installed"
```

0.4.3 Paths

```
In [78]: iPath <- "Data/Input/"
         oPath <- "Data/Output/"
```

0.5 Data Load

OPCION A: Seleccionar fichero en ventana para mayor comodidad
Data load using the {tcltk} package. Ucomment the line if using this option

```
In [79]: # file_data <- tcltk::tk_choose.files(multi = FALSE)
```

OPCION B: Especificar el nombre de archivo

```
In [80]: iFile <- "CU_55_05_02_gasto_municipio.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 leerán datos del archivo:  Data/Input/CU_55_05_02_gasto_municipio.csv
```

0.6 Task

0.6.1 File format

Verify/Obtain Tabular CSV (row x column) if is appropriate Remarks

- Se ha verificado que el CSV carga las filas y columnas sin errores

CODE

```
In [81]: data <- read.csv(file_data)
         # Verify if the data is appropriate and obtain its dimensions
         if (!is.null(dim(data))) {
         cat("\nEl archivo contiene una representación tabular.")
         # Print the dimensions (rows x columns)
         cat("\nNúmero de filas: ", dim(data)[1])
         cat("\nNúmero de columnas: ", dim(data)[2])
         } else {
         warning("Cuidado: el archivo no contiene una representación tabular.")
         }
```

```
El archivo contiene una representación tabular.
Número de filas:  50294
Número de columnas:  8
```

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

```
In [82]: data <- read_csv(file_data)
```

```
Rows: 50294 Columns: 8
 Column specification
Delimiter: ","
chr (5): mes, pais_orig_cod, pais_orig, mun_dest, CMUN
dbl (3): mun_dest_cod, turistas, gasto

 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:

```
In [83]: data |> glimpse()
```

```
Rows: 50,294
Columns: 8
$ mes           <chr> "2019-07", "2019-07", "2019-07", "2019-07", "2019-07", "
$ pais_orig_cod <chr> "000", "010", "011", "030", "110", "121", "123", "126",
$ pais_orig     <chr> "Total", "Total Europa", "Total Unión Europea", "Total A
$ mun_dest_cod  <dbl> 28002, 28002, 28002, 28002, 28002, 28002, 28002, 28002,
```

```
$ mun_dest      <chr> "Ajalvir", "Ajalvir", "Ajalvir", "Ajalvir", "Ajalvir", "
$ turistas      <dbl> 338, 290, 268, 37, 56, 54, 37, 40, 157, 116, 109, 8461,
$ CMUN          <chr> "002", "002", "002", "002", "002", "002", "002", "002",
$ gasto         <dbl> 86.78, 86.78, 86.78, 86.78, 76.36, 78.92, 93.65, 102.04,
```

Muestra de los primeros datos:

```
In [84]: data |> slice_head(n = 5)
```

A spec_tbl_df: 5 Œ 8

mes <chr>	pais_orig_cod <chr>	pais_orig <chr>	mun_dest_cod <dbl>	mun_dest <chr>	turistas <dbl>
2019-07	000	Total	28002	Ajalvir	338
2019-07	010	Total Europa	28002	Ajalvir	290
2019-07	011	Total Unión Europea	28002	Ajalvir	268
2019-07	030	Total América	28002	Ajalvir	37
2019-07	110	Francia	28002	Ajalvir	56

Change Data Separator Character in CSV files to “,” Remarks

- Se ha comprobado que el separador es "," ya que es la opción por defecto de `read_csv()`

CODE

```
In [85]: #
```

Verify decimal point in numeric data is “.” Remarks

- Se ha comprobado que el símbolo decimal es "." ya que es la opción por defecto de `read_csv()`

CODE

```
In [86]: #
```

0.6.2 Feature Names

Verify names are Simple, Usable and Recognizable Remarks

- Se ha comprobado que los nombres de columnas son simples, usables y reconocibles

CODE
Visualizo el nombre de las columnas (características)

```
In [87]: colnames(data)
```

1. ’mes’ 2. ’pais_orig_cod’ 3. ’pais_orig’ 4. ’mun_dest_cod’ 5. ’mun_dest’ 6. ’turistas’ 7. ’CMUN’ 8. ’gasto’

Cambio de nombre a columnas (si es necesario) para mejor uso

```
In [88]: # colnames(data) <- c("")
```

```
In [89]: colnames(data)
```

1. ’mes’ 2. ’pais_orig_cod’ 3. ’pais_orig’ 4. ’mun_dest_cod’ 5. ’mun_dest’ 6. ’turistas’ 7. ’CMUN’ 8. ’gasto’

Rename Target column name="Target" and always will be the last column Remarks

- La columna a predecir o estimar sería turistas
- No procede ya que no es necesario para aplicar los modelos previstos

CODE

```
In [90]: data$Target <- data$gasto
```

Verify column names are the first row of csv file Remarks

- Se ha comprobado que la primera fila del archivo son los nombres de columna, ya que es la opción por defecto de `read_csv()`

CODE

```
In [91]: #
```

Remove spaces and others characters in the column names Remarks

- Se ha comprobado que los nombres de columna solo tienen caracteres ascii y no tienen espacios

CODE
Visualizo el nombre de las columnas (características)

```
In [92]: # Columns names
         colnames(data)
```

1. 'mes' 2. 'pais_orig_cod' 3. 'pais_orig' 4. 'mun_dest_cod' 5. 'mun_dest' 6. 'turistas' 7. 'CMUN' 8. 'gasto' 9. 'Target'
Cambio espacio por '_' en nombres (si procede)

```
In [93]: colnames(data) <- str_replace_all(colnames(data), " ", "_")
```

```
In [94]: colnames(data)
```

1. 'mes' 2. 'pais_orig_cod' 3. 'pais_orig' 4. 'mun_dest_cod' 5. 'mun_dest' 6. 'turistas' 7. 'CMUN' 8. 'gasto' 9. 'Target'

0.6.3 Data Types

Verify data types Remarks

- Se ha comprobado el tipo de datos adecuados al importar los datos con `read_csv()`

CODE
Visualizo el tipo de las columnas (características)

```
In [95]: sapply(data, class)
         glimpse(data)
```

mes 'character' **pais_orig_cod** 'character' **pais_orig** 'character' **mun_dest_cod** 'numeric' **mun_dest** 'character' **turistas** 'numeric' **CMUN** 'character' **gasto** 'numeric' **Target** 'numeric'

```
Rows: 50,294
Columns: 9
$ mes           <chr> "2019-07", "2019-07", "2019-07", "2019-07", "2019-07", "
$ pais_orig_cod <chr> "000", "010", "011", "030", "110", "121", "123", "126",
$ pais_orig     <chr> "Total", "Total Europa", "Total Unión Europea", "Total A
$ mun_dest_cod  <dbl> 28002, 28002, 28002, 28002, 28002, 28002, 28002, 28002,
$ mun_dest      <chr> "Ajalvir", "Ajalvir", "Ajalvir", "Ajalvir", "Ajalvir", "
$ turistas      <dbl> 338, 290, 268, 37, 56, 54, 37, 40, 157, 116, 109, 8461,
$ CMUN          <chr> "002", "002", "002", "002", "002", "002", "002", "002",
$ gasto         <dbl> 86.78, 86.78, 86.78, 86.78, 76.36, 78.92, 93.65, 102.04,
$ Target        <dbl> 86.78, 86.78, 86.78, 86.78, 76.36, 78.92, 93.65, 102.04,
```

Change data types Remarks

- No aplica

CODE

```
In [96]: # Changing column types
         # data$xx <- as.xx(data$xx)
```

Verify that object type is changed Remarks

- No aplica

CODE

```
In [97]: sapply(data, class)
         glimpse(data)
```

mes 'character' **pais_orig_cod** 'character' **pais_orig** 'character' **mun_dest_cod** 'numeric' **mun_dest** 'character' **turistas** 'numeric' **CMUN** 'character' **gasto** 'numeric' **Target** 'numeric'

```
Rows: 50,294
Columns: 9
$ mes           <chr> "2019-07", "2019-07", "2019-07", "2019-07", "2019-07", "
$ pais_orig_cod <chr> "000", "010", "011", "030", "110", "121", "123", "126",
$ pais_orig     <chr> "Total", "Total Europa", "Total Unión Europea", "Total A
$ mun_dest_cod  <dbl> 28002, 28002, 28002, 28002, 28002, 28002, 28002, 28002,
$ mun_dest      <chr> "Ajalvir", "Ajalvir", "Ajalvir", "Ajalvir", "Ajalvir", "
$ turistas      <dbl> 338, 290, 268, 37, 56, 54, 37, 40, 157, 116, 109, 8461,
$ CMUN          <chr> "002", "002", "002", "002", "002", "002", "002", "002",
$ gasto         <dbl> 86.78, 86.78, 86.78, 86.78, 76.36, 78.92, 93.65, 102.04,
$ Target        <dbl> 86.78, 86.78, 86.78, 86.78, 76.36, 78.92, 93.65, 102.04,
```

0.6.4 Tidy Data

Verify each variable forms a column Remarks

- Se ha comprobado que cada columna se refiere a una variable

CODE

```
In [98]: #
         ncol(data)
         glimpse(data)
```

9

```
Rows: 50,294
Columns: 9
$ mes           <chr> "2019-07", "2019-07", "2019-07", "2019-07", "2019-07", "
$ pais_orig_cod <chr> "000", "010", "011", "030", "110", "121", "123", "126",
$ pais_orig     <chr> "Total", "Total Europa", "Total Unión Europea", "Total A
$ mun_dest_cod  <dbl> 28002, 28002, 28002, 28002, 28002, 28002, 28002, 28002,
$ mun_dest      <chr> "Ajalvir", "Ajalvir", "Ajalvir", "Ajalvir", "Ajalvir", "
$ turistas      <dbl> 338, 290, 268, 37, 56, 54, 37, 40, 157, 116, 109, 8461,
$ CMUN          <chr> "002", "002", "002", "002", "002", "002", "002", "002",
$ gasto         <dbl> 86.78, 86.78, 86.78, 86.78, 76.36, 78.92, 93.65, 102.04,
$ Target        <dbl> 86.78, 86.78, 86.78, 86.78, 76.36, 78.92, 93.65, 102.04,
```

Verify each observation forms a row Remarks

- Se ha comprobado que cada fila se refiere a una observación

CODE

```
In [99]: #
         nrow(data)
         head(data, 1)
```

50294

A tibble: 1 Œ 9

mes <chr>	pais_orig_cod <chr>	pais_orig <chr>	mun_dest_cod <dbl>	mun_dest <chr>	turistas <dbl>	CMUN <chr>	gasto <dbl
2019-07	000	Total	28002	Ajalvir	338	002	86.78

Verify each type of observational unit forms a table Remarks

- No aplica

CODE

```
In [100]: #
```

0.6.5 Additional Actions not Initially Contemplated

No aplica

Title Remarks

- xxxxxx
- xxxxxx
- xxxxxx

CODE

```
In [101]: #
```

0.7 Data Save

- Solo si se han hecho cambios
- No aplica

Identificamos los datos a guardar

```
In [102]: 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 "_06".
- Resto del nombre del archivo de entrada
- Extensión del archivo

Ejemplo: "CU_04_06_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) y que se ha transformado en el proceso 06

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

0.7.1 Proceso 06

```
In [103]: caso <- "CU_55"
          proceso <- '_06'
          tarea <- "_03"
          archivo <- ""
          proper <- "_gasto_municipio"
          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

```
In [104]: # file_save <- paste0(caso, proceso, tarea, tcltk::tkgetSaveFile(), proper, extension)
          # path_out <- paste0(oPath, file_save)
          # write_csv(data_to_save_xxxx, 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/.

```
In [105]: 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_55_06_03_gasto_municipio.csv'

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

```
In [106]: path_in <- paste0(iPath, file_save)
          file.copy(path_out, path_in, overwrite = TRUE)
```

TRUE

0.8 REPORT

A continuación se realizará un informe de las acciones realizadas

0.9 Main Actions Carried Out

Ejemplos - Se han comprobado todas las tareas de Data Adecuacy - No se ha tenido que realizar ninguna acción adicional

0.10 Main Conclusions

- Los datos ya se habían tratado en el proceso 05 para poder hacer las tareas de ETL y no ha sido necesaria ninguna modificación

0.11 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

```
In [107]: # incluir código
```