# 05. - Data Collection_CU_55_02_gasto_municipio_v_01

June 16, 2023

#

CU55_Modelo agregado de estimación del gasto medio por turista

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 02. Asignar gasto medio por origen en tabla de turistas

- Para cada mes y país de origen, asignar el gasto medio en la Comunidad

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")
```

'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'

### 0.1.2 Packages to use

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

[2]:
```
library(readr)
library(stringr)
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
```

### 0.1.3 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. Turismo receptor

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_45_05_03_receptor.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_45_05_03_receptor.csv
```

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

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

```
Rows: 50294 Columns: 7
  Column specification



Delimiter: ","
chr (5): mes, pais_orig_cod, pais_orig, mun_dest, CMUN
dbl (2): mun_dest_cod, turistas

 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:

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

```
Rows: 50,294
Columns: 7
$ 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", …
```

Muestra de datos:

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

A spec_tbl_df: 5 × 7

| 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 |

2. Gasto medio

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_55_05_01_gasto_comunidad.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_55_05_01_gasto_comunidad.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: 680 Columns: 4
  Column specification



Delimiter: ","
dbl (4): FK_Periodo, Anyo, Valor, cod_pais

 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: 680
Columns: 4
$ FK_Periodo <dbl> 22, 21, 20, 19, 22, 21, 20, 19, 22, 21,
20, 19, 22, 21, 20,…
$ Anyo       <dbl> 2022, 2022, 2022, 2022, 2021, 2021, 2021,
2021, 2020, 2020,…
$ Valor      <dbl> 101.480, 109.300, 122.010, 97.010, 98.500,
101.140, 112.930…
$ cod_pais   <dbl> 126, 126, 126, 126, 126, 126, 126, 126,
126, 126, 126, 126,…
```

Muestra de datos:

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

A spec_tbl_df: 5 × 4

| FK_Periodo <dbl> | Anyo <dbl> | Valor <dbl> | cod_pais <dbl> |
|---|---|---|---|
| 22 | 2022 | 101.48 | 126 |
| 21 | 2022 | 109.30 | 126 |
| 20 | 2022 | 122.01 | 126 |
| 19 | 2022 | 97.01 | 126 |
| 22 | 2021 | 98.50 | 126 |

## 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:

- Número de turistas por país y municipio
- Gasto medio por turista en la comunidad de Madrid

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.

- Función para asignar el código de trimestre al mes

[14]:
```
cod_periodo <- function(x){
  if(x %in% 1:3){
    return(19)
  } else  if(x %in% 4:6){
    return(20)
  } else  if(x %in% 7:9){
```

```
    return(21)
  } else  if(x %in% 10:12){
    return(22)
  }
}
```

Asignar a cada periodo y municipio el gasto medio.

NOTA: este proceso es bastante largo

[15]:
```
inicio <- Sys.time()
tdata_01 <- sapply(seq_along(data_01$mes), function(x){
  a <- str_sub(data_01$mes[x], 1, 4)
  m <- str_sub(data_01$mes[x], -2, -1)
  p <- data_01$pais_orig_cod[x]
  d <- data_02 |> filter(Anyo == as.numeric(a),
                         FK_Periodo == cod_periodo(as.numeric(m)),
                         cod_pais == p)
  if(nrow(d) == 1){
    return(d$Valor[1])
  } else if(nrow(d) == 0){
    d0 <- data_02 |> filter(Anyo == as.numeric(a),
                            FK_Periodo == cod_periodo(as.numeric(m)),
                            is.na(cod_pais))
    return(d0$Valor[1])
  } else(stop("Más de un valor"))
})
tdata_02 <- data_01 |>
  mutate(gasto = tdata_01)
cat("tiempo: ", Sys.time() - inicio)
```

```
tiempo:  1.972325
```

### 0.4 Synthetic Data Generation

No aplica

### 0.5 Fake Data Generation

No aplica

### 0.6 Open Data

Los datos originales venían de fuentes públicas

### 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

[16]:
```
data_to_save <- 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

[17]:
```
caso <- "CU_55"
proceso <- '_05'
tarea <- "_02"
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

[18]:
```
# 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/.

[19]:
```
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_05_02_gasto_municipio.csv'

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

[20]:
```
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, stringr 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 1.1.0 * stringr 1.5.0

### 0.8.3 Data structures

**Objeto `data`**

- Hay 50294 filas con información de las siguientes variables:
  - mes
  - pais_orig_cod
  - pais_orig
  - mun_dest_cod
  - mun_dest
  - turistas
  - CMUN
  - gasto

**Observaciones generales sobre los datos**

- El dato existente es gasto medio por turista y país de origen en toda la comunidad.
- Al no haber más información, la mejor estimación que podemos dar del gasto medio en el municipio por turista es esa media

- Lo mismo para la diferencia entre meses del mismo trimestre

#### 0.8.4 Consideraciones para despliegue en piloto

- No aplica

#### 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

### 0.9 Main Actions

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

- Se han asignado a los turistas que llegan de cada país su gasto medio

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

- El archivo está listo para ser usado en los modelos

### 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

```
[ ]: # incluir código
```