05. - Data Collection_CU_45_03_receptor_v_01

June 11, 2023

#

CU45_Planificación y promoción del destino en base a los patrones en origen de los turistas

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 03. Transformar datos de entrada de turistas no nacionales

- Generar csv consolidado de datos de número de turistas por país de origen

Los datos se descargan de aquí, copiados a Input:

- Descarga directa de https://ine.es/dynt3/inebase/es/index.htm?padre=8578&capsel=8579
- Archivo: exp_tmov_receptor_mun.xlsx

Son datos de turistas no residentes por municipio de origen a partir de los datos de antenas móviles publicados en INE

Se encuentran en INEBASE pero la descarga disrecta completa no es posible por la restricción de volumen. Se puede a futuro automatizar

[1]:
```
## 52048: RECEPTOR - Número de turistas mensuales por municipio de destino,␣
 ↪desglosados por continente y país de residencia.
# t <- 52048
# groups_id <- get_tables(t, resource = "group")
# groups_id
# s <- get_tables(t, resource = "data")
# > s
# $status
# [1] "No puede mostrarse por restricciones de volumen"
```

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.

[2]:
```
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

[3]:
```
library(readr)
library(dplyr)
library(readxl)
library(stringr)
# library(tidyr)

p <- c("tcltk", "readr", "dplyr", "readxl", "stringr")
```

```
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

[4]:
```
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

[5]:
```
# file_data <- tcltk::tk_choose.files(multi = FALSE)
```

OPCION B: Especificar el nombre de archivo

[6]:
```
iFile <- "exp_tmov_receptor_mun.xlsx"
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/exp_tmov_receptor_mun.xlsx
```

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

El fichero excel tiene varias hojas con datos. Primero se obtienen los datos de las hojas y después se importan los datos de todas las hojas

[7]:
```
hr <- excel_sheets(file_data)[-1]
```

[8]:
```
data <- lapply(hr,
                    function(x){
```

```
                    read_excel(file_data, sheet = x) |> filter(prov_dest_cod
↪== "28")
                }) |>
 bind_rows()
```

Estructura de los datos:

[9]:
```
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  <chr> "28002", "28002", "28002", "28002",
"28002", "28002", "2…
$ mun_dest      <chr> "Ajalvir", "Ajalvir", "Ajalvir",
"Ajalvir", "Ajalvir", "…
$ turistas      <dbl> 338, 290, 268, 37, 56, 54, 37, 40, 157,
116, 109, 8461, …
$ prov_dest_cod <chr> "28", "28", "28", "28", "28", "28",
"28", "28", "28", "2…
$ prov_dest     <chr> "Madrid", "Madrid", "Madrid", "Madrid",
"Madrid", "Madri…
```

Muestra de datos:

[10]:
```
data |> slice_head(n = 5)
```

A tibble: 5 x 8

mes <chr>	pais_orig_cod <chr>	pais_orig <chr>	mun_dest_cod <chr>	mun_dest <chr>	turistas <dbl>	prov_ <chr>
2019-07	000	Total	28002	Ajalvir	338	28
2019-07	010	Total Europa	28002	Ajalvir	290	28
2019-07	011	Total Unión Europea	28002	Ajalvir	268	28
2019-07	030	Total América	28002	Ajalvir	37	28
2019-07	110	Francia	28002	Ajalvir	56	28

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:

- Turismo receptor por municipio 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.

Data extract

- Quitar columnas que no hacen falta y extraer el código de municipio

[11]:
```
edata <- data |>
    mutate(CMUN = str_sub(mun_dest_cod, start = 3)) |>
    select(-c(prov_dest_cod, prov_dest))
```

0.4 Synthetic Data Generation

No aplica

0.5 Fake Data Generation

No aplica

0.6 Open Data

Los datos fueron obtenidos de datos públicos del INE

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

[12]:
```
data_to_save <- edata
```

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

[13]:
```
caso <- "CU_45"
proceso <- '_05'
tarea <- "_03"
archivo <- ""
proper <- "_receptor"
```

```
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

[14]:
```
# 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/.

[15]:
```
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_45_05_03_receptor.csv’

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

[16]:
```
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, readxl, 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 * readxl 1.4.1 * 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

Observaciones generales sobre los datos

- Además de los valores de países, vienen datos totales por regiones
- En teoría serían la suma de los países que componen las regiones, pero podría no coincidir por el secreto estadístico (pocos datos en un país determinado, que sí se suman al total pero no se publican)
- Los datos disponibles en el archivo utilizado son en este rango de meses

[22]:
```
edata |> pull(mes) |> range()
```

1. '2019-07' 2. '2022-10'

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 guardado los datos de turismo receptor

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

- Se deben unir a los datos de establecimientos por municipio para 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
```