09.3.- Data Cleansing-Outliers_CU_18_20_infra_meteo_v_01

June 13, 2023

#

CU18_Infraestructuras_eventos

Citizenlab Data Science Methodology > II - Data Processing Domain *** > # 09.3.- Data Cleansing - Outliers

Data Cleaning refers to identifying and correcting (or removing) errors in the dataset that may negatively impact a predictive model, replacing, modifying, or deleting the dirty or coarse data.

0.1 Tasks

Basic operations	Text data analysis
	Delete Needless/Irrelevant/Private Columns
	Inconsistent Data.
	Expected values
	Zeroes
	Columns with a Single Value
	Columns with Very Few Values
	Columns with Low Variance
	Duplicates (rows/samples) & (columns/features) Data
Missing Values	Missing Values Identification
	Missing Values Per Sample
	Missing Values Per Feature
	Zero Missing Values
	Other Missing Values
	Null/NaN Missing Values
	Delete Missing Values
	Deleting Rows with Missing Values in Target Column
	Deleting Rows with Missing Values
	Deleting Features with some Missing Values
	Deleting Features using Rate Missing Values
	Basic Imputation
	Imputation by Previous Row Value
	Imputation by Next Row Value
	Statistical Imputation
	Selection of Imputation Strategy
	Constant Imputation
	Mean Imputation
	Median Imputation
	Most Frequent Imputation
	Interpolation Imputation
	Prediction Imputation (KNN Imputation)
	Evaluating k-hyperparmeter in KNN Imputation
	Applying KNN Imputation
	Iterative Imputation
	Evaluating Different Imputation Order
	Applying Iterative Imputation
Outliers	Outliers - Univariate
	Visualizing Outliers
	Distribution
	Box Plots
	Isolation Forest
	Outliers Identification
	Grubbs' Test
	Z-Score
	Standard Deviation Method
	Interquartile Range Method
	Tukey's method
	Internally studentized residuals AKA z-score method
	Median Absolute Deviation method
	Outliers - MultiVariate
	Visualizing Outliers
	ScatterPlots
	Outliers Identification
	Mahalanobis Distance
	Robust Mahalanobis Distance
	DBSCAN Clustering
	PyOD Library
	Automatic Detection and Removal of Outliers
	Compare Algorithms
	LocalOutlierFactor
	IsolationForest
	Minimum Covariance Determinant

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_18_09.2_20_diario_infra
- Output File: CU_18_09.3_20_diario_infra

0.3.1 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_CTYPE=es_ES.UTF-8;LC_NUMERIC=C;LC_TIME=es_ES.UTF-8;LC_COLLATE=es_ES.UTF-8;LC_MONETARY=es_ES.UTF-8;LC_MESSAGES=en_US.UTF-8;LC_PAPER=es_ES.UTF-8;LC_NAME=C;LC_ADDRESS=C;LC_TELEPHONE=C;LC_MEASUREMENT=es_ES.UTF-8;LC_IDENTIFICATION=C'

0.4 Settings

0.4.1 Libraries to use

[2]:
```
library(readr)
library(dplyr)
# library(sf)
library(tidyr)
library(stringr)
library(ggplot2)
```

```
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
```

0.4.2 Paths

[3]:
```
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

[4]:
```
# file_data <- tcltk::tk_choose.files(multi = FALSE)
```

OPCION B: Especificar el nombre de archivo

[5]:
```
iFile <- "CU_18_09.2_20_diario_infra.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_18_09.2_20_diario_infra.csv
```

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

[6]:
```
data <- read_csv(file_data)
```

```
Rows: 377816 Columns: 10
  Column specification


Delimiter: ","
dbl  (9): id_inf, capacidad, demanda, evento_infra, evento_zona, tmed,
prec,…
date (1): fecha

 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.
```

Visualizo los datos.

Estructura de los datos:

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

```
Rows: 377,816
Columns: 10
$ id_inf       <dbl> 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    <dbl> 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, …
$ tmed         <dbl> 6.953211, 6.196420, 6.483569, 5.875797,
6.212680, 5.87854…
$ prec         <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, …
$ velmedia     <dbl> 0.6433886, 0.3417523, 0.4132169,
0.1820178, 0.2118110, 0.…
$ presMax      <dbl> 952.9357, 950.2191, 950.8051, 951.6768,
953.5118, 952.168…
```

Muestra de los primeros datos:

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

A spec_tbl_df: 5 × 10

id_inf <dbl>	fecha <date>	capacidad <dbl>	demanda <dbl>	evento_infra <dbl>	evento_zona <dbl>	tmed <dbl>	p <dbl>
1	2019-01-01	993	883	1	1	6.953211	0
2	2019-01-01	996	888	0	0	6.196420	0
3	2019-01-01	1036	922	0	0	6.483569	0
4	2019-01-01	1020	1134	1	0	5.875797	0
5	2019-01-01	992	1103	1	1	6.212680	0

0.6 Outliers - Univariate

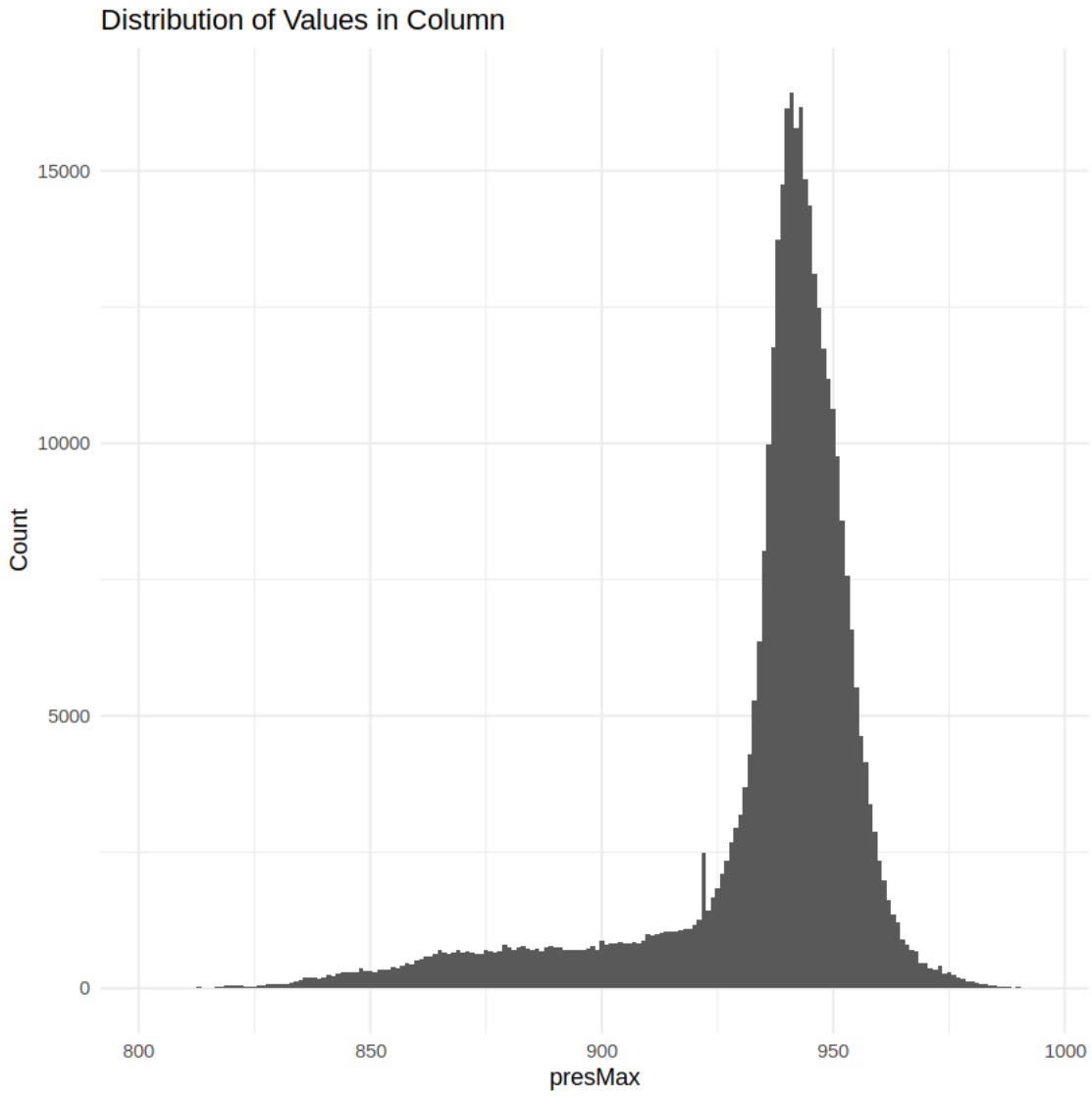
0.6.1 Visualizing Outliers

Distribution Selecting feature to analyze

[9]:
```r
# Selecting feature to analyze
column_name <- "presMax"
```

Operation

[10]:
```r
# Create a histogram plot
ggplot(data, aes(x = data[[column_name]])) +
  geom_histogram(binwidth = 1) +
  labs(x = column_name, y = "Count") +
  ggtitle("Distribution of Values in Column") +
  theme_minimal()
```



Box Plots Are great to summarize and visualize the distribution of variables easily and quickly.

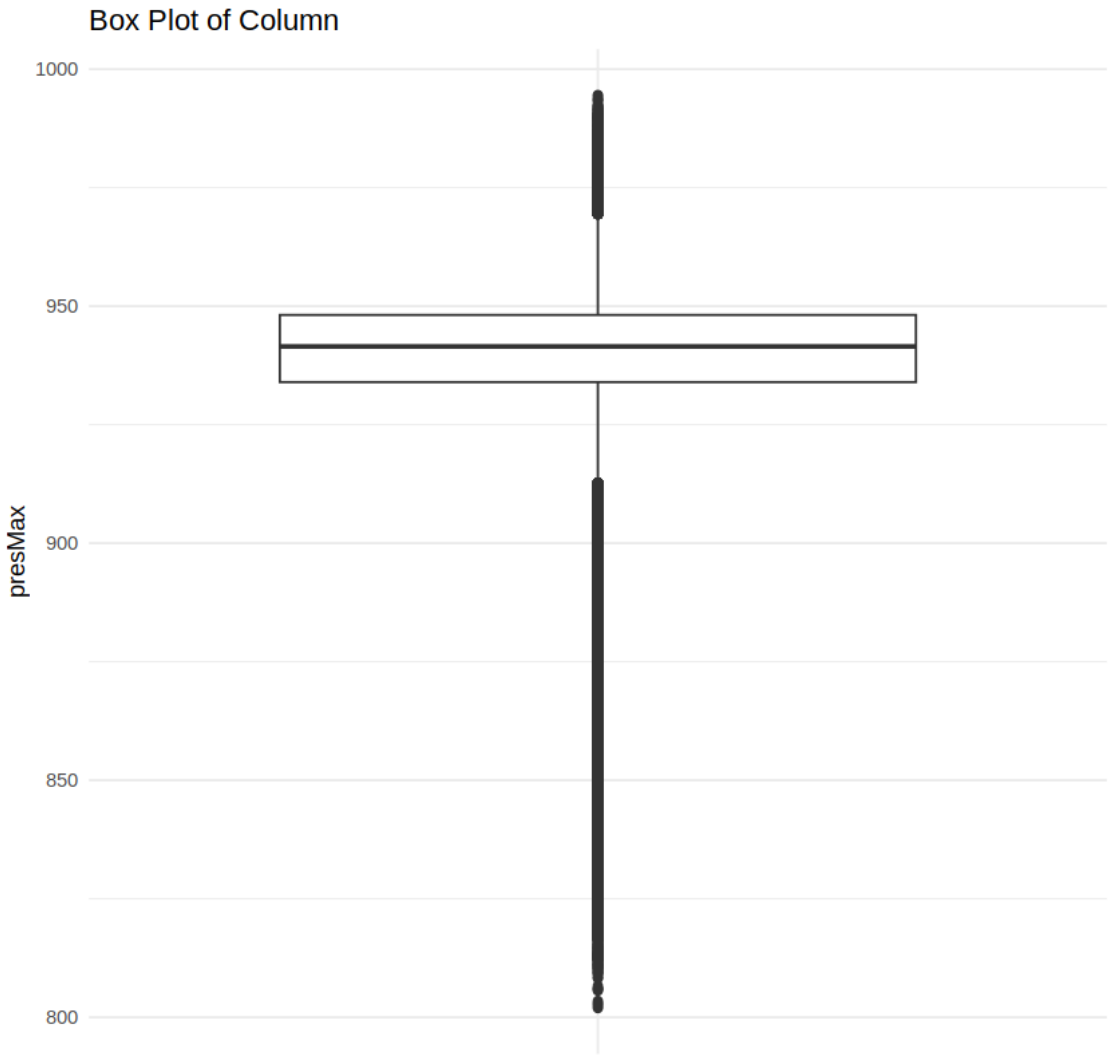
[ ]:

Selecting feature to analyze

[11]:
```r
# Selecting feature to analyze
column_name <- "presMax"
```

Operation

[12]:
```r
# Analyze specifics features
# Create a box plot
ggplot(data, aes(x = "", y = data[[column_name]])) +
  geom_boxplot() +
  labs(x = "", y = column_name) +
  ggtitle("Box Plot of Column") +
  theme_minimal()
```



Isolation Forest Selecting feature to analyze

[13]:
```
# Selecting feature to analyze
```

Operation

[ ]:
```

```

0.6.2 Outliers Identification

Grubbs' Test Selecting feature to analyze

[14]:
```
# Selecting feature to analyze
column_name <- "presMax"
```

Operation

[15]:
```
library(outliers)
outliers <- grubbs.test(data[[column_name]], opposite = FALSE)
print(outliers)
```

```
        Grubbs test for one outlier

data:  data[[column_name]]
G = 5.48406, U = 0.99992, p-value = 0.007848
alternative hypothesis: lowest value 801.911968064451 is an outlier
```

Z-Score Selecting feature to analyze

[16]:
```
# Selecting feature to analyze
column_name <- "presMax"
```

Operation

[ ]:
```

```

[18]:
```
# Define a threshold to identify an outlier.
# List of row numbers with outlier
# Choose the numeric column from your data

# Calculate the z-score
z_scores <- data %>% select(all_of(column_name)) %>% scale()
```

```
# Define a threshold for identifying outliers (e.g., z-score > 5 or z-score <␣
↪-5)
threshold <- 5

# Find the row numbers with z-scores exceeding the threshold
outlier_rows <- which(abs(z_scores) > threshold)

# Print the row numbers with outliers
print(outlier_rows)

data_cleaned <- subset(data, !(row.names(data) %in% outlier_rows))
```

```
 [1]  17176  18314  18315  19452  19453  20590  20591  22866  24004  24005
[11]  24682  25142  25820  28556  28557  29694  29695 145770 145771 146908
[21] 146909 165116 165117 166254 166255 171944 171945 173082 173083 174220
[31] 174221 305090 310780 310781 311918 311919 314194 314195 327850 327851
[41] 330126 330127 333540 333541 334678 334679 335816 335817 338092 338093
[51] 338770 339230 339231 340368 340369 341506 341507 342644 342645 343322
[61] 343782 344920 344921 346058 346059 348334 348335 349012 349472 350150
[71] 350610 350611 358576 358577 359714 359715 360392 360852 360853 361990
[81] 361991 363128 363129 364266 364267 364944 365404 365405 366082
```

Standard Deviation Method Selecting feature to analyze

[38]:
```
# Selecting all features to analyze
column_name <- "presMax"
```

Operation

[ ]:
```

```

[44]:
```
# # identify outliers with standard deviation
# # Choose the numeric column from your data
# column_name_df <- data[[column_name]]

# # Calculate the mean and standard deviation of the column
# column_mean <- mean(column_name_df)
# column_sd <- sd(column_name_df)

# # Define the threshold as a multiple of the standard deviation (e.g., 3 times␣
↪the standard deviation)
# threshold <- 3

# # Identify the outliers based on the threshold
# outlier_rows <- which(column_name_df > (column_mean + threshold * column_sd)␣
↪| column_name_df < (column_mean - threshold * column_sd))
```

```
# # Print the updated column with outliers removed
# print(outlier_rows)

# data_cleaned <- subset(data, !(row.names(data) %in% outlier_rows))
```

Interquartile Range Method Selecting factor k

[45]:
```
# Selecting factor k
column_name <- "presMax"
threshold <- 1.5
```

Operation

[48]:
```
# identify outliers with standard deviation
# Choose the numeric column from your data
# column_name_df <- data[[column_name]]

# # Calculate the first quartile (Q1) and third quartile (Q3)
# Q1 <- quantile(column_name_df, 0.25)
# Q3 <- quantile(column_name_df, 0.75)

# # Calculate the IQR (Interquartile Range)
# IQR <- Q3 - Q1

# # Identify the outliers based on the threshold
# outlier_rows <- which(column_name_df < (Q1 - threshold * IQR) |␣
 ↪column_name_df > (Q3 + threshold * IQR))
# print(outlier_rows)
# data_cleaned <- subset(data, !(row.names(data) %in% outlier_rows))
```

Tukey's method

[51]:
```
#Tukey's method
# column_name <- "presMax"

# column_name_df <- data[[column_name]]
# # Calculate the first quartile (Q1) and third quartile (Q3)
# Q1 <- quantile(column_name_df, 0.25)
# Q3 <- quantile(column_name_df, 0.75)

# # Calculate the interquartile range (IQR)
# IQR <- Q3 - Q1

# # Define the multiplier for Tukey's method (e.g., 1.5 times the IQR)
# multiplier <- 1.5

# # Calculate the lower and upper bounds for outliers
```

```
# lower_bound <- Q1 - multiplier * IQR
# upper_bound <- Q3 + multiplier * IQR

# # Identify the outliers based on the bounds
# outlier_rows <- which(column_name_df < lower_bound | column_name_df >␣
 ↪upper_bound)

# # Print the updated column with outliers removed
# print(outlier_rows)
# data_cleaned <- subset(data, !(row.names(data) %in% outlier_rows))
```

Internally studentized residuals AKA z-score method

[ ]:
```
#Internally studentized method (z-score)
```

Median Absolute Deviation method

[53]:
```
#MAD method
# column_name <- "presMax"

# column_name_df <- data[[column_name]]
# # Calculate the median absolute deviation (MAD)
# mad <- median(abs(column_name_df - median(column_name_df, na.rm = TRUE)), na.
 ↪rm = TRUE)

# # Define a threshold for identifying outliers (e.g., 3 times the MAD)
# threshold <- 3 * mad

# # Identify the outliers based on the MAD
# outliers <- which(abs(column_name_df - median(column_name_df, na.rm = TRUE))␣
 ↪> threshold)

# # Print the updated column with outliers removed
# print(outliers)
# data_cleaned <- subset(data, !(row.names(data) %in% outliers))
```

0.7 Outliers - MultiVariate

0.7.1 Visualizing Outliers

ScatterPlots: a common way to plot multivariate outliers is the scatter plot.

ScatterPlots A common way to plot multivariate outliers is the scatter plot.

[ ]:
```

```

Selecting feature to analyze

[ ]:
```
# Selecting feature to analyze
```

Operation

[ ]:
```

```

0.7.2 Outliers Identification

Mahalanobis Distance

[ ]:
```

```

Selecting feature to analyze

[ ]:
```
# Selecting features to analyze
```

Operation

[ ]:
```
# Analyze selected features
```

[ ]:
```
# Analyze all dataset
```

Robust Mahalanobis Distance

[ ]:
```
#Robust Mahalonibis Distance
```

Selecting feature to analyze

[ ]:
```
# Selecting features to analyze
```

Operation

[ ]:
```
# Analyze selected features
```

[ ]:
```
# Analyze all dataset
```

DBSCAN Clustering Selecting feature to analyze

[ ]:
```
# Selecting feature to analyze
```

Operation

[ ]:
```

```

[ ]:
```
# specify & fit model
```

[ ]:
```
# visualize outputs
```

[ ]:
```
# outliers dataframe
```

[ ]:
```
# Index of rows with outliers
```

[ ]:
```
# Outliers Dataframe
```

[ ]:
```

```

0.8 Data Save

- Solo si se han hecho cambios
- No aplica

Identificamos los datos a guardar

[22]:
```
data_to_save <- data_cleaned
```

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

0.8.1 Proceso 09.2

[21]:
```
caso <- "CU_18"
proceso <- '_09.3'
tarea <- "_20"
archivo <- ""
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

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

[23]:
```
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_18_09.3_20_diario_infra.csv'

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

[24]:
```
path_in <- paste0(iPath, file_save)
file.copy(path_out, path_in, overwrite = TRUE)
```

TRUE

0.9 REPORT

A continuación se realizará un informe de las acciones realizadas

0.10 Main Actions Carried Out

- Si eran necesarias se han realizado en el proceso 05 por cuestiones de eficiencia

0.11 Main Conclusions

- Los datos están limpios para el despliegue

0.12 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

[ ]:
```
```