09.3.- Data Cleansing-Outliers_04_19_vacunacion_completo_v_01

June 8, 2023

#

CU04_Optimización de vacunas

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_04_09.2_20_vacunacion_gripe_train_and_test_clean.csv
- Output File: No aplica

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.

[129]:
```
Sys.setlocale(category = "LC_ALL", locale = "es_ES.UTF-8")
```

```
Warning message in Sys.setlocale(category = "LC_ALL", locale = "es_ES.UTF-8"):
"OS reports request to set locale to "es_ES.UTF-8" cannot be honored"
```

”

0.4 Settings

0.4.1 Libraries to use

[130]:
```
library(readr)
library(dplyr)
library(tidyr)
library(stringr)
```

0.4.2 Paths

[131]:
```
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

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

OPCION B: Especificar el nombre de archivo

[133]:
```
iFile <- "CU_04_09.2_20_vacunacion_gripe_train_and_test_clean.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_04_09.2_20_vacunacion_gripe_train_and_test_clean.csv
```

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

[134]:
```
data <- read_csv(file_data)
```

```
Rows: 20868 Columns: 48
  Column specification



Delimiter: ","
chr  (3): GEOCODIGO, DESBDT, nombre_zona
dbl (44): ano, semana, n_vacunas, n_citas, tmed, prec, velmedia,
presMax, be…
lgl  (1): is_train

  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:

[135]:
```
data |> glimpse()
```

```
Rows: 20,868
Columns: 48
$ GEOCODIGO           <chr> "097", "128", "155", "085", "049",
"254", "264", "27…
$ DESBDT              <chr> "GALAPAGAR", "LA RIBOTA",
"MAJADAHONDA", "ENSANCHE V…
```

```
$ ano               <dbl> 2022, 2021, 2022, 2021, 2022, 2022,
2022, 2023, 2022…
$ semana            <dbl> 33, 47, 39, 46, 24, 5, 38, 1, 26,
2, 47, 18, 23, 5, …
$ n_vacunas         <dbl> 0, 451, 0, 813, 0, 250, 0, 144, 0,
282, 166, 0, 0, 1…
$ n_citas           <dbl> 0, 437, 0, 789, 0, 235, 0, 137, 0,
271, 159, 0, 0, 1…
$ tmed              <dbl> 21.768536, 6.039860, 15.436997,
9.887983, 21.108264,…
$ prec              <dbl> 0.0550769418, 1.2404689012,
0.6913641020, 0.07183897…
$ velmedia          <dbl> 2.4482484, 2.7974515, 2.7535661,
2.5478336, 3.956291…
$ presMax           <dbl> 901.1438, 936.6692, 926.6612,
952.3018, 833.8937, 89…
$ benzene           <dbl> 0.1795784, 0.3697754, 0.2254214,
0.4194085, 0.195865…
$ co                <dbl> 0.4692918, 0.3468722, 0.4797698,
0.2673996, 0.331213…
$ no                <dbl> 2.005147, 9.513899, 6.130449,
10.993518, 2.451963, 7…
$ no2               <dbl> 10.213564, 24.689603, 22.593902,
36.187953, 10.93601…
$ nox               <dbl> 13.02255, 38.42422, 31.55546,
53.19129, 13.60685, 25…
$ o3                <dbl> 88.27507, 36.57543, 58.67398,
32.54918, 77.88477, 55…
$ pm10              <dbl> 13.887308, 9.361394, 10.401526,
12.783278, 44.451891…
$ pm2.5             <dbl> 8.707578, 6.051115, 5.266344,
6.459633, 17.136398, 1…
$ so2               <dbl> 2.086115, 1.552412, 2.758390,
2.444614, 2.854909, 3.…
$ campana           <dbl> 2021.533, 2021.000, 2022.000,
2021.000, 2021.533, 20…
$ scampana          <dbl> 11.62222, 12.00000, 4.00000,
11.00000, 11.62222, 22.…
$ capacidad_zona    <dbl> 11051, 8524, 12733, 15717, 3792,
6640, 10796, 3364, …
$ prop_riesgo       <dbl> 0.14603798, 0.16062611, 0.21143809,
0.06622598, 0.20…
$ tasa_riesgo       <dbl> 0.003617039, 0.009632178,
0.005353189, 0.012969731, …
$ tasa_mayores      <dbl> 0.018360890, 0.034418204,
0.018018046, 0.026783402, …
$ poblacion_mayores <dbl> 0.13306650, 0.14633197, 0.19219091,
0.06053132, 0.18…
```

```
$ nombre_zona      <chr> "GALAPAGAR", "LA RIBOTA",
"MAJADAHONDA", "ENSANCHE V…
$ nsec             <dbl> 17, 19, 34, 28, 6, 12, 22, 11, 20,
21, 10, 12, 15, 1…
$ t3_1             <dbl> 40.03807, 39.60720, 42.19556,
34.34724, 43.62860, 41…
$ t1_1             <dbl> 44067, 34068, 51144, 62530, 15146,
26552, 43267, 134…
$ t2_1             <dbl> 0.5121733, 0.5109523, 0.5298013,
0.5077573, 0.501588…
$ t2_2             <dbl> 0.4878267, 0.4890477, 0.4701987,
0.4922427, 0.498411…
$ t4_1             <dbl> 0.17622140, 0.19623219, 0.16029496,
0.23756034, 0.14…
$ t4_2             <dbl> 0.6906908, 0.6574383, 0.6475255,
0.7018912, 0.676094…
$ t4_3             <dbl> 0.13306650, 0.14633197, 0.19219091,
0.06053132, 0.18…
$ t5_1             <dbl> 0.15387677, 0.07211496, 0.12445661,
0.12744893, 0.12…
$ t6_1             <dbl> 0.22398769, 0.11679614, 0.21183967,
0.19323644, 0.15…
$ t7_1             <dbl> 0.07342751, 0.05250060, 0.07595339,
0.04601377, 0.05…
$ t8_1             <dbl> 0.05728152, 0.03935768, 0.06703038,
0.03454148, 0.04…
$ t9_1             <dbl> 0.4408272, 0.4406703, 0.5570257,
0.4603761, 0.387025…
$ t10_1            <dbl> 0.12371972, 0.11272335, 0.08802468,
0.13945576, 0.11…
$ t11_1            <dbl> 0.5291455, 0.6094153, 0.5018791,
0.6560315, 0.515400…
$ t12_1            <dbl> 0.6040733, 0.6814646, 0.5505073,
0.7524379, 0.585228…
$ area             <dbl> 96647460.4, 1364369.5, 30837796.0,
48678625.6, 87516…
$ densidad_hab_km  <dbl> 455.95611, 24969.77491, 1658.48428,
1284.54736, 173.…
$ tuits_gripe      <dbl> 34, 280, 126, 206, 46, 144, 98, 24,
70, 508, 280, 12…
$ interes_gripe    <dbl> 11, 64, 42, 64, 21, 20, 32, 64, 20,
69, 64, 36, 26, …
$ is_train         <lgl> TRUE, TRUE, TRUE, TRUE, TRUE, TRUE,
TRUE, TRUE, TRUE…
```

Muestra de los primeros datos:

[136]:
```
data |> slice_head(n = 5)
```

A spec_tbl_df: 5 × 48

GEOCODIGO <chr>	DESBDT <chr>	ano <dbl>	semana <dbl>	n_vacunas <dbl>	n_citas <dbl>	t <
097	GALAPAGAR	2022	33	0	0	2
128	LA RIBOTA	2021	47	451	437	6
155	MAJADAHONDA	2022	39	0	0	1
085	ENSANCHE VALLECAS	2021	46	813	789	9
049	CERCEDILLA	2022	24	0	0	2

0.6 Outliers - Univariate

0.6.1 Visualizing Outliers

Distribution Selecting feature to analyze

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

Operation

[ ]:
```

```

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

[ ]:
```

```

Selecting feature to analyze

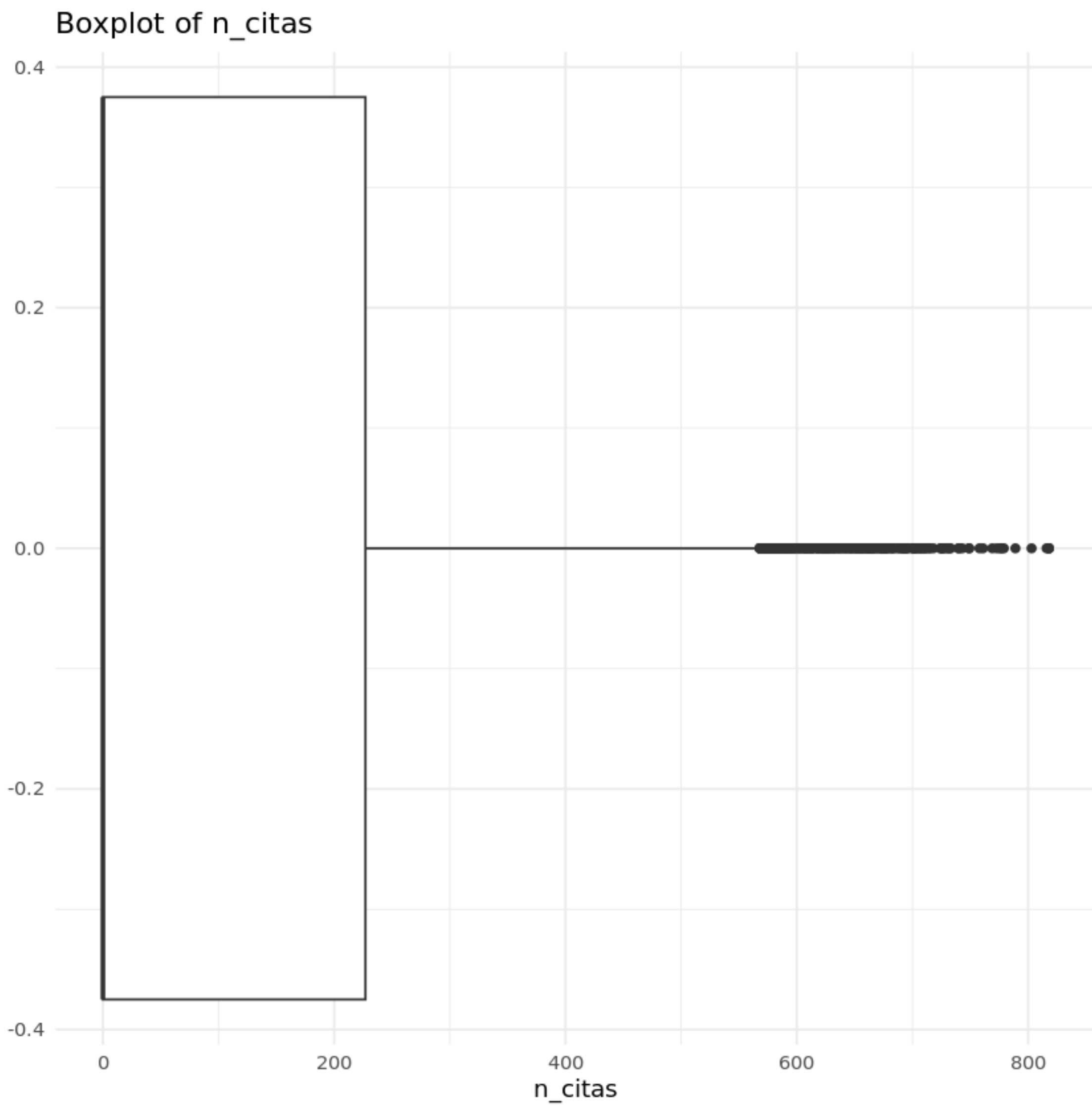
[137]:
```
# Selecting feature to analyze
column_name <- "n_citas"  # replace with your column name
```

Operation

[138]:
```
# Load the ggplot2 package
library(ggplot2)

# Visualize outliers
ggplot(data, aes_string(x = column_name)) +
  geom_boxplot() +
  theme_minimal() +
  ggtitle(paste("Boxplot of", column_name))
```

```
Warning message:
"`aes_string()` was deprecated in ggplot2 3.0.0.
 Please use tidy evaluation idioms with `aes()`.
 See also `vignette("ggplot2-in-packages")` for more information."
```



Isolation Forest Selecting feature to analyze

[142]:
```
# Selecting # Selecting feature to analyze
x <- "GEOCODIGO"  # replace with your column namefeature to plot
y <- "n_citas"  # replace with your column namefeature to plot
```

Operation

[ ]:
```
if(!require(solitude)) install.packages('solitude')
if(!require(ggplot2)) install.packages('ggplot2')
if(!require(plotly)) install.packages('plotly')

library(solitude)
```

```
library(ggplot2)
library(plotly)

model <- isolationForest$new(num_trees = 100)
model$fit(data)

anomaly_scores <- model$predict(data)
data$anomaly_scores <- anomaly_scores

# Create a scatter plot with anomaly scores as the color
plot <- ggplot(data, aes(x = x, y = y, color = anomaly_scores)) +
  geom_point() +
  scale_color_gradient(low = "blue", high = "red") +
  theme_minimal() +
  ggtitle("Isolation Forest Anomaly Scores")

# Convert to an interactive plotly plot
plotly::ggplotly(plot)
```

0.6.2 Outliers Identification

Grubbs' Test Selecting feature to analyze

[ ]:
```
# Selecting feature to analyze
column_name <- "n_citas"  # replace with your column name
```

Operation

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

Z-Score Selecting feature to analyze

[ ]:
```
# Selecting feature to analyze
column_name <- "n_citas"  # replace with your column name
```

Operation

[ ]:
```
# 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 <- scale(column_name)

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

```
threshold <- 5
```

Operation

[ ]:
```
# 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)
```

Standard Deviation Method Operation

[ ]:
```
# Selecting feature to analyze
column_name <- "n_citas"  # replace with your column name
```

[ ]:
```
# 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
outliers <- column_name_df > (column_mean + threshold * column_sd) |␣
↪column_name_df < (column_mean - threshold * column_sd)

# Remove the outliers from the column
column_name_df[!outliers] <- NA

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

Interquartile Range Method Selecting factor k

[ ]:
```
# Selecting feature to analyze
column_name <- "n_citas"  # replace with your column name

# Selecting factor k
# Define the threshold as a multiplier of the IQR (e.g., 1.5 times the IQR)
threshold <- 1.5
```

Operation

[ ]:
```
# 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
outliers <- column_name_df < (Q1 - threshold * IQR) | column_name_df > (Q3 +␣
↪threshold * IQR)

# Remove the outliers from the column
column_name_df[!outliers] <- NA

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

Tukey's method

[ ]:
```
# Selecting feature to analyze
column_name <- "n_citas"  # replace with your column name
```

[ ]:
```
#Tukey's method
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
outliers <- column_name_df < lower_bound | column_name_df > upper_bound

# Remove the outliers from the column
```

```
column_name_df[!outliers] <- NA

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

Internally studentized residuals AKA z-score method

[ ]:
```
# Selecting feature to analyze
column_name <- "n_citas"  # replace with your column name
```

[ ]:
```
#Internally studentized method (z-score)
# Calculate the z-scores for each data point
column_name_df <- data[[column_name]]
z_scores <- scale(column_name_df)

# Define a threshold for identifying outliers (e.g., z-score > 3 or z-score <␣
↪-3)
threshold <- 3

# Identify the outliers based on the z-scores
outliers <- abs(z_scores) > threshold

# Remove the outliers from the column by replacing them with NA
column_name_df[outliers] <- NA

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

Median Absolute Deviation method

[ ]:
```
# Selecting feature to analyze
column_name <- "n_citas"  # replace with your column name
```

[ ]:
```
#MAD method
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 <- abs(column_name_df - median(column_name_df, na.rm = TRUE)) >␣
↪threshold

# Remove the outliers from the column by replacing them with NA
```

```
column_name_df[outliers] <- NA

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

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
column_name <- "n_citas"  # replace with your column name
```

Operation

[ ]:
```
plot(data[[column_name]], data$Target, xlab = "Selected Variable", ylab =␣
↪"Target", main = "Scatter Plot")
```

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
column_name <- "n_citas"  # replace with your column name
```

Operation

[ ]:
```
# Select the numeric columns from the data frame
numeric_data <- data[, column_name]

# Perform DBSCAN clustering on the numeric data
dbscan_result <- dbscan(numeric_data, eps = 0.5, minPts = 5)

# Extract the cluster labels assigned by DBSCAN
cluster_labels <- dbscan_result$cluster
```

[ ]:
```
# Index of rows with outliers
# Identify the outliers as points that are not assigned to any cluster (noise␣
↪points)
outlier_indices <- which(cluster_labels == 0)
outlier_indices
```

[ ]:
```

```

0.8 Data Save

- Solo si se han hecho cambios
- No aplica

Identificamos los datos a guardar

[ ]:
```
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

0.8.1 Proceso 09.3

```
[ ]: # caso <- "CU_XX"
     # proceso <- '_09.2'
     # tarea <- "_XX"
     # archivo <- ""
     # proper <- "_xxxxx"
     # 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/.

```
[ ]: # file_save <- paste0(caso, proceso, tarea, archivo, proper, extension)
     # path_out <- paste0(oPath, file_save)
     # write_csv(data_to_save_xxxxx, path_out)

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

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

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

[ ]:
```

```