1. Calculation for each measurement (t) the period (number of months) that separates it from the previous one (t-1, obviously only starting from the second measurement in temporal order). Calculation for each measurement the total consumption of the period that separates it from the previous one (also in this case starting from the second measurement in temporal order) is included in the code

# Assuming your data frame is called dati\_ut

library(dplyr)

library(tidyr)

library(lubridate)

# Convert timestamp column to Date format

dati\_ut$data\_misura <- as.Date(dati\_ut$data\_misura, format = "%Y-%m-%d")

# 1. Calculate the time span (number of months) between consecutive measurements

dati\_ut <- dati\_ut %>%

arrange(data\_misura) %>%

mutate(time\_span\_months = c(NA, as.numeric(diff(data\_misura))),

consumption\_diff = c(NA, diff(valore\_misura)))

**consumption\_diff** is estimated as the difference in **valore\_misura** between consecutive measurements to ensure we're calculating the consumption changes correctly.

1. Calculation of average monthly consumption for all the months included in the period from t-1 to t;

# 3. Calculate the average monthly consumption for each period and each id\_utente\_ges

average\_monthly\_consumption <- dati\_ut %>%

group\_by(id\_utente\_ges, data\_misura) %>%

summarize(average\_monthly\_consumption = mean(consumption\_diff, na.rm = TRUE))

3) Creation of observation for each month included in the period from t-1 to t for which the following is indicated: the average consumption calculated in step 3; the month of consumption; the year of consumption; the handler code (id\_gestore); the user identification code (id\_utente\_ges). When creating observations for each month, we join with **average\_monthly\_consumption** to get the correct average consumption for each month.

# 3. Create observations for each month in the period from t-1 to t

observation\_data <- dati\_ut %>%

complete(data\_misura = seq(min(data\_misura), max(data\_misura), by = "month")) %>%

left\_join(average\_monthly\_consumption, by = c("data\_misura", "id\_utente\_ges"))

Once the monthly consumption database has been created for each water meter reading, the following variables contained in the user database are added to each of these records (using the id\_user\_ges variable as the linkage):

tipo\_uso ( end user typology, 1 domestic resident) x ("x" coordinates - point that originated the interruption) y ( "y" coordinates - point that originated the interruption) epsg (coordinate reference system) id\_rete (unique code of the network to which the meter refers) note (Record annotations) pro\_com (code of province given by the concentration of the province code and the three-digit municipality code) commune (name of municipality)

# Merge with other relevant data (assuming vol\_ut contains additional information)

merged\_data <- observation\_data %>%

left\_join(

vol\_ut %>%

select(

id\_utente\_ges,

tipo\_uso,

x,

y,

epsg,

id\_rete,

note,

pro\_com,

comune

),

by = "id\_utente\_ges"

)

# View the resulting data frame

View(merged\_data)

In this code:

* Finally, the resulting data is merged with additional relevant data from **vol\_ut**.

month and month\_year columns can be created from the data\_misura column using the mutate() function from the dplyr package. This code adds two new columns, month and month\_year, to the merged\_data table. The month column contains the month number extracted from the data\_misura column, and the month\_year column contains the year and month in the format YYYY-MM.

library(dplyr)

library(lubridate)

merged\_data <- merged\_data %>%

mutate(

month = month(data\_misura),

month\_year = format(data\_misura, "%Y-%m")

)

NaN values in the average\_monthly\_consumption column of the merged\_data table can be replaced with 0 using the replace\_na() function from the tidyr package. Here's how you can do it:

library(tidyr)

merged\_data <- merged\_data %>%

replace\_na(list(average\_monthly\_consumption = 0))

month and month\_year columns can be created from the data\_misura column using the mutate() function from the dplyr package for adding them to the merged\_data table. The month column contains the month number extracted from the data\_misura column, and the month\_year column contains the year and month in the format YYYY-MM

library(dplyr)

library(lubridate)

merged\_data <- merged\_data %>%

mutate(

month = month(data\_misura),

month\_year = format(data\_misura, "%Y-%m")

)

This code filters out rows where the average\_monthly\_consumption column has a value less than 0, effectively removing rows with negative values in that column.

library(dplyr)

# Filter rows with non-negative average\_monthly\_consumption

summarized\_merged\_data <- summarized\_merged\_data %>%

filter(average\_monthly\_consumption >= 0)

The code, then, runs to perform the calculation of water consumption profiles (total values per month/year) classified by the municipality, by the handler type, and by type of user. Finally, the code is used to plot histograms in R studio corresponding to the classifications of the water consumption profiles. The steps are following:

a) This code summarizes all columns in merged\_data, handling numeric columns with the sum() function and non-numeric columns by taking the first value encountered in each group. Code sums all columns for each unique value in the id\_utente\_ges column, handling both numeric and non-numeric columns. To sum all columns associated with the same id\_utente\_ges name, both numeric and non-numeric, group\_by() function is followed by summarize\_all(). The na.rm = TRUE argument ensures that any NA values are ignored during the summation process.

library(dplyr)

# Define a function to handle non-numeric columns

summarize\_non\_numeric <- function(x) {

if(is.numeric(x)) {

return(sum(x, na.rm = TRUE))

} else {

return(first(x))

}

}

# Group by id\_utente\_ges and summarize all columns

summarized\_merged\_data <- merged\_data %>%

group\_by(id\_utente\_ges) %>%

summarize(across(everything(), summarize\_non\_numeric))

b) The group\_by() and summarize() functions from the dplyr package are used to sum rows with the same value in the month\_year column and comune column in the summarized\_merged\_data table. This code groups the data by month\_year and comune, and then summarize numeric columns by summing while retaining the first value for non-numeric columns.

library(dplyr)

# Define a function to handle non-numeric columns

summarize\_non\_numeric <- function(x) {

if(is.numeric(x)) {

return(sum(x, na.rm = TRUE))

} else {

return(first(x))

}

}

# Group by month\_year and comune, then summarize numeric columns by summing

summarized\_data <- summarized\_merged\_data %>%

group\_by(month\_year, comune) %>%

summarize(across(where(is.numeric), sum, na.rm = TRUE),

across(where(is.character), ~first(.)))

# View the summarized data

View(summarized\_data)

c) This code extract the average\_monthly\_consumption associated with each month\_year for each comune in the dataset.

# Subset the summarized\_data dataframe

result\_sum <- summarized\_data[641:688, ]

d) This code plots the average\_monthly\_consumption by month\_year for each comune in the dataset.

# Plot the result\_sum dataframe

plot\_comune <- ggplot(result\_sum, aes(x = month\_year, y = average\_monthly\_consumption, fill = comune)) +

geom\_bar(stat = "identity") +

labs(title = "Water Consumption per Month/Year (Classified by Comune)",

x = "Month/Year",

y = "Total Consumption") +

theme\_minimal() +

facet\_wrap(~ comune, scales = "free\_y", ncol = 2) + # Split by comune, adjust ncol as needed

theme(axis.text.x = element\_text(angle = 45, hjust = 1)) # Rotate x-axis labels by 45 degrees

print(plot\_comune)