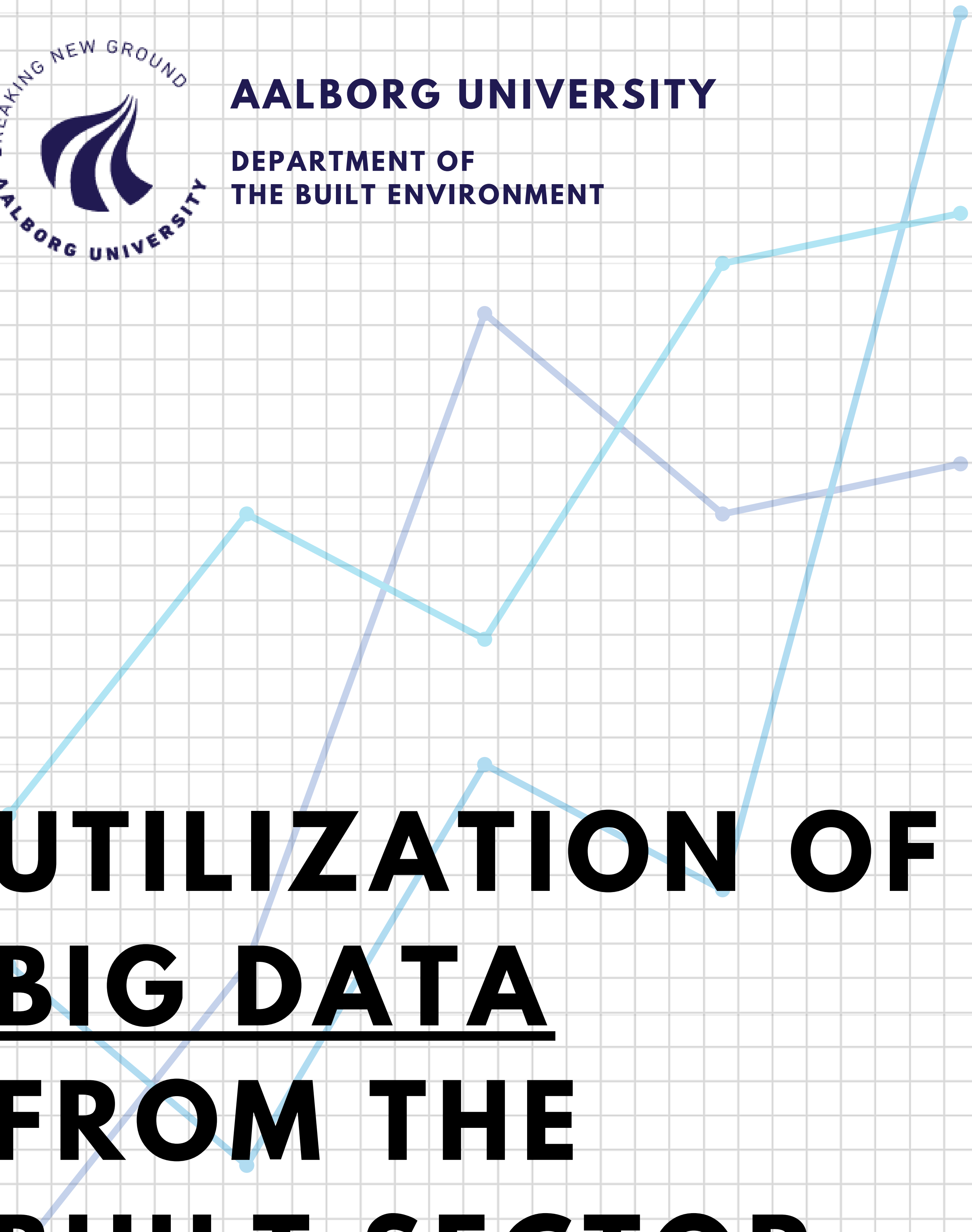
AALBORG UNIVERSITY

DEPARTMENT OF
THE BUILT ENVIRONMENT



UTILIZATION OF BIG DATA FROM THE BUILT SECTOR

MASTER THESIS

2019-2020

DANIEL LEIRIA

AALBORG UNIVERSITY
STUDENT REPORT

9th. & 10th. semesters
Indoor Environment & Energy Engineering
Byggeri og Anlæg
Thomas Manns Vej 23
9220 Aalborg Øst

Title:

Utilization of big data
from the built sector

Project:

9th. & 10th. semesters thesis

Project period:

September 2019 - June 2020

Participants:

Daniel Leiria

Supervisors:

Anna Marszal-Pomianowska
Hicham Johra
Per Heiselberg
Torben Tvedebrink

Number of pages: 159
Appendix pages: 76

Synopsis:

This master thesis is a project about the application of different data analysis methodologies in a large set of data from different smart energy meters installed in residential buildings connected to the District Heating network. The focus was to develop and use new tools to handle and study the data in order to learn more regarding the buildings energy demands and efficiency, the heating systems and the users consumption behaviours. To work with the data, it was used the software *RStudio* and this study was mainly divided into three parts: data preprocessing, data visualisation and clustering analysis.
The data preprocessing consists in a large algorithm divided in different steps, were each step assesses a specific dataset characteristic or solve a dataset problem. From it, a benchmark of imputation methods was created, revealing the most suitable methods for the DH dataset. Regarding data visualisation, different variables were calculated and different plots were made. Allowing to see different building's properties, e.g. transmission losses, solar gains, high/low temperature differences. In the clustering analysis, the K-means method was used for different variables and different datasets, where several relations between clusters of different variables was seen.

Preface

This thesis was written by Daniel Leiria as the final project from 9th. and 10th. semesters of the master program of Indoor Environmental and Energy Engineering, where the theme of this work is *Utilization of big data from the built sector.*

The project was written from the 1st. of September to the 10th. of June.

Acknowledgments

The author would like to thank the district heating company in Aalborg, *Aalborg Forsyning*, for providing the data that served to develop this thesis.

Reading guide

The references uses the Harvard method, where the name of the author and the release year is mentioned e.g. [Hansen, 2018]. All the references can be seen in the *Bibliography.* If the publishing year isn't known, the year for the last time it's visited is used.

All figures and tables are referred to as e.g. figure 1.1 or table 1.1 if it's the first figure or table in the first chapter, wheres equations are referred to as e.g. equation (1.1) if it's the first equation in the first chapter.

Daniel Leiria
<dleiri18@student.aau.dk>

Table of contents

Introduction 1

In the present century, the concern regarding the environment and the sustainability of today's choices in the future increased, where one of these concerns is related to energy usage. Currently, different projects are being carried worldwide in order to develop energy-efficient solutions based on this matter. Because according to the European Commission, the heating and cooling in buildings and industry have the share of approximately 40 % of the European's Union total energy consumption [European Commission, 2019]. Moreover, in this value, 79 % is only for heating and hot water of European households. In 2018, the Eurostat presented that 75 % of the energy used for the heating and hot water comes from fossil fuels and only 19 % is generated by renewable energy sources [European Commission, 2020]. Due to the high proportion of heating in total energy usage, more efficient and sustainable solutions must be implemented.

In Denmark, the idea of viable solutions in terms of reducing energy consumption and increasing the self-reliance in renewable energy is the basis of the new and ambitious Danish targets for 2030 and 2050 [Danish Energy Agency]. One of these solutions passes by the district heating (DH), which already covers a large area of the country to fulfil the demands of space heating and domestic hot waters (DHW) production. Therefore, a deeper understanding of all functionalities and the impact of the district heating has to meet the Danish goals regarding its network and its connected buildings is fundamental. Recently, several decisions were made for this solution, which one of them was the installation of smart energy meters in buildings that are connected to the DH network. This decision allows a better overview of the households' energy usage and the network's operation through the gathering of large amounts of data. By doing this, it grants the possibility of optimising the hot water supply, detecting present faults and boost the flexibility of the DH system for meeting the energy demands.

Due to this solution of collecting and analysing large amounts of data of building energy systems being so new and connected with other expertises, like Data Science, several investigative studies are still going in the field. These studies are based on applying different methodologies to handle the collected data and treat it, visualise patterns, cluster into groups the similar patterns and create predictive models to design energy-efficiently the District Heating network and its installations. Some of these research papers are published and served as a foundation for this thesis, which its primary goal is to find more patterns and develop new methodologies to find new information regarding District Heating data.

From this project, it was written a peer-reviewed scientific conference article that has been accepted for presentation and publication at the 12th Nordic Symposium on Building

Physics, NSB 2020, presenting the preliminary results of the data preprocessing and data clustering, and can be consulted in Appendix A.

1.1 Literature Review

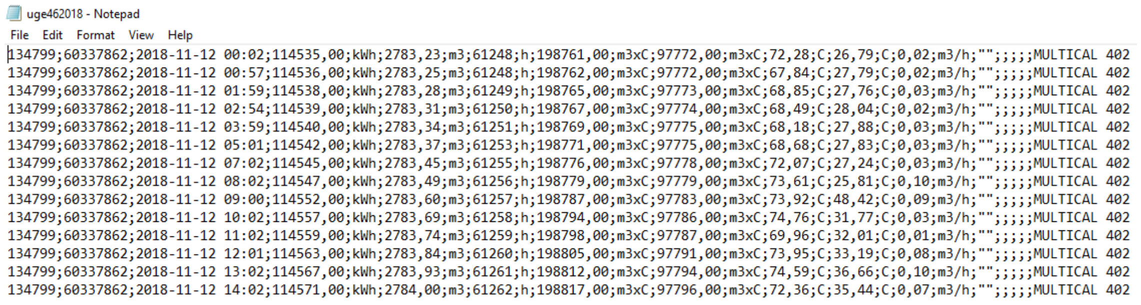
The research conducted by [Gianniou et al., 2018a] performed a clustering analysis using the K-means method in a DH dataset of 8293 single-family households in Aarhus, Denmark. This analysis allowed to group the buildings into different clusters based on their daily patterns of energy usage. It was also studied the implication of the variables: building's age, floor area and the number of adults, teenagers and children of each house in the energy usage and daily patterns of each cluster.

In the paper [Kristensen et al., 2018], it was developed a hierarchical archetype modelling and calibration framework, using 150 hourly-logged time-series data from Aarhus, Denmark. This project resulted in developing several dynamic physics-based building models which allow to design urban areas and forecast the building's energy performance when considering the several building's parameters of each archetype.

In the research [Marszal-Pomianowska et al., 2019], it was presented a simple approach to distinguish the space heating component and DHW production component from the total energy consumption measured by the DH smart meters. The approach is developed based on the assumption that in the non-heating season, most of the buildings do not use space heating. Therefore, the energy measured by the smart meters during the non-heating season is only for DHW production. The study revealed that the method predicts accurately when the ratio between DHW usage and the total heating demand is higher than 0.5 and showed as well it is more reliable for newly built houses and apartments.

1.2 Case study

The present case is from a small town in the East area of Aalborg, Denmark, where the utility company Aalborg Forsyning provided the data from the smart energy meters installed in 2018. The given data is from 1665 buildings, which mostly are residential single-family dwellings. In each building is installed a single smart energy meter which measures the total thermal energy use for space heating and production of domestic hot water (DHW), from the district heating network. In all buildings was installed one of three different models of smart meters, Multical 402 [Kamstrup A/S, 2016], Multical 403 [Kamstrup A/S, 2019] and Multical 602 [Kamstrup A/S, 2018], from the producer company Kamstrup A/S. These devices are equipped with sensors to measure temperature and flow in water-based systems with temperature ranges from 2 to 160/180 °C and have an energy calculator to measure the building's usage accurately. The initial dataset given by Aalborg Forsyning, containing the smart meter recordings, is in `.txt` format and it is separated into several files by their respective week of the year and its measurements period is one-year, from 1st of October 2018 until the 6th of October 2019. In each file, all the buildings are gathered, and each data point is an hourly timestamp measurement from a smart meter installed in a building. In figure 1.1 is represented some of the measurements from week 46 of 2018 `.txt` file:



```
134799;60337862;2018-11-12 00:02;114535,00;kWh;2783,23;m3;61248;h;198761,00;m3xC;97772,00;m3xC;72,28;C;26,79;C;0,02;m3/h;"";;;;;MULTICAL 402
134799;60337862;2018-11-12 00:57;114536,00;kWh;2783,25;m3;61248;h;198762,00;m3xC;97772,00;m3xC;67,84;C;27,79;C;0,02;m3/h;"";;;;;MULTICAL 402
134799;60337862;2018-11-12 01:59;114538,00;kWh;2783,28;m3;61249;h;198765,00;m3xC;97773,00;m3xC;68,85;C;27,76;C;0,03;m3/h;"";;;;;MULTICAL 402
134799;60337862;2018-11-12 02:54;114539,00;kWh;2783,31;m3;61250;h;198767,00;m3xC;97774,00;m3xC;68,49;C;28,04;C;0,02;m3/h;"";;;;;MULTICAL 402
134799;60337862;2018-11-12 03:59;114540,00;kWh;2783,34;m3;61251;h;198769,00;m3xC;97775,00;m3xC;68,18;C;27,88;C;0,03;m3/h;"";;;;;MULTICAL 402
134799;60337862;2018-11-12 05:01;114542,00;kWh;2783,37;m3;61253;h;198771,00;m3xC;97775,00;m3xC;68,68;C;27,83;C;0,03;m3/h;"";;;;;MULTICAL 402
134799;60337862;2018-11-12 07:02;114545,00;kWh;2783,45;m3;61255;h;198776,00;m3xC;97778,00;m3xC;72,07;C;27,24;C;0,03;m3/h;"";;;;;MULTICAL 402
134799;60337862;2018-11-12 08:02;114547,00;kWh;2783,49;m3;61256;h;198779,00;m3xC;97779,00;m3xC;73,61;C;25,81;C;0,10;m3/h;"";;;;;MULTICAL 402
134799;60337862;2018-11-12 09:00;114552,00;kWh;2783,60;m3;61257;h;198787,00;m3xC;97783,00;m3xC;73,92;C;48,42;C;0,09;m3/h;"";;;;;MULTICAL 402
134799;60337862;2018-11-12 10:02;114557,00;kWh;2783,69;m3;61258;h;198794,00;m3xC;97786,00;m3xC;74,76;C;31,77;C;0,03;m3/h;"";;;;;MULTICAL 402
134799;60337862;2018-11-12 11:02;114559,00;kWh;2783,74;m3;61259;h;198798,00;m3xC;97787,00;m3xC;69,96;C;32,01;C;0,01;m3/h;"";;;;;MULTICAL 402
134799;60337862;2018-11-12 12:01;114563,00;kWh;2783,84;m3;61260;h;198805,00;m3xC;97791,00;m3xC;73,95;C;33,19;C;0,08;m3/h;"";;;;;MULTICAL 402
134799;60337862;2018-11-12 13:02;114567,00;kWh;2783,93;m3;61261;h;198812,00;m3xC;97794,00;m3xC;74,59;C;36,66;C;0,10;m3/h;"";;;;;MULTICAL 402
134799;60337862;2018-11-12 14:02;114571,00;kWh;2784,00;m3;61262;h;198817,00;m3xC;97796,00;m3xC;72,36;C;35,44;C;0,07;m3/h;"";;;;;MULTICAL 402
```

Figure 1.1. Week 46 of 2018 file in `.txt` format

Each file has in its structure a set of parameters, known as variables, aggregated with their respective unit. The order of the parameter's disposition is listed below:

- `Link_1`: Individual identification number of a building;
- `Link_2`: Individual identification number of a smart meter;
- `TimeStep_Logging`: Time when the data was collected, with a format *Year-Month-Day Hours:Minutes*;
- `Energy`: Cumulative energy consumption measured in the building, in kWh and MWh;
- `Unit_Ener`: Unit of the energy measured at the timestamp;
- `Flow`: Cumulative water volume consumption measured in the building, in m^3;
- `Unit_Flow`: Unit of the water volume measured at the timestamp;
- `Lifetime`: Total of hours the device is in operation, in *hours*;
- `Unit_LifeT`: Unit of the lifetime measured at the timestamp;
- `Flow*T_supply`: Cumulative volume multiplied by the hourly-averaged supply temperature, in $m^3{}^\circ C$ and $0,1 \times m^3{}^\circ C$;
- `Unit_Flow&TS`: Unit of the cumulative volume multiplied by the hourly-averaged supply temperature measured at the timestamp;
- `Flow*T_return`: Cumulative volume multiplied by the hourly-averaged return temperature, in $m^3{}^\circ C$ and $0,1 \times m^3{}^\circ C$;
- `Unit_Flow&TR`: Unit of the cumulative volume multiplied by the hourly-averaged return temperature measured at the timestamp;
- `T_supply_(inst)`: Instantaneous supply temperature measured at the timestamp, in $^\circ C$;
- `Unit_TS`: Unit of the instantaneous supply temperature measured at the timestamp;
- `T_return_(inst)`: Instantaneous return temperature measured at the timestamp, in $^\circ C$;
- `Unit_TR`: Unit of the instantaneous return temperature measured at the timestamp;
- `Volume_(inst)`: Instantaneous water flow measured at the timestamp, in m^3/h;
- `Unit_Vlm`: Unit of the instantaneous water flow measured at the timestamp;
- `Error`: Type of error detected by the device during its performance;
- `SmartMeter's_Name`: Model of the smart meter.

As a remark regarding these parameters, some of the values represent a cumulative sum, like `Energy` and others are instantaneous values measured when the data is collected, e.g. `Volume_(inst)`. In the dataset, there are two different types of temperature variables measured. As an example of these measurements, `Flow*T_supply` and `T_supply_(inst)`, and not considering the cumulative volume present in the first variable, the difference between these two temperatures is that the first is an averaged temperature of the hour in which is measured and the second is the instantaneous collected value.

Another file was made available by Aalborg Forsyning, where are present the list of the house's addresses by their indicatory ID, parameter `Link_1` in the list above. This file and the specific town's location, will not be indicated in this project to assure the users' data protection.

Methodology 2

As mentioned in the Introduction chapter, the fundamental idea of this paperwork is to discover new statistical approaches and visualisation techniques to display data from the District Heating smart energy meters. The selected tool to analyse this data is the free software *RStudio*. This software is an integrated development environment (IDE) for programming with the language *R*. The language itself is designed for statistical computing and graphical display and can implement several packages that are functions made by other R-users, providing a wide range of coding functionalities, e.g. clustering analysis methods and forecasting techniques. In the current chapter, it will be presented all the methodology respectively of this project.

2.1 Data Preprocessing

The gathering of vast sets of information can open doors to understand and see relationships between the measured variables that was not seen before. The study of data belongs to the field of Data Science, but it is based on the fields of mathematics, specifically statistics. The developed methods in Data Science can be applied to see correlations between variables, sub-group datasets and classify them by their similar characteristics, develop models that can predict future conditions or variables' values. Therefore, the tools that are used in Data Science can revolutionise all other expertises when implemented.

Nevertheless, when handling large datasets, the first step to take into account is data cleansing and treatment, in order to solve or eliminate errors, outliers and missing values registered by the measurement devices. This stage is crucial in data analysis because most of the datasets have issues that must be solved. If not solved, all the adopted methodologies for the analysis and the conclusions drawn may be incorrect and not describe the reality.

The data preprocessing of the current dataset was divided into several R scripts. Each script has a different function based on the type of issue that is intended to solve or evaluate. The issues found on the DH dataset are the detected faults by the smart energy meters, the possible existence of data outliers and finally the presence of missing observations. Due to the complexity of this algorithm in solving all the data issues, it was created a flowchart, figure 2.1, where is exposed all the taken steps to prepare the District Heating dataset for analysis.

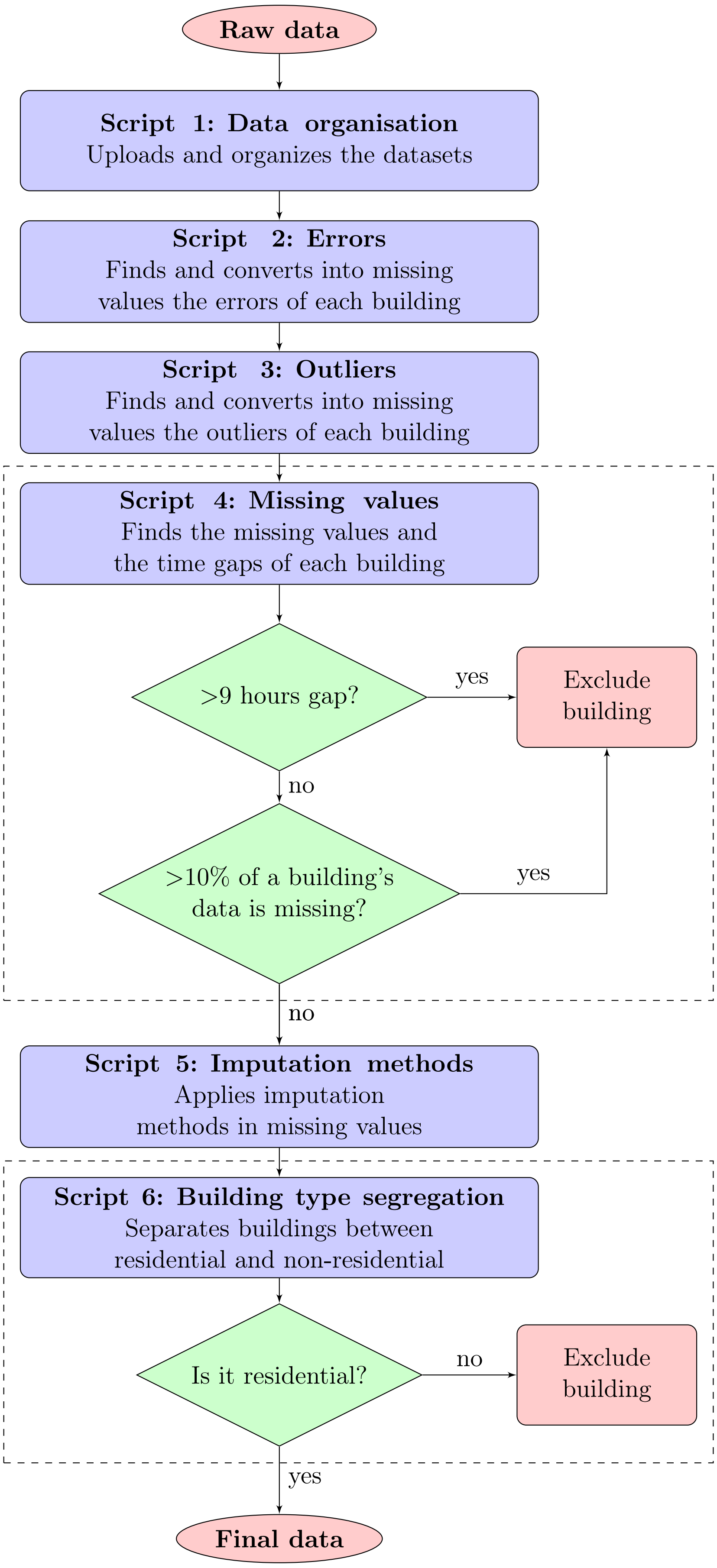

Raw data
Script 1: Data organisation
Uploads and organizes the datasets
Script 2: Errors
Finds and converts into missing values the errors of each building
Script 3: Outliers
Finds and converts into missing values the outliers of each building
Script 4: Missing values
Finds the missing values and the time gaps of each building
>9 hours gap?
yes
Exclude building
no
>10% of a building's data is missing?
yes
no
Script 5: Imputation methods
Applies imputation methods in missing values
Script 6: Building type segregation
Separates buildings between residential and non-residential
Is it residential?
no
Exclude building
yes
Final data

Figure 2.1. Data preprocessing algorithm.

2.1.1 Raw data organisation

The raw data is a term that defines the not cleansed data that comes directly from the Aalborg Forsyning that compiled all the smart meter data from each building. In this project, three different types of dataset contain the information used for this study. Aalborg Forsyning provided two of them, where the first contains the buildings and smart meters' ID with their associated addresses. The addresses information permitted to find the geolocations, latitude and longitude, of each building. The second data type is the set of .txt files named by their respective week and year of the gathered data from the DH smart energy meters, as described in section 1.2. The last data type is from Aalborg University and contains the weather data from the studied period.

In the first script, in Appendix B, all data from the different files are combined and organised in a single table and saved in a *R* file. The organisation of the dataset consists of synchronising the time samples of the measurements of each building. Because, these samples vary in a difference of minutes between buildings; therefore, the time values are rounded by the hour. Besides the time synchronisation, the data are grouped by their season of the year, as seen in the table 2.1 and day type. The day type is the distinction between weekdays, Monday to Friday, and weekends which consists of Saturdays, Sundays and official holidays in the calendar of Denmark.

Table 2.1. The year seasons and its respective time periods

Season	**Dates**	
	From	**To**
Fall	2018 - 10 - 01	2018 - 11 - 30
Winter	2018 - 12 - 01	2019 - 02 - 28
Spring	2019 - 03 - 01	2019 - 05 - 31
Summer	2019 - 06 - 01	2019 - 08 - 31
Fall	2019 - 09 - 01	2019 - 10 - 06

All the scripts in the data preprocessing finalise by saving the combined DH dataset with its modifications, to be used in the next script.

2.1.2 Errors

The second stage of the algorithm is responsible for the detection and conversion of error values into missing values. As stated in the section 1.2, one of the parameters collected by the smart meters is the variable `Errors`, which is a string value consisting in an info code description of an irregularity detected by the device. These irregularities can be a fault in one of the sensors, the detection of air in the system, weak signal, etcetera. Because these errors are recorded observations that might be wrong and influence the analysis negatively, it was created a script to handle this parameter individually. The script is called `Script_2_Errors.R` and it is in the Appendix C. The different types of errors and its descriptions may differ for each installed smart meter's model. However, some of them, even if they have a slightly different description, in the end, it might be the same type of irregularity. The script sums the number of occurrences of each error per building and evaluates its proportion regarding the overall observations' number of the building. It also eliminates each error measurement from the dataset turning this data point into a missing value that will be solved with all the other missing values posteriorly.

2.1.3 Outliers

After having the error values converted into missing values, finding the outliers of each variable's time-series of each building is the following algorithm's step. The script responsible for this task is `Script_3_Outliers.R` and it can be consulted in Appendix D. The outliers are data points in a time-series, e.g. supply temperature parameter, recorded by the smart energy meters that differ significantly from the other points. The existence of outliers can compromise the analysis due to some of the used methods are sensitive to changes in the observations, e.g. linear regression methods.

The script begins by dividing the dataset into different groups, distinguished by the building's ID number (`Link_1`), the year's season (`Season`) and the day's type (`Day_type`). This segregation of the data was made to take into account how different the values might be in different seasons or when it is a week working day or weekend.

After the division, the numerical variables, as energy, flow and temperature, were investigated by a function, to detect the outliers in the variable's time-series. The function is referred as *Mean Standard Deviation Method* in some literature, but commonly known as *Z-scores*. This method calculates the z-score of each observation by the formula 2.1:

$$Outlier_{score} = Z = \frac{X - \mu}{\sigma} \tag{2.1}$$

Where:

$Outlier_{score}$	Z-score of the observation [-]
X	Observation value - It can be Energy, Flow, Temperature [kWh or m^3/h or ºC]
μ	Mean value of the observations [kWh or m^3/h or ºC]
σ	Standard deviation of the observations [kWh or m^3/h or ºC]

The Z-score function responsible to detect the outliers in the time-series parameter `Energy` is seen in the listing 2.1 as an example:

```r
MSD_outlier_Energy <- function(x){
  mean_x <- mean(x$Energy, na.rm = TRUE)
  sd_x <- sd(x$Energy, na.rm = TRUE)
  x %>%
    mutate(
      outlier_score = (x$Energy - mean_x)/sd_x,
      outlier_energy = abs(outlier_score) > qnorm(0.9995), #If True, then it's an outlier
      outlier_energy = ifelse(is.na(outlier_energy), FALSE, outlier_energy),
      outlier_energy = ifelse(x$`Volume_(inst)` == 0, FALSE, outlier_energy)
    ) %>%
    select(-outlier_score)
  } #Mean and Standard Deviation method: Energy
```

Listing 2.1. Function to detect outliers

The value $Outlier_{score}$, in the function, is used to determine if a data point, is too distant from the expected value when it follows a normal distribution [Frost, 2020].

The used method considers an observation an outlier when $|Outlier_{score}|$ is higher than 3.291, meaning that it is a value that belongs to the area of 0.05 % of the normal distribution, as seen in the figure 2.2. The value 0.05 % is arbitrary but taking into account that higher values lead to more outliers detected.

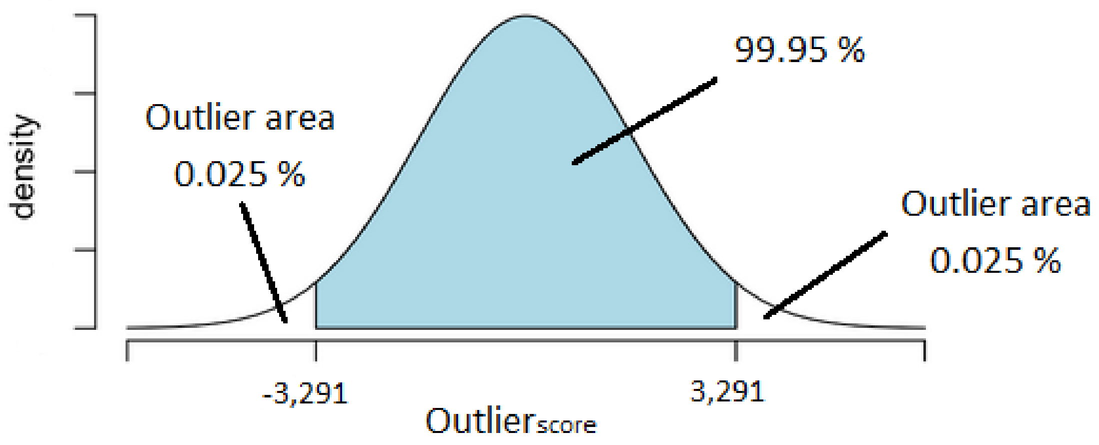


Figure 2.2. Representation of the function responsible to detect the outliers.

Nevertheless, to be considered an outlier, it must also fulfil the condition that the instantaneous flow is higher than zero. In this way, it prevents that lower values consequent of the hydraulic system being shut-off, will not be considered as outliers.

The last step of the current script is to convert all the outliers from the different variables into missing values.

2.1.4 Missing values

The fourth script in the data preprocessing is responsible for searching and dealing with the missing values. The missing values, also known as `NA` values in *R*, can be the converted values from the previous scripts or the existing ones from the smart energy meters recordings. The script `Script_4_NA.R` is the one responsible for handling the missing values, and it can be consulted in Appendix E.

The script starts by detecting the `NA` values. These `NA` values represent the missing hourly observations and can be a singular point in the time-series or belonging in a group of sequential `NA` values, denominated as gaps. The presence of significant gaps in the dataset influence the analysis' accuracy, therefore it is established that buildings with more than nine hours gaps, will be excluded from the dataset. It is also a condition to exclude a building, if the building has a variable's time-series with more than 10 % of data missing. The buildings that are not rejected will have their `NA` values solved by imputation methods, that estimate a plausible value to replace the empty observation.

2.1.5 Imputation methods

In order to solve the remaining `NA` values, it must be found a methodology that estimates credible values to substitute them. These techniques are known as imputation methods, and there are several of them that can be used to perform the values' estimation. Therefore in this algorithm's step, it was performed beforehand a test to evaluate from a list of different methods, which are the ones more suitable for this kind of data. By knowing the results from the test, the best methods were applied to all the dataset.

The list of different imputation methods, that will be tested, are from the *R* package *imputeTS* [Moritz and Gatscha, 2019] and the script responsible to evaluate the methods can be consulted in Appendix M. The final estimation of the missing values in the DH dataset by the chosen methods is performed by the script `Script_5_Imputation.R` and it is in Appendix F.

The used *R* package, *imputeTS*, is a compilation of imputation techniques and plotting functions for univariate time-series to handle missing values. From this package, the following imputation methods were compared to find the most suitable for the District Heating dataset.

Linear interpolation

This method identifies the missing observation by assuming that it is between two observations following a linear function. By recognising that the variable x is the variable time (t), then the missing variable is given by the equation 2.2:

$$y_i(t_i) = y_{i-1} + \frac{y_{i+1} - y_{i-1}}{t_{i+1} - t_{i-1}}(t_i - t_{i-1}) \tag{2.2}$$

Where:

$y_i(t_i)$	Missing observation in the period t_i
y_{i-1}, y_{i+1}	Observations where the missing value is in between
t_{i-1}, t_{i+1}	Time periods where the value t_i is in between

Spline interpolation

This technique is founded on the mathematical concept of Splines. By definition, a spline is a polynomial function divided by known points, also known as knots. In each pair of knots, the connecting spline can be a different function depending on the fitting of the curve in the studied interval. In the package *imputeTS*, the spline algorithm is referred to as cubic spline interpolation, which means that third-order polynomials connect the knots of the given dataset.

Stineman interpolation

In this method, it is used a monotone cubic interpolation to replace the missing values into values that will conserve the monotonicity of the evolution of the dataset.

Repeating with the last or next observation

Regarding this method, it can be divided into two processes. The literature calls the first **Last Observation Carried Forward**, **LOCF**, and returns each missing value as the prior value from the time-series. And the other process is called **Next Observation Carried Backwards**, **NOCB**, and consists in the reverse method of LOCF, where the closest next value from the time-series replaces the missing value. Concerning these methods, two notes must be taken into account. The first is that the nearest known point will replace the missing value from the time-series, which implies that datasets with significant missing point's gaps decrease the accuracy of this method. The second note is that if the first or last point in the time-series is a `NA` value; then the chosen technique might not work, due to the inexistence of point before the first or after the last. To overcome the issue, the method was programmed with the option of reverting from one method to another in the first and last points, if necessary. This means that the first value will be replaced by the next observation when applied LOCF. And the last value will be replaced by the observation before when used the NOCB method.

Simple moving average

This arithmetic technique is one of the most used in data science for smoothing time-series, and it consists of averaging the values within a pre-established range of points in the time-series. This range, or window-size, must be supplied beforehand, which in the script is presented by the variable k. To understand how the accuracy of this method changes with the window-size. The algorithm was tested for the k values of 2, 4, 6 and 8.

Linear weighted moving average

Relatively to this method, it is very similar to Simple Moving Average, SMA, with the particularity that the furthest points from the given window-size have a decreased weight in the average. This weighted drop has a linear evolution were the points right next to the missing value have a 1/2 influence, followed by next points with 1/3 weight continuing in this progression until the end of the window. The algorithm was tested for the k values of 2, 4, 6 and 8.

Exponential weighted moving average

Ultimately, this method also belongs to the group of the "moving average" techniques. It has the same assumption as the Linear Weighted Moving Average, LWMA, which is the reduction of the influence of the values when they are closer to the limits of the given window-size. But in this case, the weight of the observations changes exponentially, meaning that the influence of the points next to the missing value is $1/2^1$, the weight of the next points will be $1/2^2$ continuing like this until the end of the given range. The algorithm was tested for the k values of 2, 4, 6 and 8.

Replacing by mean, median, mode and random value

In these processes, the missing value is replaced by the following values: mean value of the dataset, the median, the mode or by a random number that is limited between the minimum and maximum of the existing observations in the time-series.

The developed process to compare the different methods consists in using ten of the buildings that have the lowest quantities of missing values. Each building has randomly eliminated 500 data points that will be imputed by the various methods and compared afterwards with the known values. The variations between the known points and the imputed ones are measured by the root mean square error, RMSE, where the lowest RMSE results represent the best solutions. The variables that were investigated by randomly removing their data points are the cumulative parameters of energy, volume, volume multiplied by the hourly-averaged supply temperature and volume multiplied by the hourly-averaged return temperature. And the instantaneous hourly parameters of supply temperature, return temperature and flow rate. After determining the best techniques to handle the missing data, the chosen methods are used in each variable's time-series of each building in DH dataset.

Building type segregation

To conclude the preprocessing of the DH dataset, it was created a simple script capable of differentiating from all the studied buildings the residential ones from the non-residential. The algorithm is based on the idea that a non-residential building has higher energetic properties than a residential dwelling. Therefore, when a building has high cumulative energy and volume and high instantaneous mean power, it is identified as an outlier when compared with the rest of the data. Instead of using the same methodology in subsection 2.1.3, to find the outliers in time-series. A simplified algorithm was developed in the script `Script_6_Data_Segregation.R`, in Appendix G. Through the equation 2.3, it is calculated a limit value for each variable, and the buildings that have all its three variables below the limit are considered residential dwellings.

$$X_{limit} = Q_3(X) + 1.5 \times IQR(X) \tag{2.3}$$

Where:

X	Considered variable
$Q_3(X)$	Third quartile of X
$IQR(X)$	Interquartile range of X

This method is usual in Statistics to apply in a dataset to find its outliers, where the value 1.5 is a typical arbitrary constant used in the above equation. Due to its extensive usage, its simplicity in coding in *R* and this particular dataset is not constituted by time-series, but each building has a single value per variable, this method was selected to handle the division between the buildings' types.

After concluding all the steps described above, the data is ready for analysis.

2.2 Data Visualisation

In this section, it will be presented how it was calculated new variables to enrich the dataset for the further analysis and the different visualisation techniques used to gain a more in-depth knowledge of how the different variables are related.

2.2.1 New variables

In section 1.2 in the Introduction chapter, it is shown the several parameters recorded by the DH smart energy meters. Each parameter has it is own significance of the dataset, but from them, it is possible to calculate other variables that may provide more information regarding the buildings and consumers' behaviour. These new variables are listed below:

Hourly-averaged temperatures and ΔT

The smart meters' dataset has the variable `Flow` which is the cumulative water volume consumption from the District Heating in m^3 and the variables `Flow*T_supply` and `Flow*T_return` which are the cumulative water volume consumption multiplied by the hourly-averaged temperatures of supplied and returned water in m^3°C. Therefore, by using the equation 2.4, it is possible to calculate the hourly-averaged temperatures in each data point:

$$\overline{T} = \frac{Volume \times T}{Volume} \tag{2.4}$$

Where:

$\overline{T}$	Hourly-averaged temperature [°C]
$Volume \times T$	Cumulative volume multiplied by the hourly-averaged temperature [m^3 × °C]
$Volume$	Cumulative volume [m^3]

By having these new values, it is possible to compare them with the existing parameters of instantaneous temperatures `T_supply_(inst)` and `T_return_(inst)` and calculating through the equations 2.5 and 2.6, one of the essential indicators of the buildings' energy performance, the temperature difference, known as ΔT in °C.

$$\Delta T_{inst} = T_{supply} - T_{return} \tag{2.5}$$

$$\overline{\Delta T} = \overline{T}_{supply} - \overline{T}_{return} \tag{2.6}$$

Where:

$\overline{T}_{supply}$, $\overline{T}_{return}$ and $\overline{\Delta T}$	Hourly-averaged temperatures [°C]
T_{supply}, T_{return} and ΔT_{inst}	Instantaneous temperatures [°C]

With these new variables, it is possible to assess how high is the temperature's difference in each building and therefore providing to the District Heating utility company which buildings have low ΔT, indicating that a building might have an inefficient heating system due to several causes.

Hourly-energy usage and water-volume consumption

In the dataset, it is also given the cumulative energy usage and water-volume consumption of each building. The names `Energy` and `Flow` represent these variables and the units kWh and m^3 express them. In order to obtain the hourly-consumptions, each hourly cumulated measurement is subtracted by the previous hourly cumulated measurement, as seen in equations 2.7 and 2.8:

$$E_{hourly} = E_i - E_{i-1} \tag{2.7}$$

$$V_{hourly} = V_i - V_{i-1} \tag{2.8}$$

Where:

E_{hourly} and V_{hourly}	Hourly-averaged variables [kWh and m^3]
E_i and V_i	Measurement [kWh and m^3]
E_{i-1} and V_{i-1}	Previous measurement [kWh and m^3]

By having these variables, it is possible to evaluate the hourly consumption patterns from each consumer, extracting the daily profiles from the buildings and correlate them with the ΔT values.

Power

Another parameter that is estimated through the initial parameters is power. This parameter expresses the variation of energy usage through time, therefore is an instantaneous value. It is calculated by equation 2.9 using the parameters `Volume_(inst)` which is the water flow measured in that instant and the instantaneous ΔT, mentioned above.

$$Power = \frac{c_p \cdot \rho \cdot \dot{V} \cdot \Delta T_{inst}}{3600} \tag{2.9}$$

Where:

$Power$	Power [kW]
c_p	Water isobaric specific heat [kJ/kg · K]
ρ	Water density [kg/m^3]
$\dot{V}$	Water flow [m^3/h]
ΔT_{inst}	Instantaneous ΔT [°C]

The value $c_p \cdot \rho/3600$ is considered a constant and equal to 1.145, for all observations for simplicity, even though both fluid properties are dependent on temperature. By having calculated power, it is possible to see the dependence of the water flow and instantaneous ΔT have on the overall buildings' performance.

Domestic hot water

By following the methodology from [Marszal-Pomianowska et al., 2019], it can be affirmed with some assurance that in the summer period, the ratio between DHW production and space heating usage is higher. This idea prevails since in the summer, most of the heating systems are turned off, and the DHW usage patterns remain. Therefore for each building was extracted, its daily total water consumption which gives an idea of the amount of DHW is consumed. This value was calculated by taking into account the months of June and August. Where in each day, from Monday to Friday, it was summed the water consumed during a day and finally averaged considering all the days from the two months period. Furthermore, it was extracted the $\overline{\Delta T}$ of the summer period, because it can indicate the DHW heat exchanger's efficiency.

Building's characteristics

When considering the values of energy usage and outdoor conditions, such as temperature, wind speed and solar radiation, for a given hour, it is possible to draw some of the conclusions respecting the buildings' characteristics and users' behaviour.

The building's energy demand consists of an energetic balance between the energy losses from ventilation, infiltration and transmission and the energy gains from solar radiation and heat generation from equipment or people's presence in the building. This interaction of the different energetic components is seen in figure 2.3 and can be mathematically expressed by the equation 2.10.

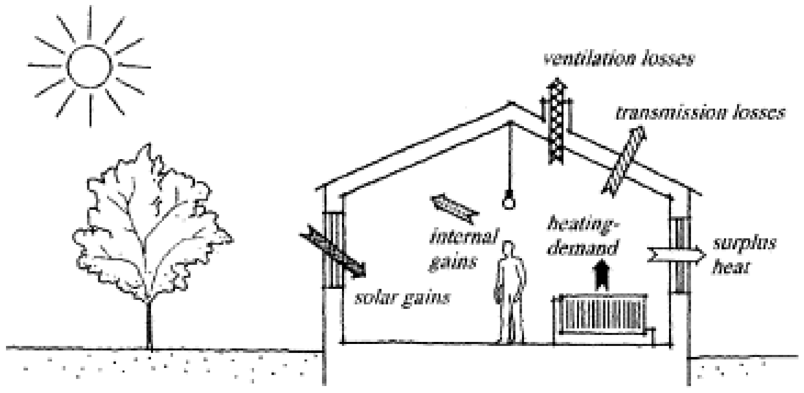


Figure 2.3. Scheme of a building's energy balance [Wittchen and Aggerholm, 2000]

Energy demand:

$$E_{demand} = E_{solar} + E_{internal} - E_{transmission} - E_{infiltration/ventilation} \qquad (2.10)$$

Where:

E_{demand}	Energy demand or balance [kWh]
E_{solar}	Solar gains [kWh]
$E_{internal}$	Internal gains [kWh]
$E_{transmission}$	Transmission losses [kWh]
$E_{infiltration/ventilation}$	Infiltration/ventilation losses [kWh]

By having the values of hourly energy, see page 13, and their respective outdoor temperature, it is possible to make a scatterplot for each building as seen in figure 2.4.

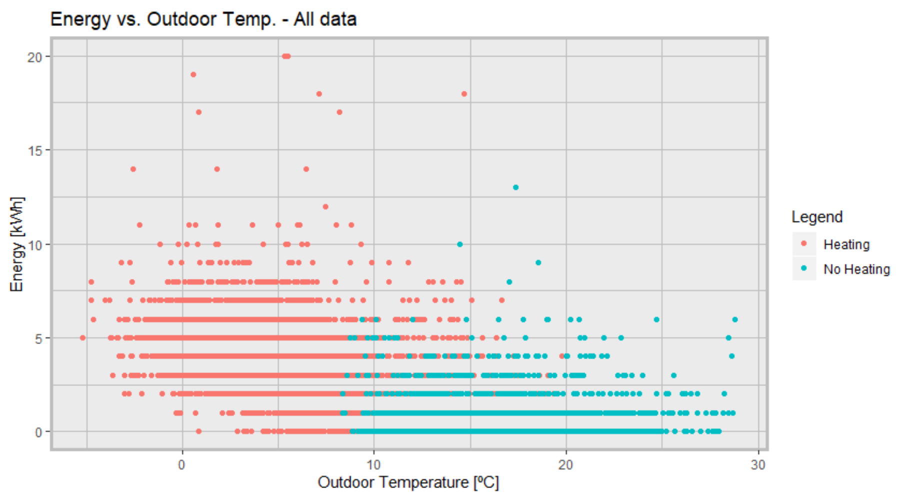


Figure 2.4. Scatterplot: Energy vs Outdoor temp. - All data

By analysing the plot itself, does not seem that there is any clear correlation between the variables, due to the widely points' spreading. The explanation behind might be due to the noise from the time-series, as seen in figure 2.5. To solve the issue, it was calculated the moving average of the time-series, in order to smooth the dataset and clarify the value's trend. The figure 2.6, shows the same time-series when applied the simple moving average with a window-size of 33 data points. The figure 2.7 is the same scatterplot as in figure 2.4, where the dataset used is the values generated by the moving average method, seen in figure 2.6.

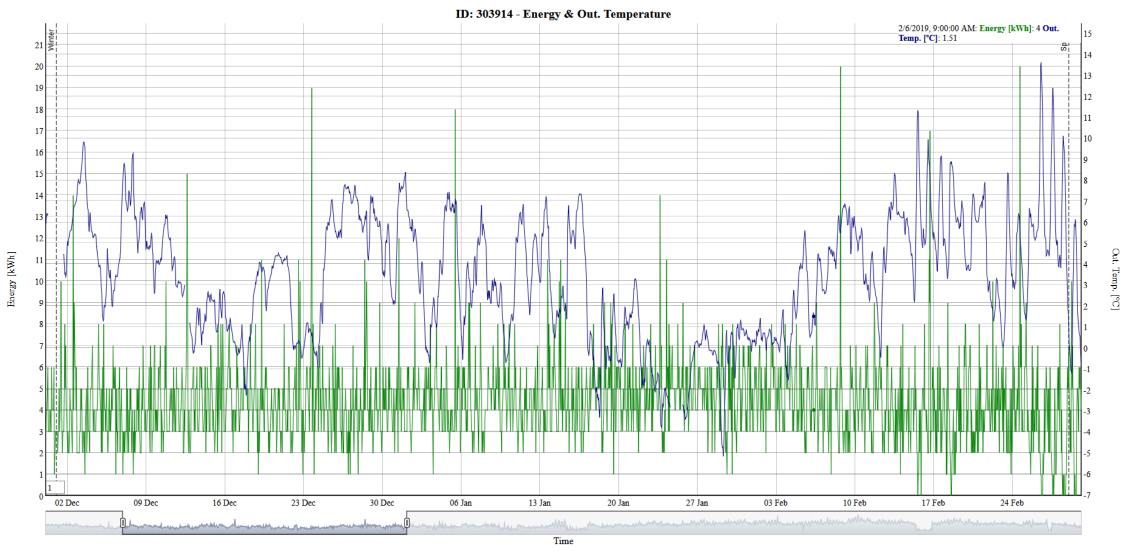


Figure 2.5. Time-series: Energy (Left - Green) vs Outdoor temp. (Right - Blue) - All data

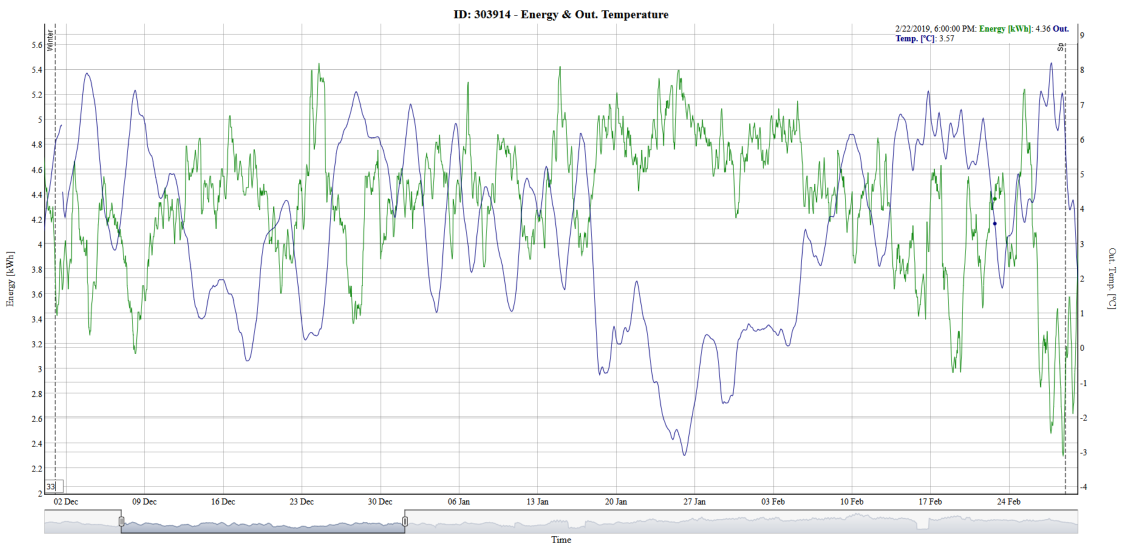


Figure 2.6. Time-series: Energy (Left - Green) vs Outdoor temp. (Right - Blue) - SMA, k = 33

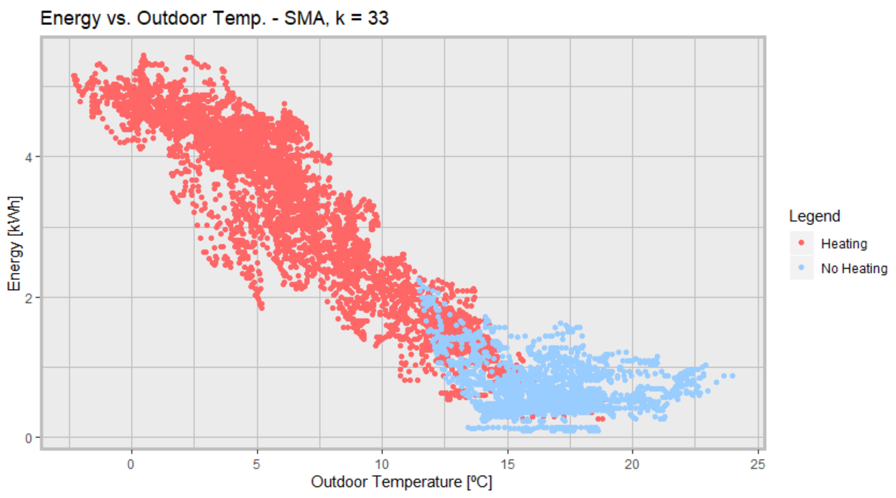


Figure 2.7. Scatterplot: Energy vs Outdoor temp. - Applied simple moving average algorithm (SMA) of k = 33

When using the smoothed values in the scatterplot of hourly-energy demand and outdoor temperature, a clear relationship between these two variables is observed. The plot shows two different linear behaviours, where the first (left side - red colour) represents the heating season, where the energy is highly dependent on the outside temperature, due to the heating system is operating. And the second (right side - blue colour) concerns the non-heating season, where the energy demand is not so dependent on the outside temperature, and more dependent on the domestic hot water production. From these two linear relations, it was performed a linear regression. Where the data points used for the heating season regression are from the Fall, Winter and Spring periods and for the non-heating season regression, only the Summer period observations. These linear regressions are expressed by the equations:

Heating season:

$$E(t_{out}) = m \times t_{out} + E(t_{out} = 0) \tag{2.11}$$

Non-heating season:

$$E(t_{out}) \approx E(t_{out} = t_{no\,influence}) \tag{2.12}$$

Where:

$E(t_{out})$	Hourly energy [kWh]
t_{out}	Outdoor temperature [°C]
m	Linear slope [kWh/°C]

The values $E(t_{out} = 0)$ and $E(t_{out} = t_{no\,influence})$ are the hourly energy when the outdoor temperature is equal to zero and equal to a temperature value when from that moment on, the energy is no longer significantly influenced by the outdoor temperature.

The value m is a variable that is dependent on the different energy components enumerated in the building's energy balance, see equation 2.10. Therefore, it is dependent on the infiltration/ventilation rates, the solar gains as the internal gains. The last one, which is based on the heat generated from the people and equipment cannot be measured or estimated from the dataset. However, the other components can be evaluated because they are related to the wind speed and solar radiation, which are given by the weather file.

To understand how significant each energy component is in the total balance of the building. The dataset is subdivided, taking into account specific external conditions. This subdivision is a filter applied in the dataset, where the influence of only one or two of the energy components can be extracted. Furthermore, from that filtered points, it is generated different linear regressions which will have different values of m. The different filters are the following:

- Heating and Non-heating regressions: Filtered by the seasons of the year;
- Temperature regression: Solar radiation is zero, and the wind speed is lower than 1 m/s;
- Infiltration/Ventilation regression: Solar radiation is zero and the wind speed is higher than 1 m/s;
- Radiation regression: Solar radiation is not zero, and the wind speed is lower than 1 m/s;

The first case reveals the overall relationship between energy and outdoor temperature for the heating and non-heating seasons. The second case, by having the data points where there is no solar radiation, night period, and the wind speed is lower than 1 m/s, then it is possible to see how the slope changes only influenced by the outdoor temperature, discovering, therefore, the influence of the transmission losses in the building. The third case reveals how the building reacts with outdoor temperature and wind speed, showing the infiltration/ventilation energy component influence. The last case considers the building's energy demand relatively the outdoor temperature and solar radiation fluctuations throughout the day, showing consequently how significant is the solar gains component in the building's heat balance. In all these three last cases, the linear regressions are calculated only for the heating season. Unfortunately, the m-value is dependent on the building's envelope area, which is information not provided in the dataset. However, to overcome the issue, it was compared the different regression's slopes of the building to each other, by calculating the following ratios:

Infiltration and Ventilation:

$$R_{inf} = \frac{m_{inf}}{m_{temp}} \tag{2.13}$$

Solar gains:

$$R_{rad} = \frac{m_{rad}}{m_{temp}} \tag{2.14}$$

Where:

R_{inf} and R_{rad}	Ratios of the energy components [-]
m_{temp}	Slope of temperature regression [kWh/°C]
m_{inf}	Slope of infiltration/ventilation regression [kWh/°C]
m_{rad}	Slope of radiation regression [kWh/°C]

By comparing them, it is possible to find how influential each outdoor condition is in a building. In figure 2.8, it is presented, as an example, a temperature regression with two different cases of a radiation regression. The first case is when the radiation's slope, m_{rad}, is lower than the temperature's slope, m_{temp}. And the second case is when the radiation's slope is higher than the temperature's slope.

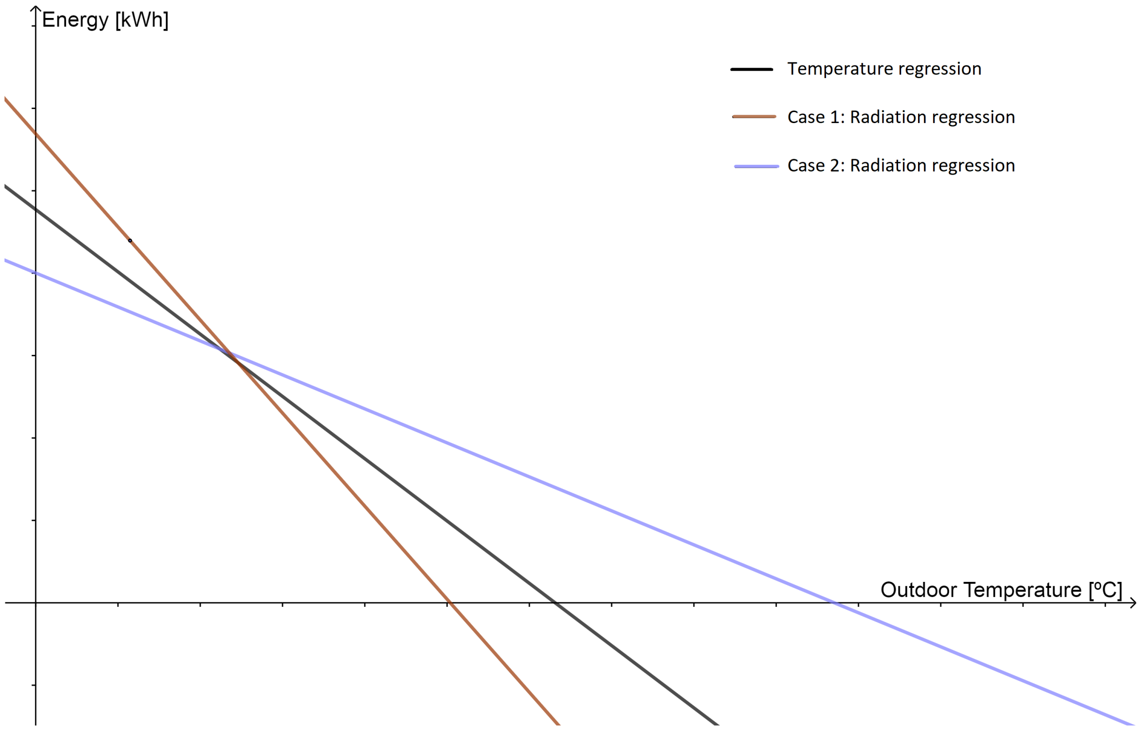


Figure 2.8. Example of the m-value changes

When m_{rad} is lower than m_{temp} (Case 1), it means that the energy demand is more dependent on the outdoor temperature. In the second case, where m_{rad} is higher than m_{temp}, the linear function is less dependent on outdoor temperature. Because the origins of the dataset responsible for the radiation regression is based only on the outdoor temperature and solar radiation, a ratio $R_{rad} < 1$ shows a building where the dataset is influenced by the solar radiation. And the lower R_{rad} is, higher is the influence of the solar radiation on the building. Based in this concept, it is defined the condition 2.15 which will be defined as a building indicator:

$$\begin{cases} R_{rad} < 1, \text{ High dependency of radiation} \\ R_{rad} \geq 1, \text{ Low dependency of radiation} \end{cases} \tag{2.15}$$

In the case of the infiltration/ventilation component, the higher the R_{inf}-value is the more influence the wind speed has on a building. Due to when the outside air enters in the space, it is at temperature of the outside. Therefore the R_{inf} is expressed by the relation 2.16 as a building indicator:

$$\begin{cases} R_{inf} < 1, \text{ Low dependency of wind} \\ R_{inf} \geq 1, \text{ High dependency of wind} \end{cases} \tag{2.16}$$

Regarding the temperature regression, the slope, m_{temp}, itself is the indicator used to measure the influence of the transmission losses in the building's energy demand even though this value is also dependent on the envelope's surface area.

By having these indicators calculated, several conclusions can be drawn from the dataset. The buildings that have large m_{temp}-values might have high transmission losses through their envelope, which might mean that they are old buildings with high transmission losses coefficients, U-value. High R_{inf} means the heating demands change proportionally with the increase of wind speed, which might mean that the building has high infiltration or ventilation rates. This can be a building, therefore, exposed significantly to the wind or that has significant infiltration/ventilation rates due to the building's construction, ventilation system or the users' behaviour in constantly opening the windows with the heating system operating. The ratio R_{rad}, when it is lower than 1, represents a building that solar radiation exposure influences its heating demands. This might mean that it is a building with large glazing areas, or that most of the windows are facing South, or that the users have their curtains opened most of the time. All the regressions were calculated by the script `Calculate_Uvalue.R` and it is in Appendix H.

2.2.2 Plotting

In this subsection, it clarifies which types of plots where used and the reason why they were chosen. The code responsible for this subsection is an *R* markdown document, instead of a typical script, which allows printing the results in an HTML file and sending them to be visualised for other people without the need to share the code and dataset files.

Bubble map

There are several ways to display a geographical area with valuable information on it, and the bubble map is one of these ways used in data visualisation. The bubble map displays a chosen geographical region, and according to the points' locations, in this case, buildings, it presents a circle in each location where their colour or size is proportional to a variable's value. With this kind of plot, it is possible to the District Heating company have an overview of the town in terms of high/low energy consumers, buildings with low ΔT, buildings where the supplied temperature is too low compared with the mean network's temperature, meaning significant heat losses in that network branch. In order to create this type of graph, the dataset must be provided with the latitude and longitude coordinates of each building. The main disadvantage of this plot is if the coordinates are to close, the circles will overlap, giving the impression of only one point instead of the real number of points. The figure 2.9 is an example of a bubble map.

Time-series plot

A time-series is a sequence of time-dependent values separated evenly in time. Each building's variable in the DH dataset is a time-series with discrete values separated by an hour interval. These type of dataset are widespread in all areas of society, e.g. from economics to engineering. A time-series is usually displayed in a line graph where the studied variable is plotted in a continuous-time span. Some plots can be with more than one variable's time-series to understand the correlation between these variables. Due to the existence of noise in specific datasets, it can be performed several techniques to smooth the graphs. A technique used in the DH dataset is the simple moving-average method which generates another time-series where its values are calculated from the average of a specific point with its neighbouring points. By using the moving-average technique, it is possible to smooth the fluctuations from the plots to have a better trend's overview of the dataset. In figure 2.10 is an example of a time-series plot.

Box and whisker plot

The box and whiskers plot, commonly known as box plot, it is a different way of displaying the variable's distribution in the dataset. The figure 2.11 is an example of a box plot, where the box zone is composed of the data comprehended between the lower quartile (25th percentile) and upper quartile (75th percentile). In the middle of the box, it can have two lines where one is located at the median value and the other at the mean value. The lines below/above the lower/upper quartiles are denominated whiskers and represent the data's distribution before/after the percentiles. These types of plots also show the lower and upper extremes of the distribution and the possible outliers. The advantages of using this plot are the display of the different distribution characteristics, e.g. median, mean, quartiles, skewness. And the possibility of grouping in the same plot various numerical or categorical variables.

Parallel coordinates plot

This plotting method is used for numerical multivariate datasets. The main idea behind this method is to attribute an axis to each variable and placed them parallel between each other. For each case, or multivariate observation, compound by a numerical vector of variables, it draws a line intersecting each parallel axis. By having all these lines distributed over the axes, it is possible to have a global overview of each case by comparing with the others. In the parallel coordinates plot, it is necessary to be aware that the axis order influences the perception of the dataset. Therefore a good practice besides carefully thinking the axis order is to change the order itself attempting in discovering new patterns. The major drawback of this plot is by having a high volume of cases turns the plot extremely dense in lines. In order to overcome the issue, the plot can be interactive when selecting a range of lines which will be highlighted while the others fade, this selective process is called brushing. In figure 2.12, it can be seen an example of a parallel coordinates plot.

Scatterplot

The scatterplots are formed by placing the data points into a cartesian coordinate system, where two variables are the x and y-axis. By displaying the variables in this kind of plot, it allows unravelling the relationship between the studied variables. These plots are also behind the concepts of correlation and regression, which are the processes of quantifying the strength of a variables relationship and estimating the mathematical relation between them. The scatterplots possibilities the understanding that each variable has on the others when compared. The drawback from this plotting method is that because it can only consider two variables, the correlation dependence between the variables might be due to another non-considered variable. The figure 2.13 exemplifies a scatterplot.

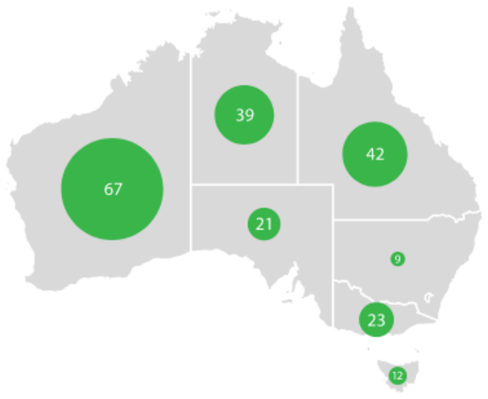


Figure 2.9. Example of a bubble map [Ribecca, 2020].

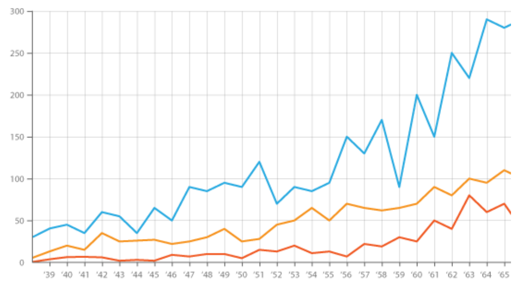


Figure 2.10. Example of a time-series [Ribecca, 2020].

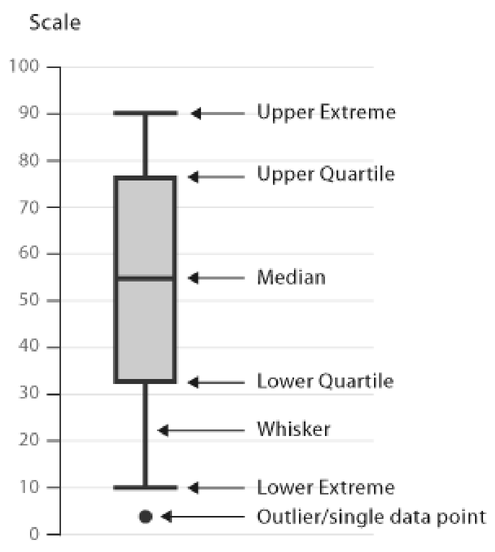


Figure 2.11. Example of a box and whisker plot [Ribecca, 2020].

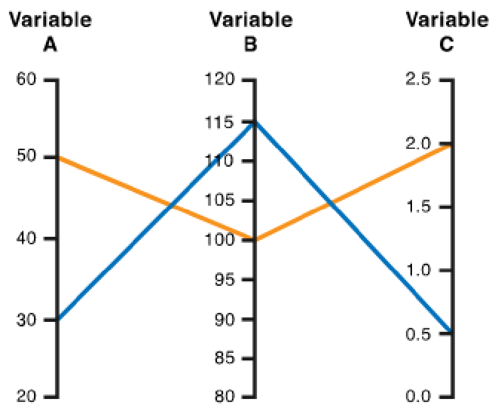


Figure 2.12. Example of a parallel coordinates plot [Ribecca, 2020].

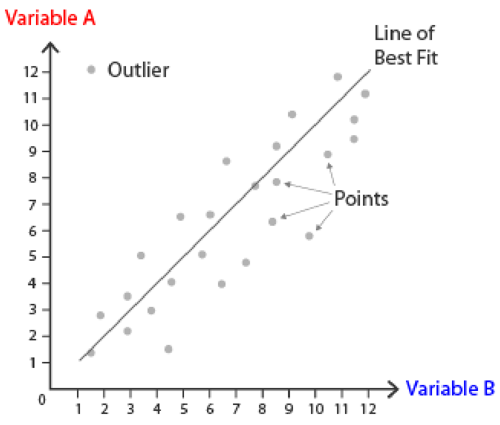


Figure 2.13. Example of a scatterplot [Ribecca, 2020].

2.3 Clustering Analysis

In Data Science, one of the methodologies used to study and find patterns in a given dataset is through clustering analysis. Clustering methods are used to identify, divide, group and categorise into subgroups the data points by their similarities in terms of their variables. Different clustering methods vary in their complexity and mathematical procedure and when applied, can achieve different results. Therefore, it is intrinsically necessary to base the choice of a specific method in terms of the data's nature. This means that the size of the data, the variables' type, the number of variables and other aspects from the data must be taken into account when choosing a suitable clustering method. From all the existing methods, the one chosen for this thesis is the **K-means method**. The decision was made taking into account, the simplicity in coding this method in a *R* script, the versatility in applying it in the different variables and that can be used for cluster time-series.

The K-means method is a Partitioning clustering method. The Partitioning methods group consists of clustering techniques where their total number of clusters must be known *a priori*, and from the number of clusters, the analysis can be carried on. This analysis subsists in dividing the time-series in the clusters' number specified in the beginning. Similar time-series form each cluster, and each cluster is identified by their centre, resembling the mean value of the points from all the time-series.

The concept behind this method is the minimisation of the total within-cluster variation; these variations intra- and inter-clusters can be measured by different dissimilarity measures processes, also known as distance matrices, where the one used in K-means is the **Euclidean distance**. The equation 2.17 defines the Euclidean distance, by measuring the distance between two observations:

$$d_{Euclidean} = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{2.17}$$

Where:

$d_{Euclidean}$	Euclidean distance
x_i, y_i	Data points
n	Number of total points in the dataset

Before running the K-means algorithm, all the values of each variable must be standardised through the equation 2.18. Another important note is that for the K-means being applied to a time-series, all its dataset must be divided according to a specific variable and the table must be transposed. The process of standardisation and transposition of the dataset is performed by the script `Script_Prepare_Clustering_DT.R` and can be consulted in Appendix I.

$$X_{standardised} = \frac{X - \mu}{\sigma} \tag{2.18}$$

Where:

$X_{standardised}$	Standardised observation [-]
X	Observation value - It can be Energy, Flow, Temperature [kWh or m^3/h or ºC]
μ	Mean value of the observations [kWh or m^3/h or ºC]
σ	Standard deviation of the observations [kWh or m^3/h or ºC]

In some datasets, the number of existing clusters is revealed. In this case of the smart energy meter's measurements, there is no information about the number of groups that the buildings can be grouped in. Therefore to predetermine the total number of clusters to begin the iterative relocation process of the cluster centres, it must be used some methods that determine the optimal number of clusters for a given dataset.

The **Elbow method** is one of the most famous methods, and it is one of the most suitable for the Partitional clustering methods due to evaluates how minimised is the total intra-cluster variation per number of clusters assuring well-compacted clusters. The Elbow method generates a function where the best solution, the optimal number of clusters, is the inflexion point, also known as the "knee", of the function. In figure 2.14, it is seen an example of the Elbow method function, wherein this case, the optimal number of clusters is four.

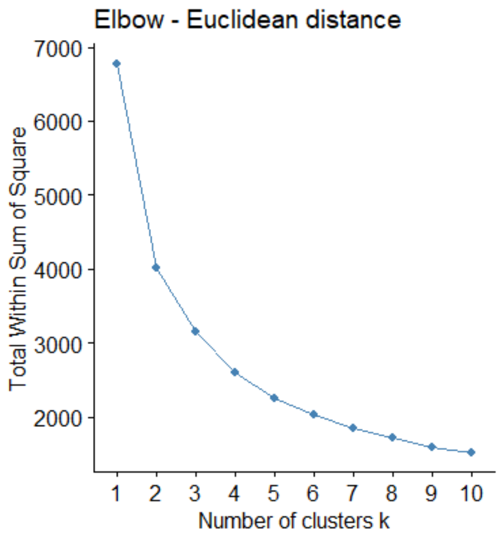


Figure 2.14. Example of the application of the Elbow method to find the optimal number of clusters.

Another approach that is used to determine the optimal number of clusters is by evaluating how well each observation occupies in its cluster. The **Average Silhouette method** measures this suitability of the points in their cluster, where a high average silhouette coefficient represents that the cluster which a data point was assigned to is the most suitable for it. The method itself is performed for several cases by changing the total number of clusters. For each case, it calculates the mean silhouette coefficient of all the observations, where the case with the highest coefficient is the optimal solution. In figure 2.14, it is seen an example of the Average Silhouette method, wherein this case, the optimal number of clusters is two, followed by the number four.

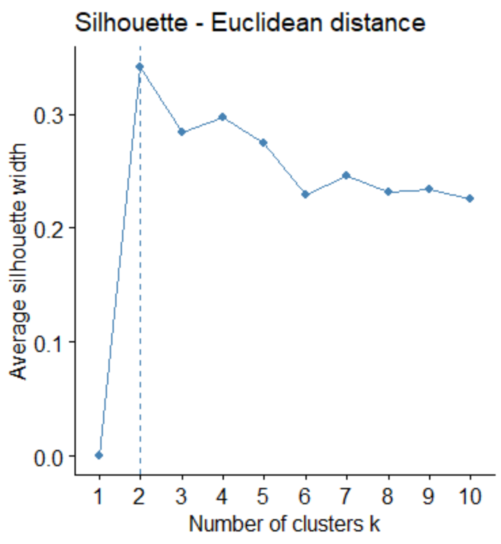


Figure 2.15. Example of the application of the Average Silhouette method to find the optimal number of clusters.

These two methods were performed by the scripts `Script_Elbow_Kmeans_3.R` and `Script_Silhouette_Kmeans_3.R`, seen in Appendix J and K, to find an optimal number of clusters for the DH dataset. From this chosen value, the K-means algorithm can be run by the script `Script_Kmeans_3.R`, also in the Appendix L.

The K-means clustering method was applied in different datasets in order to analyse the time-series by in different conditions. The first dataset is the variables cumulative energy and cumulative water-volume and the hourly-averaged temperature difference in a year span. The reason to cluster this dataset is to identify any groups of energy or volume consumption. Or even to see if the groups can be divided by their temperature difference and if the yearly mean temperature difference pattern has any influence on the overall consumption.

The second dataset consists of the daily profiles for each year's season of the variables hourly-energy, hourly-volume, flow and instantaneous temperature difference. The daily profiles were made by calculating the average variable's value for each hour of the day, only considering the working days of the week (from Monday to Friday). The working days were considered due to the constant routine that the users have during the week, which will be revealed in their heating usage.

The last dataset used in the K-means is the same as the one before but with the particularity that the buildings before clustered were divided by their yearly mean temperature difference. The temperature difference groups were divided into the groups 10 - 30 °C, 30 - 40 °C, 40 - 50 °C and 50 - 60 °C. And the considered variables were the water flow and instantaneous temperature difference, for the Winter and Summer season.

Results 3

As described before, several methods from data analysis were considered to investigate the District Heating data from Aalborg Forsyning. This chapter is composed of the collection of all the results from the methods that are described in chapter 2.

3.1 Data Preprocessing

All the study begins with the raw data from Aalborg Forsyning. The data comes from 1665 District Heating smart energy meters installed in different residential and non-residential buildings near Aalborg. In the measurements, some observations consist of detected errors by the devices, out-of-range values and missing values, which are empty data entries that were not measured in the first place. All the methodology used to cleanse and treat the data from the smart meters is described in section 2.1.

After uploading all the data from the different sources and organising it, the dataset was evaluated regarding the existing errors. In table 3.1 are all the different types of errors present in the DH dataset. For every kind of error, it was given an ID number to be used in the plots, afterwards.

Table 3.1. The different types of errors in the dataset and its identification number.

Type of Error	Number ID
Temperature difference has wrong polarity	1
V1 Wrong flow direction	2
Flow sensor with wrong flow direction	3
Air has been detected in the flow meter (only indicated as long as the fault exists)	4
Flow sensor with weak signal or air	5
T2 Below measuring range or short-circuited	6

From table 3.1, it is concluded that all faults recognised by the devices are from the sensor's performance of flow and temperatures. After identifying all the different kind of faults, it is necessary to see how many errors were detected in a full-year period. The dataset has 14,855,339 observations, where 30,785 are faults recognised by the smart meters, therefore in all dataset, 0.21 % of the data are errors. This is a small percentage when seeing the overall dataset. However, it is necessary to evaluate the number of errors per building. In figure 3.1, it is the plot with the percentage of the total sum of errors per buildings in the ten buildings with the highest values in the dataset.

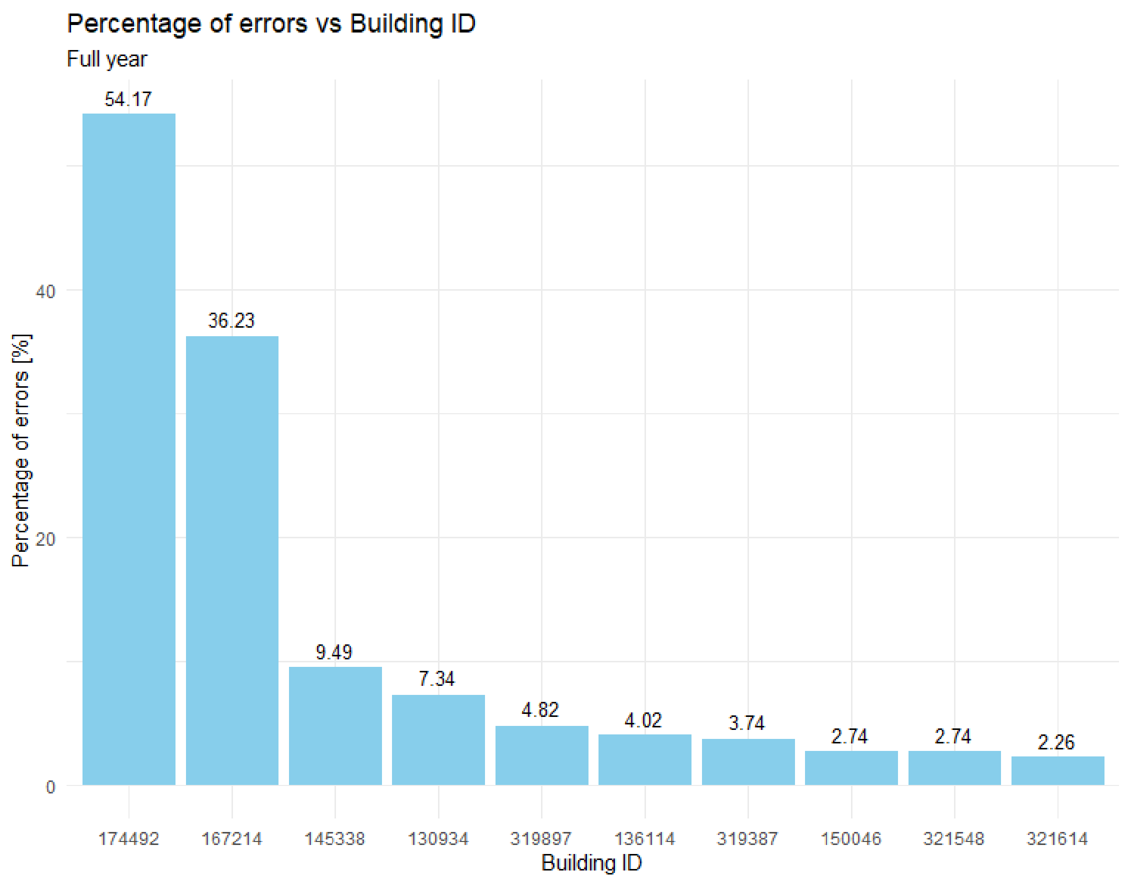


Figure 3.1. Barplot of the quantity's percentage of errors per building.

By analysing the plot, it is concluded two buildings stand out. These buildings are the 174492 and 167214 where 54.17 % and 36.23 % of their data are registered as an error. The other eight buildings have percentages below 10 %. For a better understanding of these mistakes, the script `Script_2_Errors.R` also generates bar plots of all the different types of errors per building of the figure 3.1. It was used the numeric IDs from table 3.1 to identify the error's type. The bar plots are in figure 3.2:

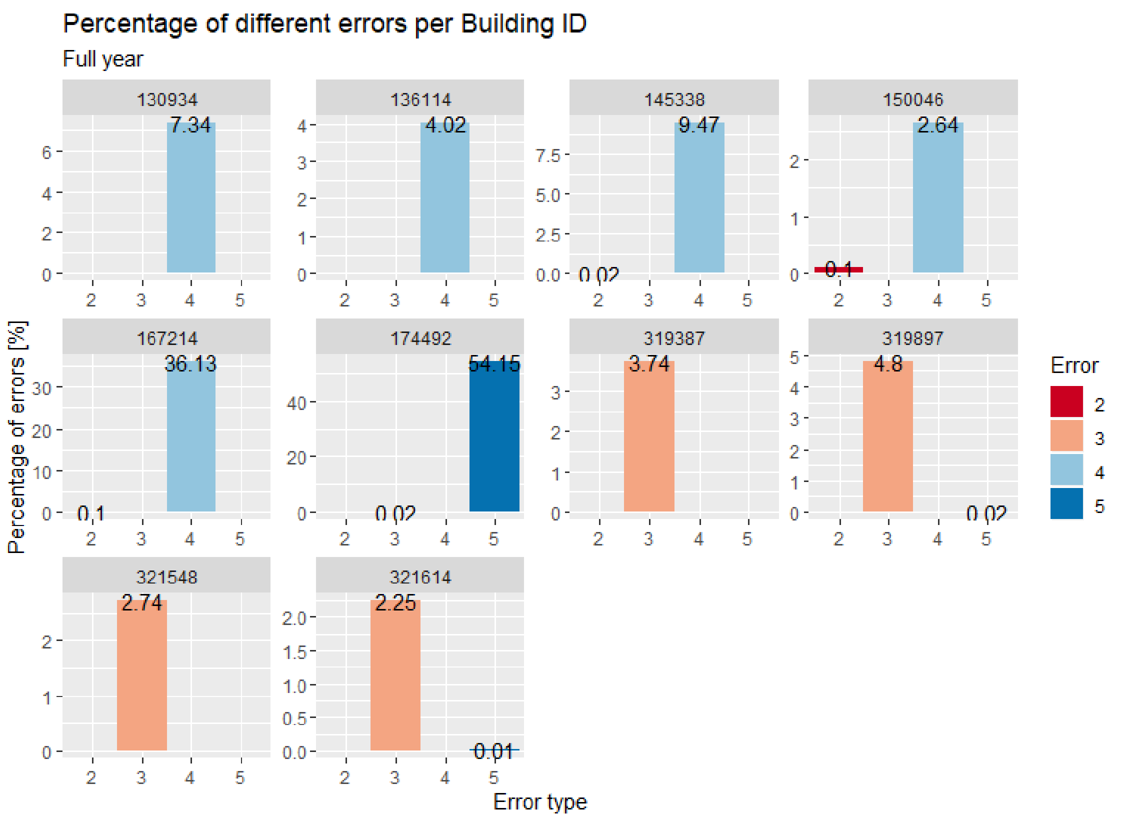


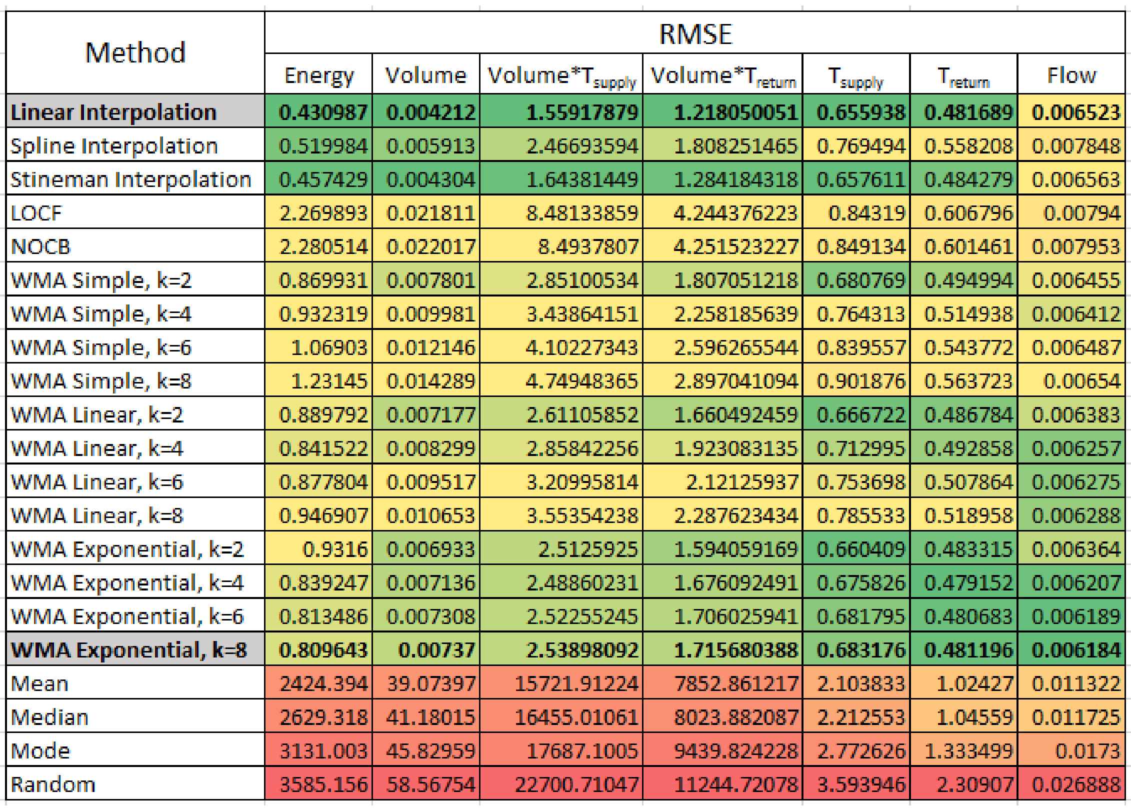
Figure 3.2. Barplot of the quantity's percentage of errors by type per building.

By examining the plot, it is concluded from these buildings, the only errors that appear are the types: 2, 3, 4 and 5. These types are the ones only from the flow sensor, which means that in this dataset is much more susceptible to an issue occur in a flow sensor than in a temperature sensor. It is also observed that the studied buildings have most of its data with only one type of error than an even distribution between types. The buildings 174492 and 167214, present the error type 5 and 4, respectively. This error type is regarding the detection of air in the system or that the smart meter's signal is weak. Therefore the systems of the two residences should be evaluated to solve the origin of the errors.

Regarding the results from the detection of outliers by the script `Script_3_Outliers.R`, when considering the set of all the variables described in subsection 2.1.3. It is observed that in the dataset, 1.99 % of the observations are outliers. Having identified all errors and outliers, both of these observations will be considered as missing values and consequently replaced as `NA` values.

The missing values, classified as `NA` values in *R*, are spread in all dataset, and they were evaluated by the script `Script_4_NA.R`. In a preliminary overview, it was concluded that most of the buildings had significant missing data gaps at the beginning of the measurements. This is due to the time synchronisation made by the `Script_1_Raw_Data.R`, where most of the buildings' recording started after 2018 - 10 - 01. Therefore to overcome this issue, all the data from 2018 - 10 - 01, 00:00h until 2018 - 11 - 04, 23:00h was discarded. In subsection 2.1.4, there are two criteria that if one of them failed by a building, the building is excluded from the dataset. The first criterium is that a building cannot have more than nine hours gap of missing data, to prevent having buildings with significant missing data gaps. The second criterium is that all buildings composed with more than 10 % of NA values must be discarded from the dataset. When following these criteria, 637 residences are eliminated from the initial number of 1665 buildings, representing an exclusion of 38.3 % of the original set of buildings. From this point, the dataset is ready to be solved by the imputation methods described in subsection 2.1.5 in the Methodology chapter.

The application of the imputation methods was made in two parts. The first part consists of finding the best techniques to apply in the DH dataset. As explained in the chapter before, several methods were applied in ten buildings time-series, in all their set of variables. And they were compared through the root mean square error, RMSE, between the known values and the imputed ones from the different techniques. The final results of this benchmark can be seen in figure 3.3, which is a coloured Excel table where the lowest values are represented by green colour and gradually become red when they increase. Because there are ten investigated buildings described in the script, it was calculated the average of the RMSE results and inserted in the Excel table.



Method	RMSE						
	Energy	Volume	Volume*T_{supply}	Volume*T_{return}	T_{supply}	T_{return}	Flow
Linear Interpolation	**0.430987**	**0.004212**	**1.55917879**	**1.218050051**	**0.655938**	**0.481689**	**0.006523**
Spline Interpolation	0.519984	0.005913	2.46693594	1.808251465	0.769494	0.558208	0.007848
Stineman Interpolation	0.457429	0.004304	1.64381449	1.284184318	0.657611	0.484279	0.006563
LOCF	2.269893	0.021811	8.48133859	4.244376223	0.84319	0.606796	0.00794
NOCB	2.280514	0.022017	8.4937807	4.251523227	0.849134	0.601461	0.007953
WMA Simple, k=2	0.869931	0.007801	2.85100534	1.807051218	0.680769	0.494994	0.006455
WMA Simple, k=4	0.932319	0.009981	3.43864151	2.258185639	0.764313	0.514938	0.006412
WMA Simple, k=6	1.06903	0.012146	4.10227343	2.596265544	0.839557	0.543772	0.006487
WMA Simple, k=8	1.23145	0.014289	4.74948365	2.897041094	0.901876	0.563723	0.00654
WMA Linear, k=2	0.889792	0.007177	2.61105852	1.660492459	0.666722	0.486784	0.006383
WMA Linear, k=4	0.841522	0.008299	2.85842256	1.923083135	0.712995	0.492858	0.006257
WMA Linear, k=6	0.877804	0.009517	3.20995814	2.12125937	0.753698	0.507864	0.006275
WMA Linear, k=8	0.946907	0.010653	3.55354238	2.287623434	0.785533	0.518958	0.006288
WMA Exponential, k=2	0.9316	0.006933	2.5125925	1.594059169	0.660409	0.483315	0.006364
WMA Exponential, k=4	0.839247	0.007136	2.48860231	1.676092491	0.675826	0.479152	0.006207
WMA Exponential, k=6	0.813486	0.007308	2.52255245	1.706025941	0.681795	0.480683	0.006189
WMA Exponential, k=8	**0.809643**	**0.00737**	**2.53898092**	**1.715680388**	**0.683176**	**0.481196**	**0.006184**
Mean	2424.394	39.07397	15721.91224	7852.861217	2.103833	1.02427	0.011322
Median	2629.318	41.18015	16455.01061	8023.882087	2.212553	1.04559	0.011725
Mode	3131.003	45.82959	17687.1005	9439.824228	2.772626	1.333499	0.0173
Random	3585.156	58.56754	22700.71047	11244.72078	3.593946	2.30907	0.026888

Figure 3.3. Results from the imputation methods benchmark.

The methods where the missing values are replaced by the mean, median or mode of the dataset or by a random number limited between the minimum and maximum values of the data presented the worst results. For the cumulative variables, the best technique found is the linear interpolation; this might be due to the fact these variables show a monotonically increasing evolution, similarly as a linear function. Regarding the instantaneous variables, the best method is the exponential weighted moving average with a window-size of 8 data points, k. The explanation behind this result might be due to its exponential nature. The replaced missing value is dependent of several neighbouring points attributing robustness to the solution when comparing other moving average algorithms. Based on these results, the chosen methods to solve the missing data points are the linear interpolation for the cumulative variables and the exponential weighted moving average with k = 8, for the instantaneous variables. Accordingly, the second part consists of applying these methods to the District Heating dataset to solve the issue of the remaining missing data.

When the missing values are estimated, it is possible to separate the buildings that are residential from the non-residential. The separation procedure is defined in subsection 2.1.5 from the Methodology chapter. The calculated limits that divide the buildings are in table 3.2:

Table 3.2. Limits to differentiate between residential and commercial building.

Variable name	**Limits**
Maximum energy [kWh]	30,286
Maximum volume [m^3]	669
Mean power [kW]	3.47

The exclusion criteria from the missing values script, it excluded 637 buildings from the original 1665 buildings, due to large quantities of missing data. In this case, the buildings with values higher than the ones from the table 3.2, will be considered non-residential and consequently excluded also from the dataset. The results are that from 1028 buildings, 59 buildings were excluded. By considering all these values, the data preprocessing algorithm finished with 969 buildings that are considered the final dataset to be used for the analysis, and represents 58.20 % of the original data from Aalborg Forsyning.

3.2 Data Visualisation

This section is based on section 2.2 from the Methodology chapter where are described several new calculated variables to supplement the dataset and the plotting techniques used to unravel some of the building and systems characteristics and consumers behaviour. The section is divided into two main parts, which the first part consists of viewing the buildings by one variable at the time. And the second part consists of a multivariable study of all the buildings when considering all their variables.

3.2.1 Single-variable overview

This part is responsible for giving an overview of the set of buildings near Aalborg regarding one of its variables. By having this general perspective, it is possible to compare them with each other. The first step in the visualisation is to map all buildings by their geolocations and differentiate them by their properties. When evaluating the District Heating network, the considered performance outputs from the buildings are the energy, water-volume and the temperature difference between the supply and the return temperatures. By having these variables, it was made a scatterplot, before the bubble mapping, to understand any possible correlations between the variables. But also to prevent that if two of the parameters are strongly correlated, there is no need for mapping both parameters. The figures 3.4, 3.5 and 3.6, are the scatterplots between total energy usage, total water-volume consumption and the mean hourly-average temperature difference in a year span for each building. Each plot has coloured the buildings according to the range of their mean hourly-average ΔT.

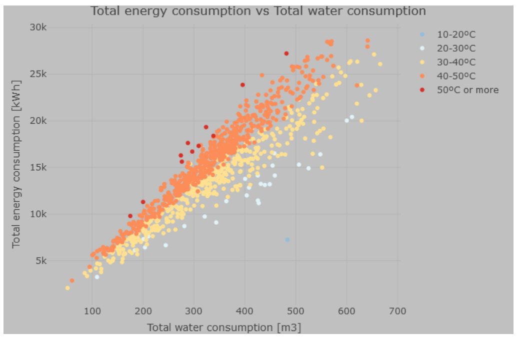


Figure 3.4. Scatterplot between total energy consumption and total water consumption.

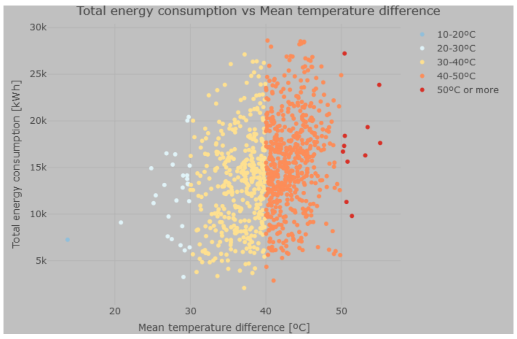


Figure 3.5. Scatterplot between total energy consumption and mean temperature difference.

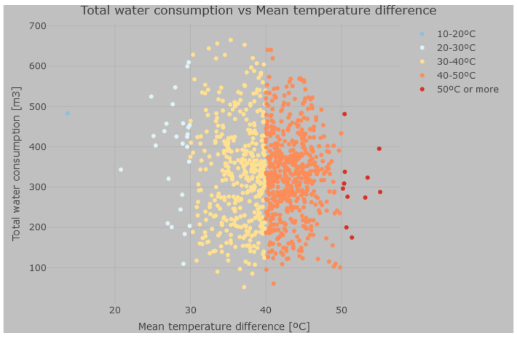


Figure 3.6. Scatterplot between total water consumption and mean temperature difference.

Regarding the plots, it can be concluded that the total energy and water usage are highly correlated and that there is a relationship where it is seen that $\overline{\Delta T}$ represents the slope of a line passing through the points marked with different colours. This relation was already expected from the results, due to the energy be a product of the water-volume with $\overline{\Delta T}$, as seen in equation 3.1.

$$E = \frac{1}{3600} \cdot c_p \cdot \rho \cdot V \cdot \overline{\Delta T} \tag{3.1}$$

Where:

E	Energy [kWh]
c_p	Water isobaric specific heat [kJ/kg · K]
ρ	Water density [kg/m^3]
V	Water volume [m^3]
$\overline{\Delta T}$	Mean hourly-average ΔT [°C]

In figure 3.6, it is not seen any correlation between the water-volume and the $\overline{\Delta T}$. The plot shows that high or low values of $\overline{\Delta T}$ have higher and lower values of total water consumption. This might mean that these two variables are dependent by other factors that are not being considered in this scatterplot or that they are independent of each other. Because of the relation between energy and the variables water-volume and $\overline{\Delta T}$, only these two last variables will be the ones used for the bubble maps.

Bubble map: Temperature

The bubble maps are a powerful tool to represent the DH dataset by spreading all the considered variables by their locations, allowing the DH company to understand if the building's proximity to each other influences the dataset. This idea of proximity comes from the premiss that most of the residential houses when close to each other might share the same characteristics, e.g. heating/DHW systems or building materials. In terms of temperature, the smart meters measure the supply and return temperatures, both in hourly-average and instantaneous measurement. For the bubble maps, it was considered the hourly-averaged values. The first variable considered is the mean supply temperature in a year's span. This value is important for the utility company because it is not dependent on the building but the DH network. By mapping the supply temperature, it is possible to see which are the buildings with low values, representing the locations on the network with high heat losses. In figure 3.7 it is represented this bubble map. It is also plotted as a bubble map, the mean hourly-averaged return temperature and $\overline{\Delta T}$ in figures 3.8 and 3.9. These two variables are dependent on the building's system and user's behaviour and therefore used by the DH company as an indicator to understand which buildings have better overall energy performance when compared.

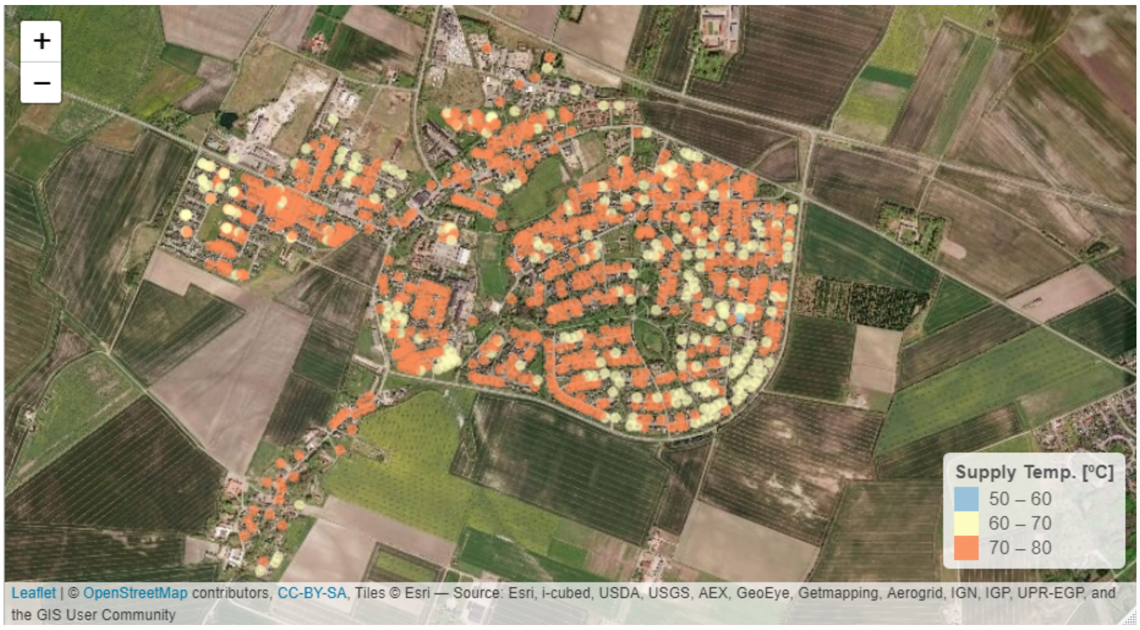

+
−
Supply Temp. [°C]
50 – 60
60 – 70
70 – 80
Leaflet | © OpenStreetMap contributors, CC-BY-SA, Tiles © Esri — Source: Esri, i-cubed, USDA, USGS, AEX, GeoEye, Getmapping, Aerogrid, IGN, IGP, UPR-EGP, and the GIS User Community

Figure 3.7. Bubble map of the mean supply temperature.

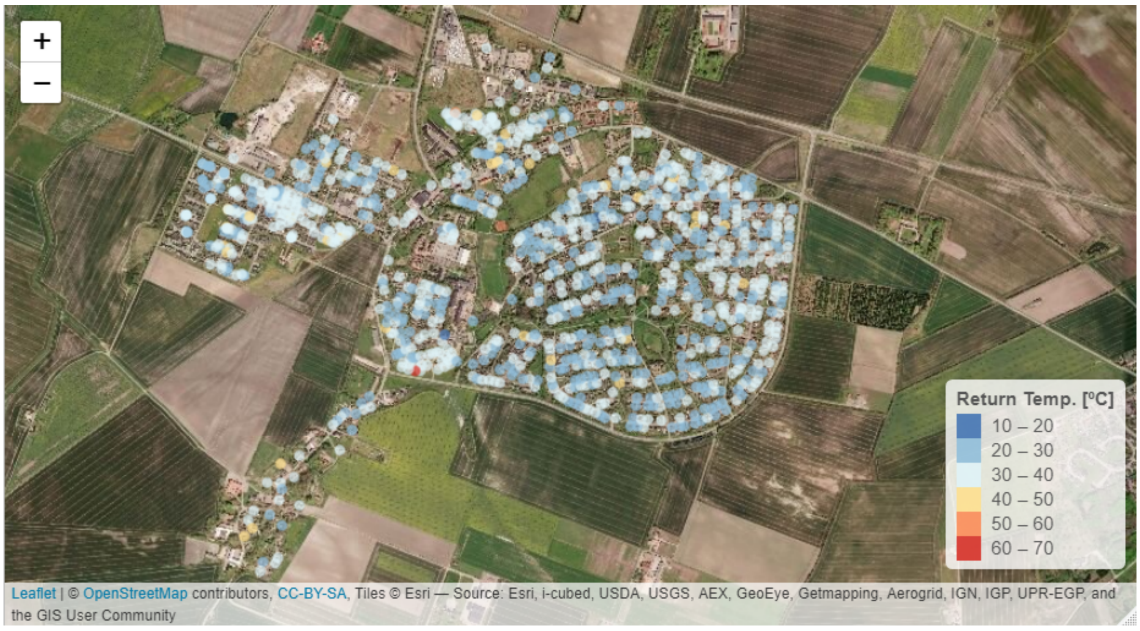

+
−
Return Temp. [ºC]
10 – 20
20 – 30
30 – 40
40 – 50
50 – 60
60 – 70
Leaflet | © OpenStreetMap contributors, CC-BY-SA, Tiles © Esri — Source: Esri, i-cubed, USDA, USGS, AEX, GeoEye, Getmapping, Aerogrid, IGN, IGP, UPR-EGP, and the GIS User Community

Figure 3.8. Bubble map of the mean return temperature.

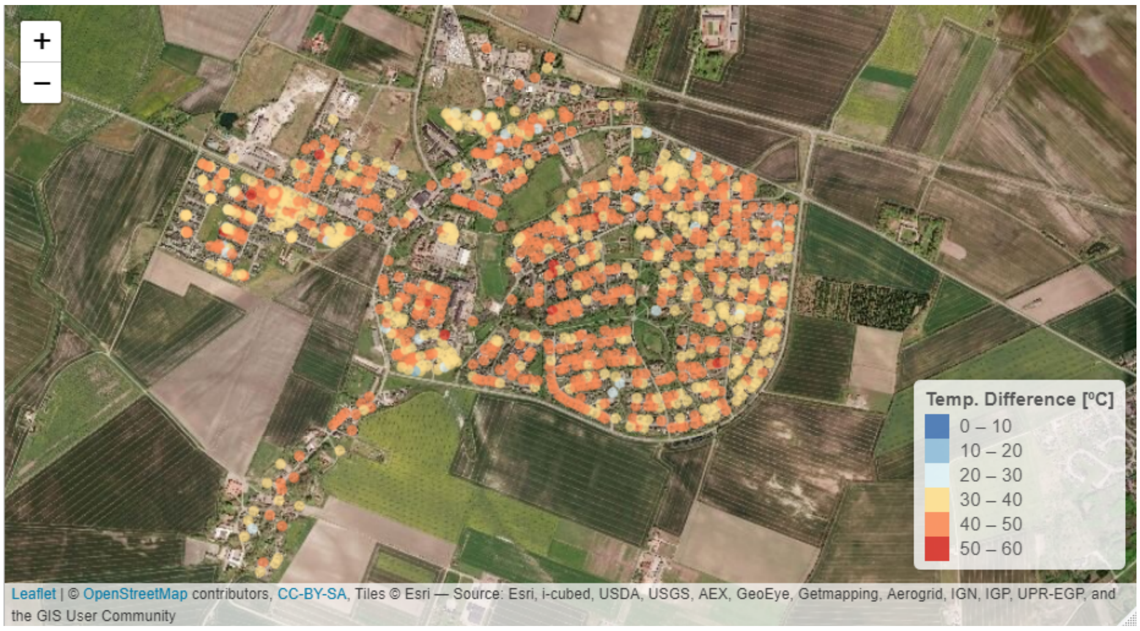

+
−
Temp. Difference [ºC]
0 – 10
10 – 20
20 – 30
30 – 40
40 – 50
50 – 60
Leaflet | © OpenStreetMap contributors, CC-BY-SA, Tiles © Esri — Source: Esri, i-cubed, USDA, USGS, AEX, GeoEye, Getmapping, Aerogrid, IGN, IGP, UPR-EGP, and the GIS User Community

Figure 3.9. Bubble map of the mean temperature difference.

From the plot in figure 3.7, it is seen the mean supply temperature of all the houses is uniform where the mean value for all the residences is 72 °C, and the standard deviation is 2.93 °C. These results were already expected since the supply temperature is a variable entirely dependent on the DH company and not from the buildings. In the plot, it is also noticed a building with a supply temperature of 59 °C (blue colour), and this value might indicate that distribution branch has significant heat losses and should be investigated by the DH company. By evaluating the mean return temperature and the mean $\overline{\Delta T}$, it is seen that the constancy from the first plot is not seen. In these last two plots, more buildings have a spreader temperature distribution, where for the return temperature has a mean value of 31 °C and standard deviation of 4.39 °C, and for the $\overline{\Delta T}$, the mean is 40 °C, and the standard deviation is 4.85 °C. The residences that have large $\overline{\Delta T}$ are seen as energetically efficient because they return to the DH network the water at low temperature. This also means the buildings that have low $\overline{\Delta T}$ are performing poorly and must be studied carefully to define a solution to increase their $\overline{\Delta T}$. The figure 3.10 is a box plot with the temperatures' distributions of the town's DH network. From it is possible to see the overall distribution and recognise which buildings are situated outside the expected range when compared with the dataset.

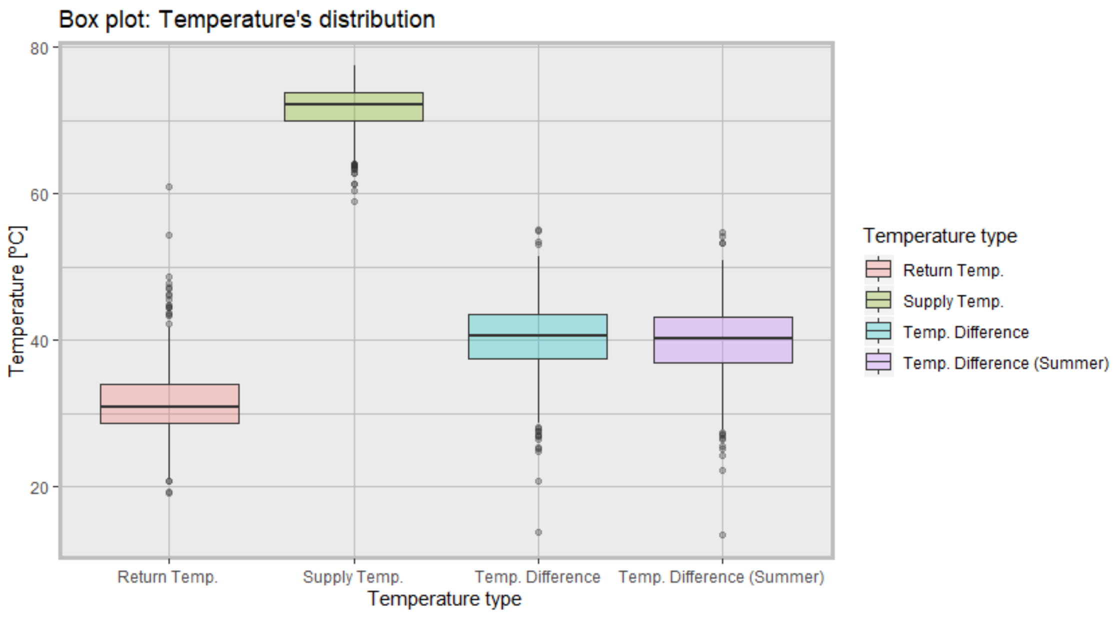


Figure 3.10. Box plot of the temperatures' distribution.

From the plot in figure 3.10, it is possible to see that the supply temperature has a compact box, which means that most of its values are similar. Furthermore, in its distribution, some points are identified as outliers. Therefore all network connecting these buildings should be investigated. The return temperature has a broader distribution with more outliers. This is connected with the different factors, e.g. users behaviour/consumption or the type/efficiency of the building's system. Most of the outliers are for temperatures higher than 40 °C, which is not energetically efficient because the water is returning to the network with high temperatures. All these buildings, therefore, should be investigated. Because of this wide distribution on the return temperature, the building's temperature difference will also have the same distribution characteristics, as the plot reveals. The outliers of the supply temperature and $\overline{\Delta T}$ were plotted in a bubble map, as seen in figure 3.11 and 3.12 to see if any pattern emerges by having fewer points on the map. The temperature difference in the summer season was also considered in the box plot in an attempt to see which buildings are performing worst due to their DHW production system. As seen in the box plot, the Summer's temperature difference is very similar to the year's temperature difference, in terms of its distribution. This result means that the $\overline{\Delta T}$ might not change significantly between the summer period and year span. But to have a better understanding of their relationship, it was plotted another box plot, figure 3.13, where its values are the ratio $\overline{\Delta T}_{summer}/\overline{\Delta T}$. From this ratio, all the buildings that have values smaller than 1, may represent buildings where the DHW system has some installation or design problem that may affect their performance.

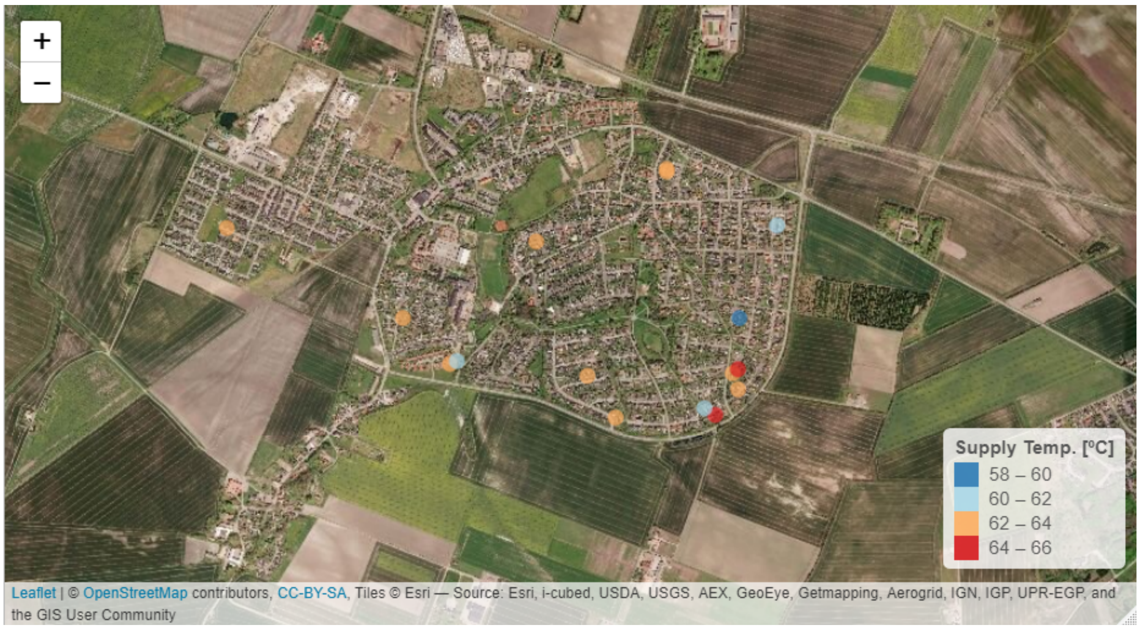

+
−
Supply Temp. [ºC]
🟦 58 – 60
🟦 60 – 62
🟧 62 – 64
🟥 64 – 66
Leaflet | © OpenStreetMap contributors, CC-BY-SA, Tiles © Esri — Source: Esri, i-cubed, USDA, USGS, AEX, GeoEye, Getmapping, Aerogrid, IGN, IGP, UPR-EGP, and the GIS User Community

Figure 3.11. Bubble map of the mean supply temperature - Outliers.

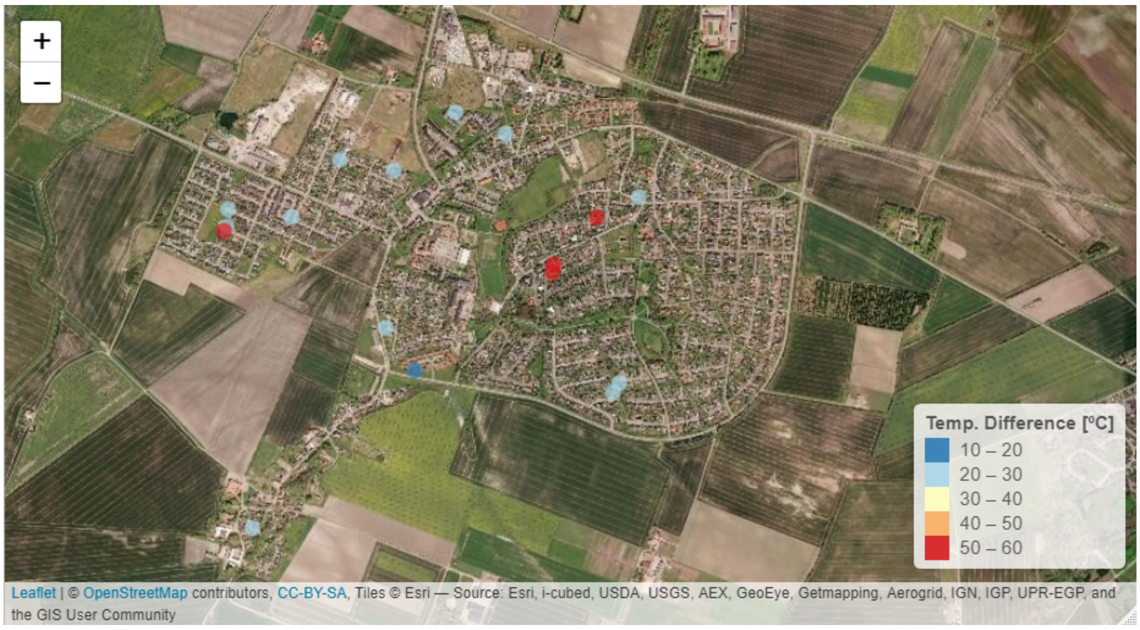

+
−
Temp. Difference [ºC]
🟦 10 – 20
🟦 20 – 30
🟨 30 – 40
🟧 40 – 50
🟥 50 – 60
Leaflet | © OpenStreetMap contributors, CC-BY-SA, Tiles © Esri — Source: Esri, i-cubed, USDA, USGS, AEX, GeoEye, Getmapping, Aerogrid, IGN, IGP, UPR-EGP, and the GIS User Community

Figure 3.12. Bubble map of the mean temperature difference - Outliers.

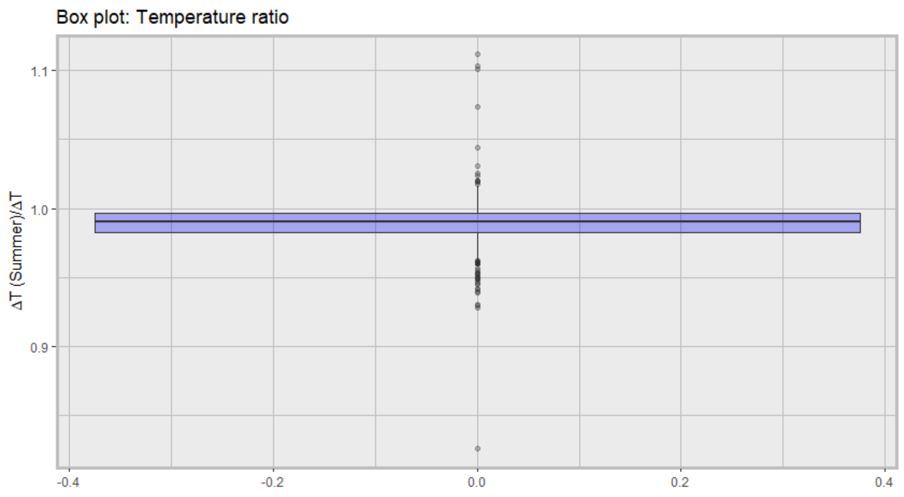


Figure 3.13. Ratio between temperature differences.

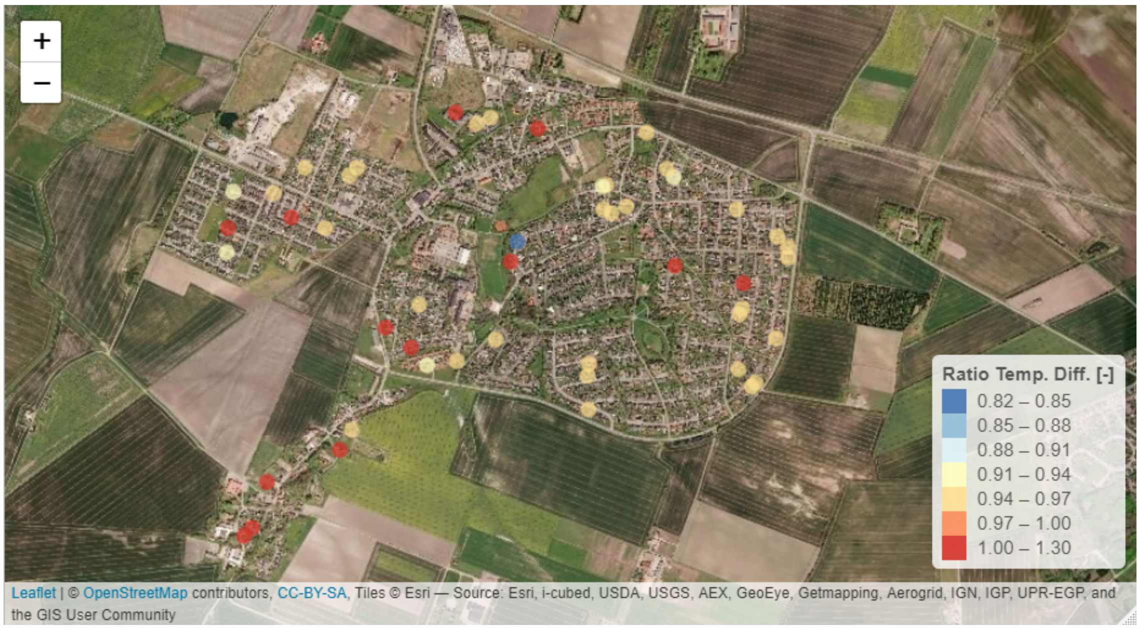

+
−
Ratio Temp. Diff. [-]
0.82 – 0.85
0.85 – 0.88
0.88 – 0.91
0.91 – 0.94
0.94 – 0.97
0.97 – 1.00
1.00 – 1.30
Leaflet | © OpenStreetMap contributors, CC-BY-SA, Tiles © Esri — Source: Esri, i-cubed, USDA, USGS, AEX, GeoEye, Getmapping, Aerogrid, IGN, IGP, UPR-EGP, and the GIS User Community

Figure 3.14. Bubble map of the ratio between temperature differences - Outliers.

The figure 3.11 shows the buildings where their mean supply temperature is low when considering the average from all the dataset. Even though their locations are spread throughout the town, it is possible to see some clusters, which might mean a local network section with higher heat losses. The figure 3.12 it is represented the outliers from the mean temperature difference. In this bubble map, it is presented some of the buildings that have high $\overline{\Delta T}$ and the ones with low $\overline{\Delta T}$. As seen, the high values are between 50 and 60 °C, and the low values are between 10 and 30 °C. Due to the significance of the temperature difference variable in describing the building's energy usage, these buildings were studied more in-depth in section 3.3.

Regarding the Summer's temperature measurements from the plots in the figures 3.13 and 3.14 it can be stated that the mean value of the ratio $\overline{\Delta T}_{summer}/\overline{\Delta T}$ is 0.989 and the standard deviation is 0.015. This means that most of the buildings have a uniform ratio with a lower $\overline{\Delta T}_{summer}$ than the year's $\overline{\Delta T}$ which is caused by a decrease of the operating temperatures in the non-heating season. The outliers of this ratio are presented in figure 3.14, where exists low-value outliers as high-value outliers. The low-value ratio might represent a building that their DHW production system is not that efficient when compared with the heating system. Due to the premise that during the non-heating season the dwelling's heating system is not operating or that it is operating but not that significant as in the heating season. The high-values ratio might represent the residences that have a more efficient DHW heat exchanger than its building's space heating system.

Bubble map: Water consumption

Another important variable that can describe the building is the water-volume consumption. This variable, as seen from the plot in figure 3.4 is highly correlated with energy usage. To have an overview of the year's water consumption of the town, it is also a plotted a bubble map in figure 3.15, where all the buildings are represented.

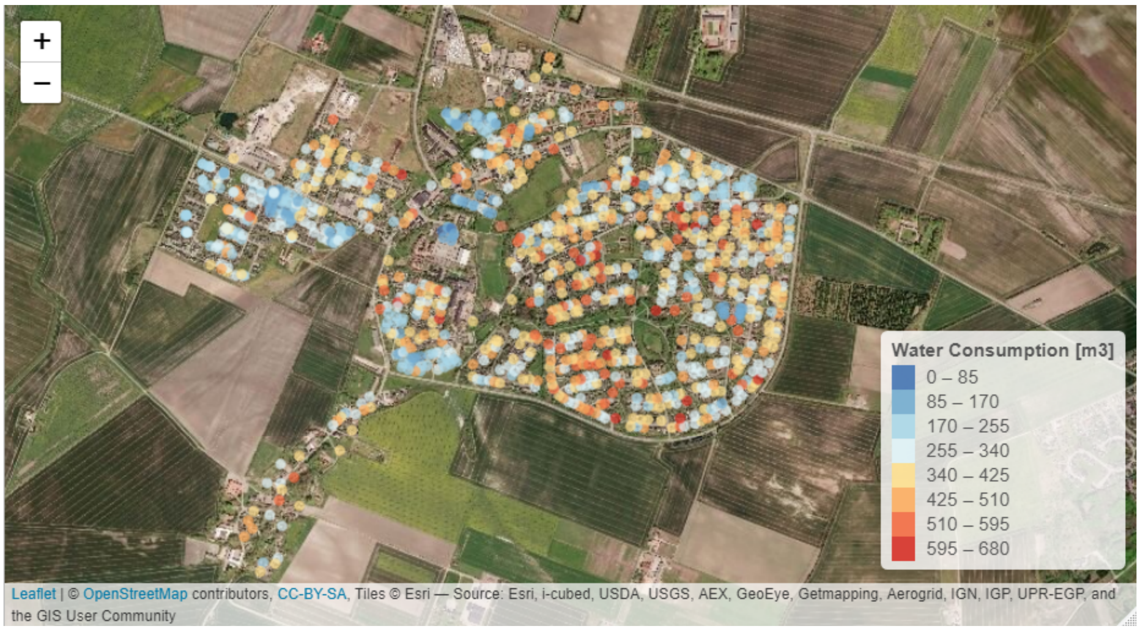

+
−
Water Consumption [m3]
0 – 85
85 – 170
170 – 255
255 – 340
340 – 425
425 – 510
510 – 595
595 – 680
Leaflet | © OpenStreetMap contributors, CC-BY-SA, Tiles © Esri — Source: Esri, i-cubed, USDA, USGS, AEX, GeoEye, Getmapping, Aerogrid, IGN, IGP, UPR-EGP, and the GIS User Community

Figure 3.15. Bubble map of the total water consumption.

The figure 3.15 show that most of the colours are grouped similarly, indicating that buildings that are close to each other have similar consumptions. This idea corroborates the theory that the adjacent buildings have similar characteristics, e.g. heating/DHW systems or the number of rooms that can affect directly the water consumption. The figure 3.16 is another bubble map, where it is represented the mean daily water consumption of each building in the Summer period.

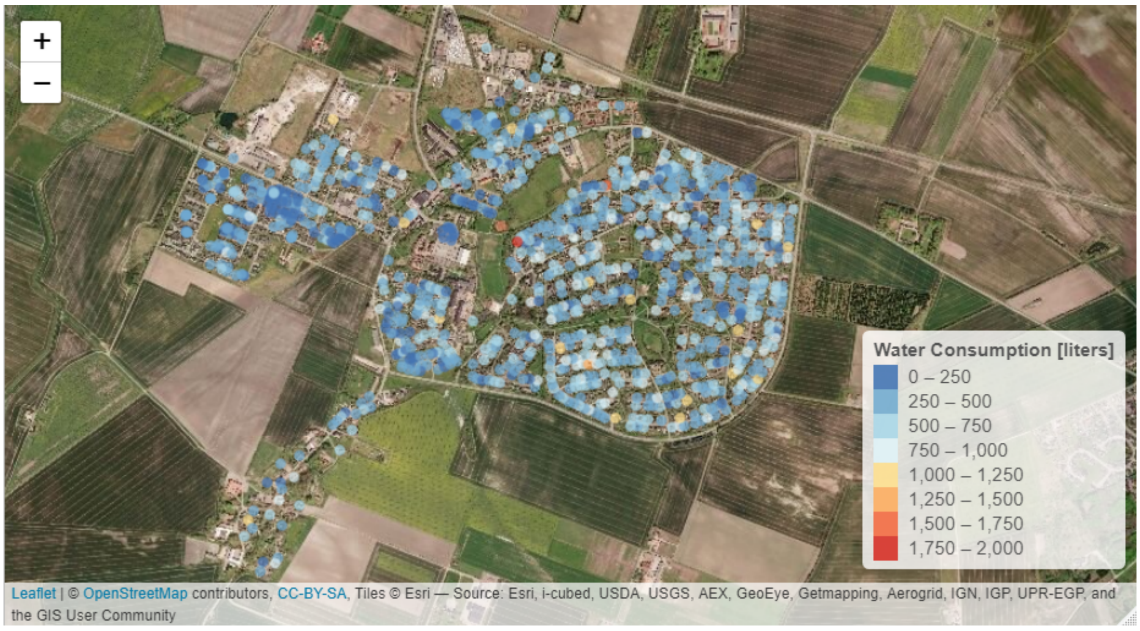

+
−
Water Consumption [liters]
0 – 250
250 – 500
500 – 750
750 – 1,000
1,000 – 1,250
1,250 – 1,500
1,500 – 1,750
1,750 – 2,000
Leaflet | © OpenStreetMap contributors, CC-BY-SA, Tiles © Esri — Source: Esri, i-cubed, USDA, USGS, AEX, GeoEye, Getmapping, Aerogrid, IGN, IGP, UPR-EGP, and the GIS User Community

Figure 3.16. Bubble map of the daily water consumption at the Summer period.

The figure 3.16 presents that most of the buildings in the dataset have similar daily water consumptions during the Summer. The buildings in the map that have higher values than compared with the average might be buildings where their heating system continues operating, e.g. for floor heating in bathrooms, or that DHW production system as any fault which requires to have a higher consumption to fulfil the daily energy demands. In figures 3.17 and 3.18, it is presented the same bubble maps but only with the buildings where their values stand out as outliers from the distribution.

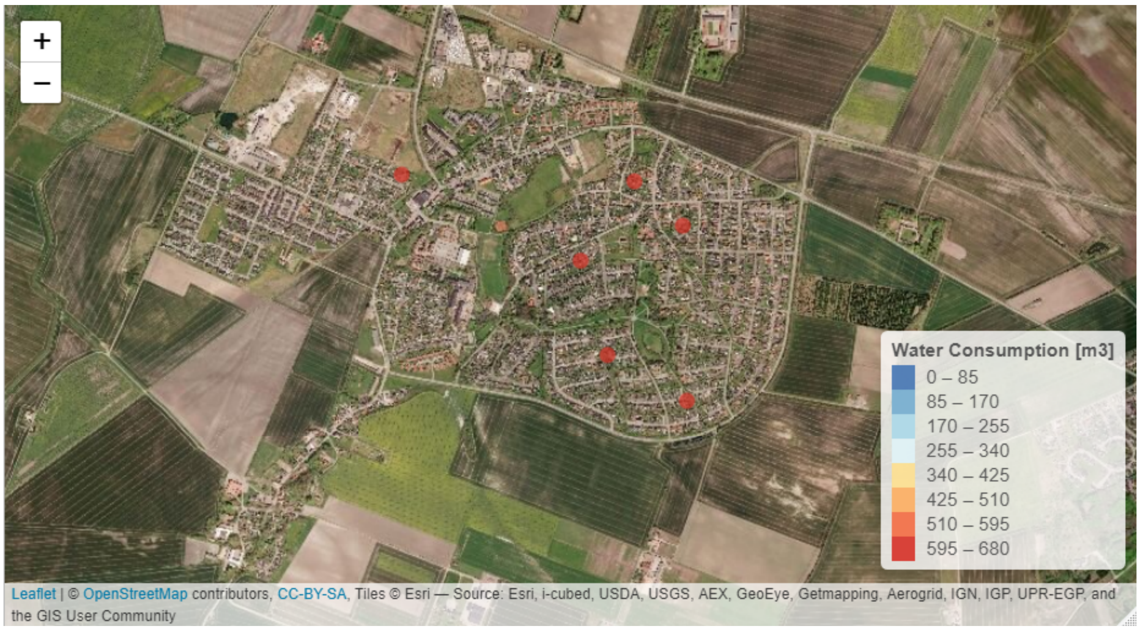

+
−
Water Consumption [m3]
0 – 85
85 – 170
170 – 255
255 – 340
340 – 425
425 – 510
510 – 595
595 – 680
Leaflet | © OpenStreetMap contributors, CC-BY-SA, Tiles © Esri — Source: Esri, i-cubed, USDA, USGS, AEX, GeoEye, Getmapping, Aerogrid, IGN, IGP, UPR-EGP, and the GIS User Community

Figure 3.17. Bubble map of the total water consumption - Outliers.

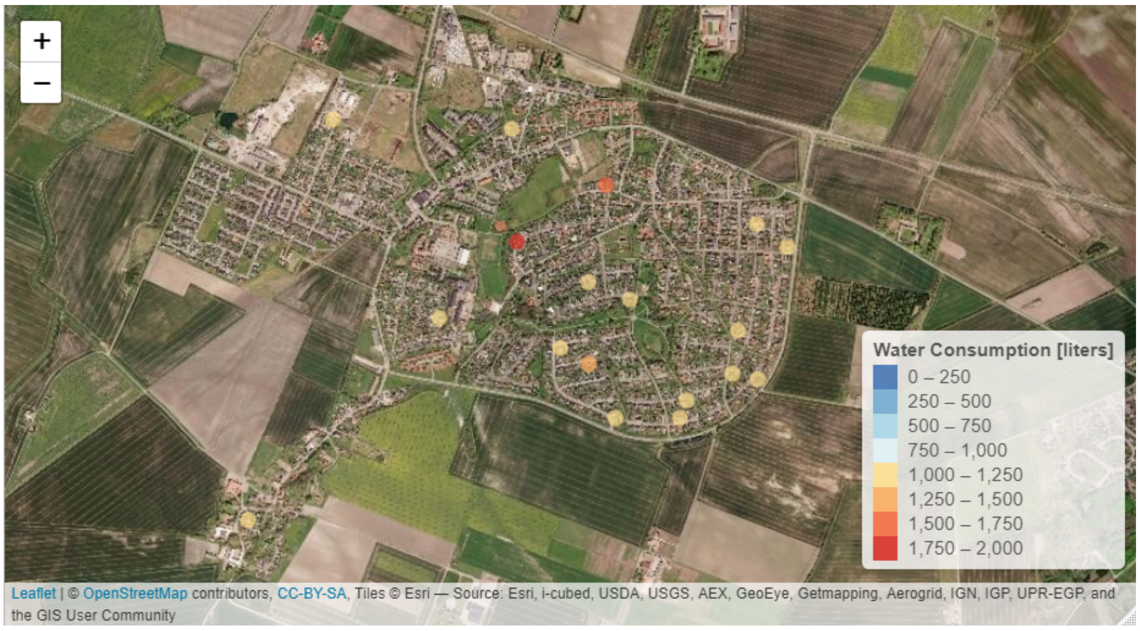

+
−
Water Consumption [liters]
0 – 250
250 – 500
500 – 750
750 – 1,000
1,000 – 1,250
1,250 – 1,500
1,500 – 1,750
1,750 – 2,000
Leaflet | © OpenStreetMap contributors, CC-BY-SA, Tiles © Esri — Source: Esri, i-cubed, USDA, USGS, AEX, GeoEye, Getmapping, Aerogrid, IGN, IGP, UPR-EGP, and the GIS User Community

Figure 3.18. Bubble map of the daily water consumption at the Summer period - Outliers.

From figure 3.17 it is concluded that are a few dwellings that are considered as outliers. These outliers are highlighted due to their high water consumption. A dwelling can have high consumption, due to numerous factors as having an inefficient heat exchange system which requires more water to fulfil the energy demands. Also, it can be due to a big number of DHW consumers in the house, e.g. people or equipment. If provided more information concerning the building like the number of rooms or the energy labelling, it is possible to see if these values are reasonable or not. The figure 3.18, presents more outliers than the data from the figure 3.17. Each highlighted dwelling present a significant daily water consumption in the Summer. This indicates to the DH company which buildings might consume large amounts of DHW or these particular buildings use their heating system the full-year.

3.2.2 Multi-variable overview

In the subsection above, the measured temperatures and water-volume consumption were mapped. The mapping of each building provides to the utility company the identification of the buildings that have peculiar values in their dataset. But as shown in the Methodology chapter, other variables were calculated and can be used to provide important information about the dwellings. These variables could also be plotted as bubble maps, but because there are so many variables that might be relevant to consider when studying the DH network with their buildings, the amount of generated maps would turn very difficult the analysis. Also, the bubble maps can express at most two different variables when plotted. Therefore when having more variables, it is necessary to associate them to understand how the parameters influence each other and how they affect the building's energy efficiency. The solution found to plot all the variables was through a Parallel Coordinates plot, as described in page 21 of the Methodology chapter. This type of plot, due to its characteristics allows the plot of the several variables' values of the buildings. The figure 3.19 shows the parallel coordinates plot made from the DH dataset.

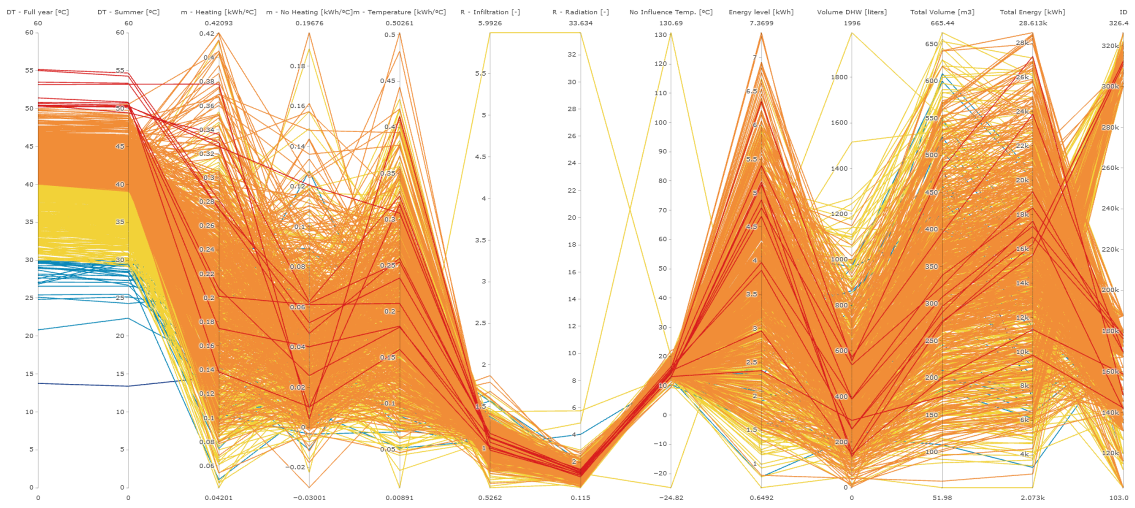


Figure 3.19. Parallel coordinates plot from DH dataset.

In the plot, it was considered several variables. Each variable was considered due to specific information that provides to the utility company. The temperature colour gradient used is the same as the one from the figure 3.9, where the high mean yearly temperature differences, $\overline{\Delta T}$, are red and the lowest temperature differences are represented in blue. Each variable is described below:

1. **DT - Full year:** The mean temperature difference, in °C, calculated in a year's span;
2. **DT - Summer:** The mean temperature difference, in °C, calculated at the Summer's season, as described in page 14 in the Methodology chapter;
3. **m - Heating:** The slope value of the linear regression made for all three seasons: Fall, Winter and Spring. These values are negative and expressed in kWh/°C, but in the plot are described as positive values;
4. **m - No Heating:** The slope value of the linear regression made for the Summer's season. These values are negative and expressed in kWh/°C, but in the plot are described as positive values;
5. **m - Temperature:** The slope value of the linear regression made when considering the outdoor conditions where there is no solar radiation and the wind speed is low. These values are negative and expressed in kWh/°C, but in the plot are described as positive values;
6. **R Infiltration:** The ratio regarding the infiltration/ventilation component of the heating demand, calculated from the slopes of the linear regressions for all the three seasons Fall, Winter and Spring, as described in page 15 in the Methodology chapter;
7. **R Radiation:** The ratio regarding the solar gains component of the heating demand, calculated from the slopes of the linear regressions for all the three seasons Fall, Winter and Spring, as described in page 15 in the Methodology chapter;
8. **No Influence Temp.:** The estimated outdoor temperature, in °C, when the people from that moment on turn off their space heating systems, as described in page 15 in the Methodology chapter;
9. **Energy Level:** The estimated energy usage, in kWh, when the outdoor temperature is equal to 0°C, as described in page 15 in the Methodology chapter;

10. **Volume - Summer:** The estimated daily hot water consumption, in litres, at the Summer season, as described in page 14 in the Methodology chapter;
11. **Total Volume:** The total hot water consumption, in m^3, in a year's span of each building;
12. **Total Volume:** The total energy usage, in kWh, in a year's span of each building from the smart energy meters;
13. **Building's ID:** The identification number attributed by the utility company to distinguish the buildings;

The variable 1. and 2. are the mean temperature differences between the supply and return temperature of the hot water distributed by the DH network for a year's span and the Summer's season. The yearly mean $\overline{\Delta T}$ quantifies how significant is the temperature drop by the heating/DHW production systems of each building, where large drops may imply energy-efficient systems in the dwellings. And small drops can indicate dwellings where their systems do not perform efficiently. The Summer period averaged $\overline{\Delta T}$ indicates in most of the cases the temperature efficiency of the DHW heat exchanger, due to, in Summer, most of the heating systems are not in operation; therefore the temperature drop occurs only on the DHW heat exchanger. By having small temperature differences in this last variable or a big difference between these variables, it can indicate to the utility company that the problem of a specific building for their energy inefficiency is from the DHW production system.

The variables 3. and 4. are the slopes of the linear regressions originated from the scatterplots between the hourly-energy usage and their outdoor temperature for the heating and non-heating seasons. The slope of the heating season indicates the influence that the outdoor temperature has in the building's energy usage, where high values may imply that a dwelling is highly dependent on the outside's temperature, which can also indicate that this type of building may have large transmission losses and/or high infiltration/ventilation rates. The second variable represents the linear's regression slope for the Summer season. For most of the buildings, this value is close to zero, 0, which indicates a dwelling that its energy demand in the Summer is not dependent on the outdoor temperature. This variable when significantly higher than zero, provides to DH company the information of the dwellings which do not shut-off their space heating system during Summer, which can be one of the reasons why a specific building may have high total energy usage measured by their smart energy meter. The variable 5. indicates the linear regression slope when the outdoor conditions of no solar radiation and low wind speed are met. Because of these conditions, the slope will indicate only the transmission losses component in the overall energy demand, allowing to identify the buildings with large transmission losses through their envelope.

The variables 6. and 7. are the ratios described in page 15 in the Methodology chapter. These ratios are a relation between different slopes from linear regressions of the heating season by filtering the dataset when certain conditions are met. These ratios are calculated in order to quantify the influence of the infiltration/ventilation, and solar gains have on the energy demand. When the infiltration ratio is equal or above to one, it indicates that specific residence energy demand is highly dependent on the existing infiltration/ventilation component. From these large values, the utility company can see which buildings may have high infiltration/ventilation rates or even if the people have the practice of opening their windows with the heating systems operating. The radiation ratio indicates to the utility company, which are buildings have their energy usage is highly/lowly dependent on solar radiation. The ratios below one represent buildings that are influenced by solar gains, which may imply that these buildings have large window-to-wall ratio and/or that most of their windows may be facing South. By knowing beforehand that the more energy-efficient buildings or the newly constructed buildings conform with the new building regulations that focus in buildings being built more airtight and by having large windows to receive more radiation during the day, with these ratios, it is possible to identify which are these types of buildings.

The variables 8. and 9. are also from the linear regressions originated from the scatterplots between the hourly-energy usage and outside temperature. The first variable is the outdoor temperature value that separates the heating season from the non-heating season. Where the buildings with high values represent houses when the heating systems were shut-off with high outdoor temperatures. The second variable is denominated "Energy level" and represents the required energy when the outdoor temperature is equal to zero. These two variables can also be used to indicate which buildings are more energy-efficient by having low no-influence outdoor temperatures (variable 8) and low energy levels (variable 9).

The variable 10. is the daily hot water consumption of each building in the Summer season, which was also plotted in bubble map in the figure 3.16. This variable indicates to the DH company, which are the buildings that are high/low water consumers, which in Summer, are mainly dependent on domestic hot water. If the DH company has access to the number of rooms per dwelling it may also find which buildings are consuming large amounts of hot water, when comparing what it should be expected. The last three variables are the total water consumption and total energy usage in a year and the building's identification number.

In order to demonstrate how advantageous it is to apply this type of plot in District Heating dataset, it was selected two buildings, where one of them have a high mean yearly temperature difference, and the other has a low $\overline{\Delta T}$ but with similar total energy usage. These two buildings are seen in figure 3.20.

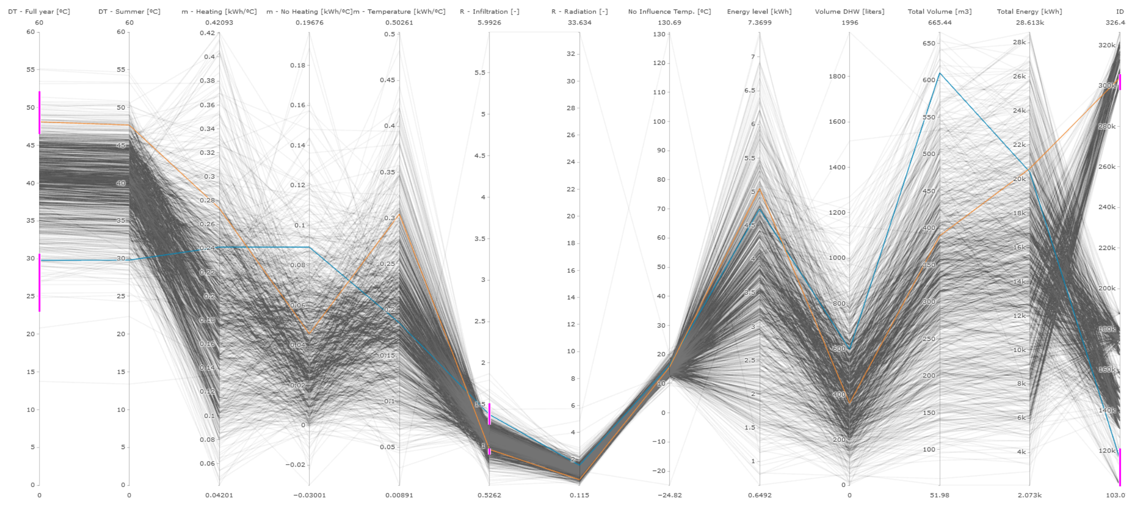


Figure 3.20. Parallel coordinates plot from DH dataset.

These two buildings have the building's ID: 116030 and 303914. When these two dwellings are selected in the parallel coordinates plot, their variables have the following values, as seen in table 3.3:

Table 3.3. Results from the buildings 116030 and 303914.

Building's ID	**116030**	**303914**
$\overline{\Delta T}$ (Year) [°C]	29.8	48.1
$\overline{\Delta T}$ (Summer) [°C]	29.8	47.7
m - Heating [kWh/°C]	0.241	0.274
m - No Heating [kWh/°C]	0.0891	0.0457
m - Temperature [kWh/°C]	0.186	0.304
R - Infiltration [-]	1.38	0.96
R - Radiation [-]	1.58	0.52
No Influence Out. Temp. [°C]	16.4	15.7
Energy Level [kWh]	4.75	5.05
Volume (Summer) [liters]	598	361
Total Volume [m^3]	611	389
Total Energy [kWh]	20,419	20,614

In table 3.3, the buildings' year $\overline{\Delta T}$ are significantly different from each other, which indicates that the low-temperature difference case has an inefficient heating system when comparing with the high $\overline{\Delta T}$ dwelling. This is concluded since both buildings have approximately the same amount of total energy usage but different total water consumption. The building 116030 needs more than 1.5 times of water to fulfil the same energy demands than building 303914. Regarding their summer $\overline{\Delta T}$, both cases have similar $\overline{\Delta T}$ when considering the different time periods. Which indicates that, in both cases, the DHW production system is not responsible for the large differences between the houses in terms of yearly $\overline{\Delta T}$ but the house 116030 should be studied in terms of its DHW heat exchanger due to its low $\overline{\Delta T}$. The daily water consumption for the Summer is also larger for the 116030 case than for the 303914 case, which may imply that the residence 116030 has a larger DHW consumption than comparing with the residence 303914, due to more people living in the house, having more DWH consumers or just because of its DHW production system's efficiency.

Regarding the linear regressions obtained when the energy demands are correlated with the weather conditions, it is seen that the energy demands of the building 303914 are influenced more by the outdoor temperature than the building 116030. This is concluded due to the building 303914 has higher slope values on the Heating and Temperature regressions, which might indicate that the building 303914 has an overall transmission losses coefficient higher than the building 116030. From the small slope values of the regression made for the non-heating season, it is seen that both buildings do not use their heating system in this system. And the dwelling 303914 turned off their heating system before than the dwelling 116030, which is also confirmed by the no influence outdoor temperature being lower for 303914 than for the 116030.

The infiltration/ventilation ratio and radiation ratio are lower than one for the 303914 case, indicating that this dwelling is highly dependent on the solar gains and not that much influenced by the wind speed. In this case, the solar gains affect the energy demand of the dwelling greatly because the value is not only below one, it is 0.52 demonstrating that the solar radiation in the building decreases the dependence between the outdoor temperature and the energy considerably. This case might be a dwelling have large window's areas or that most of them are facing South. The building 116030 is the opposite of 303914, where it is highly dependent on wind speed and not much influenced by solar radiation. The 116030 case might be a dwelling does not receive much solar radiation, due to most of its windows are facing North or that may be covered most of the time by internal blinds or external objects that shadow them. Also due to R_{inf} is 1.38, it may also imply that this building has significant infiltration/ventilation rates because of the building might not be airtight enough or that the people in the house open their windows often with space heating system operating.

From an online research about these two dwellings, it is verified that most of these conclusions are correct. The building 303914 has large windows exposed to the South, and the building 116030 have just a few small windows. Regarding the year of construction, both buildings were built before 1970, which may be the reason for high transmission losses (303914) and high infiltration rates (116030). These two dwellings have the same number of rooms, but this information does not deny totally the assumption that high daily water consumption during Summer may be due to more users, can just be that the house

303914 does not have currently that many people as the dwelling 116030, even though is just a supposition and cannot be confirmed by the data. Another information is that the building 116030 was not built initially connected to the DH network but was provided with an individual heating system, and the building 303914 since its construction was directly connected to the DH network. This might be the reason for the large difference between the dwellings' $\overline{\Delta T}$ where the residence 116030 after being connected with district heating, their space heating and DHW production systems were not optimised for this type of solution.

3.3 Data Clustering

This section of the thesis is regarding the application of the k-means clustering method to identify existing groups of buildings that are similar in their characteristics. This method's results section was divided into three parts, depending on different filtered datasets from the original DH dataset or by a specific variable, as described in the Methodology chapter.

3.3.1 One-year variable

The first part concerns how many clusters are identified in yearly variable time-series. This is performed with the finality of detecting the buildings that have their parameters similar throughout all year's dataset. From section 3.2, it was seen that the variables that were not relatable to each other and that are both measured by the smart energy meters are the water-volume and the temperature difference between the supply and return water temperature of the DH network. Therefore both time-series variables of each building were used as a dataset to perform the K-means.

To apply the method, it is necessary to determine beforehand the optimal number of clusters. For this it was applied the Elbow method and Average Silhouette method which determine the best number of clusters for a dataset by taking into account different clustering properties. The results from these methods are from the figures 3.21 to 3.23:

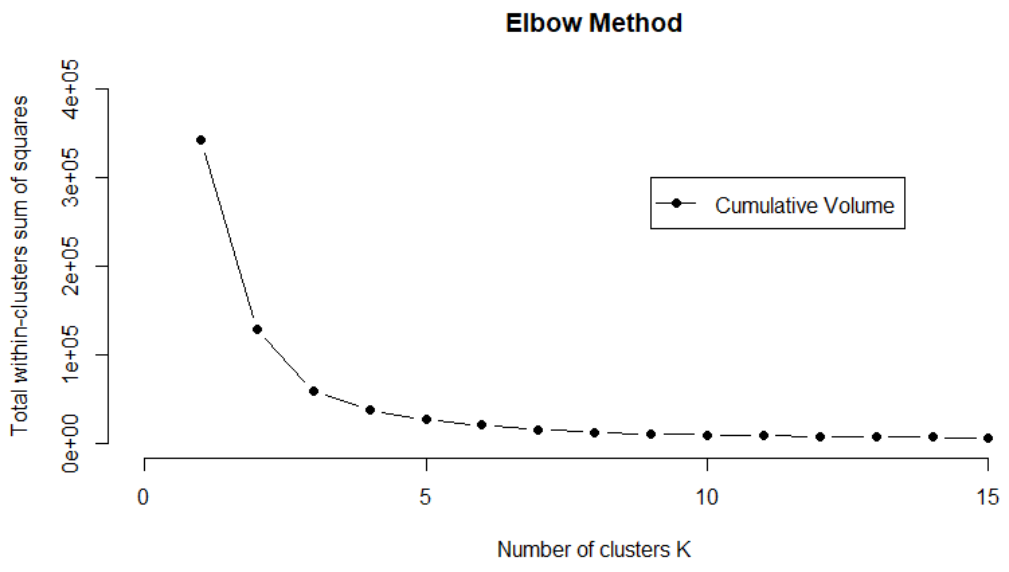


Figure 3.21. Results of Elbow method for cumulative water-volume.

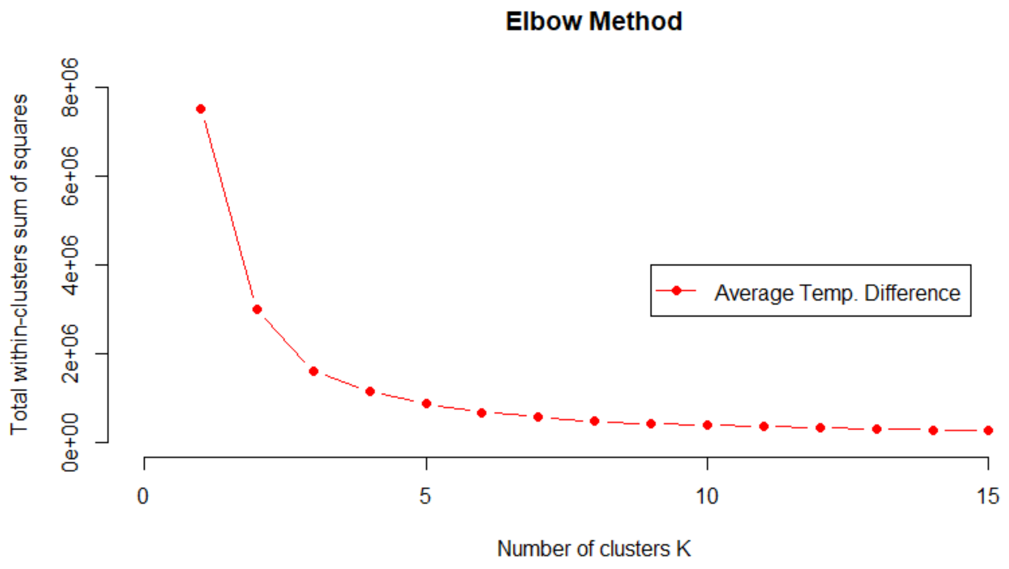


Figure 3.22. Results of Elbow method for hourly-average temperature difference.

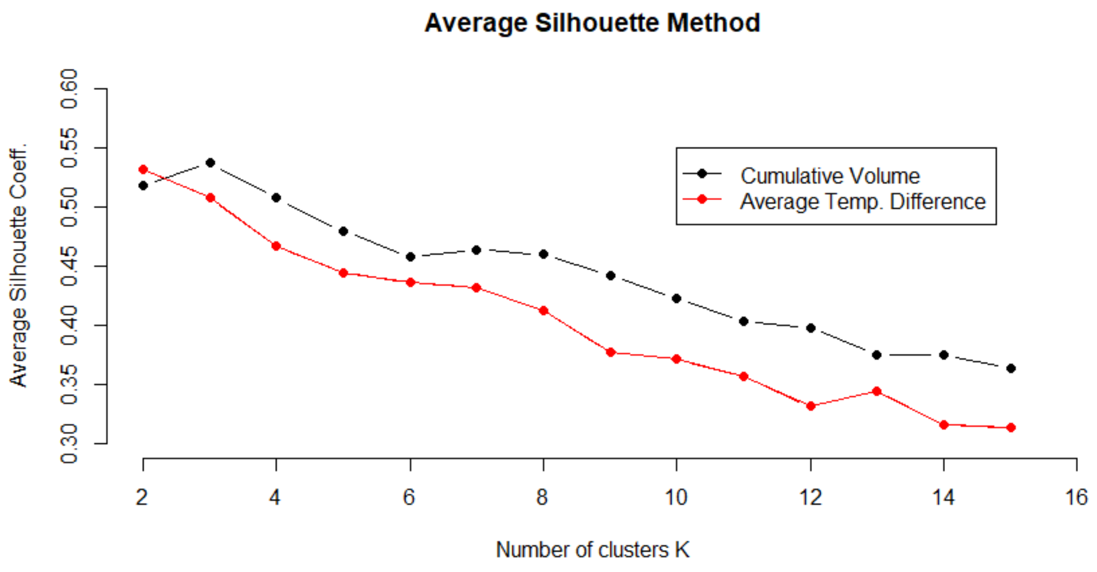


Figure 3.23. Results of Average Sillhouette method for cumulative water-volume and hourly-average temperature difference.

The figures 3.21 and 3.22 show that three is the number of clusters that will assure well-compacted clusters for both of the variables, due to the function's "knee" is when the number of clusters is equal to 3. The figure 3.23 shows the results from the average silhouette method in both variables. For this method, the most suitable cluster number is the one that maximises the average silhouette coefficient. The cumulative volume has its maximum average silhouette coefficient when that dataset is divided into three clusters, and for the temperature difference variable, its coefficient is maximum when the dataset is divided into two clusters. From these results, the K-means was applied considering 3 clusters. The K-means results are plotted in the figures 3.24 and 3.25:

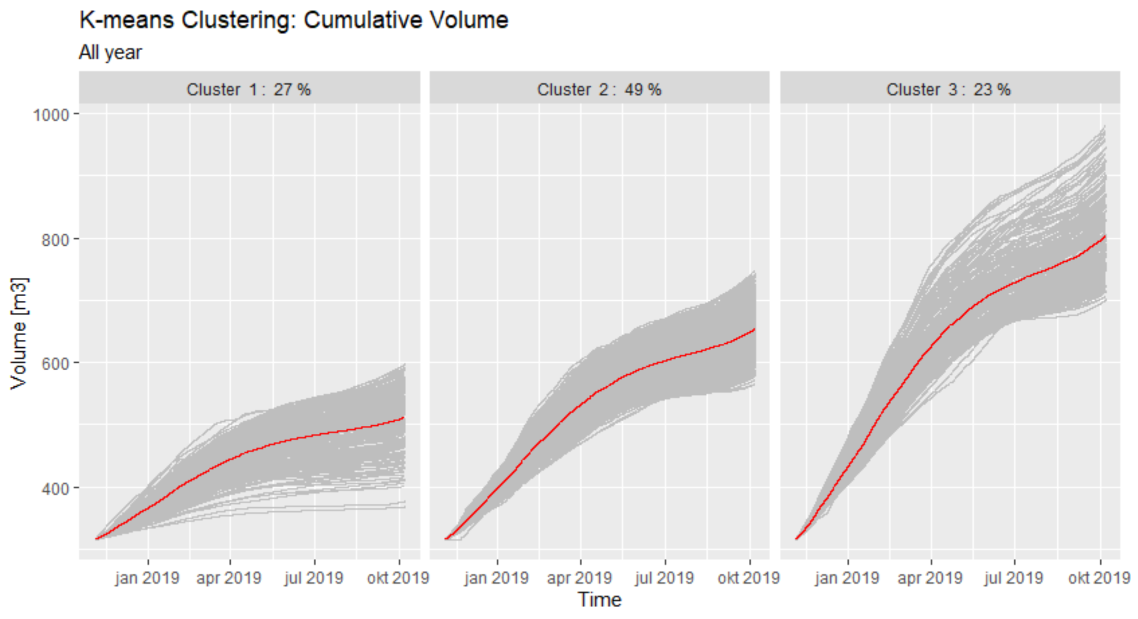


Figure 3.24. Results of K-means clustering for cumulative water-volume.

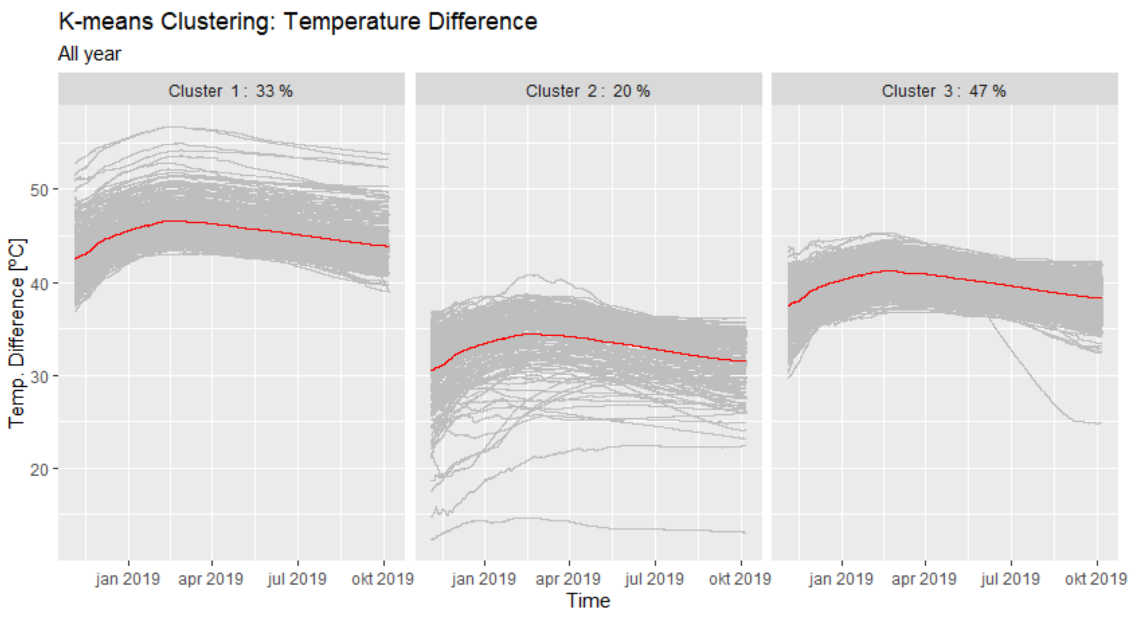


Figure 3.25. Results of K-means clustering for hourly-average temperature difference.

In both figures, the red lines are the centres of the clusters, which are the average values of the time-series existing in their specific cluster. On top of each cluster plot, it is given a number to identify the cluster group and the percentage of buildings inside the group. As a note, the Cluster 1 in figure 3.24 may not be constituted by the buildings that are present in Cluster 1 of figure 3.25, each method applied on each variable is performed independently from the other studied variables. The cumulative water-volume is divided into three clusters, as seen in figure 3.24, the results show that these three main groups vary in terms of the incremental slope of the cumulative volume, where the most significant difference is in the inclination of the heating season. The cluster 1 is represented by the buildings that have the lowest cumulative values in the DH dataset. Cluster 2 is the group of buildings with the medium cumulative values, and it groups most of the buildings present in the dataset, 49 %. The last group is cluster 3, and it is constituted by the buildings with the highest cumulative values.

The figure 3.25 is the results from the K-means of the hourly-average temperature difference in year's span. The results, when compared with the cumulative volume, show the clusters have slight differences in the percentages of the number of buildings present in each group. The most compact cluster in the results is the Cluster 3, that has 47 % of the buildings connected to the DH network, which might indicate a resemblance with cluster 2 of the cumulative volume. Cluster 1 is represented by the highest $\overline{\Delta T}$ values and cluster 2 is the group of clusters with the lowest $\overline{\Delta T}$ values, and it is also the group where the most irregular time-series are situated. This might mean that the heating systems with lower $\overline{\Delta T}$ values may have the more fluctuating $\overline{\Delta T}$ measurements.

3.3.2 Daily profiles

The K-means was also applied to determine the dwellings that have similar daily profiles of energy, volume, instantaneous temperature difference and flow. The profiles were made for each year's season and only considering the working days, Monday to Friday, where users' routine is more constant throughout the year. The optimal number of clusters was determined by the Elbow and Average Silhouette methods, and the results are the figures from 3.26 to 3.33:

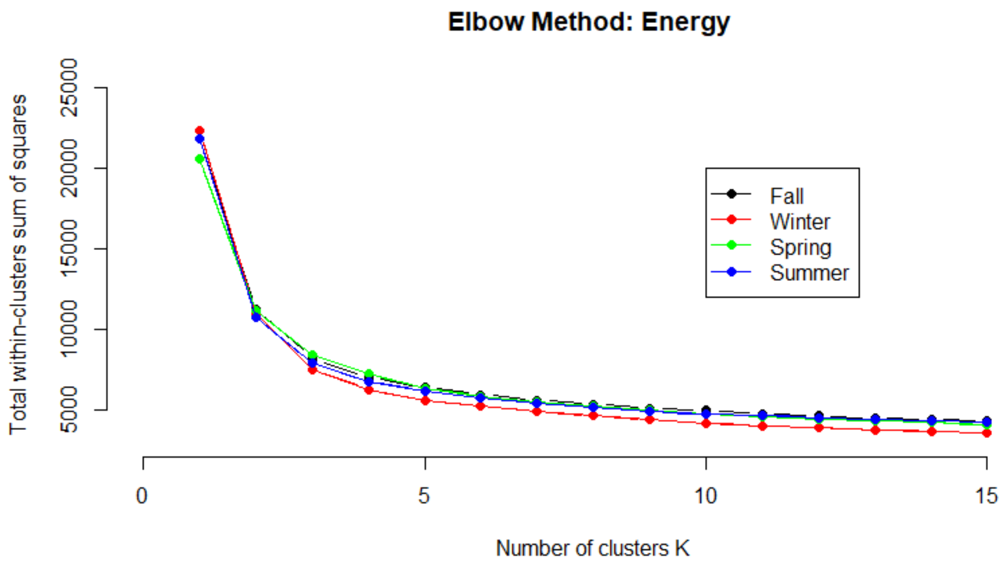


Figure 3.26. Results of the Elbow method for the energy profiles.

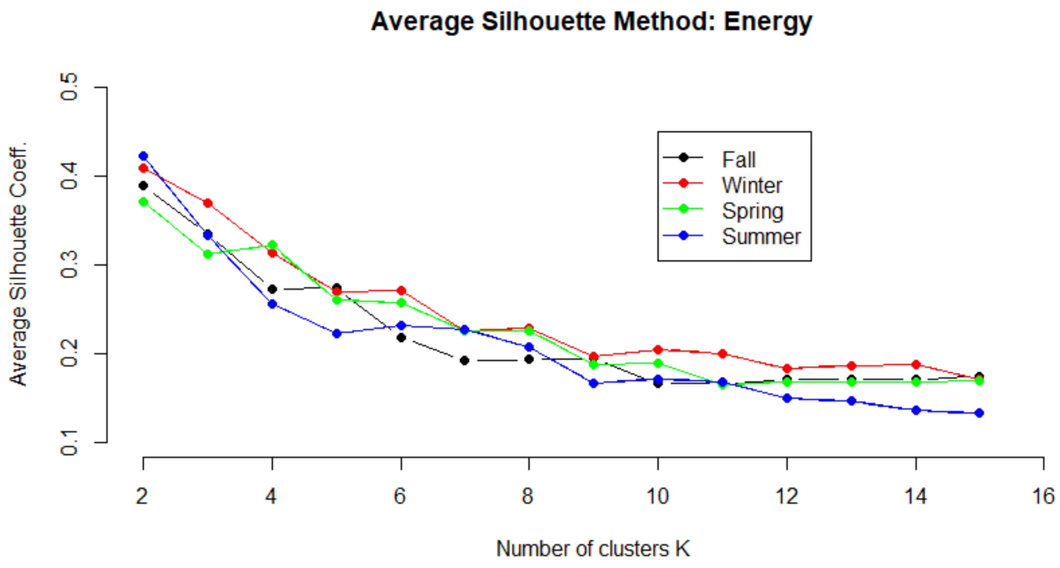


Figure 3.27. Results of the Average Silhouette method for the energy profiles.

The Elbow method for the energy profile in all seasons presented similar functions, where the optimal number of clusters is three. For the Average Silhouette method, the best number is two clusters, but because this the minimum number of considered clusters for the method, it is usually not considered if the other numbers are not that different. Therefore the best number of clusters is three for all the seasons with the exception of Spring, which the best number is four.

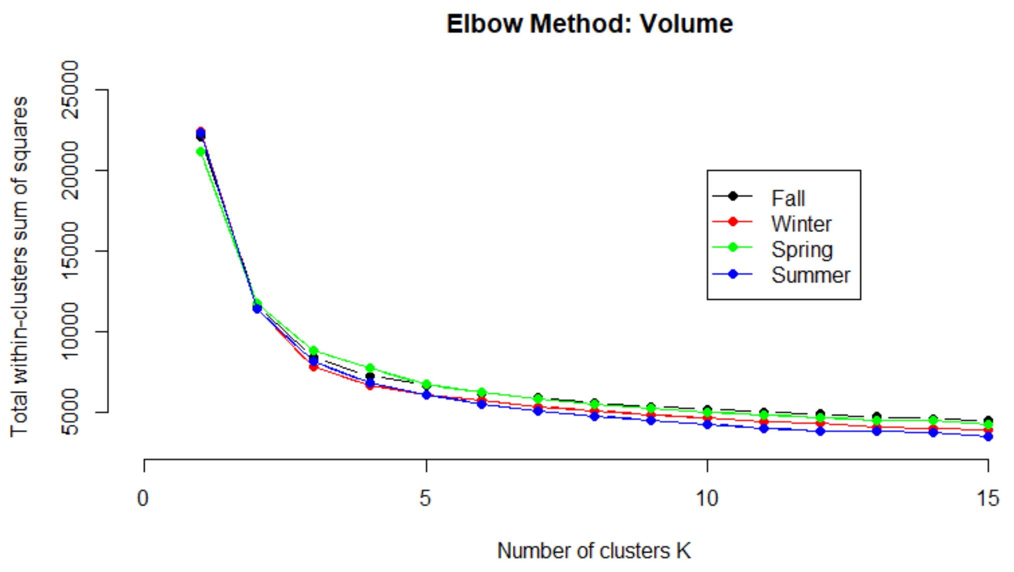


Figure 3.28. Results of the Elbow method for the volume profiles.

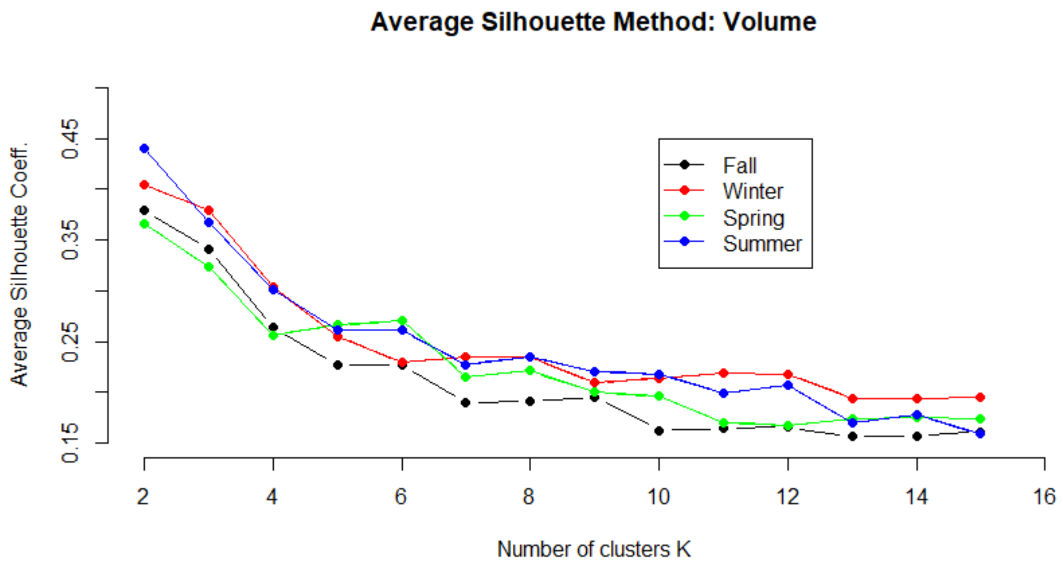


Figure 3.29. Results of the Average Silhouette method for the volume profiles.

The Elbow method for the volume variable in all seasons presented similar functions as in energy, where the optimal number of clusters is three. For the Average Silhouette method, the optimal solution is three clusters.

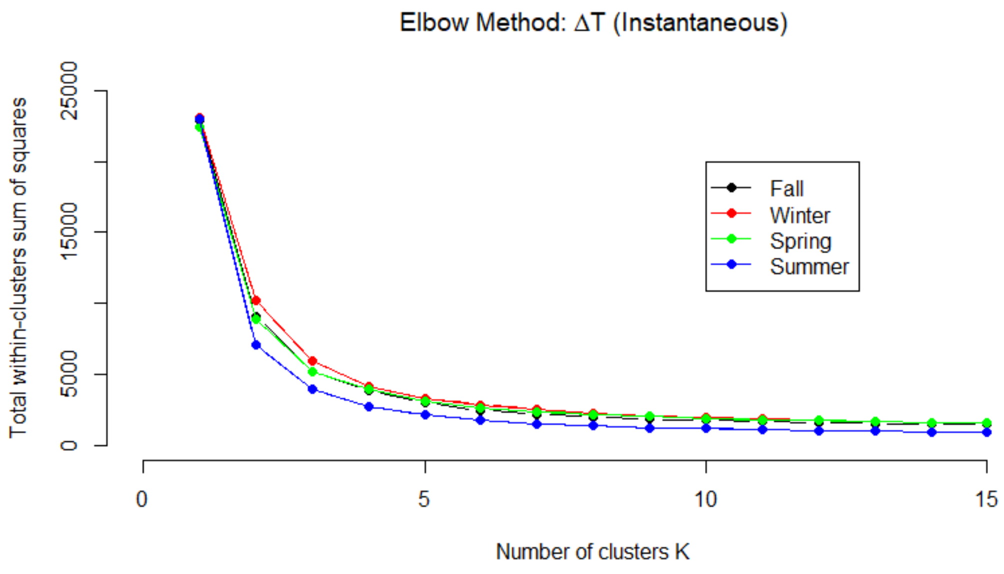


Figure 3.30. Results of the Elbow method for the temperature difference profiles.

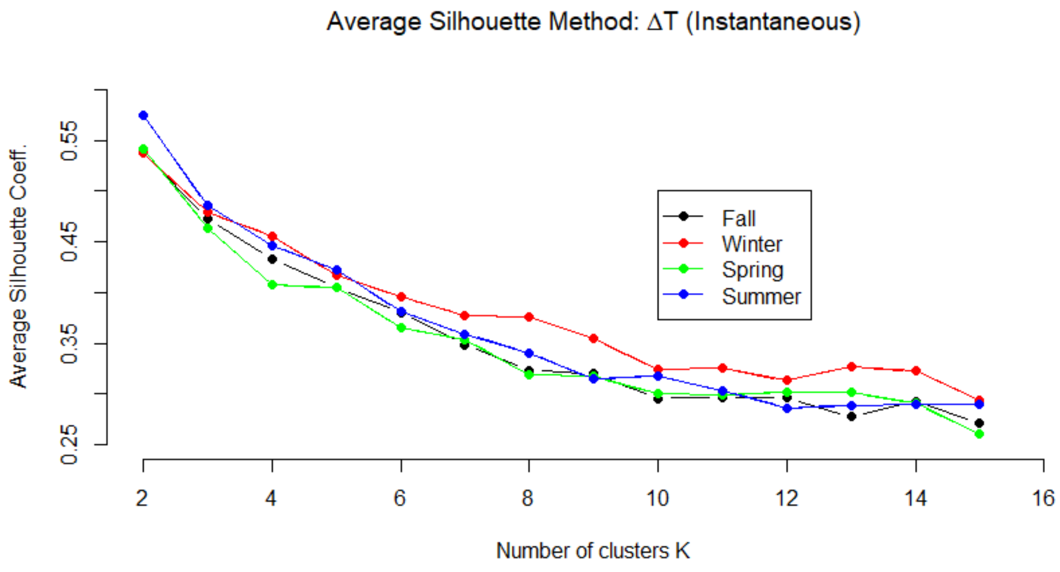


Figure 3.31. Results of the Average Silhouette method for the instantaneous temperature difference profiles.

The Elbow method for the instantaneous temperature difference profile in all seasons presented similar functions, where the optimal result of clusters is three. For the Average Silhouette method, the best number is three clusters as well, where their average silhouette coefficients are identical for each case.

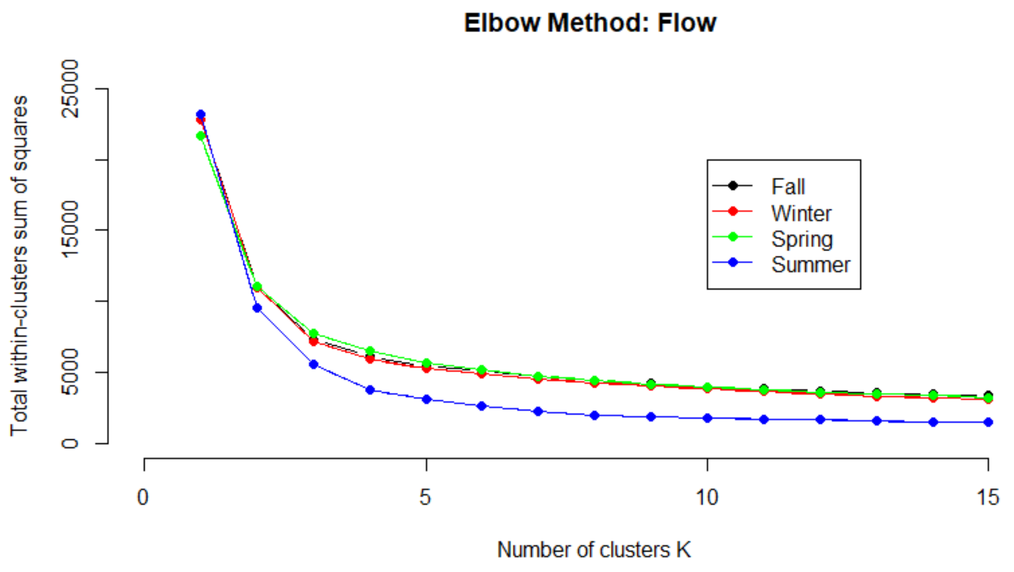


Figure 3.32. Results of the Elbow method for the flow profiles.

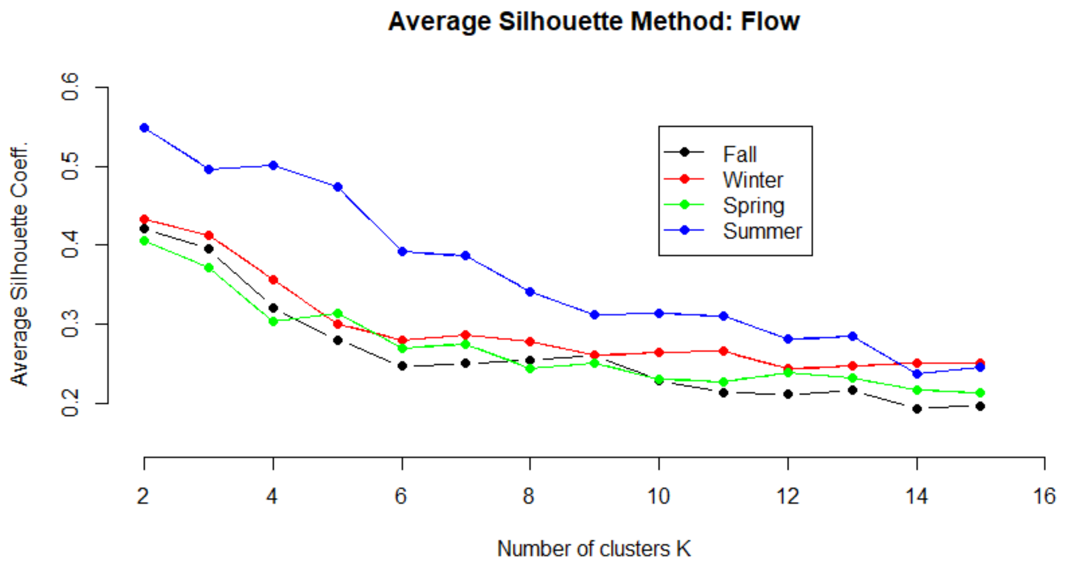


Figure 3.33. Results of the Average Silhouette method for the flow profiles.

The Elbow method for the flow in all seasons presented similar functions, where the optimal number of clusters is three. For the Average Silhouette method, the best number is three clusters for all the seasons except Summer, which the best clusters' number is four. From all applied methods, it was decided to use three clusters for all the K-means application cases, as it was used for the yearly variables. The results of the clustering method is in figures from 3.34 to 3.49 :

Daily profiles: Energy

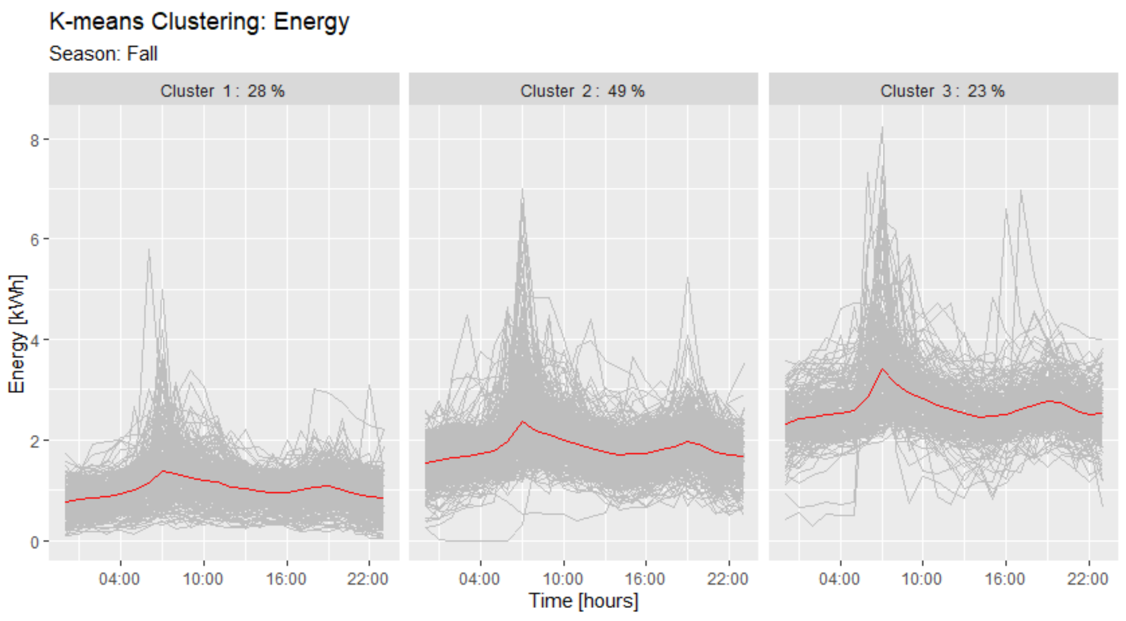


Figure 3.34. Results of K-means clustering for energy profiles in the Fall.

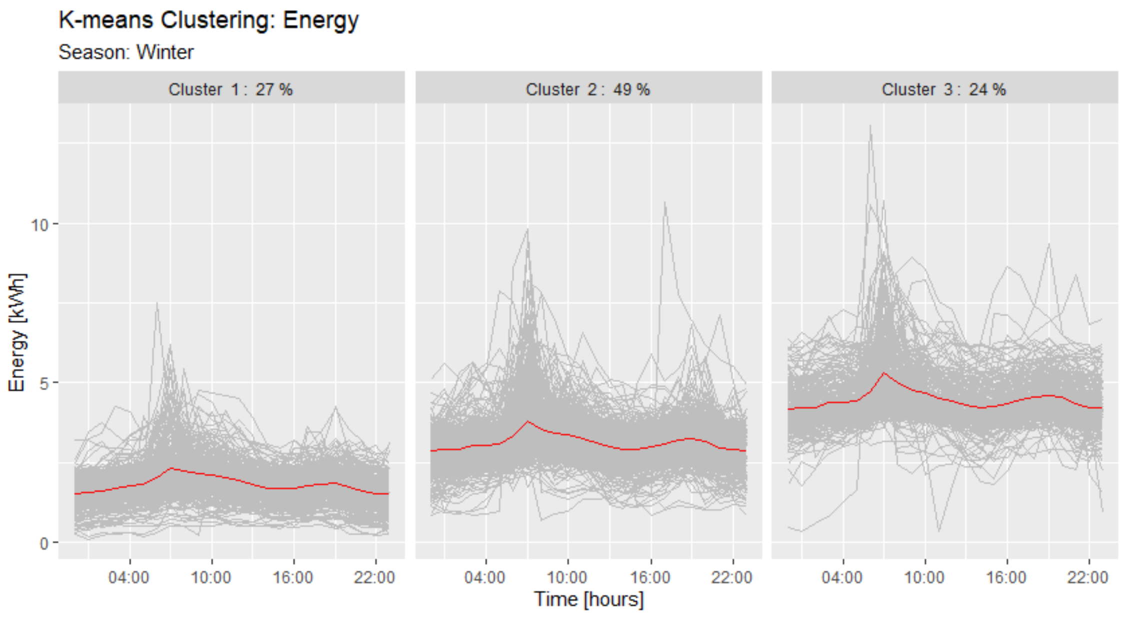


Figure 3.35. Results of K-means clustering for energy profiles in the Winter.

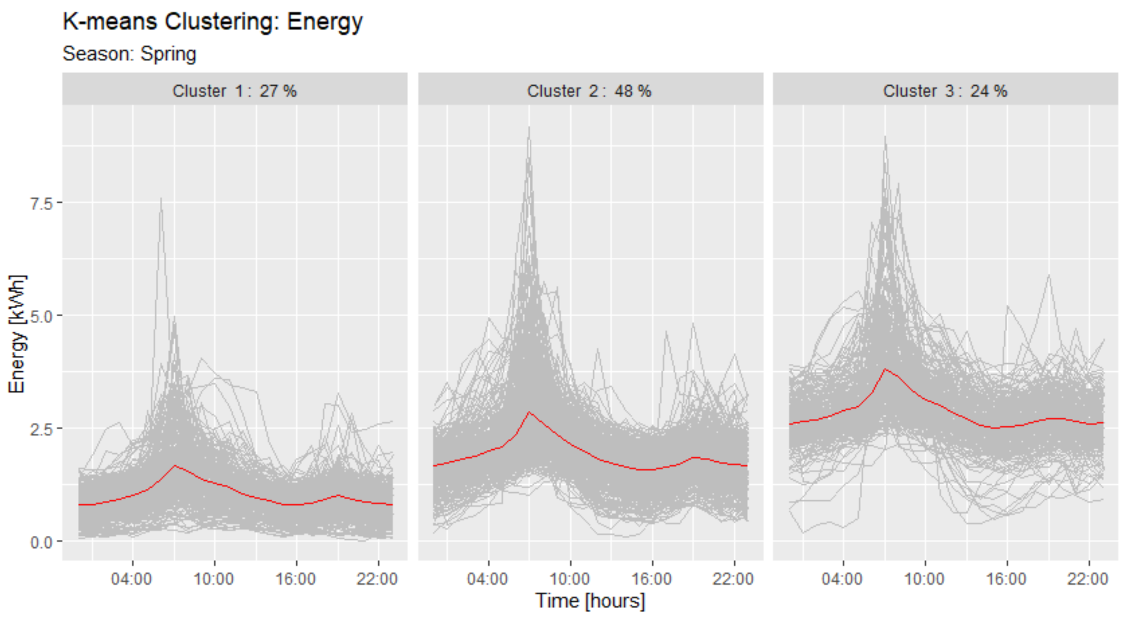


Figure 3.36. Results of K-means clustering for energy profiles in the Spring.

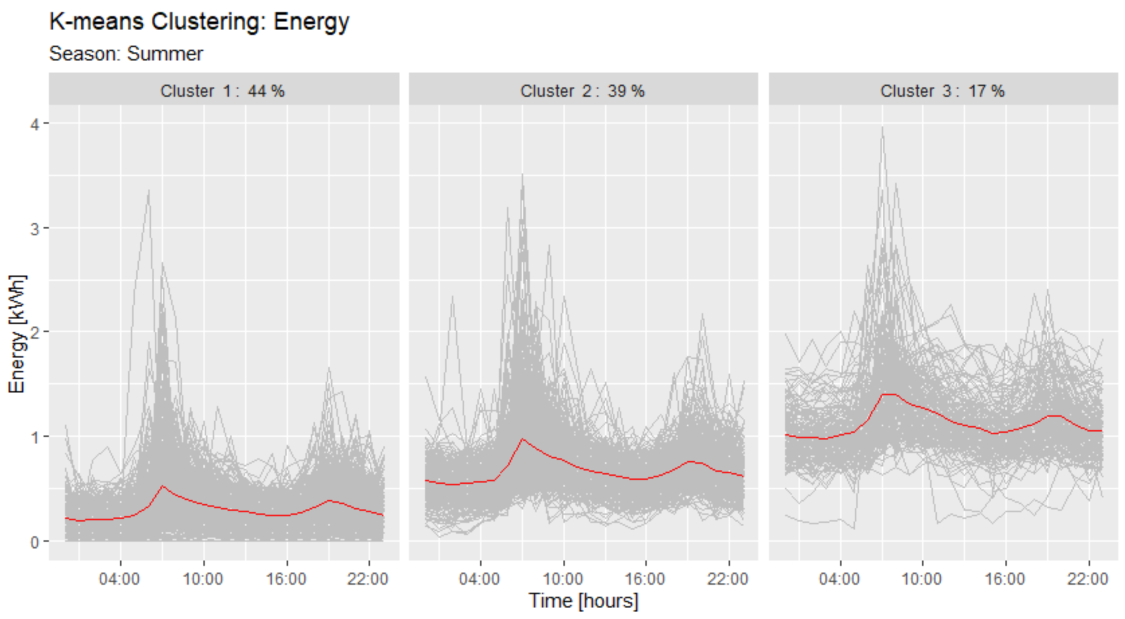


Figure 3.37. Results of K-means clustering for energy profiles in the Summer.

The energy profiles show that during working days for all seasons exists two energy peaks, where the first is the most significant peak situated at 7:00 hours and the second smaller peak is between 18:00-20:00 hours. The first is originated from the usual awakening hour of a person and the second is produced due to home arrival of the users after work. Each cluster group presents the same peaks with different magnitudes, where Cluster 1 is composed by the buildings with the lowest daily energy demands and the Cluster 3 is the one with the buildings with the largest demands. All the seasons presented similar percentages of buildings' amount in each cluster where the largest cluster is the number 2 (medium energy demand), except the Summer season which the largest cluster is 1 (low energy demand). This might be due to some of the users turn off the space heating during the non-heating season.

Daily profiles: Volume

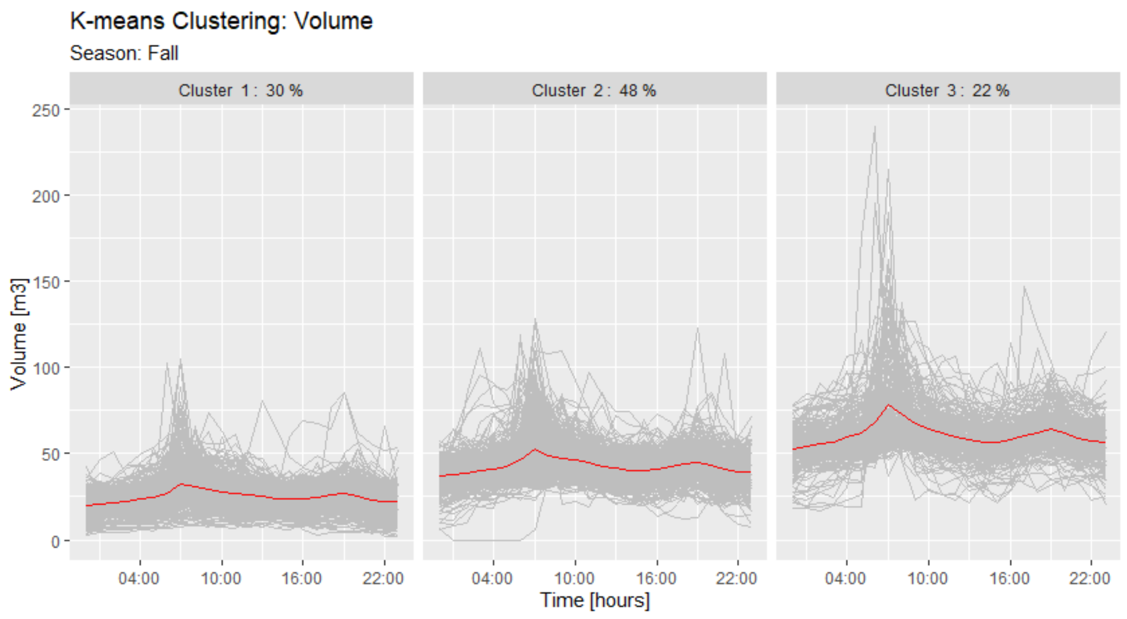


Figure 3.38. Results of K-means clustering for volume profiles in the Fall.

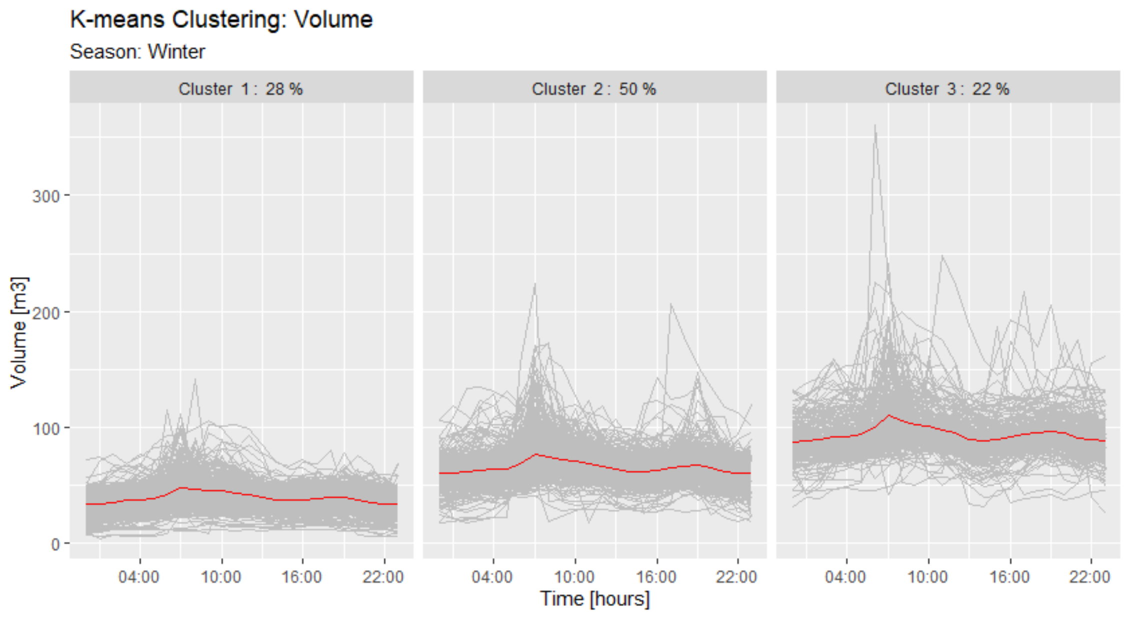


Figure 3.39. Results of K-means clustering for volume profiles in the Winter.

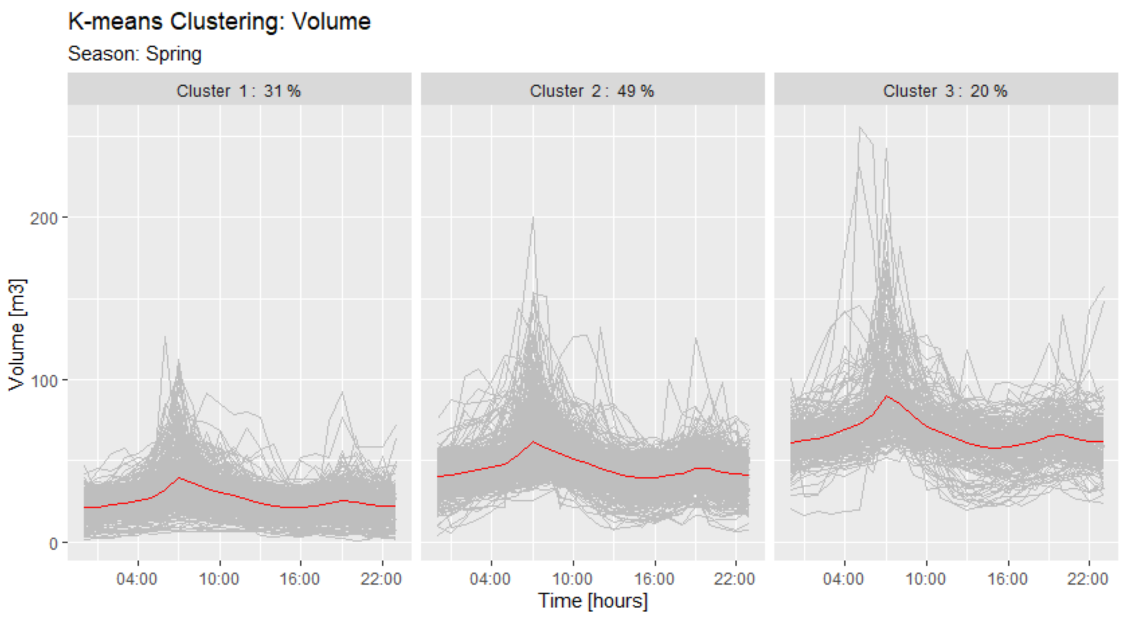


Figure 3.40. Results of K-means clustering for volume profiles in the Spring.

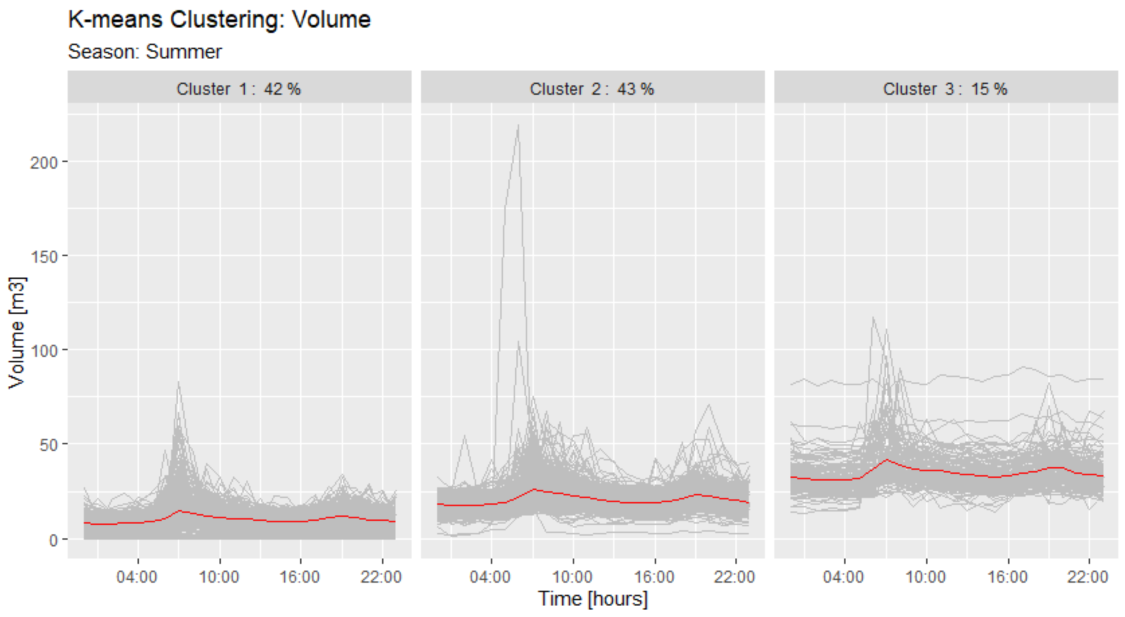


Figure 3.41. Results of K-means clustering for volume profiles in the Summer.

The profiles generated from the water-volume consumption show that during working days for all seasons exists two major consumption peaks, as it happens in the energy profiles. This corroborates the concept in figure 3.4 at the page 32, where the volume and the energy are highly correlated. Each cluster group presents the same peaks with different magnitudes, where cluster 1 is composed by the buildings with the lowest daily volume demands and the cluster 3 is the one with the buildings that have the largest demands. All the seasons presented similar percentages of buildings' amount in each cluster where the largest cluster is the number 2 (medium energy demand), except the Summer season which the largest cluster is also 2 (medium volume consumption) but followed by the Cluster 1 (low consumption) with a difference of 1 %. This might be due to some of the users turn off the space heating during the non-heating season, as explained for the daily energy profiles.

Daily profiles: ΔT

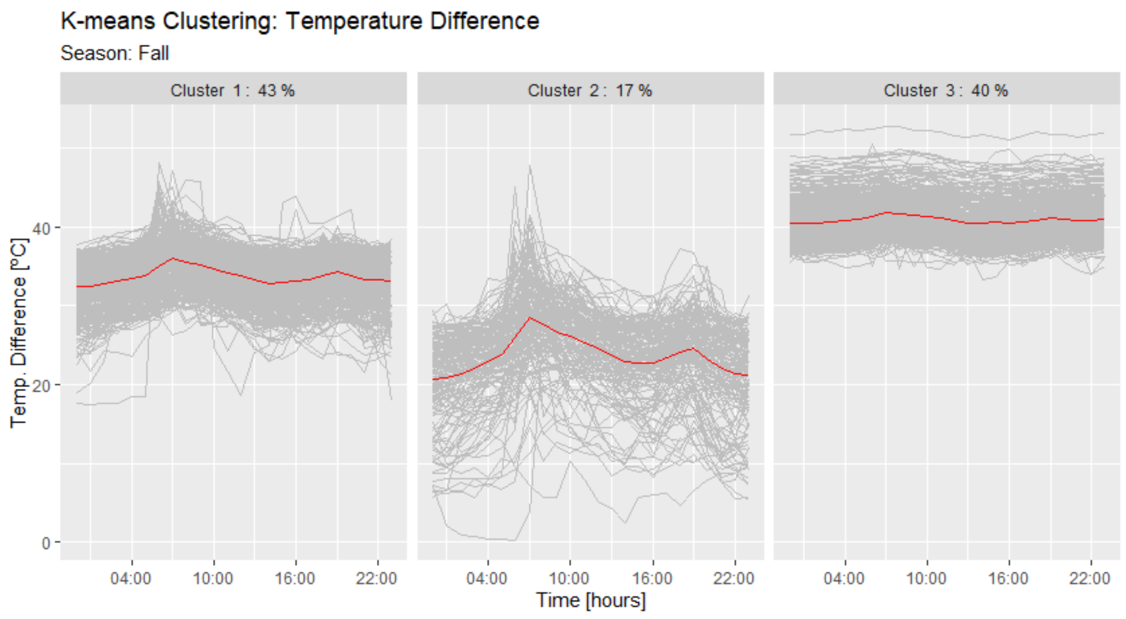


Figure 3.42. Results of K-means clustering for instantaneous temperature difference profiles in the Fall.

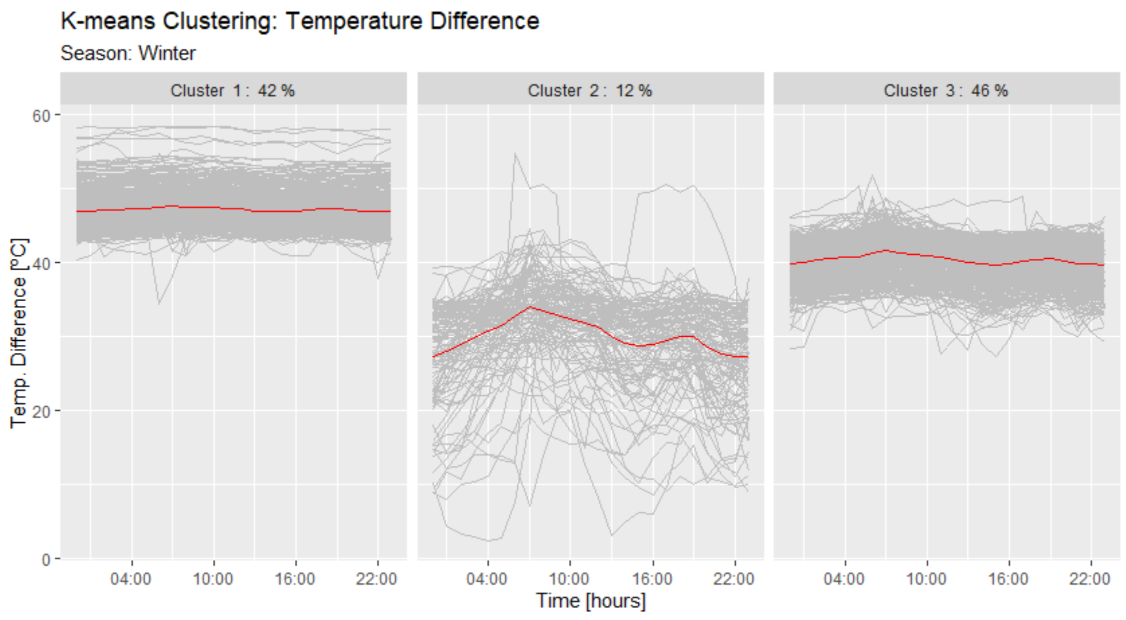


Figure 3.43. Results of K-means clustering for instantaneous temperature difference profiles in the Winter.

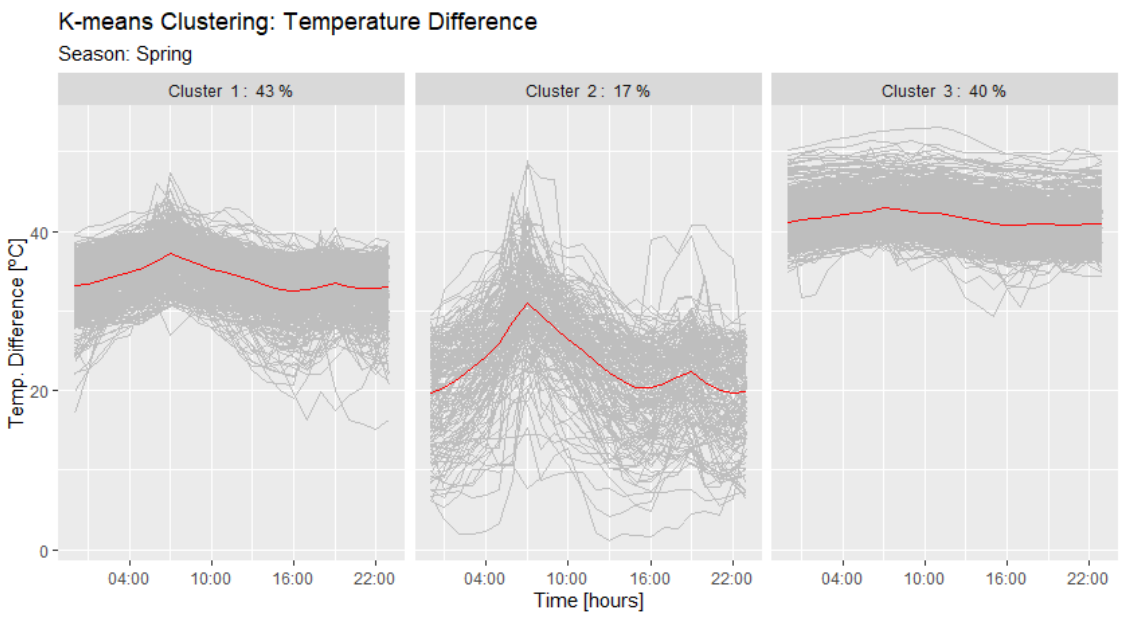


Figure 3.44. Results of K-means clustering for instantaneous temperature difference profiles in the Spring.

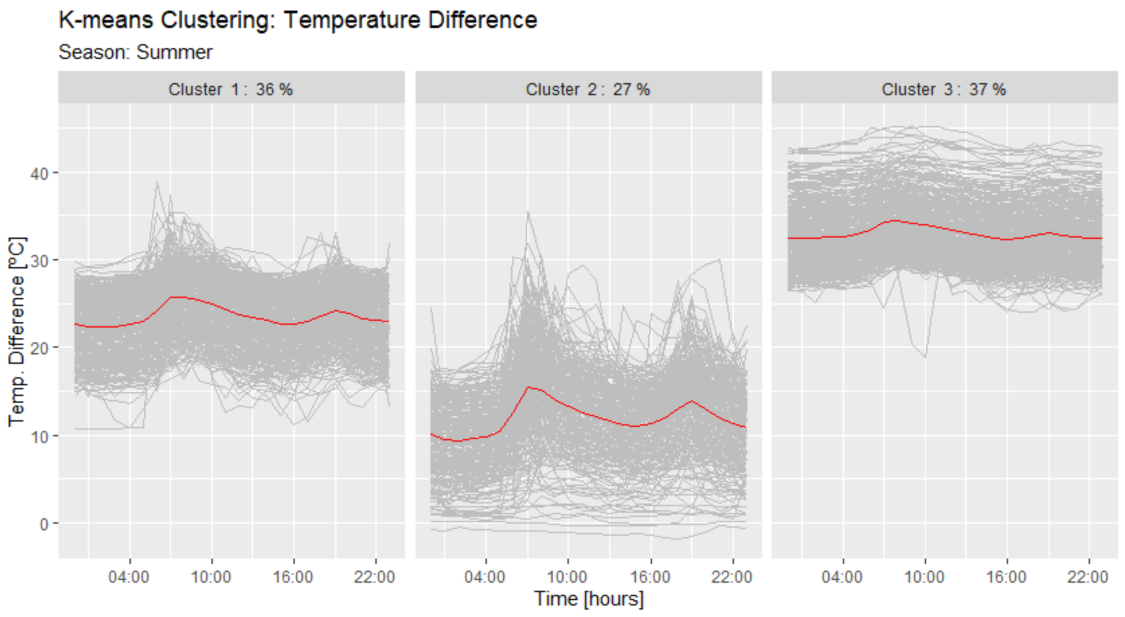


Figure 3.45. Results of K-means clustering for instantaneous temperature difference profiles in the Summer.

By analysing the temperature difference profiles, the same patterns seen in the other variables are generated. Revealing therefore that the energy peaks form consequently proportionally temperature difference peaks. This means that when the energy demands increase, the water consumption and temperature difference increases, consequently. And to have an increase on ΔT to meet the demands, the return temperature decreases. Each cluster group presents their peaks ate the same hours with different ΔT values. Regarding the clusters' distinction, it is related by the magnitude of ΔT, where cluster 1 represents the medium ΔT levels, followed by low ΔT buildings in cluster 2 and the highest ΔT

profiles in cluster 3. In all seasons, the majority of buildings are in the clusters 1 and 3, medium and high ΔT profiles. At Summer, it is seen that more buildings changed from a higher ΔT level cluster to the lowest level, cluster 2. From the results, it is also seen that the size of the temperature's peak is related to the ΔT level. Where the lower the ΔT profile is, the higher the peaks' magnitude is, and consequently, the more fluctuating is the ΔT throughout the day.

Daily profiles: Flow

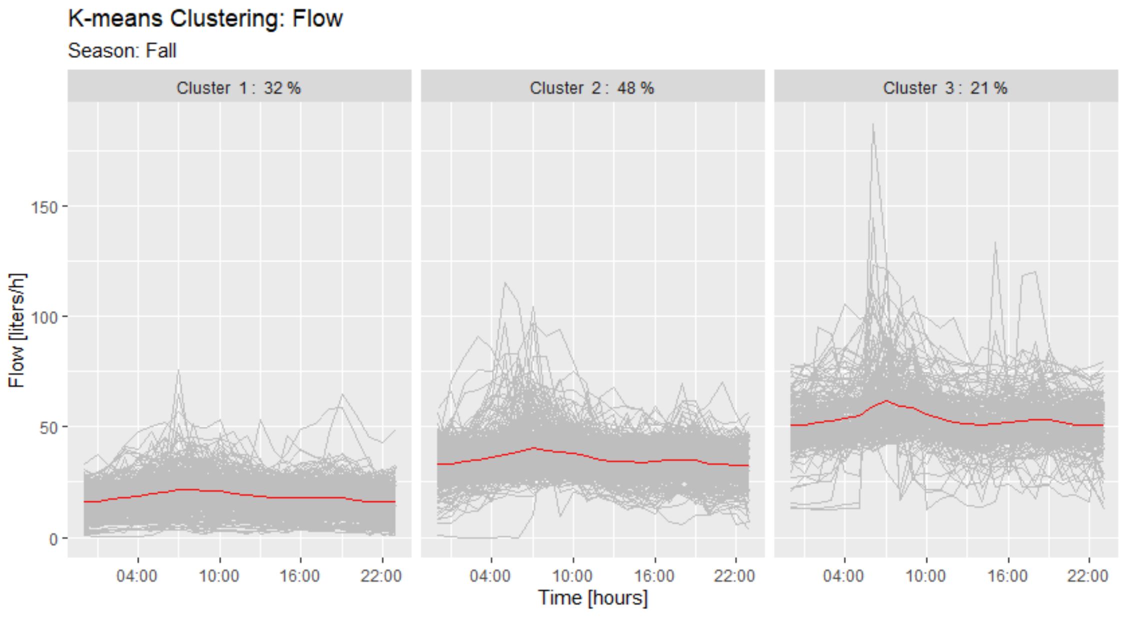


Figure 3.46. Results of K-means clustering for flow profiles in the Fall.

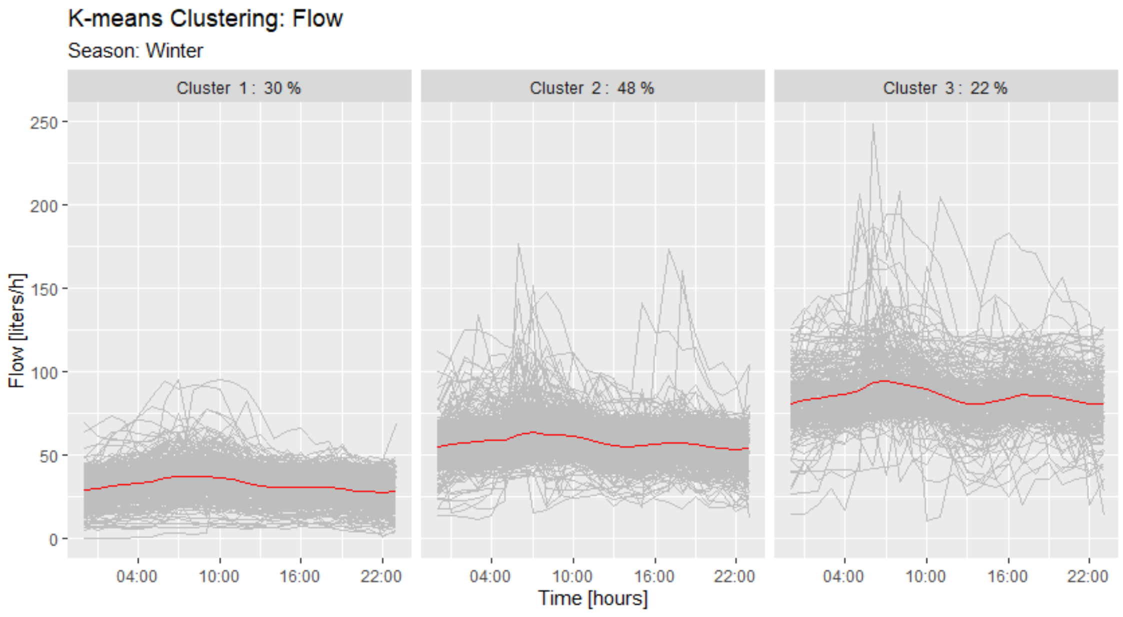


Figure 3.47. Results of K-means clustering for flow profiles in the Winter.

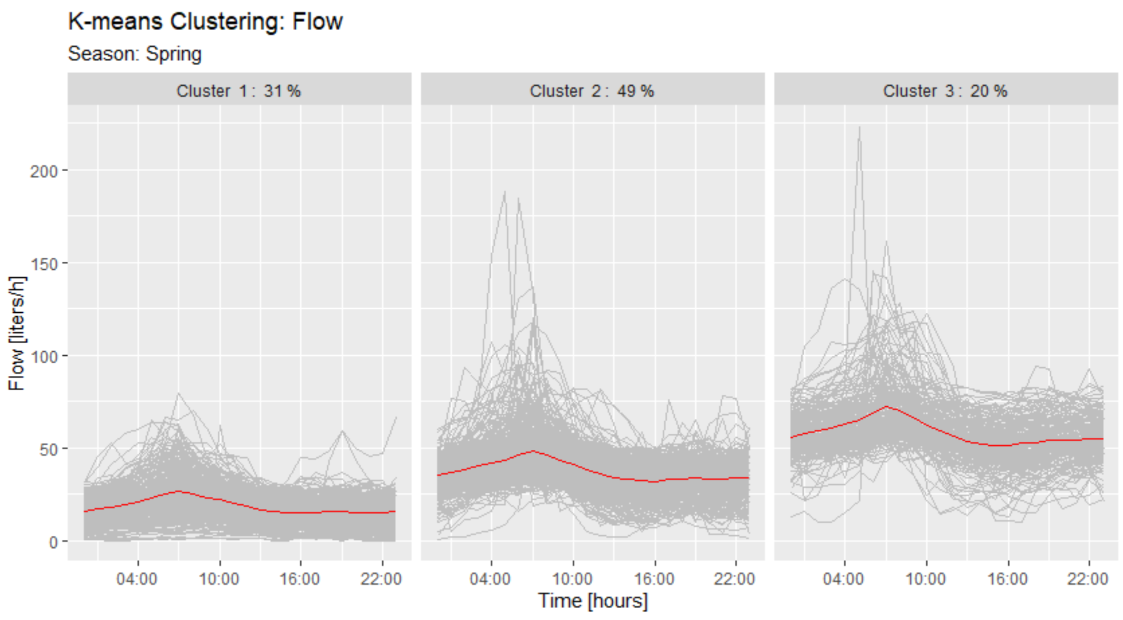


Figure 3.48. Results of K-means clustering for flow profiles in the Spring.

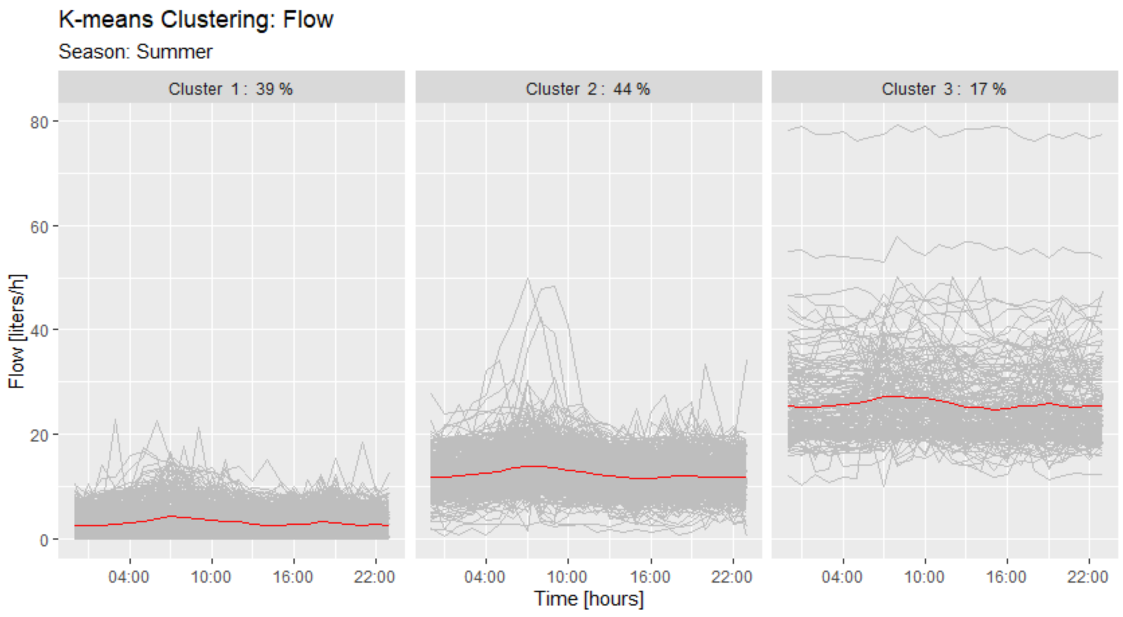


Figure 3.49. Results of K-means clustering for flow profiles in the Summer.

The flow profiles, the same patterns are seen as in the other variables. Where the two major flow peaks are at 7:00 hours and 18:00-20:00 hours. Each cluster group presents the same peaks with different magnitudes, where cluster 1 is composed by the buildings with the lowest flow measurements and the cluster 3 is the one with the buildings that have the highest water flows. Regarding the flow profiles, the amount of buildings per cluster is constant in the Fall, Winter and Spring, where in the Summer some of the buildings moved from high flow clusters to the lowest flow cluster, cluster 1. From the results, it is also seen that the size of the flow peak is related to the flow level. Where the higher the flow profile is, the higher the peaks' magnitude are, and consequently, the more fluctuating is the flow throughout the day. This observation is the opposite of what is seen in the temperature difference profiles.

3.3.3 Daily profiles - ΔT level

In section 3.2, the buildings are categorised by their average temperature difference throughout a year, $\overline{\Delta T}$. In order to see if exists any significant differences that may exist in each temperature level, it was applied the K-means method for each subset of buildings that have similar $\overline{\Delta T}$. As seen from the results above, the Winter, Spring and Fall have similar results. Therefore it was only performed this analysis for the Winter and Summer seasons. The variables chosen in this analysis are the instantaneous temperature difference and flow. The Elbow and Average Silhouette methods determined the optimal number of clusters, and the results are the figures from 3.50 to 3.57:

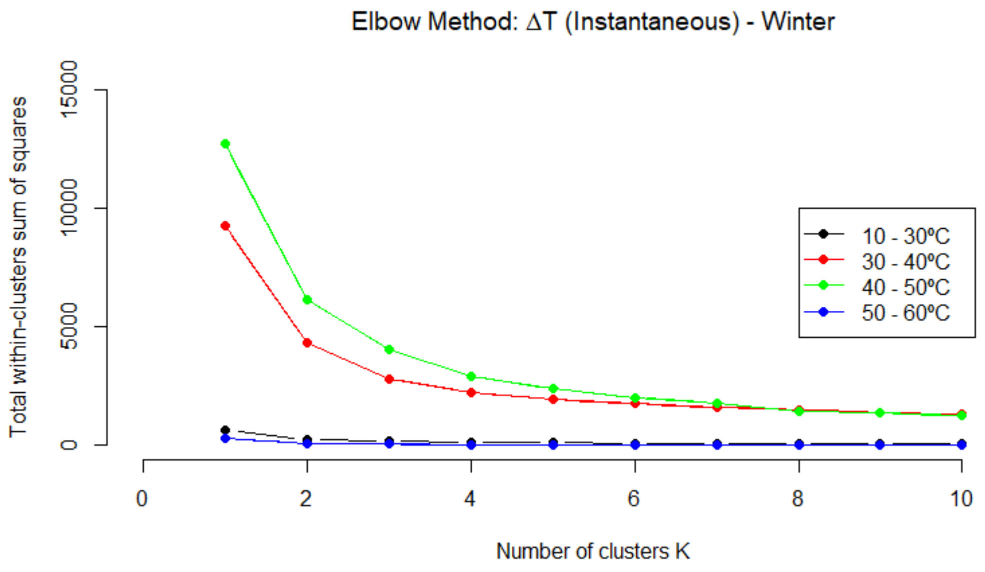


Figure 3.50. Results of the Elbow method for the instantaneous temperature difference profiles in Winter.

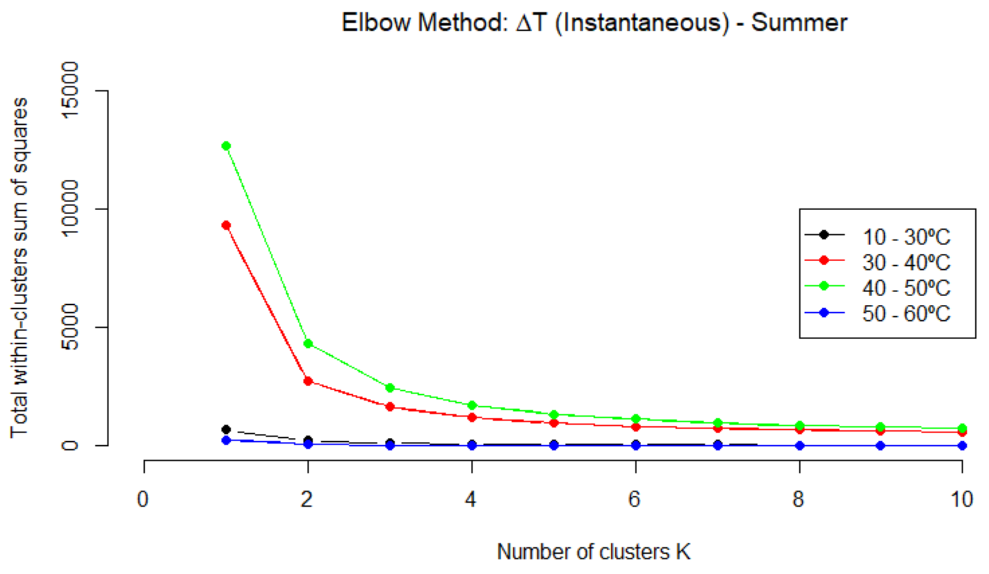


Figure 3.51. Results of the Elbow method for the instantaneous temperature difference profiles in Summer.

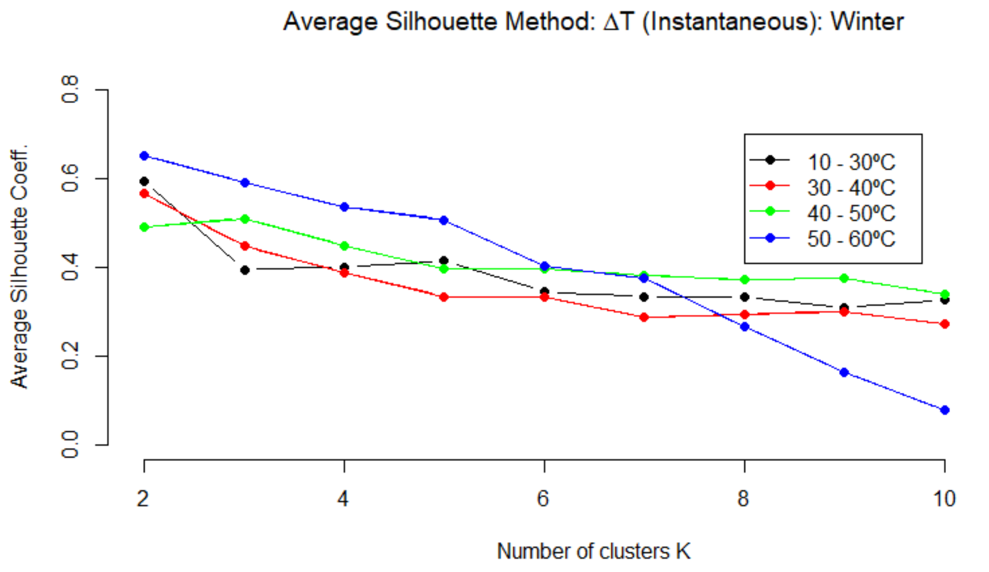


Figure 3.52. Results of the Average Silhouette method for the instantaneous temperature difference profiles in Winter.

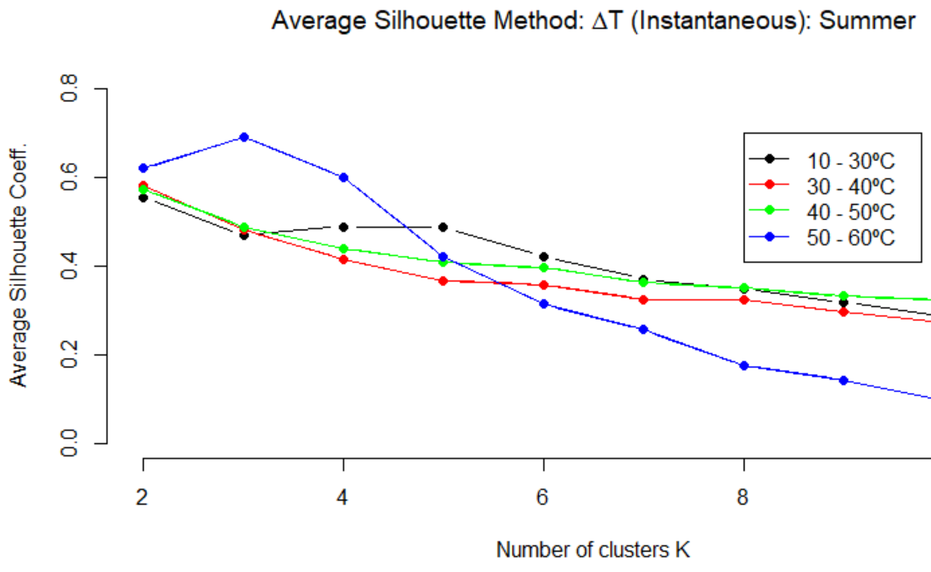


Figure 3.53. Results of the Average Silhouette method for the instantaneous temperature difference profiles in Summer.

The methods for the instantaneous temperature difference shown that the most suitable number of clusters is two. But because the number three also presented good results and to follow the above K-means results, it was also decided to use three to run the K-means.

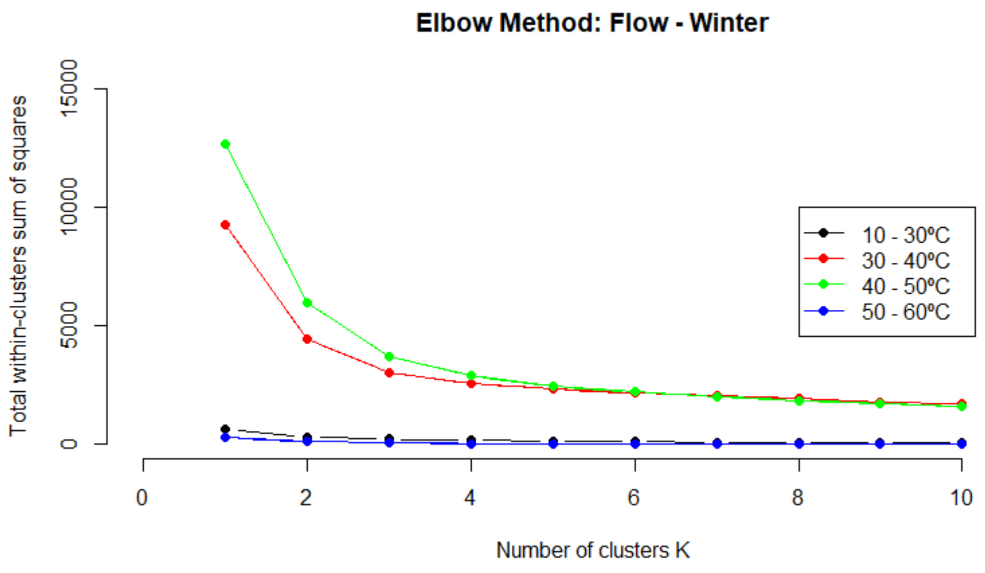


Figure 3.54. Results of the Elbow method for the flow profiles in Winter.

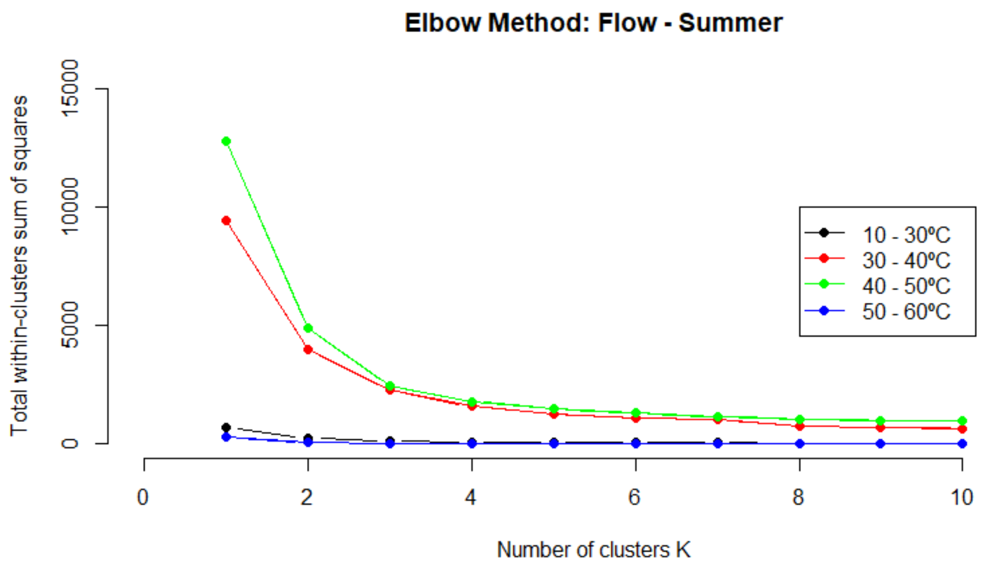


Figure 3.55. Results of the Elbow method for the flow profiles in Summer.

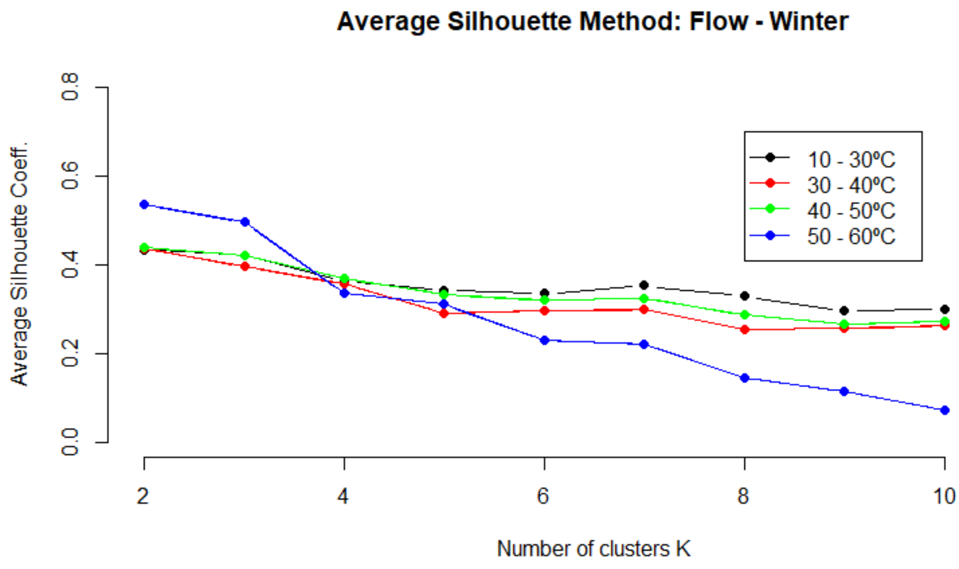


Figure 3.56. Results of the Average Silhouette method for the flow profiles in Winter.

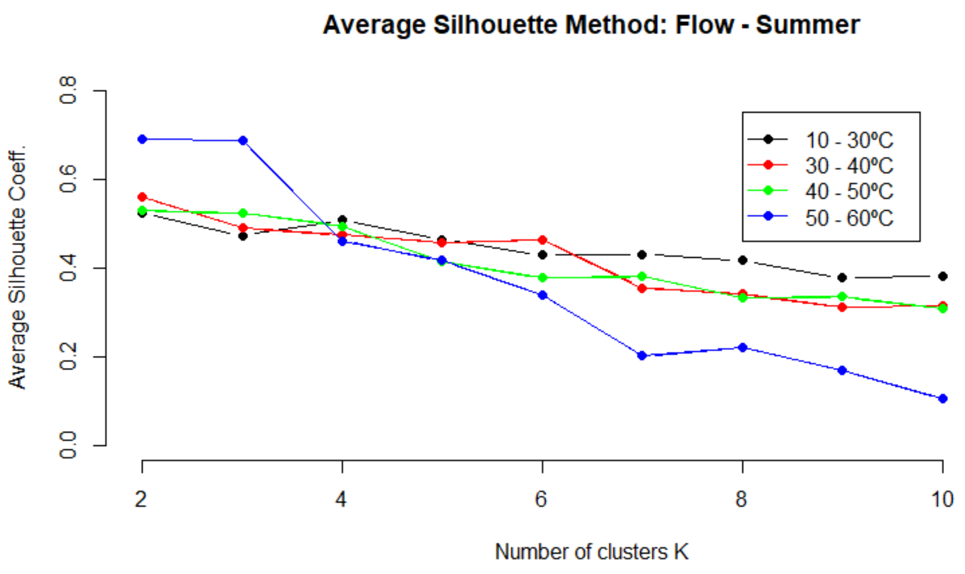


Figure 3.57. Results of the Average Silhouette method for the flow profiles in Summer.

The results for the variable flow also shown that the most suitable number of clusters is two. But for all the K-means applications be congruent with each other, it was decided to use three clusters, instead of two. By having these values determined the K-means method can be applied.

Daily profiles in Winter - Instantaneous ΔT

This set of K-means results are for the Winter season, and it is divided according to the different average ΔT levels during a year measured in the buildings.

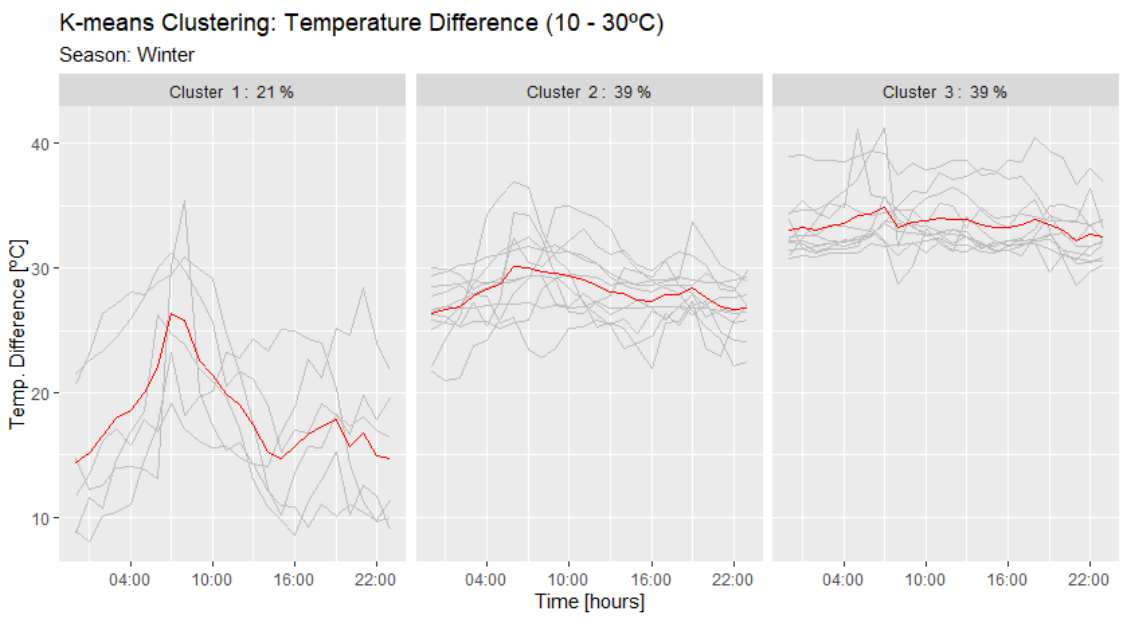


Figure 3.58. ΔT = 10 - 30 °C: Results of K-means clustering for instantaneous temperature difference profiles in the Winter.

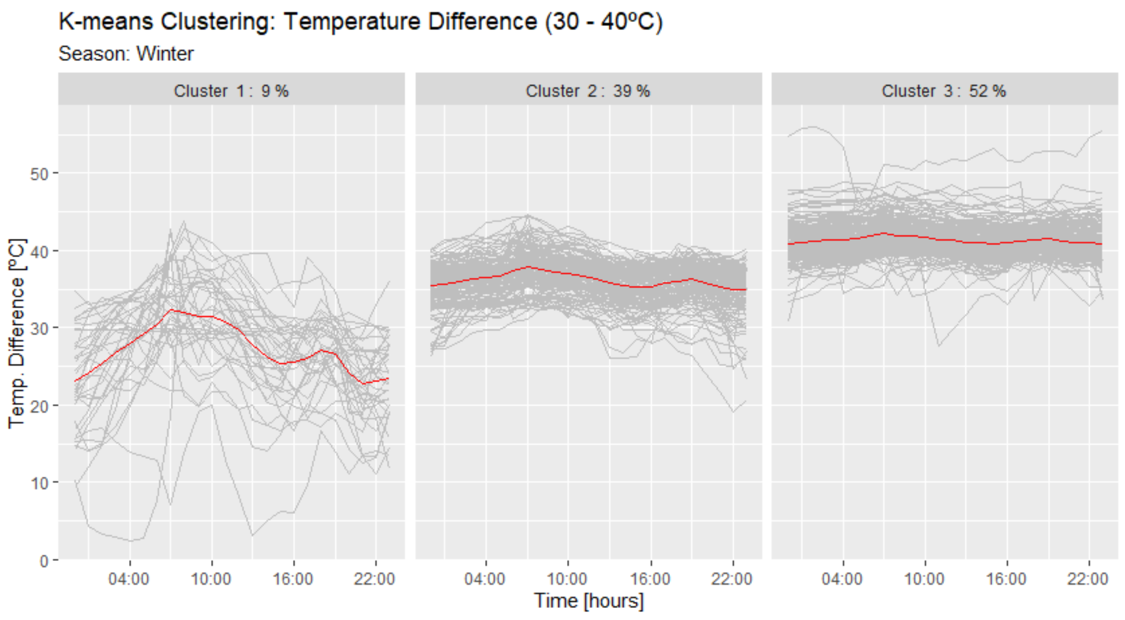


Figure 3.59. ΔT = 30 - 40 °C: Results of K-means clustering for instantaneous temperature difference profiles in the Winter.

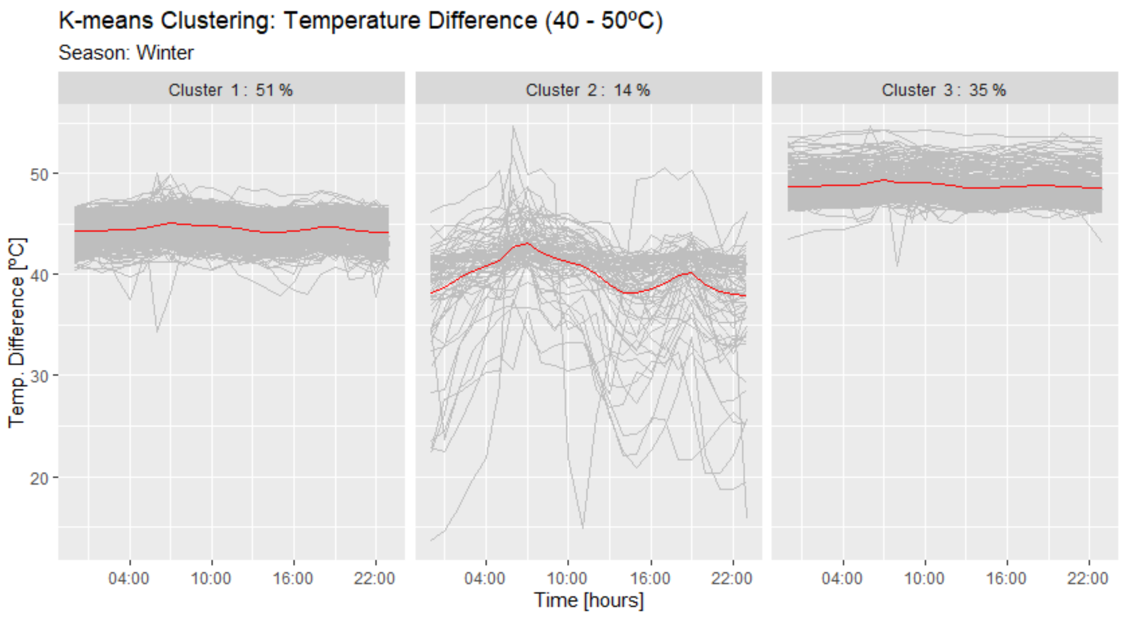


Figure 3.60. ΔT = 40 - 50 °C: Results of K-means clustering for instantaneous temperature difference profiles in the Winter.

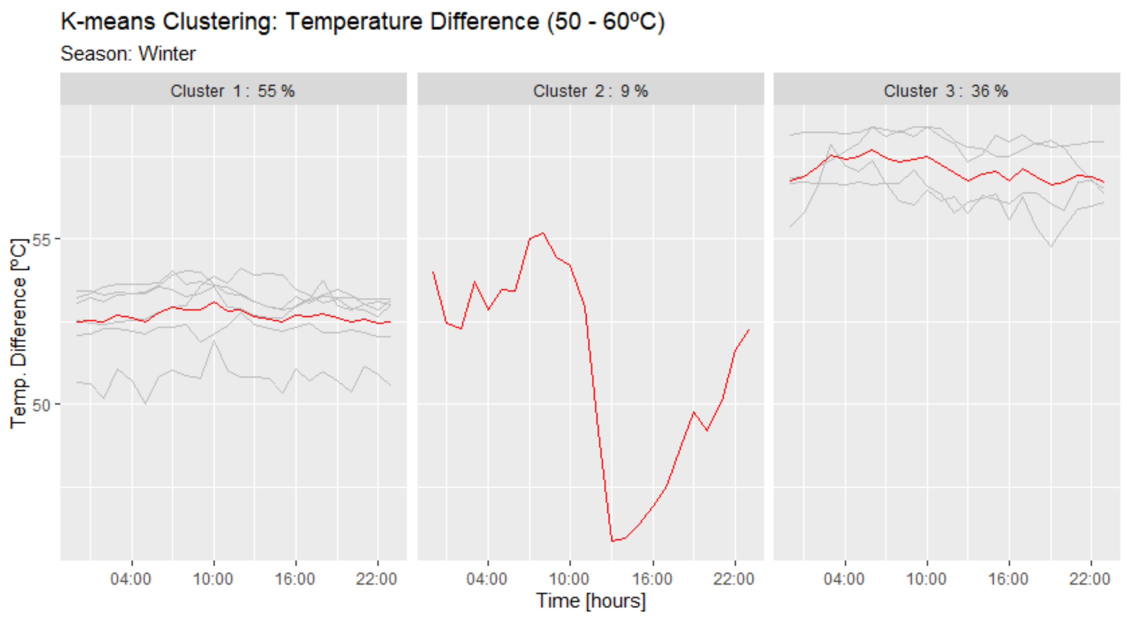


Figure 3.61. ΔT = 50 - 60 °C: Results of K-means clustering for instantaneous temperature difference profiles in the Winter.

The figure 3.58 is the buildings that their ΔT is between 10 °C and 30 °C. The two major clusters are the 2 and 3, where its profiles are similar to each other only varying its ΔT magnitude. Cluster 1, is composed of the buildings with the most fluctuating profiles and that have the largest peaks of the subset. The figures 3.59 and 3.60 are constituted by the buildings that their ΔT are in the temperature difference groups 30 - 40 °C and 40 - 50 °C. These two datasets presented similar results in terms of their clusters, where each cluster have similar profiles when the two groups are compared. The figure 3.61 is constituted by the buildings with an annual mean temperature difference between 50 and 60 °C. This group is composed of only a few buildings of the DH dataset. The clusters 1 and 3 have similar profiles, and the cluster 2 is composed of only one building which its daily ΔT profile is extremely different from the other building's profiles.

Daily profiles in Summer - Instantaneous ΔT

This set of K-means results are for the Summer season and it is divided according to the different average ΔT levels during a year measured in the buildings.

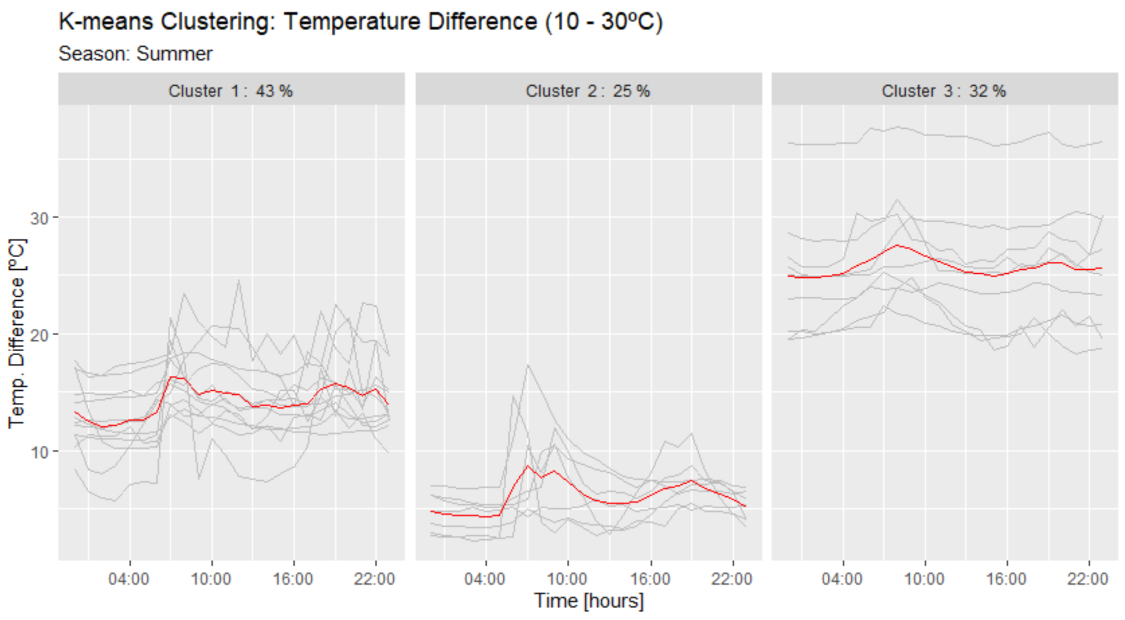


Figure 3.62. ΔT = 10 - 30 °C: Results of K-means clustering for instantaneous temperature difference profiles in the Summer.

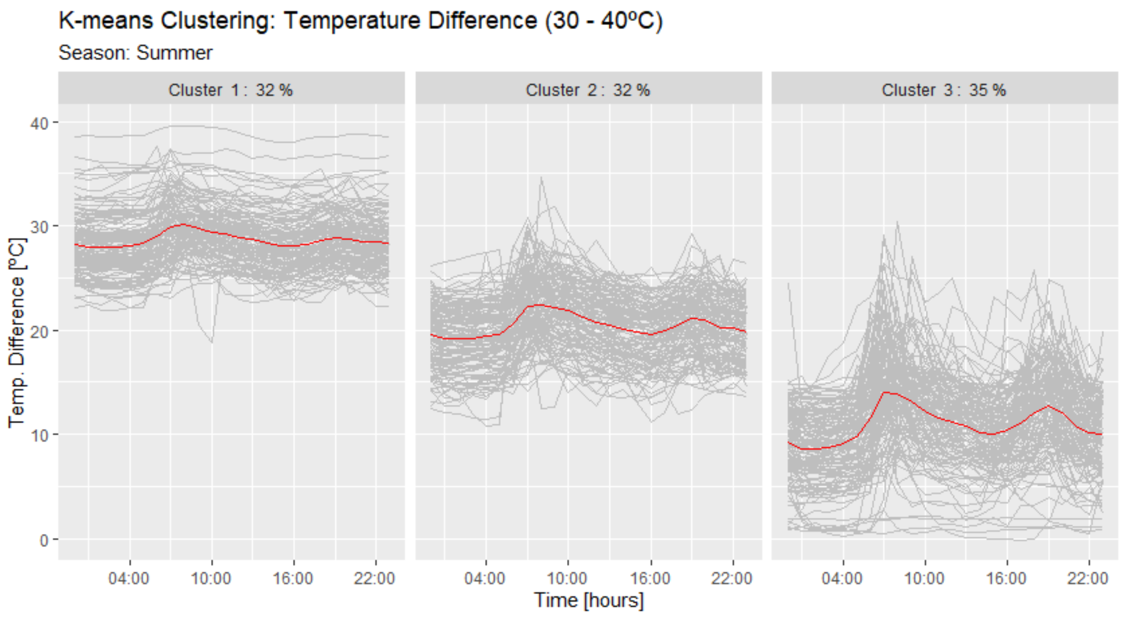


Figure 3.63. ΔT = 30 - 40 °C: Results of K-means clustering for instantaneous temperature difference profiles in the Summer.

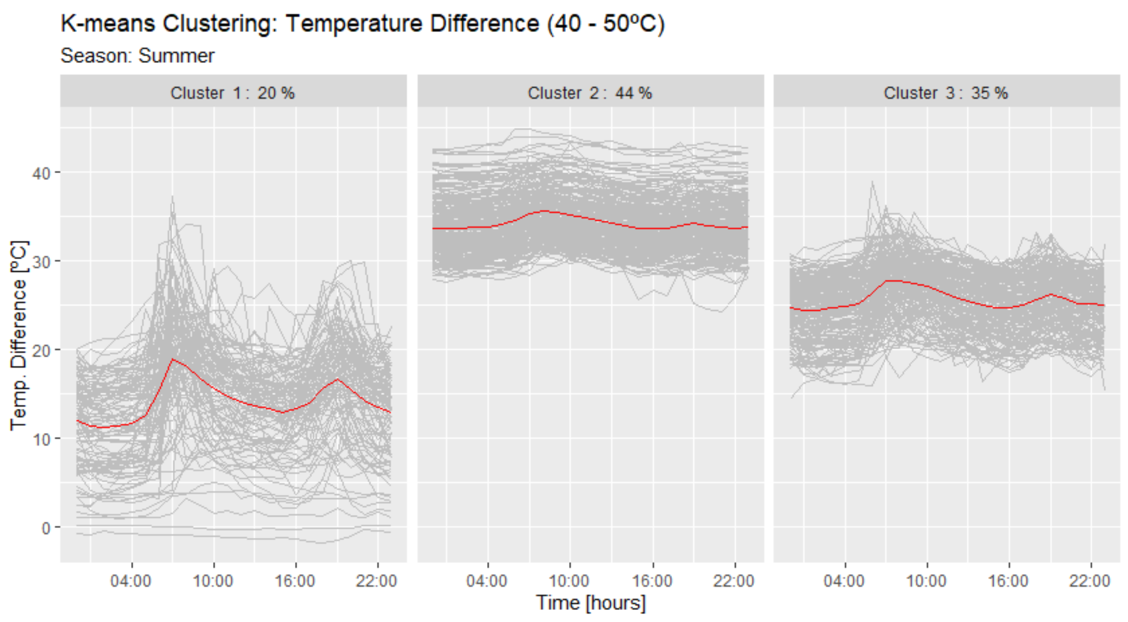


Figure 3.64. ΔT = 40 - 50 °C: Results of K-means clustering for instantaneous temperature difference profiles in the Summer.

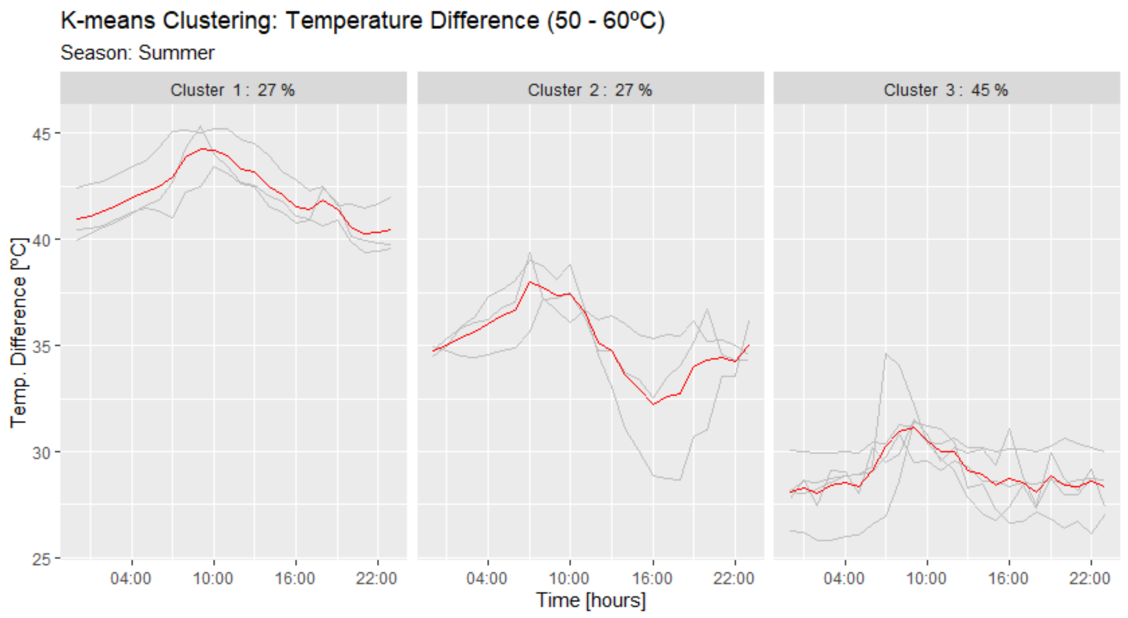


Figure 3.65. ΔT = 50 - 60 °C: Results of K-means clustering for instantaneous temperature difference profiles in the Summer.

The figure 3.62 is the buildings that their ΔT is between 10 °C and 30 °C. The two major clusters are the 1 and 3, but the clusters 2 and 3 have similar profiles where the only difference is the ΔT magnitudes. Cluster 1, is composed of the buildings with the most fluctuating profiles. The figures 3.63 and 3.64 are constituted by the buildings that their year's $\overline{\Delta T}$ are in the temperature difference groups 30 - 40 °C and 40 - 50 °C. These two datasets presented similar results in terms of their clusters, where each cluster have similar profiles with two distinct peaks, morning and evening when the two groups are compared.

The figure 3.65 is constituted by the buildings with an annual mean temperature difference between 50 and 60 °C. All the cluster groups have similar profiles. When evaluating this last dataset, it is verified that even though the buildings have high year average temperature difference (50 - 60 °C), their Summer daily ΔT profiles are lower. The main reason for this is that year average temperature difference is calculated from the variable hourly-averaged ΔT and the daily profiles are made from the instantaneous ΔT measurements from the smart energy meters.

Daily profiles in Winter - Flow

This set of K-means results are for the Winter season, and it is divided according to the different average ΔT levels during a year measured in the buildings.

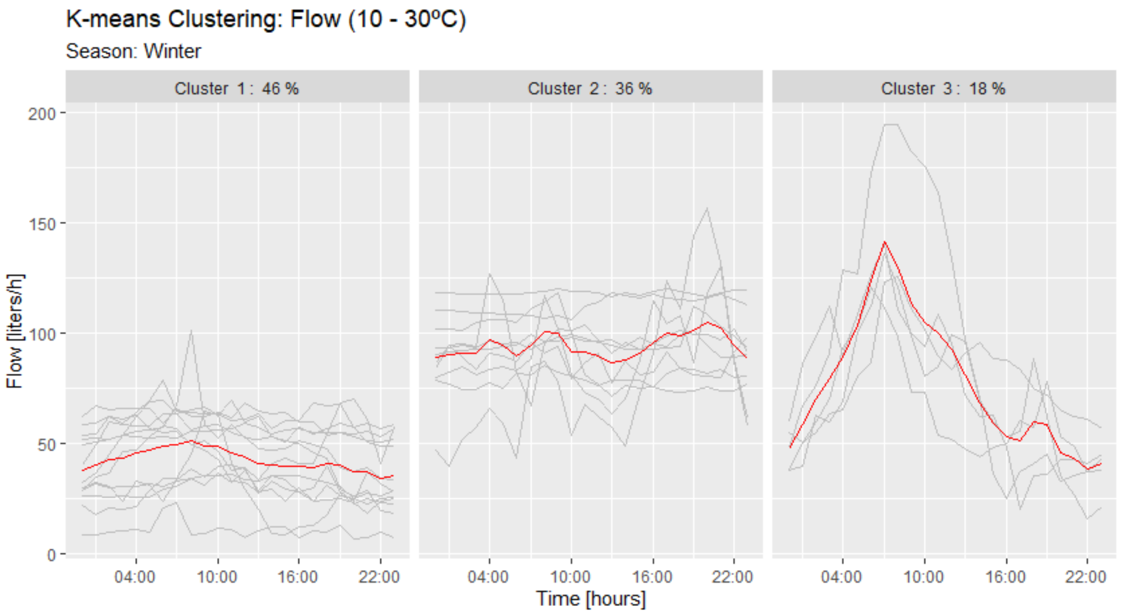


Figure 3.66. ΔT = 10 - 30 °C: Results of K-means clustering for flow profiles in the Winter.

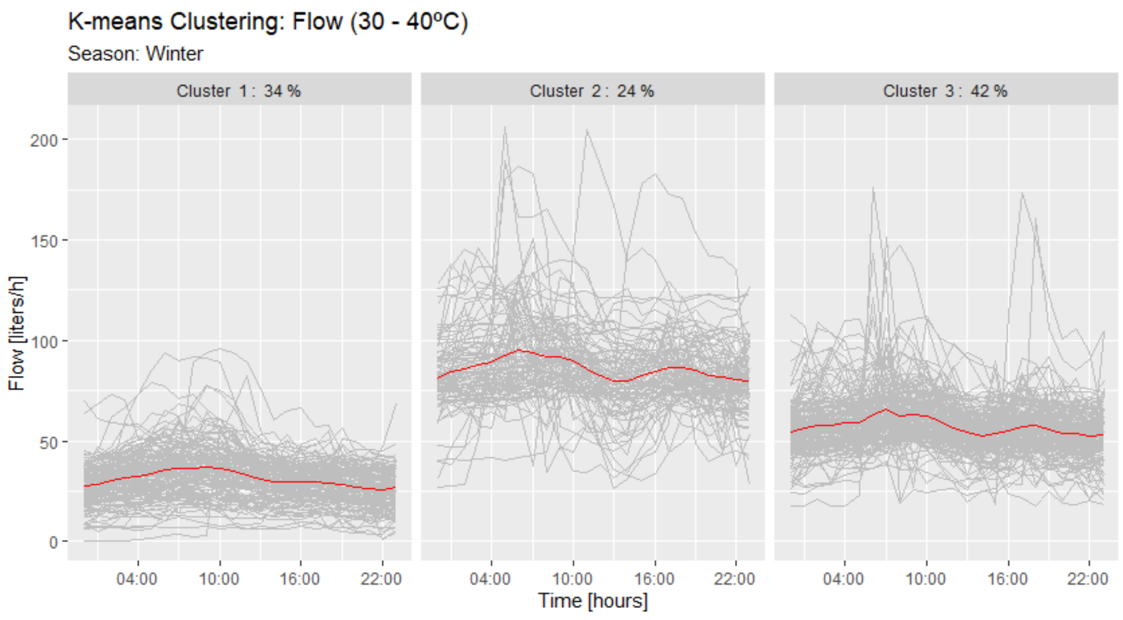


Figure 3.67. ΔT = 30 - 40 °C: Results of K-means clustering for flow profiles in the Winter.

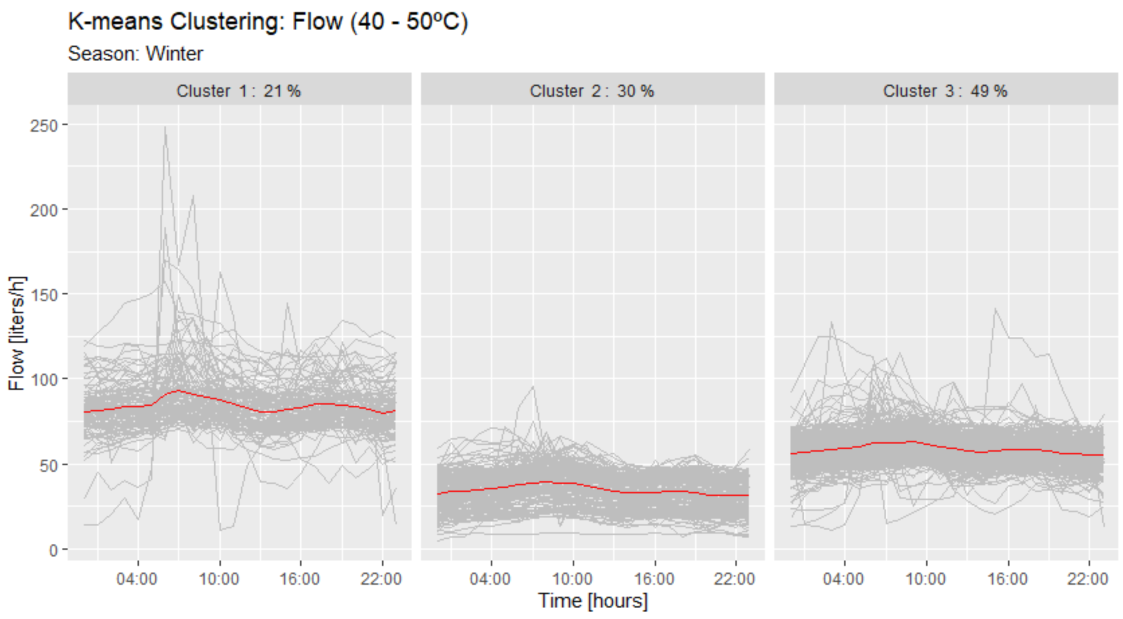


Figure 3.68. ΔT = 40 - 50 °C: Results of K-means clustering for flow profiles in the Winter.

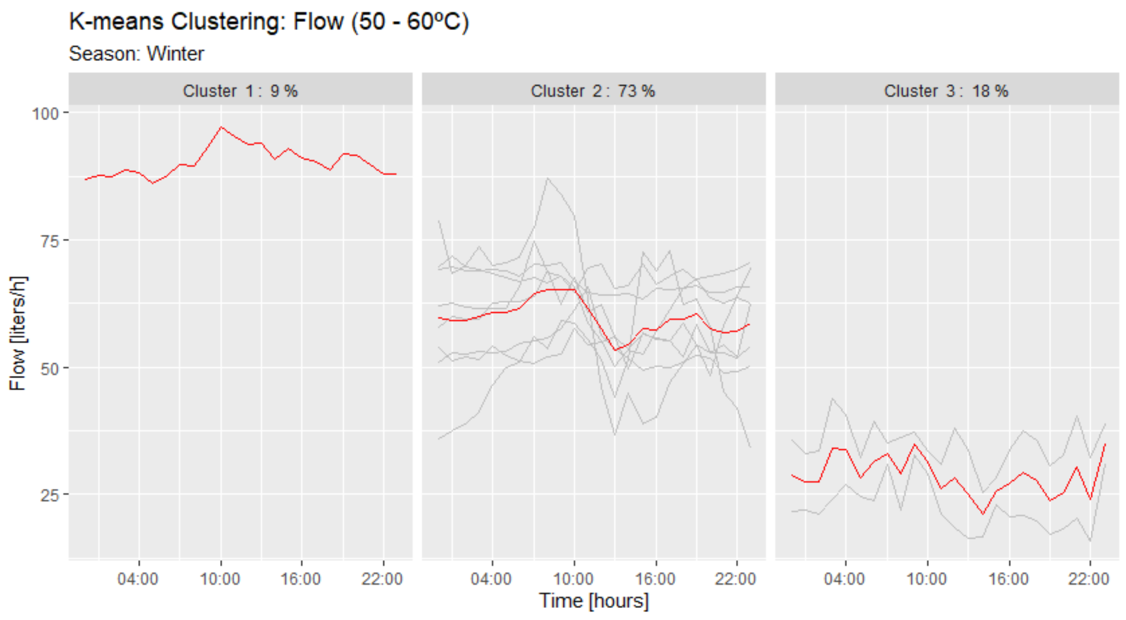


Figure 3.69. ΔT = 50 - 60 °C: Results of K-means clustering for flow profiles in the Winter.

The figure 3.66 is the buildings that their ΔT is between 10 °C and 30 °C. The two major clusters are the 1 and 2, which also have similar profiles. Cluster 3, is composed of the buildings with the most fluctuating and different profiles when compared with the clusters 1 and 2. The figures 3.67 and 3.68 are constituted by the buildings that their year's $\overline{\Delta T}$ are in the temperature difference groups 30 - 40 °C and 40 - 50 °C. These two datasets presented similar results in terms of their clusters, where each cluster have similar profiles with two distinct peaks, morning and evening when the two ΔT groups are compared. The figure 3.69 is constituted by the buildings with an annual mean temperature difference between 50 and 60 °C. This group is composed of only a few buildings of the DH dataset. All the clusters have different flow profiles, and cluster 1 is composed of only one building which its daily flow profile is highest in terms of measured flow.

Daily profiles in Summer - Flow

This set of K-means results are for the Summer season, and it is divided according to the different average ΔT levels during a year measured in the buildings.

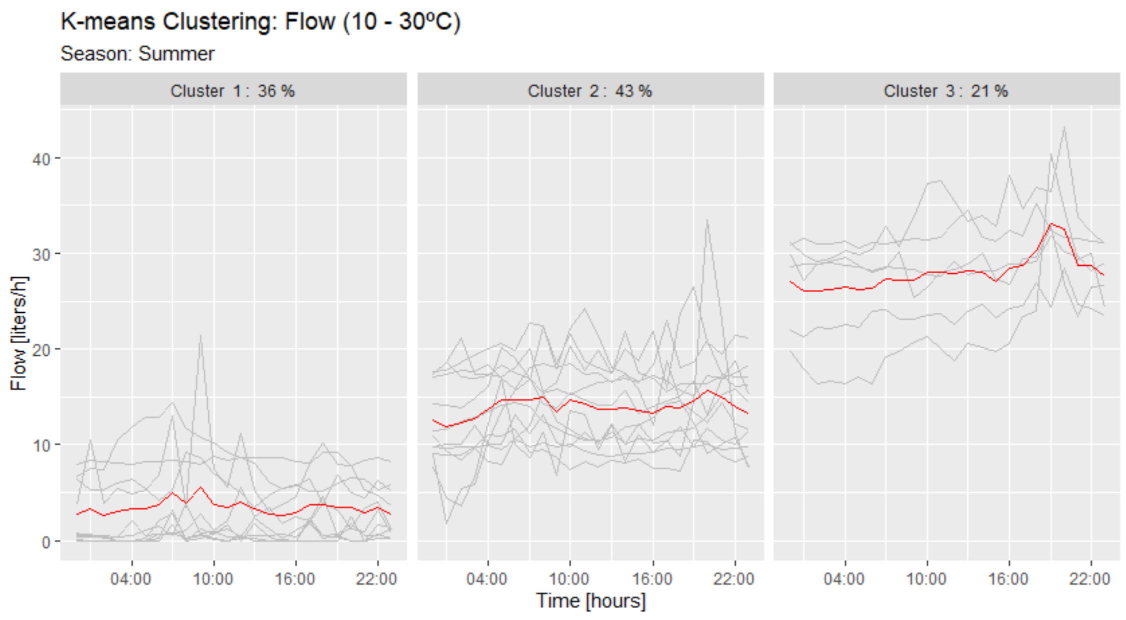


Figure 3.70. ΔT = 10 - 30 °C: Results of K-means clustering for flow profiles in the Summer.

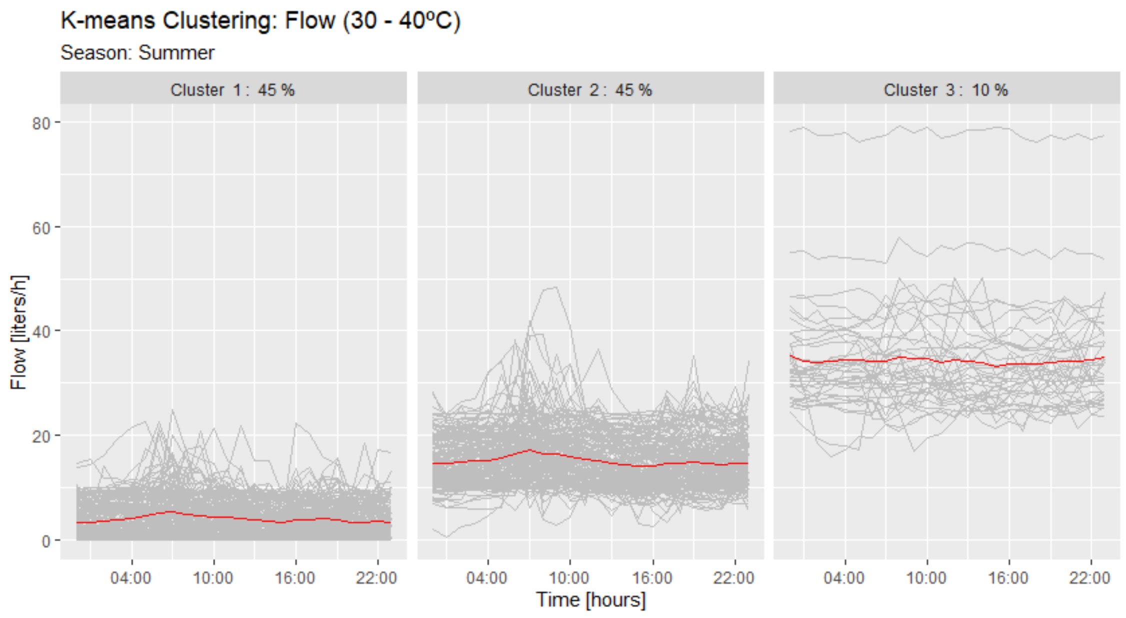


Figure 3.71. ΔT = 30 - 40 °C: Results of K-means clustering for flow profiles in the Summer.

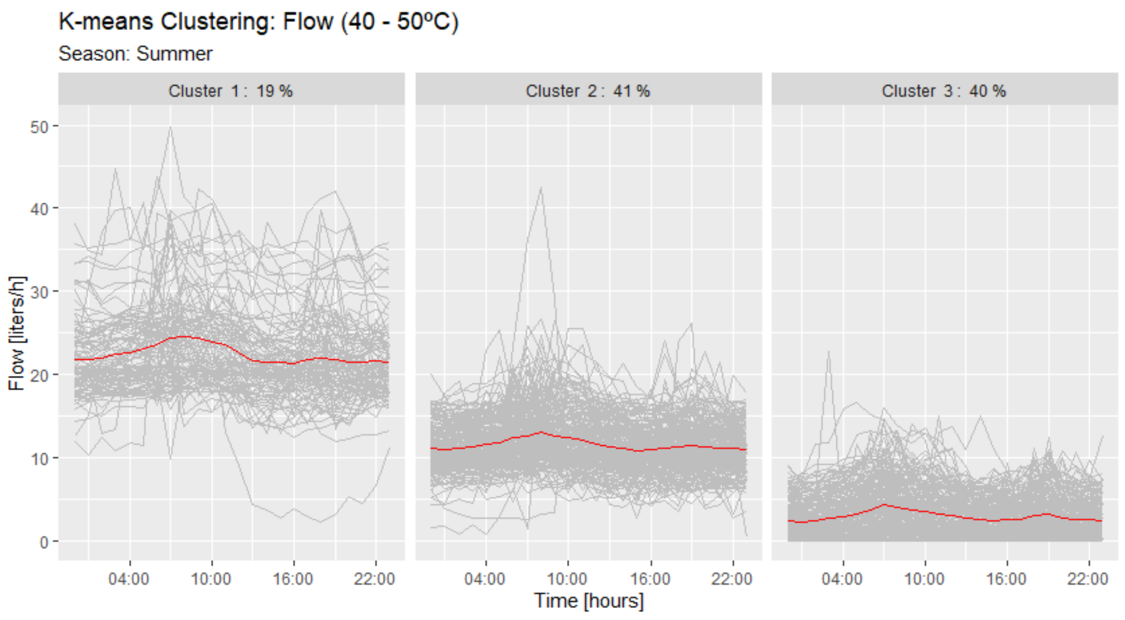


Figure 3.72. ΔT = 40 - 50 °C: Results of K-means clustering for flow profiles in the Summer.

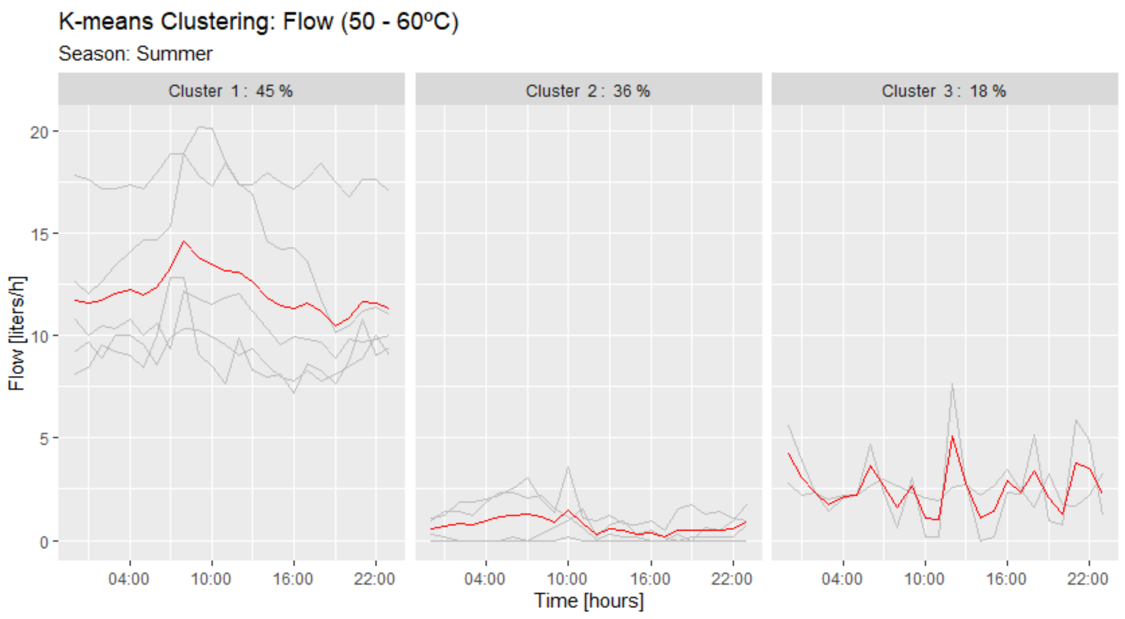


Figure 3.73. ΔT = 50 - 60 °C: Results of K-means clustering for flow profiles in the Summer.

The figure 3.70 is the buildings that their ΔT is between 10 °C and 30 °C. All the clusters have a similar percentage of buildings' amount, where the biggest cluster is number 2. Each cluster has a different profile, where cluster 1, has a predominant flow peak in the morning where the other has it in the evening. The main difference between cluster 2 and 3, is that cluster 3 is composed of the buildings that have a larger evening peak than the buildings from cluster 2. The figures 3.71 and 3.72 are constituted by the buildings that their year's $\overline{\Delta T}$ are in the temperature difference groups 30 - 40 °C and 40 - 50 °C. These two datasets presented similar results in terms of their clusters. Cluster 3 from the figure 3.71 is the cluster from this dataset that has a different flow profile from the others, with several peaks throughout the day, more concentrated in the morning period. The figure 3.73 is constituted by the buildings with an annual mean temperature difference between 50 and 60 °C. This group is composed of only a few buildings of the DH dataset, and all its clusters have different flow profiles.

Conclusions 4

In this thesis, it was presented an algorithm composed of several steps that must be taken when handling raw data from District Heating smart energy meters, in order to cleanse and prepare the data for analysis. The overall dataset's analysis was based on calculating several new variables, plotting and visualisation of these variables and clustering analysis using the K-means method. The K-means clustering method was selected due to its simplicity to code in *R*, the possibility to apply it in time-series and that it is commonly used for data clustering including similar topics of investigation

The study case is a small town in Aalborg, Denmark, constituted by 1665 smart energy meters that measure the total heat energy use, hot water consumption, fluid flow, supply temperature, and return temperature of each building connected to the district heating network. All the data recorded by the smart meters was gathered by the utility company *Aalborg Forsyning.*

From this project, it was obtained several results in different parts of the analysis. In the data preprocessing, it was determined the most suitable methods to solve the problem of missing entries in the dataset. It was concluded for variables with cumulative values the missing data can be well estimated by linear interpolation method and the instantaneous measured variables by the exponential weighted moving average with a high window-range. In this part, it was also verified that the most usual error type collect by the smart energy meters is related to the flow measuring sensor. Initially, the dataset was constituted by 1665 buildings, where only 969 of them were used for analysis after the data preprocessing algorithm, due to the other buildings having too many missing observations or were considered non-residential buildings.

Regarding the visualisation and new variables' calculation, the different generated plots and new determining variables from weather files enriched the information about the buildings in the DH network. From this part, it was possible to locate in the network the high heat losses and extract the dwellings with high and low-temperature differences and water consumption. It was also found a simplistic way to evaluate buildings in terms of DHW consumption and the impact that the outdoor temperature, solar radiation and wind speed has on the residence's heating energy demands.

The clustering analysis performed by the K-means method identified three particular clusters for all the different types of datasets. The clusters in all the analysis showed a certain resemblance between each other variable's clusters. Where in the first dataset the percentage of buildings in the two variables are similar. Regarding the other two datasets, it was seen a high similarity between the percentage of buildings for the seasons Fall, Winter and Spring. Regarding the daily profiles from the second dataset, it presents clear patterns such as morning and evening peaks of heat energy demand, corresponding in peaks in the profiles of water volume, temperature difference and flow. The last analysed dataset, presents different profiles for the Winter and Summer, in terms of instantaneous temperature difference and flow where their yearly average temperature differences are different. In the Winter season both variables are seen more correlated to each other, where in Summer the generated profiles are different. Using this temperature difference distinction, it was possible to identify buildings that have daily profiles utterly different from the rest of the DH dataset, which might be to the heating system or the user consumption behaviour.

Further work 5

Despite the several applied methods and obtained results in this thesis, there are much more methodologies and concepts that can be investigated to uncover more information and patterns regarding District Heating by the smart energy meters data. Therefore several suggestions are presented in this chapter:

- Optimisation of the preprocessing algorithm, investigation of better building's exclusion conditions and application of different missing data imputation methods;

- The integration of more local weather information and with the Danish national building register in order to optimise the methods used in this project, to determine more and accurate building characteristics and correlate the smart meters data with the type of building's space/DHW systems. For this topic the research paper [Gianniou et al., 2018b] is suggested;

- Create a methodology to distinguish between the space heating demands and the DHW demands of each building from its overall measured energy usage. For this topic the research papers [Marszal-Pomianowska et al., 2019] and [Bacher et al., 2016] are suggested;

- The application of other clustering analysis methods in the different measured parameters or other Machine Learning algorithms, e.g. forecasting and classification. For this topic the research paper [Kristensen et al., 2018] is suggested;

- Anonymized socio-economic studies to assess possible correlations between types of occupants and heating/DHW usage profiles;

- Building an interactive web-based interface to present and share the processed smart energy meter data with professionals of the building sector and with the DH utility companies. In *R* it can be made by the application *Shiny*, but other tools like *Tableau* and *Power BI* can also be used.

Bibliography

Bacher et al., 2016. Peder Bacher, Philip Anton de Saint-Aubain, Lasse Engbo Christiansen and Henrik Madsen. *Non-parametric method for separating domestic hot water heating spikes and space heating.* Energy and Buildings, 130, 107 – 112, 2016. ISSN 0378-7788. doi: https://doi.org/10.1016/j.enbuild.2016.08.037. URL `http://www.sciencedirect.com/science/article/pii/S0378778816307332`.

Danish Energy Agency. Danish Energy Agency. *The Danish Energy Model*, Danish Energy Agency.

European Commission, March 2019. European Commission. *Clean energy for all Europeans*, 2019. Available at: `https://ec.europa.eu/energy/topics/energy-strategy/clean-energy-all-europeans_en` [Accessed 21 May 2020].

European Commission, April 2020. European Commission. *Heating and cooling*, 2020. Available at: `https://ec.europa.eu/energy/topics/energy-efficiency/heating-and-cooling_en?redir=1` [Accessed 21 May 2020].

Frost, 2020. Jim Frost. *5 Ways to Find Outliers in Your Data*, 2020. Available at: `https://statisticsbyjim.com/basics/outliers/` [Accessed 20 Mar. 2020].

Gianniou et al., 2018a. Panagiota Gianniou, Xiufeng Liu, Alfred Heller, Per Sieverts Nielsen and Carsten Rode. *Clustering-based analysis for residential district heating data.* Energy Conversion and Management, 165, 840 – 850, 2018. ISSN 0196-8904. doi: https://doi.org/10.1016/j.enconman.2018.03.015. URL `http://www.sciencedirect.com/science/article/pii/S019689041830236X`.

Gianniou et al., 2018b. Panagiota Gianniou, Christoph Reinhart, David Hsu, Alfred Heller and Carsten Rode. *Estimation of temperature setpoints and heat transfer coefficients among residential buildings in Denmark based on smart meter data.* Building and Environment, 139, 125 – 133, 2018. ISSN 0360-1323. doi: https://doi.org/10.1016/j.buildenv.2018.05.016. URL `http://www.sciencedirect.com/science/article/pii/S0360132318302762`.

Kamstrup A/S, 2016. Kamstrup A/S. *MULTICAL® 402 - Technical Description*, Kamstrup A/S, 2016.

Kamstrup A/S, 2019. Kamstrup A/S. *MULTICAL® 403 - Technical Description*, Kamstrup A/S, 2019.

Kamstrup A/S, 2018. Kamstrup A/S. *MULTICAL® 602 - Technical Description*, Kamstrup A/S, 2018.

Kristensen et al., 2018. Martin Heine Kristensen, Rasmus Elbæk Hedegaard and Steffen Petersen. *Hierarchical calibration of archetypes for urban building energy* *modeling*. Energy and Buildings, 175, 219 – 234, 2018. ISSN 0378-7788. doi: https://doi.org/10.1016/j.enbuild.2018.07.030. URL `http://www.sciencedirect.com/science/article/pii/S0378778818312532`.

Marszal-Pomianowska et al., **2019**. Anna Marszal-Pomianowska, Chen Zhang, Michal Pomianowski, Per Heiselberg, Kirsten Gram-Hanssen and Anders Rhiger Hansen. *Simple methodology to estimate the mean hourly and the daily profiles of domestic hot water demand from hourly total heating readings.* Energy and Buildings, 184, 53 – 64, 2019. ISSN 0378-7788. doi: https://doi.org/10.1016/j.enbuild.2018.11.035. URL `http://www.sciencedirect.com/science/article/pii/S0378778818321261`.

Moritz and Gatscha, **2019**. Steffen Moritz and Sebastian Gatscha. *Time Series Missing Value Imputation*, 2019. URL `https://cran.r-project.org/web/packages/imputeTS/imputeTS.pdf`.

Ribecca, **2020**. Severino Ribecca. *The Data Visualisation Catalogue*, 2020. Available at: `https://datavizcatalogue.com/about.html` [Accessed 10 Jun. 2020].

Wittchen and Aggerholm, **2000**. Kim B Wittchen and Søren Aggerholm. *Calculation of building heating demand in EPIQR*. Energy and Buildings, 31(2), 137 – 141, 2000. ISSN 0378-7788. doi: https://doi.org/10.1016/S0378-7788(99)00027-4. URL `http://www.sciencedirect.com/science/article/pii/S0378778899000274`.

Part I

Appendix

Conference article NSB 2020 A

Treatment and analysis of smart energy meter data from a cluster of buildings connected to district heating: A Danish case

Hicham Johra[1,*], *Daniel* Leiria[1], *Per* Heiselberg[1], *Anna* Marszal-Pomianowska[1], and *Torben* Tvedebrink[2]

[1]Aalborg University, Department of Civil Engineering, Thomas Manns Vej 23, 9220 Aalborg Ø, Denmark
[2]Aalborg University, Department of Mathematical Sciences, Skjernvej 4A, 9220 Aalborg Ø, Denmark

Abstract. District heating has been found to be a key component of future and reliable smart energy grids comprising 100% of renewable energy sources for countries with dominant heating season. However, these systems face challenges that require a deeper understanding of the coupling between the distribution networks and the connected buildings, to enable demand-side management and balance the intermittence of renewables. In recent years, many smart energy meters have been installed on the heating systems of Danish dwellings connected to district heating, and the first yearly measurement data sets of large building clusters are now available. This article presents the methodology for the pre-processing and cluster analysis (K-means clustering) of a one-year-long smart energy meter measurement data from 1665 Danish dwellings connected to district heating. The aim is to identify typical household daily profiles of heat energy use, return temperature, and temperature difference between the supply and the return fluid. The study is performed with the free software environment "R", which enables the rapid extraction of information to be shared with professionals of the building and energy sectors. After presenting the preliminary results of the clustering analysis, the article closes with the future work to be conducted on this study case.

1 Introduction

For countries of temperate and cold climates with a dominant heating season, district heating (DH) has been found to be the most efficient, sustainable and cost-effective solution to provide heat to buildings in urban areas. It is also a key component for future and reliable smart grid systems with 100% renewable energy sources [1,2]. However, the current DH systems are facing several production and distribution challenges:

- Detecting faults in the network.
- Identifying critical distribution points, congestion distribution bottlenecks, problematic buildings, and energy-intensive user behaviors.
- Optimizing the system's efficiency by decreasing the overall temperature in the DH network (from typically 80 °C / 45 °C supply/return temperature, down to 50 °C / 25 °C in some extreme cases) to reduce the heat losses in the distribution pipes.
- Decreasing the supply temperature to allow the integration of low-temperature waste heat input from the surrounding industry ecosystem.
- Minimizing the return temperature of the DH network to increase the energy efficiency of the heat production units.
- Diminishing the demand peaks (typically occurring in the morning) to avoid using or installing fossil-fuel peak boilers or temperature boosters with high operation and maintenance costs, and to enable downsizing new distribution networks.
- Matching the heat energy demand with the heat energy supply and bear the increasing share of intermittent renewable energy sources.

Smart-meter monitoring and demand-side management strategies can greatly help to address the aforementioned challenges. However, large efforts are still needed to gain a better understanding of the dynamic interactions between the DH networks and the connected buildings, in order to develop and calibrate efficient numerical urban-scale models of those integrated systems. The latter, together with the feedback information from the Smart Grid meters, will ease planning and optimum control of the DH plants and networks, making use of the aggregated energy flexibility potential of building clusters to perform load shifting and peak shaving at a district level and tackle some of the aforementioned issues.

Denmark has a long history of district heating use with networks that are widespread over the entire country. Indeed, Denmark has one of the largest shares of heating production covered by DH systems. 63% of all private houses and citizens of Denmark are connected to the district heating network for space heating and domestic hot water (DHW) production. For comparison, in Europe, only Iceland and Latvia have a higher share of DH coverage (90% and 65%, respectively), followed by Finland, Lithuania, Poland and Sweden (more than 50%). DH systems are primarily found in urban and suburban areas where the heating need density is high. However, smaller towns and villages (500 households)

* Corresponding author: hj@civil.aau.dk

can be also be equipped with a DH network. In addition to the vast central DH networks implemented in the 6 largest urban areas of Denmark, around 400 smaller ones can be found in other towns of modest size around the country. Those various DH installations present a large diversity of distribution network sizes, configurations, fuels mixes, and production plants, comprising some short term heat storages (typically 12 hours of full load heat production from the plant), supplementary solar heating or electric boilers, and integrating parts of the surplus heat generation from local industries. In 2013, more than 70% of all DH energy in Denmark was produced in cogeneration with electricity by efficient combined heat and power (CHP) plants. Consequently, more than 60% of the national electricity was produced by CHPs [3].

For all of those aforementioned reasons, Denmark is a particularly interesting case to study the many aspects of the district heating technologies. Fortunately, Denmark has recently started a massive and systematic campaign of smart energy meters installation in all buildings connected to DH. In addition, Denmark has a consistent national building and household information database, which enables statistical studies about the correlations between socio-economic context and energy-related practices, for instance.

The first complete yearly data sets of large clusters of buildings connected to DH networks in different Danish cities are now available for treatment and analysis. Although big data mining has now become a common concept and practice in many fields of research and in the industry such as electricity grid operators (because data from electricity smart meters have been readily available for several years now), this is still a new thing for urban-scale district heating systems.

In recent years, some Danish research projects started investigating how to extract valuable information for the buildings and energy sector from those smart energy meters' data sets.

Gianniou et al. [4] performed a clustering analysis on the heating usage data of 8293 Danish dwellings connected to a district heating network. The researchers could thus classify the different households into specific groups with defined consumption intensity and representative patterns. They also looked at the correlations between the energy intensity, the building's characteristics, the type of occupants, the load profiles of the households, their consumption behavior, and the changes of the latter over time.

In another publication, Gianniou et al. [5] have estimated the indoor set-point temperature (a very sensitive assumption for building energy models) and the building heat losses of 14,000 dwellings by applying linear regression and heat balance calculation to a large data set of DH smart energy meters, weather data and information from the national building register.

Kristensen et al. [6] used smart energy meter data (50 training buildings and 100 test buildings) to create dynamic physics-based building energy models of archetypes using the Bayesian calibration framework. Those building models can produce a good prediction of the aggregated energy use of single-family houses connected to DH.

Hedegaard et al. [7] adopted a bottom-up modeling approach to create, calibrate and validate an urban-scale model of the district heating consumption of 159 houses, using data from public building registers, local weather stations, and smart energy meters of this building cluster. This model has then enabled testing the effectiveness of a simple price-based aggregated demand response strategy to reduce demand peaks in the district heating network.

In this paper, the preliminary result analysis of a one-year data collection of 1665 smart energy meters in a Danish town connected to a large DH network is presented. The data is analyzed with the K-means clustering method to help identify typical household daily profiles of heat energy usage, return temperature to the DH network, and the temperature difference between the supply and the return fluid.

2 Study case

2.1. Building cluster

The study case is a building cluster consisting of 1665 residential dwellings (mostly single-family houses), all located in the town of Klarup (Denmark). Klarup has a population of 4733 inhabitants. It is situated 11 km to the east of Aalborg (the fourth largest city in Denmark).

For space heating and domestic hot water supply, the entire building cluster is connected to the large district heating network of the municipality of Aalborg (see Figure 1). For space heating supply, the local network inside each building comprising the heating terminals (radiators or under-floor heating) is directly connected to the main DH network. On the other hand, the instantaneous domestic hot water production is performed by a heat exchanger (indirect connection to the main DH network).

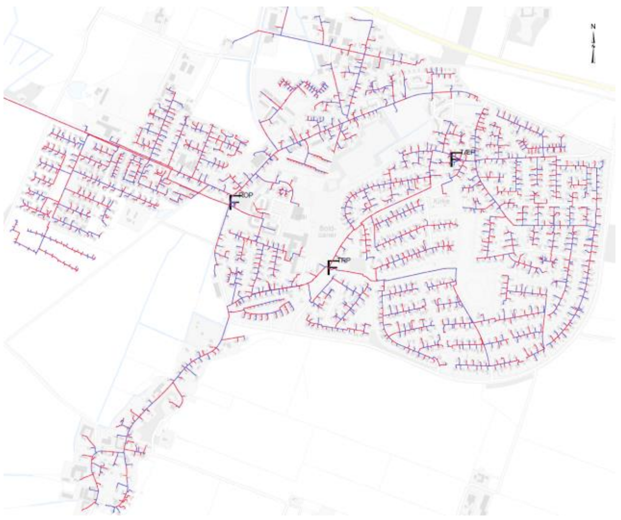


Fig. 1. Overview of the town study case, Klarup, and its district heating distribution network.

Aalborg Forsyning [8] is the utility company in charge of the heat production and distribution, development, operation and maintenance of the DH distribution network. In 2018, Aalborg Forsyning has installed smart energy meters in every dwelling of Klarup: one smart energy meter per building, measuring the aggregated energy usage for space heating and domestic hot water production.

2.2. Smart energy metering system

The smart energy meters installed in the buildings of the study case are state-of-the-art metering devices for hydronic heating systems: Multical® 402, Multical® 403, and Multical® 602 from the specialized manufacturer Kamstrup A/S [9]. The typical uncertainties on the temperature measurements, the fluid flow rate measurements, and the estimates of the energy usage are ± 0.53%, ± 2% and ± 0.22 %, respectively. The different measurement variables recorded by the smart energy meters are as follows: timestamp, fluid flow rate, supply temperature, return temperature, cumulative volume usage, and cumulative heat energy usage. In addition, the smart energy meter indicates and records any specific error message if it detects the following problems: wrong flow direction in the fluid flow sensor, air detected in the fluid flow sensor, weak signal from the fluid flow sensor, return temperature is higher than supply temperature, return temperature is too low. The measurements are integrated (averaged) over a time period of around one hour. Consequently, there are around 24 measurements per day for each building.

2.3. Data set

The study of this paper analyses the data collected from the building cluster case during an entire year: from the 1st of October 2018 until the 7th of October 2019. Although the raw data contains more than 14 million lines, the entire data loading, pre-processing, processing, and analysis algorithm developed for this study can be run on a standard computer station in around 2 hours.

3 Methodology

In this study, the entire data processing and analysis are performed with the free software environment and language for statistical computing and graphics “R” [10]. The pre-processing of the raw data files consists in the concatenation of the different log files corresponding to different time periods, followed by resampling to obtain coherent and synchronized hourly times series of the different measurement variables for all the buildings. The quality of the data set is then assessed to determine what criteria is used to discard the sub-sets corresponding to buildings with too many or too large information gaps in the time series. In order to prepare the remaining data subset for clustering analysis, it is necessary to operate an imputation (interpolation/estimation) of the measurement gaps in the time series recorded by the smart energy meters.

The clustering analysis method used in this study to identify subgroups of similar households within the observations of the data set is named “K-means clustering” [11]. This method is simple to employ, widely used, fast to compute, and can be applied to time series, which makes it a perfect tool for the analysis of data from large building clusters.

Antecedently to running the K-means clustering algorithm, the input data set must be standardized, i.e., scaled. The optimum number of clusters (subgroups) is then determined with the “Average Silhouette” method. Finally, the K-means clustering algorithm is executed with a Euclidean distance measure [11].

4 Data analysis, preliminary results, and discussions

4.1. Data set quality and data cleansing

A preliminary data quality assessment is conducted to estimate the amount of missing information in the raw data collected from the 1665 building cases. The missing information is either erroneous measurement points (flagged as an error by the smart energy meter, or containing *NaN* values, or out of realistic range for this DH system) or missing measurements (no measurement has been logged for more than an hour which creates a gap in the time series). It is found that the raw data set has an overall 3.7 % of missing values. One could consider that this percentage of missing information is negligible and use the entire raw data as is. However, consecutive missing data can form large continuous gaps in the time series, which can be problematic when imputing/estimating the latter by means of interpolation.

One can see in Figure 2 the distribution of the gaps in the time series as a function of their length. It can be observed that the vast majority of the data gaps are short, with the one-hour gaps, two-hour gaps and three-hour gaps representing 70 %, 13.7 % and 5.5 % of the time series gap occurrences, respectively.

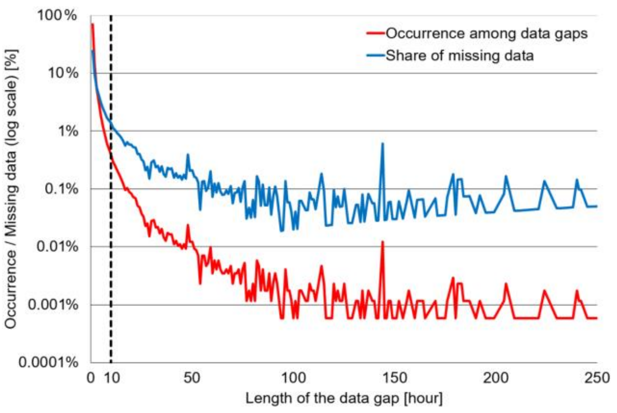


Fig. 2. Occurrence distribution and contribution to the total missing information of the time series gaps as a function of their length in the raw data set.

In order to avoid the interpolation of very large data gaps, all buildings having information gaps larger than 9 consecutive hours have been discarded. The data subset of the 1028 remaining buildings (out of 1665) has an overall 2.49 % of missing information, with a maximum of 10.7 % missing values for the time series with the most data gaps. The quality of this subset is considered satisfactory and it is thus used for the clustering analysis.

4.2. Imputation of the missing data

The next step in the data pre-processing is the imputation of the missing values (gaps) in the time series. Because there are numerous methods suitable for the replacement of missing values in univariate time series [12, 13], the latter are benchmarked against each other. One can see in Table 1 the results of this benchmarking. Each method has been tested to estimate information gaps artificially introduced into 10 building time series that originally did not have any missing data. These information gaps have a length distribution that is equal to that of the entire data set (see the previous subsection) and are randomly placed within the 10 building time series. The Root Mean Square Error (RMSE) is then calculated for each measurement variable and used to determine the most suited imputation methods for the current data set, i.e., the ones with the lowest RMSE.

According to the benchmarking results, it is chosen to use linear interpolation for the cumulative measurements (energy, volume, volume x supply temperature, volume x return temperature), and exponential weighted moving average (k=8) for instantaneous measurements (supply temperature, return temperature, fluid flow rate).

Table 1. Benchmarking results for the different missing data imputation methods.

Imputation method	Root Mean Square Error (RMSE)						
	Cumulative measurements				instantaneous measurements		
	Cumulative energy	Cumulative volume	Volume x supply temperature	Volume x return temperature	Supply temperature	Return temperature	Fluid flow
Linear interpolation	4.05E-01	7.47E-03	2.39E+00	1.76E+00	9.50E-01	7.08E-01	9.45E-03
Spline interpolation	6.99E-01	1.37E-02	4.09E+00	2.64E+00	1.12E+00	8.44E-01	1.13E-02
Stineman interpolation	4.22E-01	7.56E-03	2.50E+00	1.88E+00	9.57E-01	7.13E-01	9.52E-03
Last observation carried forward	1.52E+00	3.27E-02	1.24E+01	6.13E+00	1.25E+00	8.83E-01	1.15E-02
Next Observation Carried Backward	1.60E+00	3.43E-02	1.24E+01	6.17E+00	1.20E+00	8.73E-01	1.11E-02
Simple moving average, k=2	6.39E-01	1.32E-02	4.91E+00	2.65E+00	9.77E-01	7.06E-01	9.39E-03
Simple moving average, k=4	8.27E-01	1.77E-02	6.55E+00	3.44E+00	1.06E+00	7.16E-01	9.14E-03
Simple moving average, k=6	9.98E-01	2.20E-02	7.91E+00	4.28E+00	1.17E+00	7.52E-01	9.18E-03
Simple moving average, k=8	1.15E+00	2.54E-02	8.77E+00	4.75E+00	1.26E+00	7.79E-01	9.24E-03
Linear weighted moving average, k=2	5.93E-01	1.22E-02	4.47E+00	2.47E+00	9.60E-01	6.98E-01	9.30E-03
Linear weighted moving average, k=4	6.77E-01	1.43E-02	5.21E+00	2.81E+00	9.98E-01	6.92E-01	8.99E-03
Linear weighted moving average, k=6	7.74E-01	1.68E-02	6.01E+00	3.29E+00	1.05E+00	7.08E-01	8.96E-03
Linear weighted moving average, k=8	8.53E-01	1.87E-02	6.50E+00	3.58E+00	1.10E+00	7.21E-01	8.95E-03
Exp. weighted moving average, k=2	5.79E-01	1.19E-02	4.32E+00	2.41E+00	9.55E-01	6.97E-01	9.28E-03
Exp. weighted moving average, k=4	5.84E-01	1.21E-02	4.35E+00	2.42E+00	9.59E-01	6.82E-01	8.99E-03
Exp. weighted moving average, k=6	5.95E-01	1.24E-02	4.44E+00	2.47E+00	9.64E-01	6.82E-01	8.94E-03
Exp. weighted moving average, k=8	5.99E-01	1.25E-02	4.46E+00	2.49E+00	9.66E-01	6.82E-01	8.93E-03
Mean value	2.36E+03	5.28E+01	2.08E+04	1.04E+04	2.96E+00	1.45E+00	1.56E-02
Median value	2.56E+03	5.56E+01	2.17E+04	1.06E+04	3.13E+00	1.48E+00	1.61E-02

4.3. Problem decomposition and optimum number of clusters

As mentioned before, the aim of the current clustering analysis is to identify typical household daily profiles of energy usage, return temperature to the DH network, and temperature difference between the supply and the return fluid. For the sake of simplicity, the current study is restricted to only working days: all weekend days and official bank holidays are removed. It is assumed that households have a more regular pattern of energy usage during working days compared to rest days. In addition, the data set is divided into four time periods: Spring, Summer, Autumn, Winter. For each household, the seasonal typical daily profile is thus calculated as the average of all the days over the respective season time period.

In order to perform the K-means clustering analysis, an optimum number of clusters has to be chosen beforehand. Three popular methods, i.e., the Elbow method, the Silhouette method, and the Gap method, have been tested to estimate cluster compactness and clustering quality as a function of the total number of clusters. It is chosen to use the Silhouette method to assess the optimum number of clusters for the analysis of energy usage profiles, return temperature profiles, and temperature difference profiles. One can see in Figure 3 that the optimum number of clusters (highest average silhouette width) is 2 for all the cases. However, it is decided to set the number of clusters to 4 for the analysis of all the measured variables.

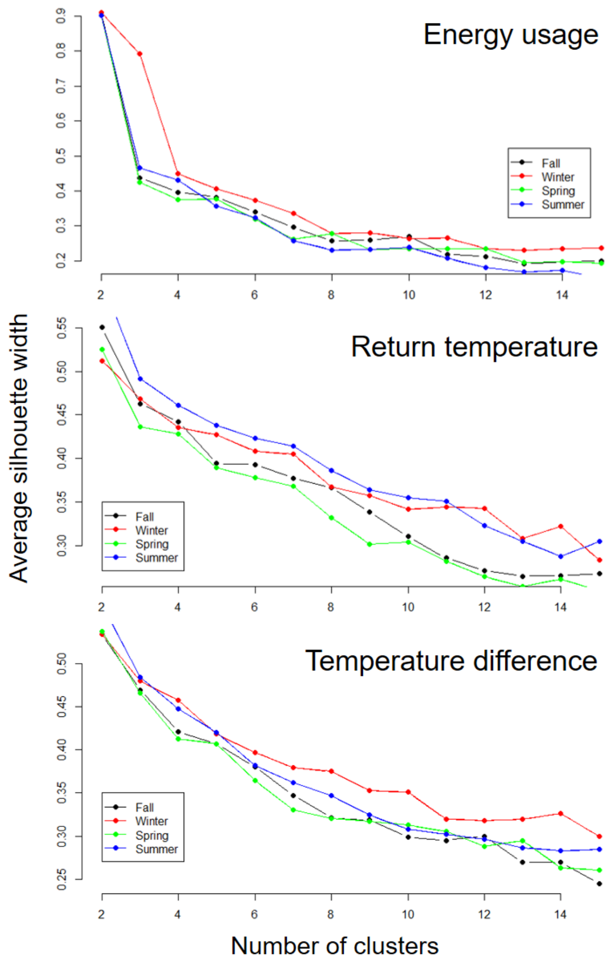


Fig. 3. Average silhouette width (clustering quality) as a function of the total number of clusters.

4.4. K-means clustering results

After all the above mentioned pre-processing steps have been completed, the K-means clustering algorithm can be run. One can see in Figure 4, Figure 5 and Figure 6, the results of this clustering analysis.

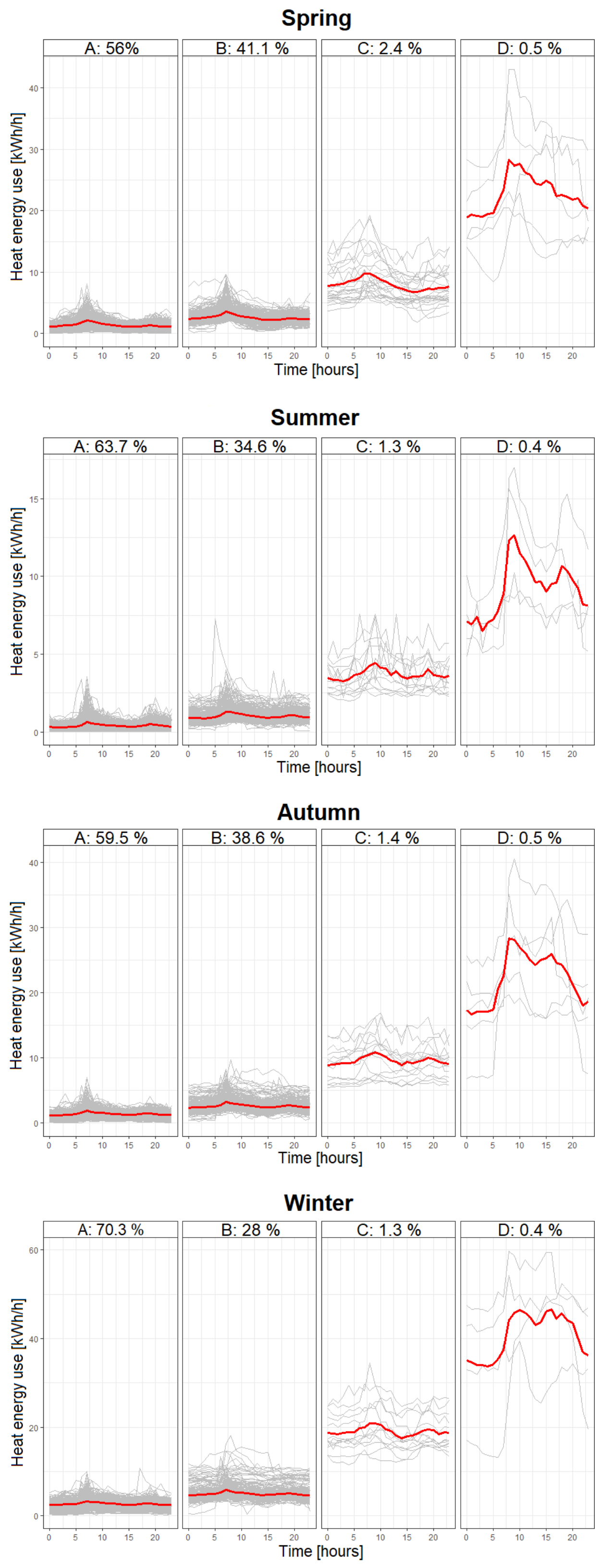


Fig. 4. K-means clustering analysis results for the daily profiles of the heat energy use for working days only during the four seasons of Spring, Summer, Autumn, and Winter, respectively.

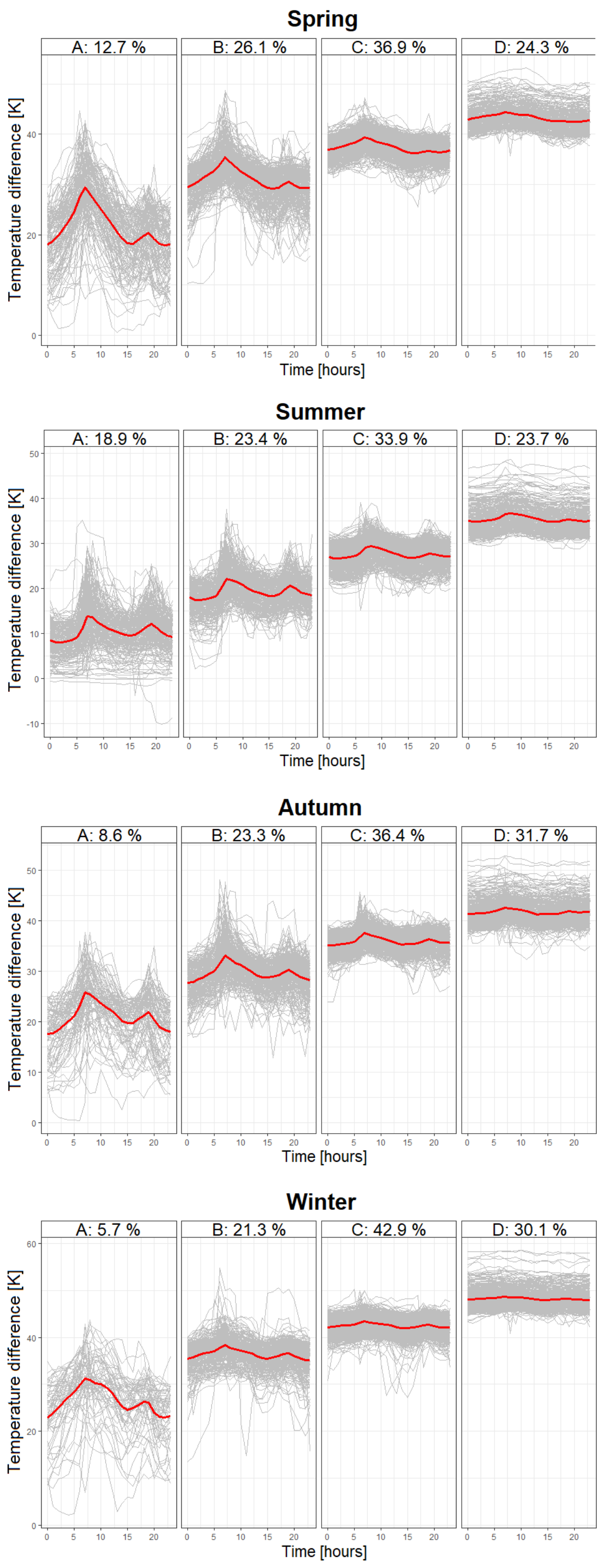


Fig. 5. K-means clustering analysis results for the daily profiles of the temperature difference between the supply and the return fluid for working days only during the four seasons of Spring, Summer, Autumn, and Winter, respectively.

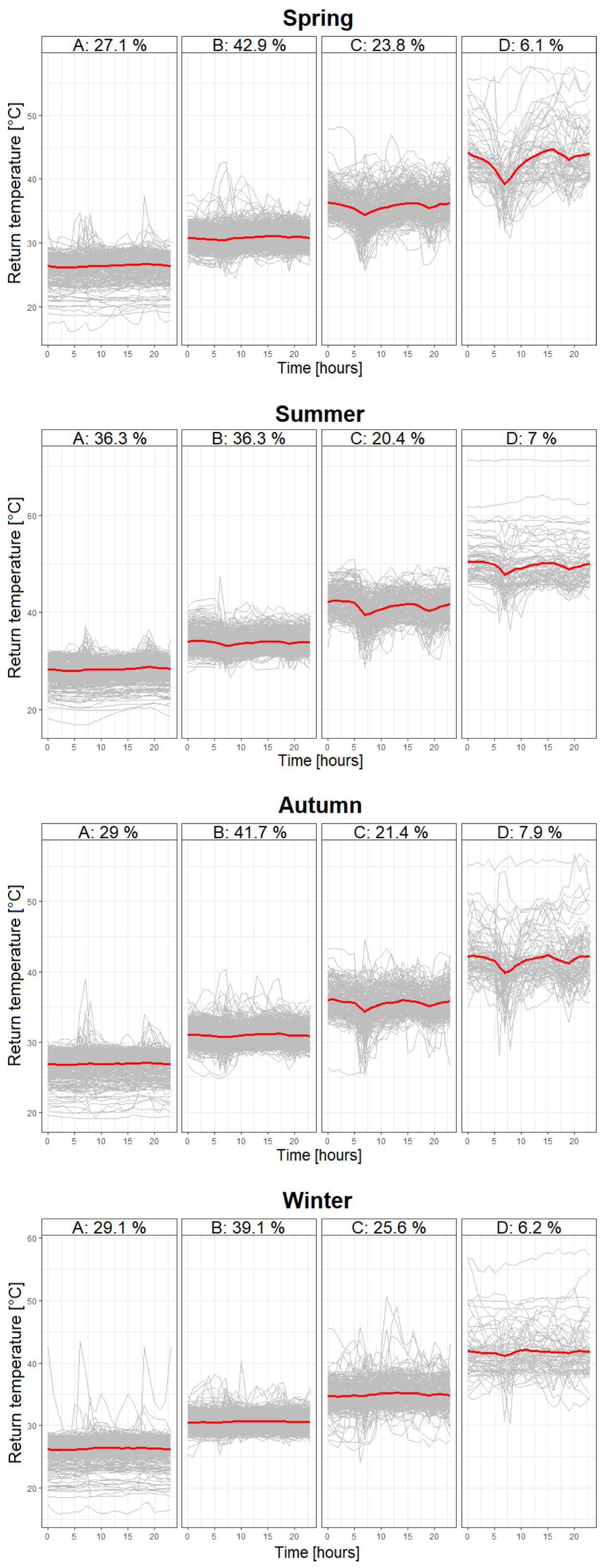


Fig. 6. K-means clustering analysis results for the daily profiles of the return temperature to the DH network for working days only during the four seasons of Spring, Summer, Autumn, and Winter, respectively.

For each type of daily profile, namely, heat energy use, temperature difference between the supply and the return fluid, and return temperature to the DH network, the 1028 buildings are classified into 4 different groups (clusters): A, B, C, and D. The subplots present the daily profiles of all the buildings (grey lines) in their respective group. The centroid of each cluster is illustrated by the thick red line. The distribution of each group within the total population of buildings is indicated on the top of each subplot.

One can observe in Figure 4 that the vast majority of the buildings belong to the clusters A and B, which present a rather similar daily profile heat energy use with a significant morning peak (morning shower time on working days) and a smaller evening peak. The same trends can be observed in clusters C and D, but with a much larger overall energy use. One explanatory hypothesis could be that the buildings in clusters A and B are single-family houses (as expected), whereas the buildings in clusters C and D are larger residential buildings with several apartment blocks but a single main smart energy meter. Those results are in agreement with the general description of the study case city and are very similar to what was observed in another Danish city by Giannou et al. [4].

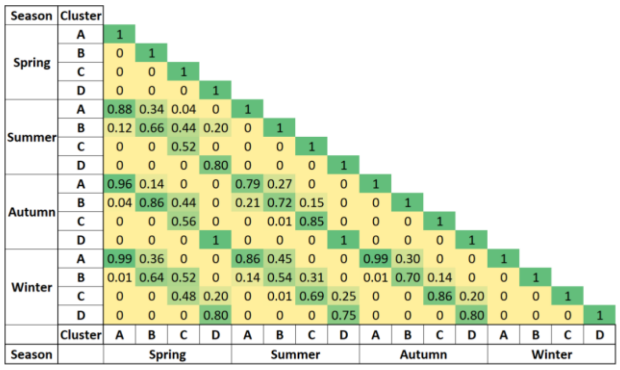
One can see in Figure 5 that concerning the daily profile of temperature difference between the supply and the return fluid, the distribution among the different categories is more balanced. For all seasons, the categories with a relatively low overall temperature difference (clusters A and B) present 2 clear peaks in the morning and in the evening, which is similar to is observed on all heat energy use profiles. However, clusters C and D have a significantly larger temperature difference but with a much more regular profile.

The return temperature profiles presented in Figure 6 appear as a direct consequence of what has been previously described for the temperature difference profiles. Because the supply temperature is expected to be rather stable throughout the entire study case DH subnetwork, the return temperature from the building should be directly correlated with the temperature difference. It is thus logical to observe that clusters A and B have low and stable return temperature profiles, whereas clusters C and D have higher return temperature profiles with more variations and 2 clear temperature drops at morning peak and at evening peak.

4.5. Correlations in between clusters and measured variables

In this section, a simple correlation analysis is performed by calculating the percentage of shared buildings between two clusters from different seasons or different measurement variables. An element of the correlation matrices is marked 1 if 100% of the buildings are shared between the two clusters, and it is marked 0 if there are no common buildings in between those clusters.

One can see in Figure 7 the matrix of correlations between the different clusters for heat energy use profile during the four seasons. One can notice that most of the terms of the matrix are zero or close to it, except the terms corresponding to the same cluster. This clearly indicates that most of the buildings do not change heat energy use categories in between the different seasons.



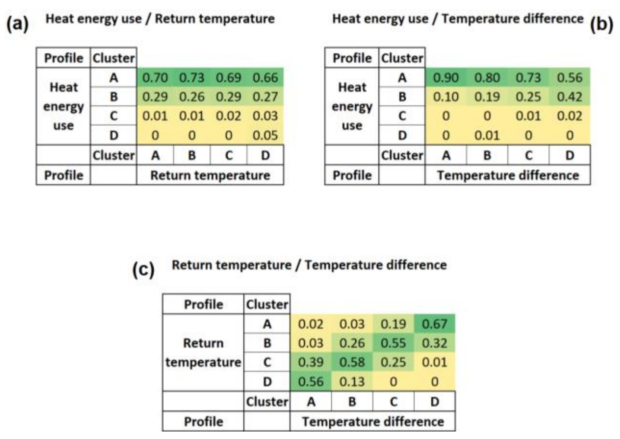
Season	Cluster																
Spring	A	1															
	B	0	1														
	C	0	0	1													
	D	0	0	0	1												
Summer	A	0.88	0.34	0.04	0	1											
	B	0.12	0.66	0.44	0.20	0	1										
	C	0	0	0.52	0	0	0	1									
	D	0	0	0	0.80	0	0	0	1								
Autumn	A	0.96	0.14	0	0	0.79	0.27	0	0	1							
	B	0.04	0.86	0.44	0	0.21	0.72	0.15	0	0	1						
	C	0	0	0.56	0	0	0.01	0.85	0	0	0	1					
	D	0	0	0	1	0	0	0	1	0	0	0	1				
Winter	A	0.99	0.36	0	0	0.86	0.45	0	0	0.99	0.30	0	0	1			
	B	0.01	0.64	0.52	0	0.14	0.54	0.31	0	0.01	0.70	0.14	0	0	1		
	C	0	0	0.48	0.20	0	0.01	0.69	0.25	0	0	0.86	0.20	0	0	1	
	D	0	0	0	0.80	0	0	0	0.75	0	0	0	0.80	0	0	0	1
	Cluster	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D
Season		Spring				Summer				Autumn				Winter			

Fig. 7. Matrix of correlations between the different clusters for daily profiles of the heat energy use (working days only) during the four seasons of Spring, Summer, Autumn, and Winter, respectively.

One can see in Figure 8 the matrices of correlations between the different clusters of daily profiles for heat energy use, return temperature from the building, and temperature difference between the supply and the return fluid. The correlation analysis is performed for the working days of the winter season only.

One can observe in Figure 8 (a) that there is no particular pattern for the correlation matrix of the heat energy use and the return temperature. The distribution of the return temperature clusters within the heat energy ones is very similar to that of the energy profile among all buildings, which would suggest that there is no particular correlation between those two variables.

On the other hand, one can see in Figure 8 (b) that there is a slight trend of buildings with a low heat energy use (towards cluster A) having also lower temperatures and vice versa.



(a) Heat energy use / Return temperature

Profile	Cluster				
Heat energy use	A	0.70	0.73	0.69	0.66
	B	0.29	0.26	0.29	0.27
	C	0.01	0.01	0.02	0.03
	D	0	0	0	0.05
	Cluster	A	B	C	D
Profile		Return temperature			

Heat energy use / Temperature difference (b)

Profile	Cluster				
Heat energy use	A	0.90	0.80	0.73	0.56
	B	0.10	0.19	0.25	0.42
	C	0	0	0.01	0.02
	D	0	0.01	0	0
	Cluster	A	B	C	D
Profile		Temperature difference			

(c) Return temperature / Temperature difference

Profile	Cluster				
Return temperature	A	0.02	0.03	0.19	0.67
	B	0.03	0.26	0.55	0.32
	C	0.39	0.58	0.25	0.01
	D	0.56	0.13	0	0
	Cluster	A	B	C	D
Profile		Temperature difference			

Fig. 8. Matrices of correlations between the different clusters of daily profiles for different measurement variables for the winter season only and for the working days only.

In agreement with what was assumed in the previous section, one can clearly identify in Figure 8 (c) the correlation between the temperature difference and the return temperature from the building.

5 Conclusions

In this paper, the authors have presented the different steps necessary for the pre-processing and cleansing of the raw data from smart energy meters in order to perform a K-means clustering analysis. The K-means clustering method is fairly simple to use, fast to compute and can be applied to time series, which makes it ideal to analyze, categorize and identify typical profiles or patterns in large measurement data sets from vast building clusters.

The study case of the current investigations is a small Danish town connected to the large district heating network of the local regional capital Aalborg. 1665 smart energy meters have been installed in all the dwellings of this locality, and are recording the total heat energy use, fluid flow, supply temperature, and return temperature to the district heating network.

The preliminary results from the study of a one-year data collection of those 1665 smart energy meters (out of which 1028 are kept for analysis) identified 4 distinct typical daily profiles of heat energy use, temperature difference between the supply and the return fluid, and return temperature from the building. Those profiles present clear patterns such as morning and evening peaks of heat energy demand, which lead to corresponding peaks in the profiles of temperature difference, and drops of the return temperature from the building. Those results are in agreement with previous observations of district heating systems in Denmark. In addition, a simple correlation analysis showed that the buildings of the study case tend to stay in the same heat energy use category from one season to the other. Finally, since the supply temperature is expected to be homogenous throughout the subnetwork of this building cluster, there is a strong correlation between the return temperature from the building and the temperature difference between supply and return fluid.

6 Future work

As the current paper only presents the preliminary results of this clustering analysis, many follow-up studies will be conducted on the smart energy meter measurement data set:

- Other pre-processing, missing data imputation and clustering methods will be tested.
- Further analysis of the data and coupling with the local weather information and the Danish national building register will be performed to identify building characteristics such as the envelope thermal performance, or the type of heating system installed.
- A challenging task will be to identify certain traits of the occupants' behavior and practices such as the domestic hot water usage profile, and the indoor temperature set point.
- Anonymized socio-economic studies would also be of great interest to assess the correlation between types of occupant and heating usage profiles.
- The "Shiny" package [14] will be used in the R environment to develop interactive web-based

interfaces to present and share the processed smart energy meter data with professionals of the building sector and with the DH utility companies.

- Further data analysis could also help to detect problems in the DH network or identify problematic and critical buildings in the network.
- Combined with high-quality GIS information about this DH network study case, this data set analysis will be used to generate, calibrate and validate an urban-scale numerical model based on the Modelica language, and sub-model component libraries and tools developed by the IBPSA Project 1 [15]. This detailed model will be able to simulate both the thermodynamics of the DH network and of every single building, which makes it suitable for testing urban-scale demand-side management and energy flexibility strategies for building clusters.

This work was financed by the InterHUB project which is funded by Aalborg University's strategic funding for interdisciplinary research. InterHUB project: Intermittent energy – Integrating Households, Utilities, and Buildings (www.interhub.aau.dk).

References

1. H. Lund, B. Möller, B.V. Mathiesen, A. Dyrelund, Energy **35**, 1381-1390 (2010)
2. B.V. Mathiesen, H. Lund, D. Connolly, H. Wenzel, P.A. Østergaard, B. Möller, S. Nielsen, I. Ridjan, P. Karnøe, K. Sperling, F.K. Hvelplund, Appl. Energy **145**, 139-154 (2015)
3. Danish Energy Agency, *Regulation and planning of district heating in Denmark* (2015)
4. P. Giannou, X. Liu, A. Heller, P.S. Nielsen, C. Rode, Energy Convers. Manag. **165**, 840-850 (2018)
5. P. Giannou, C. Reinhart, D. Hsu, A. Heller, C. Rode, Build. Environ. **139**, 125-133 (2018)
6. M.H. Kristensen, R.E. Hedegaard, S. Petersen, Energ. Buildings **175**, 219-234 (2018)
7. R.E. Hedegaard, M.H. Kristensen, T.H. Pedersen, A. Brun, S. Petersen, Appl. Energy **242**, 181-204 (2019)
8. Aalborg Forsyning, https://aalborgforsyning.dk
9. Kamstrup A/S, Smart heat meters & devices, https://www.kamstrup.com/en-en/heat-solutions/heat-meters
10. R environment for statistical computing, https://www.r-project.org/about.html
11. K-means clustering analysis with R, https://uc-r.github.io/kmeans_clustering
12. S. Moritz, T. Bartz-Beielstein, R J. **9**, 207-218 (2017)
13. Time Series Missing Value Imputation, R package, https://cran.r-project.org/web/packages/imputeTS/imputeTS.pdf
14. Shiny, R package, https://shiny.rstudio.com
15. IBPSA Project 1, https://ibpsa.github.io/project1

Script 1: Raw Data - Data Preprocessing B

```r
### Script 1: Organizes the Raw Data (Preprocessing)
    ----------------------------------------------------------------------------

### 1. Setting the main directory -----------------------------------------------------------------
## Setting the data for the folder where is the file with the data.

# Aalborg University
setwd("C:/Users/LAB-PC/OneDrive - Aalborg Universitet/Master Thesis/7. DH
    Analysis/Scripts/S1_Preprocessing")

### 2. Installing important packages
    -----------------------------------------------------------------

if (!require('data.table') & require('magrittr')==FALSE & require('imputeTS')==FALSE &
    require('lubridate')==FALSE & require('dplyr')==FALSE & require('readr')==FALSE &
    require('ggplot2')==FALSE & require('tictoc')==FALSE
   & require('DataExplorer')==FALSE & require('ggpubr')==FALSE & require('lemon')==FALSE &
       require('gghighlight')==FALSE & require('xts')==FALSE
   & require('tidyr')==FALSE & require('tibble')==FALSE & require('purrr')==FALSE &
       require('cluster')==FALSE & require('factoextra')==FALSE & require('leaflet')==FALSE){ #If
       NOT installed then install and check it to be used
  install.packages('data.table')
  install.packages('dplyr')
  install.packages('readr')
  install.packages('ggplot2')
  install.packages('tictoc')
  install.packages('imputeTS')
  install.packages('magrittr')
  install.packages('DataExplorer')
  install.packages('ggpubr')
  install.packages('lemon')
  install.packages('gghighlight')
  install.packages('xts')
  install.packages('tidyr')
  install.packages('tibble')
  install.packages('purrr')
  install.packages('cluster')
  install.packages('factoextra')
  install.packages("leaflet")

  library(data.table)
  library(dplyr)
  library(readr)
  library(ggplot2)
  library(tictoc)
  library(lubridate)
  library(imputeTS)
  library(magrittr)
  library(DataExplorer)
  library(ggpubr)
  library(lemon)
  library(gghighlight)
  library(viridis)
  library(hrbrthemes)
  library(xts)
  library(tidyr)
  library(tibble)
  library(purrr)
  library(cluster)
  library(factoextra)
  library(leaflet)
} else {                        #If already installed then "check it" in the menu "Packages" to be used
  library(data.table)
  library(dplyr)
  library(readr)
  library(ggplot2)
  library(tictoc)
  library(lubridate)
  library(imputeTS)
  library(magrittr)
  library(DataExplorer)
  library(ggpubr)
  library(lemon)
  library(gghighlight)
  library(viridis)
  library(hrbrthemes)
  library(xts)
  library(tidyr)
  library(tibble)
  library(purrr)
  library(cluster)
  library(factoextra)
  library(leaflet)
}

### 3. Import all the *.txt files in the main directory to a single data frame ---------------------------

tic('Time to load data in *.txt format')

temp = list.files(pattern="*.txt") #These function produce a character vector of the names of files or directories in the main directory.
DH_data <- rbindlist(sapply(temp, fread, simplify = FALSE, header = FALSE, sep = ";", dec = ","))

Addresses <- read.csv("List_Addresses_ID_Geocode.csv", header = FALSE, sep = ";", dec = ".", stringsAsFactors=FALSE) # File with DH_data addresses

## Header = FALSE -> There is no header's name on the given data.
## sep = ";" -> The data is separated by semicolons (;).
## fread -> Function that reads faster the table.
## dec = ',' -> The decimals are distinguished by commas.

toc()

### 4. Reshape the data ------------------------------------------------------------------------------------

tic('Time to reshape the data')

## Giving a name for each variable (columns' names) - Header
colnames(DH_data) <- c("Link_1","Link_2","TimeStep_Logging","Energy","Unit_Ener","Flow","Unit_Flow","Lifetime","Unit_LifeT","Flow*T_supply",

colnames(Addresses) <- c("Link_1","Address","Latitude","Longitude")

## All empty spaces '' convert to NA values
```

```
DH_data[DH_data==''] <- NA #Must be before the conversion of Time_Logging from character to date class, if not it will give a mistake!

## Convert TimeStep logging variable from 'character' data type to 'date' type
DH_data$TimeStep_Logging <- as.POSIXct(DH_data$TimeStep_Logging, tz = "", "%Y-%m-%d %H:%M") #Be careful with the difference between "as.POSIXct" and "as.POSIXlt"

## Force the data to be a specific class (ex: a column of numbers to be read as an number and not as a string)
DH_data$Link_1=as.numeric(DH_data$Link_1)
DH_data$Link_2=as.integer(DH_data$Link_2)
DH_data$Energy=as.numeric(DH_data$Energy)
DH_data$Unit_Ener=as.character(DH_data$Unit_Ener)
DH_data$Flow=as.numeric(DH_data$Flow)
DH_data$Unit_Flow=as.character(DH_data$Unit_Flow)
DH_data$Lifetime=as.integer(DH_data$Lifetime)
DH_data$Unit_LifeT=as.character(DH_data$Unit_LifeT)
DH_data$`Flow*T_supply`=as.numeric(DH_data$`Flow*T_supply`)
DH_data$`Unit_Flow&TS`=as.character(DH_data$`Unit_Flow&TS`)
DH_data$`Flow*T_return`=as.numeric(DH_data$`Flow*T_return`)
DH_data$`Unit_Flow&TR`=as.character(DH_data$`Unit_Flow&TR`)
DH_data$`T_supply_(inst)`=as.numeric(DH_data$`T_supply_(inst)`)
DH_data$Unit_TS=as.character(DH_data$Unit_TS)
DH_data$`T_return_(inst)`=as.numeric(DH_data$`T_return_(inst)`)
DH_data$Unit_TR=as.character(DH_data$Unit_TR)
DH_data$`Volume_(inst)`=as.numeric(DH_data$`Volume_(inst)`)
DH_data$Unit_Vlm=as.character(DH_data$Unit_Vlm)
DH_data$Error=as.character(DH_data$Error)
DH_data$Col21=as.numeric(DH_data$Col21)
DH_data$Col22=as.character(DH_data$Col22)
DH_data$Col23=as.numeric(DH_data$Col23)
DH_data$Col24=as.character(DH_data$Col24)
DH_data$`SmartMeter's_Name`=as.character(DH_data$`SmartMeter's_Name`)


toc()

### 5. Convert all values for the same units ----------------------------------------------------------
## Some values have different units, e.g. kWh and MWh in the Energy column

DH_data$Energy <- ifelse(DH_data$Unit_Ener == 'MWh' & !is.na(DH_data$Energy), DH_data$Energy*1000, DH_data$Energy)
DH_data$Unit_Ener[DH_data$Unit_Ener == 'MWh'] <- 'kWh'
DH_data$`Flow*T_supply` <- ifelse(DH_data$`Unit_Flow&TS` == 'm3xC x 0,1' & !is.na(DH_data$`Flow*T_supply`), DH_data$`Flow*T_supply`/0.1, DH_data$`Flow*T_supply`)
DH_data$`Unit_Flow&TS`[DH_data$`Unit_Flow&TS` == 'm3xC x 0,1'] <- 'm3xC'
DH_data$`Flow*T_return` <- ifelse(DH_data$`Unit_Flow&TR` == 'm3xC x 0,1' & !is.na(DH_data$`Flow*T_return`), DH_data$`Flow*T_return`/0.1, DH_data$`Flow*T_return`)
DH_data$`Unit_Flow&TR`[DH_data$`Unit_Flow&TR` == 'm3xC x 0,1'] <- 'm3xC'

### 6. Resample data according to time and creates columns for weekdays and seasons' names ----------------------------------------------------------

attr(DH_data$TimeStep_Logging, "tzone") <- "Europe/Copenhagen" #Convert to Europe/Copenhagen time

DH_data <- DH_data%>%
  mutate(Time_round = as.POSIXct(round(DH_data$TimeStep_Logging, "hours")))
toc()

### 4. Creates sequence of time -----------------------

DH_data_update <- split(DH_data,DH_data$Link_1)

add_time <- function(DH_data, Time_min, Time_max){

  Sequence_time <- as.data.frame(seq(Time_min, Time_max, by = "hour"))
  Sequence_time <- as.data.frame(Sequence_time[c(-651),]) #Issue: 2019-10-28 02:00:00 -> Eliminates it
  colnames(Sequence_time) <- c("Time_round") #Atributes another name in the column

  DH_data_ID <- mean(DH_data$Link_1, na.rm = TRUE)
  DH_data_Smeter <- mean(DH_data$Link_2)

  DH_data <- Sequence_time %>%
    transmute(Time_round = as.POSIXct(Time_round), Link_1 = DH_data_ID) #Function returns a dataframe with 2 columns: Date and Link_1

}

Time_min <- as.POSIXct(round(min(DH_data$TimeStep_Logging, na.rm = T), "hours")) #Minimum date in all data
Time_max <- as.POSIXct(round(max(DH_data$TimeStep_Logging, na.rm = T), "hours")) #Maximum date in all data

DH_data_update_list <- lapply(DH_data_update, add_time, Time_min = Time_min, Time_max = Time_max) #Applies the function "add_time" which has more than one variable

DH_data_update <- rbindlist(DH_data_update_list) #Binds all the buildings together in one dataframe

DH_data <- DH_data_update %>% #Joins dataframe of the time sequence with the DH_data dataset
  left_join(DH_data)


DH_data <- DH_data %>%
  mutate(Time_hour = hour(Time_round)+(minute(Time_round)/60)) #Adds a column with only the hours and minutes from the date

DH_data <- DH_data %>%                      #Create columns with day, month and year
  mutate(Day = day(DH_data$Time_round), Month = month(DH_data$Time_round), Year = year(DH_data$Time_round))

DH_data <- DH_data %>%
  mutate(Week_day = weekdays(DH_data$Time_round)) #Add a column with weekdays name
DH_data$Week_day=as.character(DH_data$Week_day) #Converts it to a string

DH_data <- DH_data %>% #Separate days regarding weekday or weekend
  mutate(Day_type = 0)
DH_data$Day_type[DH_data$Week_day == 'lrdag'] <- 'Weekend'
DH_data$Week_day[DH_data$Week_day == 'lrdag'] <- 'Saturday'
DH_data$Day_type[DH_data$Week_day == 'sndag'] <- 'Weekend'
DH_data$Week_day[DH_data$Week_day == 'sndag'] <- 'Sunday'
DH_data$Day_type[DH_data$Week_day == 'mandag'] <- 'Weekday'
DH_data$Week_day[DH_data$Week_day == 'mandag'] <- 'Monday'
DH_data$Day_type[DH_data$Week_day == 'tirsdag'] <- 'Weekday'
DH_data$Week_day[DH_data$Week_day == 'tirsdag'] <- 'Tuesday'
DH_data$Day_type[DH_data$Week_day == 'onsdag'] <- 'Weekday'
DH_data$Week_day[DH_data$Week_day == 'onsdag'] <- 'Wednesday'
DH_data$Day_type[DH_data$Week_day == 'torsdag'] <- 'Weekday'
DH_data$Week_day[DH_data$Week_day == 'torsdag'] <- 'Thursday'
DH_data$Day_type[DH_data$Week_day == 'fredag'] <- 'Weekday'



DH_data <- DH_data %>%
  mutate(Week_number = strftime(DH_data$Time_round, format = "%V")) #Add a column with week number
DH_data$Week_number=as.numeric(DH_data$Week_number) #Convert to a numeric value

DH_data <- DH_data %>%
  mutate(Season = month(DH_data$Time_round)) #Create column where the season is given regarding the month
DH_data$Season[DH_data$Season==month(as.POSIXct(’01/12/1996’, format="%d/%m/%Y"))] <- ’Winter’
DH_data$Season[DH_data$Season==month(as.POSIXct(’01/01/1996’, format="%d/%m/%Y"))] <- ’Winter’
DH_data$Season[DH_data$Season==month(as.POSIXct(’01/02/1996’, format="%d/%m/%Y"))] <- ’Winter’
DH_data$Season[DH_data$Season==month(as.POSIXct(’01/03/1996’, format="%d/%m/%Y"))] <- ’Spring’
DH_data$Season[DH_data$Season==month(as.POSIXct(’01/04/1996’, format="%d/%m/%Y"))] <- ’Spring’
DH_data$Season[DH_data$Season==month(as.POSIXct(’01/05/1996’, format="%d/%m/%Y"))] <- ’Spring’
DH_data$Season[DH_data$Season==month(as.POSIXct(’01/06/1996’, format="%d/%m/%Y"))] <- ’Summer’
DH_data$Season[DH_data$Season==month(as.POSIXct(’01/07/1996’, format="%d/%m/%Y"))] <- ’Summer’
DH_data$Season[DH_data$Season==month(as.POSIXct(’01/08/1996’, format="%d/%m/%Y"))] <- ’Summer’
DH_data$Season[DH_data$Season==month(as.POSIXct(’01/09/1996’, format="%d/%m/%Y"))] <- ’Fall’
DH_data$Season[DH_data$Season==month(as.POSIXct(’01/10/1996’, format="%d/%m/%Y"))] <- ’Fall’
DH_data$Season[DH_data$Season==month(as.POSIXct(’01/11/1996’, format="%d/%m/%Y"))] <- ’Fall’
DH_data$Season=as.character(DH_data$Season) #Convert to a string


### 7. Attribute "Weekend" for holidays in Denmark -------------------------------

DH_data <- DH_data %>%
  mutate(Holiday_day = day(DH_data$Time_round), Holiday_month = month(DH_data$Time_round), Holiday_year = year(DH_data$Time_round)) #Create column where the where the year/month/day is given

DH_data$Day_type[DH_data$Holiday_day==day(as.POSIXct(’01/01/2018’, format="%d/%m/%Y")) & DH_data$Holiday_month==month(as.POSIXct(’01/01/2018’, format="%d/%m/%Y")) & DH_data$Holiday_year==year(as.POSIXct(’01/01/2018’, format="%d/%m/%Y"))] <- ’Weekend’
DH_data$Day_type[DH_data$Holiday_day==day(as.POSIXct(’29/03/2018’, format="%d/%m/%Y")) & DH_data$Holiday_month==month(as.POSIXct(’29/03/2018’, format="%d/%m/%Y")) & DH_data$Holiday_year==year(as.POSIXct(’29/03/2018’, format="%d/%m/%Y"))] <- ’Weekend’
DH_data$Day_type[DH_data$Holiday_day==day(as.POSIXct(’30/03/2018’, format="%d/%m/%Y")) & DH_data$Holiday_month==month(as.POSIXct(’30/03/2018’, format="%d/%m/%Y")) & DH_data$Holiday_year==year(as.POSIXct(’30/03/2018’, format="%d/%m/%Y"))] <- ’Weekend’
DH_data$Day_type[DH_data$Holiday_day==day(as.POSIXct(’01/04/2018’, format="%d/%m/%Y")) & DH_data$Holiday_month==month(as.POSIXct(’01/04/2018’, format="%d/%m/%Y")) & DH_data$Holiday_year==year(as.POSIXct(’01/04/2018’, format="%d/%m/%Y"))] <- ’Weekend’
DH_data$Day_type[DH_data$Holiday_day==day(as.POSIXct(’02/04/2018’, format="%d/%m/%Y")) & DH_data$Holiday_month==month(as.POSIXct(’02/04/2018’, format="%d/%m/%Y")) & DH_data$Holiday_year==year(as.POSIXct(’02/04/2018’, format="%d/%m/%Y"))] <- ’Weekend’
DH_data$Day_type[DH_data$Holiday_day==day(as.POSIXct(’27/04/2018’, format="%d/%m/%Y")) & DH_data$Holiday_month==month(as.POSIXct(’27/04/2018’, format="%d/%m/%Y")) & DH_data$Holiday_year==year(as.POSIXct(’27/04/2018’, format="%d/%m/%Y"))] <- ’Weekend’
DH_data$Day_type[DH_data$Holiday_day==day(as.POSIXct(’10/05/2018’, format="%d/%m/%Y")) & DH_data$Holiday_month==month(as.POSIXct(’10/05/2018’, format="%d/%m/%Y")) & DH_data$Holiday_year==year(as.POSIXct(’10/05/2018’, format="%d/%m/%Y"))] <- ’Weekend’
DH_data$Day_type[DH_data$Holiday_day==day(as.POSIXct(’20/05/2018’, format="%d/%m/%Y")) & DH_data$Holiday_month==month(as.POSIXct(’20/05/2018’, format="%d/%m/%Y")) & DH_data$Holiday_year==year(as.POSIXct(’20/05/2018’, format="%d/%m/%Y"))] <- ’Weekend’
DH_data$Day_type[DH_data$Holiday_day==day(as.POSIXct(’21/05/2018’, format="%d/%m/%Y")) & DH_data$Holiday_month==month(as.POSIXct(’21/05/2018’, format="%d/%m/%Y")) & DH_data$Holiday_year==year(as.POSIXct(’21/05/2018’, format="%d/%m/%Y"))] <- ’Weekend’
DH_data$Day_type[DH_data$Holiday_day==day(as.POSIXct(’24/12/2018’, format="%d/%m/%Y")) & DH_data$Holiday_month==month(as.POSIXct(’24/12/2018’, format="%d/%m/%Y")) & DH_data$Holiday_year==year(as.POSIXct(’24/12/2018’, format="%d/%m/%Y"))] <- ’Weekend’
DH_data$Day_type[DH_data$Holiday_day==day(as.POSIXct(’25/12/2018’, format="%d/%m/%Y")) & DH_data$Holiday_month==month(as.POSIXct(’25/12/2018’, format="%d/%m/%Y")) & DH_data$Holiday_year==year(as.POSIXct(’25/12/2018’, format="%d/%m/%Y"))] <- ’Weekend’
DH_data$Day_type[DH_data$Holiday_day==day(as.POSIXct(’26/12/2018’, format="%d/%m/%Y")) & DH_data$Holiday_month==month(as.POSIXct(’26/12/2018’, format="%d/%m/%Y")) & DH_data$Holiday_year==year(as.POSIXct(’26/12/2018’, format="%d/%m/%Y"))] <- ’Weekend’
DH_data$Day_type[DH_data$Holiday_day==day(as.POSIXct(’31/12/2018’, format="%d/%m/%Y")) & DH_data$Holiday_month==month(as.POSIXct(’31/12/2018’, format="%d/%m/%Y")) & DH_data$Holiday_year==year(as.POSIXct(’31/12/2018’, format="%d/%m/%Y"))] <- ’Weekend’

DH_data$Day_type[DH_data$Holiday_day==day(as.POSIXct(’01/01/2019’, format="%d/%m/%Y")) & DH_data$Holiday_month==month(as.POSIXct(’01/01/2019’, format="%d/%m/%Y")) &
```

```
    DH_data$Holiday_year==year(as.POSIXct('01/01/2019', format="%d/%m/%Y"))] <- 'Weekend'
DH_data$Day_type[DH_data$Holiday_day==day(as.POSIXct('18/04/2019', format="%d/%m/%Y")) &
    DH_data$Holiday_month==month(as.POSIXct('18/04/2019', format="%d/%m/%Y")) &
    DH_data$Holiday_year==year(as.POSIXct('18/04/2019', format="%d/%m/%Y"))] <- 'Weekend'
DH_data$Day_type[DH_data$Holiday_day==day(as.POSIXct('19/04/2019', format="%d/%m/%Y")) &
    DH_data$Holiday_month==month(as.POSIXct('19/04/2019', format="%d/%m/%Y")) &
    DH_data$Holiday_year==year(as.POSIXct('19/04/2019', format="%d/%m/%Y"))] <- 'Weekend'
DH_data$Day_type[DH_data$Holiday_day==day(as.POSIXct('21/04/2019', format="%d/%m/%Y")) &
    DH_data$Holiday_month==month(as.POSIXct('21/04/2019', format="%d/%m/%Y")) &
    DH_data$Holiday_year==year(as.POSIXct('21/04/2019', format="%d/%m/%Y"))] <- 'Weekend'
DH_data$Day_type[DH_data$Holiday_day==day(as.POSIXct('22/04/2019', format="%d/%m/%Y")) &
    DH_data$Holiday_month==month(as.POSIXct('22/04/2019', format="%d/%m/%Y")) &
    DH_data$Holiday_year==year(as.POSIXct('22/04/2019', format="%d/%m/%Y"))] <- 'Weekend'
DH_data$Day_type[DH_data$Holiday_day==day(as.POSIXct('17/05/2019', format="%d/%m/%Y")) &
    DH_data$Holiday_month==month(as.POSIXct('17/05/2019', format="%d/%m/%Y")) &
    DH_data$Holiday_year==year(as.POSIXct('17/05/2019', format="%d/%m/%Y"))] <- 'Weekend'
DH_data$Day_type[DH_data$Holiday_day==day(as.POSIXct('30/05/2019', format="%d/%m/%Y")) &
    DH_data$Holiday_month==month(as.POSIXct('30/05/2019', format="%d/%m/%Y")) &
    DH_data$Holiday_year==year(as.POSIXct('30/05/2019', format="%d/%m/%Y"))] <- 'Weekend'
DH_data$Day_type[DH_data$Holiday_day==day(as.POSIXct('09/06/2019', format="%d/%m/%Y")) &
    DH_data$Holiday_month==month(as.POSIXct('09/06/2019', format="%d/%m/%Y")) &
    DH_data$Holiday_year==year(as.POSIXct('09/06/2019', format="%d/%m/%Y"))] <- 'Weekend'
DH_data$Day_type[DH_data$Holiday_day==day(as.POSIXct('10/06/2019', format="%d/%m/%Y")) &
    DH_data$Holiday_month==month(as.POSIXct('10/06/2019', format="%d/%m/%Y")) &
    DH_data$Holiday_year==year(as.POSIXct('10/06/2019', format="%d/%m/%Y"))] <- 'Weekend'
DH_data$Day_type[DH_data$Holiday_day==day(as.POSIXct('24/12/2019', format="%d/%m/%Y")) &
    DH_data$Holiday_month==month(as.POSIXct('24/12/2019', format="%d/%m/%Y")) &
    DH_data$Holiday_year==year(as.POSIXct('24/12/2019', format="%d/%m/%Y"))] <- 'Weekend'
DH_data$Day_type[DH_data$Holiday_day==day(as.POSIXct('25/12/2019', format="%d/%m/%Y")) &
    DH_data$Holiday_month==month(as.POSIXct('25/12/2019', format="%d/%m/%Y")) &
    DH_data$Holiday_year==year(as.POSIXct('25/12/2019', format="%d/%m/%Y"))] <- 'Weekend'
DH_data$Day_type[DH_data$Holiday_day==day(as.POSIXct('26/12/2019', format="%d/%m/%Y")) &
    DH_data$Holiday_month==month(as.POSIXct('26/12/2019', format="%d/%m/%Y")) &
    DH_data$Holiday_year==year(as.POSIXct('26/12/2019', format="%d/%m/%Y"))] <- 'Weekend'
DH_data$Day_type[DH_data$Holiday_day==day(as.POSIXct('31/12/2019', format="%d/%m/%Y")) &
    DH_data$Holiday_month==month(as.POSIXct('31/12/2019', format="%d/%m/%Y")) &
    DH_data$Holiday_year==year(as.POSIXct('31/12/2019', format="%d/%m/%Y"))] <- 'Weekend'

DH_data <- DH_data %>%
  select(-Holiday_day, -Holiday_month, -Holiday_year)

DH_data <- DH_data %>% #Join DH_data data with addressess and geocodes data
  inner_join(Addresses)

DH_data <- DH_data %>% #Eliminates possible repeated lines
  distinct()

### 8. Save the data.frame into a *.Rda file and store it in the main directory
    ----------------------------------------------------------

tic('Time to save the data.frame in the main directory')

save(DH_data, file="DH_data_20182019.Rda")

z <- DH_data %>%                      # Data frame without units information and organized by date
    and Building ID
  arrange(Time_round) %>%
  arrange(Link_1) %>%
  select(-Unit_Ener,-Unit_Flow,-Unit_LifeT,-`Unit_Flow&TS`,-`Unit_Flow&TR`,-Unit_TS,-Unit_TR,-Unit_Vlm)

save(z, file="z_1.Rda") #Save dataframe to be used in the next scripts

toc()
```

Listing B.1. Script 1: Raw Data

Script 2: Errors - Data Preprocessing C

```r
### Script 2: Errors (Preprocessing)

### 1. Setting the main directory -----------------------------------------------------------------
## Setting the data for the folder where is the file with the data.

# Aalborg University - Full year
setwd("C:/Users/LAB-PC/OneDrive - Aalborg Universitet/Master Thesis/7. DH
    Analysis/Scripts/S1_Preprocessing")

### 2. Installing important packages
    -----------------------------------------------------------------

if (!require('data.table') & require('magrittr')==FALSE & require('imputeTS')==FALSE &
    require('lubridate')==FALSE & require('dplyr')==FALSE & require('readr')==FALSE &
    require('ggplot2')==FALSE & require('tictoc')==FALSE
   & require('DataExplorer')==FALSE & require('ggpubr')==FALSE & require('lemon')==FALSE &
       require('gghighlight')==FALSE & require('xts')==FALSE
   & require('tidyr')==FALSE & require('tibble')==FALSE & require('purrr')==FALSE &
       require('cluster')==FALSE & require('factoextra')==FALSE){ #If NOT installed then install
       and check it to be used
  install.packages('data.table')
  install.packages('dplyr')
  install.packages('readr')
  install.packages('ggplot2')
  install.packages('tictoc')
  install.packages('imputeTS')
  install.packages('magrittr')
  install.packages('DataExplorer')
  install.packages('ggpubr')
  install.packages('lemon')
  install.packages('gghighlight')
  install.packages('xts')
  install.packages('tidyr')
  install.packages('tibble')
  install.packages('purrr')
  install.packages('cluster')
  install.packages('factoextra')

  library(data.table)
  library(dplyr)
  library(readr)
  library(ggplot2)
  library(tictoc)
  library(lubridate)
  library(imputeTS)
  library(magrittr)
  library(DataExplorer)
  library(ggpubr)
  library(lemon)
  library(gghighlight)
  library(viridis)
  library(hrbrthemes)
  library(xts)
  library(tidyr)
  library(tibble)
  library(purrr)
  library(cluster)
  library(factoextra)
} else {                       #If already installed then "check it" in the menu "Packages" to be used
  library(data.table)
  library(dplyr)
  library(readr)
  library(ggplot2)
  library(tictoc)
  library(lubridate)
  library(imputeTS)
  library(magrittr)
  library(DataExplorer)
  library(ggpubr)
  library(lemon)
  library(gghighlight)
  library(viridis)
  library(hrbrthemes)
  library(xts)
  library(tidyr)
  library(tibble)
  library(purrr)
  library(cluster)
  library(factoextra)
}

### 3. Load the *.Rda file from the main directory ------------------------------------------------------------

tic('Time to load the data.frame')

load("z_1.Rda") #Load dataframe

toc()

### 4. List of functions ------------------------------------------------------------

convert_error_DK_EN <- function(df){

  df$Error[df$Error == "Temperaturforskel har forkert polaritet"] <- "Temperature difference has wrong polarity"
  df$Error[df$Error == "Flowmler V1, forkert flowretning"] <- "V1 Wrong flow direction"
  df$Error[df$Error == "Flowmler har forkert flowretning"] <- "Flow sensor with wrong flow direction"
  df$Error[df$Error == "Der er fundet luft i flowmleren (angives kun, s   lnge   fejlen findes)"] <- "Air has been detected in the flow meter (only indicated as long as the fault exists)"
  df$Error[df$Error == "Flowmler med svagt signal eller luft"] <- "Flow sensor with weak signal or air"
  df$Error[df$Error == "Temperaturfler t2 under  grnse  eller kortsluttet"] <- "t2 Below measuring range or short-circuited"

  return(df)
} #Function: Converts the errors written in Danish to English

convert_error_to_number<- function(df){

  df$Error[df$Error == "Temperature difference has wrong polarity"] <- 1
  df$Error[df$Error == "V1 Wrong flow direction"] <- 2
  df$Error[df$Error == "Flow sensor with wrong flow direction"] <- 3
  df$Error[df$Error == "Air has been detected in the flow meter (only indicated as long as the fault exists)"] <- 4
  df$Error[df$Error == "Flow sensor with weak signal or air"] <- 5
  df$Error[df$Error == "t2 Below measuring range or short-circuited"] <- 6

  return(df)
} #Function: Converts the errors types into numbers


convert_value_to_NA <- function(df){
  df$Energy[!is.na(df$Error)] <- NA
  df$Flow[!is.na(df$Error)] <- NA
  df$‘Flow*T_supply‘[!is.na(df$Error)] <- NA
  df$‘Flow*T_return‘[!is.na(df$Error)] <- NA
  df$‘T_supply_(inst)‘[!is.na(df$Error)] <- NA
  df$‘T_return_(inst)‘[!is.na(df$Error)] <- NA
  df$‘Volume_(inst)‘[!is.na(df$Error)] <- NA

  return(df)
} #Function: When a data point presents an error, converts all its significant variables into NA
     values

### 5. Reshape dataframe ---------------------------------------------------------------------------

tic('Time to reshape dataframe')

z <-  convert_error_DK_EN(z) # Convert error types from Danish language to English

z$Link_1 <- as.character(z$Link_1) #Converts to a string Link_1
z$Link_2 <- as.character(z$Link_2) #Converts to a string Link_2

toc()


### 6. Errors Plots
     ---------------------------------------------------------------------------------------

n_filter <- 10 #Number of buildings to use in plots

z_error_total <- z %>% #Creates table with all the total number of rows per building ID
  group_by(Link_1) %>%
  summarise(N_total=n())

z_error_not_NA <- z %>% #Table with values without NA values, meaning table only with errors
  filter(!is.na(Error))

z_error_NA_total <- z_error_not_NA %>% #Total number of errors per ID building
  group_by(Link_1) %>%
  summarise(N_error=n())

z_error <- z_error_NA_total %>% #Joins above tables with a column of percentage of errors per
     building ID, organized by descending order
  inner_join(z_error_total) %>%
  mutate(Percentage_error = round((N_error/N_total)*100,digits = 2)) %>%
  arrange(desc(Percentage_error))

z_error_filter <- z_error %>%
  head(n_filter) #Filter by a number of buildings that was established before to be used for the
       plots!

## Total percentage of errors per building

Plt_error_percentage <- ggplot(data = z_error_filter, aes(x=reorder(Link_1, -Percentage_error),
     y=Percentage_error)) + #Bar plot with the total percentage of errors
  geom_bar(stat = "identity", fill="sky blue") +
  labs(title="Percentage of errors vs Building ID", subtitle = "Full year", y="Percentage of errors
       [%]", x="Building ID") +
  geom_text(aes(label=Percentage_error), vjust=-0.6, color="black",
          position = position_dodge(0.9), size=3.5) +
  theme_minimal()

## Percentage of different types of errors per ID building

z_error_diff <- z_error_not_NA %>% #Total number of the different types of errors per building
  group_by(Link_1, Error) %>%
  summarise(N_total_diff=n())

z_error_diff_filter <- z_error_filter %>% #To use the same number of selected buildings for plotting
  inner_join(z_error_diff) %>%
  mutate(Percentage_error_diff = round((N_total_diff/N_total)*100,digits = 2))

# Plot: Bar plot with total number of each error per building stacked together
Plt_error_diff_percent <- ggplot(z_error_diff_filter, aes(fill=Error, y=Percentage_error_diff,
     x=reorder(Link_1, -Percentage_error_diff))) +
  geom_bar(position="stack", stat="identity") +
  labs(title="Percentage of different errors vs Building ID", subtitle = "Full year", y="Percentage
       of different errors [%]", x="Building ID")

z_error_diff_filter_1 <- convert_error_to_number(z_error_diff_filter)

# Plot: Bar plot with total number of each error per building separated with percentage above
Plt_error_diff_percent_1 <- ggplot(z_error_diff_filter_1, aes(fill=Error, y=Percentage_error_diff,
     x=Error)) +
  geom_bar(position="dodge", width=1, stat="identity") +
  labs(title = "Percentage of different errors per Building ID", subtitle = "Full year",
       y="Percentage of errors [%]", x="Error type") +
  facet_rep_wrap(~Link_1, scales = "free_y", repeat.tick.labels = TRUE) +
  geom_text(aes(label = Percentage_error_diff), vjust = 0.5) +
  scale_fill_brewer(palette = "RdBu")

### 7. Convert values into NA --------------------------------

z <- convert_value_to_NA(z) #Convert error into NA

z <- z %>%
  mutate(data_changed_error = ifelse(!is.na(z$Error), TRUE, FALSE)) #Knows if the value was an
       error before conversion
percentage_of_conversion <-
     (length(z$data_changed_error[z$data_changed_error==TRUE])/length(z$data_changed_error))*100
     #Percentage of errors converted into NA

save(z, file="z_2.Rda")
```

Listing C.1. Script 2: Errors

Script 3: Outliers - Data Preprocessing D

```r
### Script 3: Outliers (Preprocessing)

### 1. Setting the main directory -----------------------------------------------------------------
## Setting the data for the folder where is the file with the data.

# Aalborg University
setwd("C:/Users/LAB-PC/OneDrive - Aalborg Universitet/Master Thesis/7. DH
     Analysis/Scripts/S1_Preprocessing")

### 2. Installing important packages
     -----------------------------------------------------------------

if (!require('data.table') & require('magrittr')==FALSE & require('imputeTS')==FALSE &
     require('lubridate')==FALSE & require('dplyr')==FALSE & require('readr')==FALSE &
     require('ggplot2')==FALSE & require('tictoc')==FALSE
    & require('DataExplorer')==FALSE & require('ggpubr')==FALSE & require('lemon')==FALSE &
        require('gghighlight')==FALSE & require('xts')==FALSE
    & require('tidyr')==FALSE & require('tibble')==FALSE & require('purrr')==FALSE &
        require('cluster')==FALSE & require('factoextra')==FALSE){ #If NOT installed then install
        and check it to be used
  install.packages('data.table')
  install.packages('dplyr')
  install.packages('readr')
  install.packages('ggplot2')
  install.packages('tictoc')
  install.packages('imputeTS')
  install.packages('magrittr')
  install.packages('DataExplorer')
  install.packages('ggpubr')
  install.packages('lemon')
  install.packages('gghighlight')
  install.packages('xts')
  install.packages('tidyr')
  install.packages('tibble')
  install.packages('purrr')
  install.packages('cluster')
  install.packages('factoextra')

  library(data.table)
  library(dplyr)
  library(readr)
  library(ggplot2)
  library(tictoc)
  library(lubridate)
  library(imputeTS)
  library(magrittr)
  library(DataExplorer)
  library(ggpubr)
  library(lemon)
  library(gghighlight)
  library(viridis)
  library(hrbrthemes)
  library(xts)
  library(tidyr)
  library(tibble)
  library(purrr)
  library(cluster)
  library(factoextra)
} else {                        #If already installed then "check it" in the menu "Packages" to be
    used
  library(data.table)
  library(dplyr)
  library(readr)
  library(ggplot2)
  library(tictoc)
  library(lubridate)
  library(imputeTS)
  library(magrittr)
  library(DataExplorer)
  library(ggpubr)
  library(lemon)
  library(gghighlight)
  library(viridis)
  library(hrbrthemes)
  library(xts)
  library(tidyr)
  library(tibble)
  library(purrr)
  library(cluster)
  library(factoextra)
}

### 3. Load the *.Rda file from the main directory
    ------------------------------------------------------------

tic('Time to load the data.frame')

# Full year
load("z_2.Rda")  #Load data from previous script

toc()

### 4. List of functions ------------------------------------------------------------

convert_value_to_NA <- function(df){
  df$Energy[df$outlier_energy == TRUE] <- NA
  df$Flow[df$outlier_flow == TRUE] <- NA
  df$‘Flow*T_supply‘[df$outlier_flow_ts == TRUE] <- NA
  df$‘Flow*T_return‘[df$outlier_flow_tr == TRUE] <- NA
  df$‘T_supply_(inst)‘[df$outlier_tsupply == TRUE] <- NA
  df$‘T_return_(inst)‘[df$outlier_treturn == TRUE] <- NA
  df$‘Volume_(inst)‘[df$outlier_volume == TRUE] <- NA


  return(df)
} #Function: When a data point presents an outlier, converts into NA value


### 5. Group data according to ID, season and day type
    ...............................................

z_group <- split(z,list(z$Link_1, z$Season, z$Day_type))


### 6. Find outliers: Functions --------------------------------

MSD_outlier_Energy <- function(x){
  mean_x <- mean(x$Energy, na.rm = TRUE) #Mean value
  sd_x <- sd(x$Energy, na.rm = TRUE) #Standard deviation
  x %>%
    mutate(
      outlier_score = (x$Energy - mean_x)/sd_x, #Z-score
      outlier_energy = abs(outlier_score) > qnorm(0.9995), #If True, then it's an outlier:
          conditions
      outlier_energy = ifelse(is.na(outlier_energy), FALSE, outlier_energy),
      outlier_energy = ifelse(x$`Volume_(inst)` == 0, FALSE, outlier_energy)
    ) %>%
    select(-outlier_score)

} #Mean and Standard Deviation method: Energy


MSD_outlier_Flow <- function(x){
  mean_x <- mean(x$Flow, na.rm = TRUE)
  sd_x <- sd(x$Flow, na.rm = TRUE)
  x %>%
    mutate(
      outlier_score = (x$Flow - mean_x)/sd_x,
      outlier_flow = abs(outlier_score) > qnorm(0.9995), #If True, then it's an outlier
      outlier_flow = ifelse(is.na(outlier_flow), FALSE, outlier_flow),
      outlier_flow = ifelse(x$`Volume_(inst)` == 0, FALSE, outlier_flow)
    ) %>%
    select(-outlier_score)

} #Mean and Standard Deviation method: Flow

MSD_outlier_Flow_TS <- function(x){
  mean_x <- mean(x$`Flow*T_supply`, na.rm = TRUE)
  sd_x <- sd(x$`Flow*T_supply`, na.rm = TRUE)
  x %>%
    mutate(
      outlier_score = (x$`Flow*T_supply` - mean_x)/sd_x,
      outlier_flow_ts = abs(outlier_score) > qnorm(0.9995), #If True, then it's an outlier
      outlier_flow_ts = ifelse(is.na(outlier_flow_ts), FALSE, outlier_flow_ts),
      outlier_flow_ts = ifelse(x$`Volume_(inst)` == 0, FALSE, outlier_flow_ts)
    ) %>%
    select(-outlier_score)

} #Mean and Standard Deviation method: Flow*T_supply

MSD_outlier_Flow_TR <- function(x){
  mean_x <- mean(x$`Flow*T_return`, na.rm = TRUE)
  sd_x <- sd(x$`Flow*T_return`, na.rm = TRUE)
  x %>%
    mutate(
      outlier_score = (x$`Flow*T_return` - mean_x)/sd_x,
      outlier_flow_tr = abs(outlier_score) > qnorm(0.9995), #If True, then it's an outlier
      outlier_flow_tr = ifelse(is.na(outlier_flow_tr), FALSE, outlier_flow_tr),
      outlier_flow_tr = ifelse(x$`Volume_(inst)` == 0, FALSE, outlier_flow_tr)
    ) %>%
    select(-outlier_score)

} #Mean and Standard Deviation method: Flow*T_return

MSD_outlier_TS <- function(x){
  mean_x <- mean(x$`T_supply_(inst)`, na.rm = TRUE)
  sd_x <- sd(x$`T_supply_(inst)`, na.rm = TRUE)
  x %>%
    mutate(
      outlier_score = (x$`T_supply_(inst)` - mean_x)/sd_x,
      outlier_tsupply = abs(outlier_score) > qnorm(0.9995), #If True, then it's an outlier
      outlier_tsupply = ifelse(is.na(outlier_tsupply), FALSE, outlier_tsupply),
      outlier_tsupply = ifelse(x$`Volume_(inst)` == 0, FALSE, outlier_tsupply)
    ) %>%
    select(-outlier_score)

} #Mean and Standard Deviation method: T_supply (instantaneous)

MSD_outlier_TR <- function(x){
  mean_x <- mean(x$`T_return_(inst)`, na.rm = TRUE)
  sd_x <- sd(x$`T_return_(inst)`, na.rm = TRUE)
  x %>%
    mutate(
      outlier_score = (x$`T_return_(inst)` - mean_x)/sd_x,
      outlier_treturn = abs(outlier_score) > qnorm(0.9995), #If True, then it's an outlier
      outlier_treturn = ifelse(is.na(outlier_treturn), FALSE, outlier_treturn),
      outlier_treturn = ifelse(x$`Volume_(inst)` == 0, FALSE, outlier_treturn)
    ) %>%
    select(-outlier_score)

} #Mean and Standard Deviation method: T_return (instantaneous)

MSD_outlier_Volume <- function(x){
  mean_x <- mean(x$`Volume_(inst)`, na.rm = TRUE)
  sd_x <- sd(x$`Volume_(inst)`, na.rm = TRUE)
  x %>%
    mutate(
      outlier_score = (x$`Volume_(inst)` - mean_x)/sd_x,
      outlier_volume = abs(outlier_score) > qnorm(0.9995), #If True, then it's an outlier
      outlier_volume = ifelse(is.na(outlier_volume), FALSE, outlier_volume),
      outlier_volume = ifelse(x$`Volume_(inst)` == 0, FALSE, outlier_volume)
    ) %>%
    select(-outlier_score)

} #Mean and Standard Deviation method: Volume (instantaneous)

### 7. Run functions to detect outliers and join values in the tables -------------

z_group <- lapply(z_group, MSD_outlier_Energy) %>% #Detect outliers
  lapply(MSD_outlier_Flow) %>%
  lapply(MSD_outlier_Flow_TS) %>%
  lapply(MSD_outlier_Flow_TR) %>%
  lapply(MSD_outlier_TS) %>%
  lapply(MSD_outlier_TR) %>%
  lapply(MSD_outlier_Volume)

z <- rbindlist(z_group) #Join values in a single table

z <- convert_value_to_NA(z) #Run function to convert outliers into NA values

z <- z %>%  #Create column that shows if any value was an outlier changed to NA, returns value TRUE
  mutate(data_changed_outlier = ifelse((outlier_energy | outlier_flow | outlier_flow_ts |
       outlier_flow_tr | outlier_tsupply | outlier_treturn | outlier_volume) == TRUE, TRUE, FALSE))
       %>%
  select(-outlier_energy, -outlier_flow, -outlier_flow_ts, -outlier_flow_tr, -outlier_tsupply,
       -outlier_treturn, -outlier_volume)

z$data_changed_outlier[is.na(z$data_changed_outlier)] <- FALSE

# Calculate percentage of outliers
percentage_of_conversion <-
     (length(z$data_changed_outlier[z$data_changed_outlier==TRUE])/length(z$data_changed_outlier))*100


save(z, file="z_3.Rda") #Save dataframe
```

Listing D.1. Script 3: Outliers

Script 4: Missing Values - Data Preprocessing E

```r
### Script 4: NA Values - Missing Values (Preprocessing)

### 1. Setting the main directory -----------------------------------------------------------------
## Setting the data for the folder where is the file with the data.

# Aalborg University
setwd("C:/Users/LAB-PC/OneDrive - Aalborg Universitet/Master Thesis/7. DH
    Analysis/Scripts/S1_Preprocessing")

### 2. Installing important packages
    -----------------------------------------------------------------

if (!require('data.table') & require('magrittr')==FALSE & require('imputeTS')==FALSE &
    require('lubridate')==FALSE & require('dplyr')==FALSE & require('readr')==FALSE &
    require('ggplot2')==FALSE & require('tictoc')==FALSE
    & require('DataExplorer')==FALSE & require('ggpubr')==FALSE & require('lemon')==FALSE &
        require('gghighlight')==FALSE & require('xts')==FALSE
    & require('tidyr')==FALSE & require('tibble')==FALSE & require('purrr')==FALSE &
        require('cluster')==FALSE & require('factoextra')==FALSE){ #If NOT installed then install
        and check it to be used
  install.packages('data.table')
  install.packages('dplyr')
  install.packages('readr')
  install.packages('ggplot2')
  install.packages('tictoc')
  install.packages('imputeTS')
  install.packages('magrittr')
  install.packages('DataExplorer')
  install.packages('ggpubr')
  install.packages('lemon')
  install.packages('gghighlight')
  install.packages('xts')
  install.packages('tidyr')
  install.packages('tibble')
  install.packages('purrr')
  install.packages('cluster')
  install.packages('factoextra')

  library(data.table)
  library(dplyr)
  library(readr)
  library(ggplot2)
  library(tictoc)
  library(lubridate)
  library(imputeTS)
  library(magrittr)
  library(DataExplorer)
  library(ggpubr)
  library(lemon)
  library(gghighlight)
  library(viridis)
  library(hrbrthemes)
  library(xts)
  library(tidyr)
  library(tibble)
  library(purrr)
  library(cluster)
  library(factoextra)
} else {                        #If already installed then "check it" in the menu "Packages" to be
    used
  library(data.table)
  library(dplyr)
  library(readr)
  library(ggplot2)
  library(tictoc)
  library(lubridate)
  library(imputeTS)
  library(magrittr)
  library(DataExplorer)
  library(ggpubr)
  library(lemon)
  library(gghighlight)
  library(viridis)
  library(hrbrthemes)
  library(xts)
  library(tidyr)
  library(tibble)
  library(purrr)
  library(cluster)
  library(factoextra)
}


### 3. Load the *.Rda file from the main directory
    ------------------------------------------------------------

tic('Time to load the data.frame')

# Full year
load("z_3.Rda")  #Load data from previous script

toc()

z <- z %>%  #Detect NA values originated by errors, outliers or normal NA
  mutate(data_changed_NA = ifelse(
    data_changed_error == FALSE & data_changed_outlier == FALSE & (is.na(Energy) | is.na(Flow) |
        is.na(`Flow*T_supply`) | is.na(`Flow*T_return`) | is.na(`T_supply_(inst)`) |
        is.na(`T_return_(inst)`) | is.na(`Volume_(inst)`)),
    TRUE,
    FALSE
    ), NA_value = ifelse(data_changed_error == TRUE | data_changed_outlier == TRUE |
        data_changed_NA == TRUE, TRUE, FALSE))

z$data_changed_NA[is.na(z$TimeStep_Logging)] <- TRUE #Detect NA values in Time column

z <- z %>%
  mutate(z_day = day(z$Time_round), z_month = month(z$Time_round), z_year = year(z$Time_round)) %>%
                  # Data frame without units information and organized by date and Building ID
  arrange(Time_round) %>%
  arrange(Link_1)  #Create columns where the where the year/month/day are given

z <- z %>%
  filter(z_year == 2019 | (z_day >= 5 & z_month == 11 & z_year == 2018) | (z_month == 12 & z_year
      == 2018)) #Remove October until beginning of November



```

```r
## Detect NA Gaps

NA_vector <- rle(z$NA_value == TRUE)

z <- z %>%
  mutate(Gap = rep(NA_vector$values*NA_vector$lengths, NA_vector$lengths))

Gap_max <- z %>%      #Dataframe with highest gap in a building
  group_by(Link_1) %>%
  summarize(max_gap=max(Gap)) %>%
  arrange(desc(max_gap))

z <- z %>%           #Join with dataset
  inner_join(Gap_max)

z <- z %>%
  filter(max_gap < 10) #Less than 10 hours gaps (More will be removed)



### 4. NA values ---------------------------------------------

#NA percentage for each building
n_filter <- 30 #Number of buildings for plot

#Number of rows by ID
n_rows_total_ID <- z %>%
  group_by(Link_1) %>%
  summarise(N_total=n())

# Number of rows with NA values by ID
n_rows_NA_ID <- z %>%
  filter(NA_value == TRUE) %>%
  group_by(Link_1) %>%
  summarise(N_total_NA=n())

# Percentage of NA values by ID
z_NA_percent1 <- n_rows_NA_ID %>%
  inner_join(n_rows_total_ID) %>%
  mutate(Percentage_NA = ifelse(!is.na(N_total_NA), round((N_total_NA/N_total)*100,digits = 2), 0))
     %>%
  arrange(desc(Percentage_NA))

z_NA_percent <- z_NA_percent1 %>%
  head(n_filter)


# Bar plot of percentages vs Building ID
plt_NA_percentage <- ggplot(data = z_NA_percent, aes(x=reorder(Link_1, -Percentage_NA),
    y=Percentage_NA)) +
  geom_bar(stat = "identity", fill=z_NA_percent$Link_1) +
  labs(title="Percentage of NA vs Building ID", subtitle = "All variables - Full year",
     y="Percentage of NA [%]", x="Building ID") +
  geom_text(aes(label = Percentage_NA), vjust = -0.5)


Remove_percentage <- 10 #Percentage of remove the data according to NA values

z <- z %>%                      #Remove buildings with high NA percentages
   inner_join(z_NA_percent1) %>%
   filter(Percentage_NA < Remove_percentage) %>%
   select(-N_total_NA, -N_total, -Percentage_NA)

 z$Link_1 <- as.numeric(z$Link_1) #Converts to a string Link_1
 z$Link_2 <- as.numeric(z$Link_2) #Converts to a string Link_2

 z <- z %>%                       # Final dataset organized by date and Building ID
   arrange(Time_round) %>%
   arrange(Link_1) %>%
   select(-z_day, -z_month, -z_year, -Gap, -max_gap)


save(z, file="z_4.Rda")
```

Listing E.1. Script 4: Missing values

F Script 5: Imputation - Data Preprocessing

```r
### Script 5: Imputation (Preprocessing)

### 1. Setting the main directory -----------------------------------------------------------------
## Setting the data for the folder where is the file with the data.

# Aalborg University
setwd("C:/Users/LAB-PC/OneDrive - Aalborg Universitet/Master Thesis/7. DH
    Analysis/Scripts/S1_Preprocessing")

### 2. Installing important packages
    -----------------------------------------------------------------

if (!require('data.table') & require('magrittr')==FALSE & require('imputeTS')==FALSE &
    require('lubridate')==FALSE & require('dplyr')==FALSE & require('readr')==FALSE &
    require('ggplot2')==FALSE & require('tictoc')==FALSE
   & require('DataExplorer')==FALSE & require('ggpubr')==FALSE & require('lemon')==FALSE &
       require('gghighlight')==FALSE & require('xts')==FALSE
   & require('tidyr')==FALSE & require('tibble')==FALSE & require('purrr')==FALSE &
       require('cluster')==FALSE & require('factoextra')==FALSE){ #If NOT installed then install
       and check it to be used
  install.packages('data.table')
  install.packages('dplyr')
  install.packages('readr')
  install.packages('ggplot2')
  install.packages('tictoc')
  install.packages('imputeTS')
  install.packages('magrittr')
  install.packages('DataExplorer')
  install.packages('ggpubr')
  install.packages('lemon')
  install.packages('gghighlight')
  install.packages('xts')
  install.packages('tidyr')
  install.packages('tibble')
  install.packages('purrr')
  install.packages('cluster')
  install.packages('factoextra')

  library(data.table)
  library(dplyr)
  library(readr)
  library(ggplot2)
  library(tictoc)
  library(lubridate)
  library(imputeTS)
  library(magrittr)
  library(DataExplorer)
  library(ggpubr)
  library(lemon)
  library(gghighlight)
  library(viridis)
  library(hrbrthemes)
  library(xts)
  library(tidyr)
  library(tibble)
  library(purrr)
  library(cluster)
  library(factoextra)
} else {                        #If already installed then "check it" in the menu "Packages" to be
    used
  library(data.table)
  library(dplyr)
  library(readr)
  library(ggplot2)
  library(tictoc)
  library(lubridate)
  library(imputeTS)
  library(magrittr)
  library(DataExplorer)
  library(ggpubr)
  library(lemon)
  library(gghighlight)
  library(viridis)
  library(hrbrthemes)
  library(xts)
  library(tidyr)
  library(tibble)
  library(purrr)
  library(cluster)
  library(factoextra)
}

### 3. Load the *.Rda file from the main directory
    ------------------------------------------------------------

tic('Time to load the data.frame')

# Full year
load("z_4.Rda") #Load dataframe

toc()

z_group <- split(z,z$Link_1) #Split dataframe into small dataframes by their Link_1


### 4. Convert variables columns into time series and imputes values
    -----------------------------------

Imputation_function <- function(z){

  TS_Energy <- xts(z$Energy, z$Time_round)
  TS_Volume <- xts(z$Flow, z$Time_round)
  TS_Volume_Tsupply <- xts(z$`Flow*T_supply`, z$Time_round)
  TS_Volume_Treturn <- xts(z$`Flow*T_return`, z$Time_round)
  TS_Tsupply <- xts(z$`T_supply_(inst)`, z$Time_round)
  TS_Treturn <- xts(z$`T_return_(inst)`, z$Time_round)
  TS_Flow <- xts(z$`Volume_(inst)`, z$Time_round)

  TS_Energy <- na_interpolation(TS_Energy, option = "linear")
  TS_Volume <- na_interpolation(TS_Volume, option = "linear")
  TS_Volume_Tsupply <- na_interpolation(TS_Volume_Tsupply, option = "linear")
  TS_Volume_Treturn <- na_interpolation(TS_Volume_Treturn, option = "linear")
  TS_Tsupply <- na_ma(TS_Tsupply, k = 8, weighting = "exponential")
  TS_Treturn <- na_ma(TS_Treturn, k = 8, weighting = "exponential")
  TS_Flow <- na_ma(TS_Flow, k = 8, weighting = "exponential")

  z %>%
    mutate(
      Energy_TS = as.numeric(TS_Energy),
      Flow_TS = as.numeric(TS_Volume),
      Flow_Tsupply_TS = as.numeric(TS_Volume_Tsupply),
      Flow_Treturn_TS = as.numeric(TS_Volume_Treturn),
      T_supply_TS = as.numeric(TS_Tsupply),
      T_return_TS = as.numeric(TS_Treturn),
      Volume_TS = as.numeric(TS_Flow)
      )
}  #Function that converts data into Time Series type and applies the chosen imputation methods per
     building

z_group <- lapply(z_group, Imputation_function) #Applies function

z <- rbindlist(z_group) #Binds all the data together

z$Energy <- z$Energy_TS #Results will be the new data in each variable
z$Flow <- z$Flow_TS
z$`Flow*T_supply` <- z$Flow_Tsupply_TS
z$`Flow*T_return` <- z$Flow_Treturn_TS
z$`T_supply_(inst)` <- z$T_supply_TS
z$`T_return_(inst)` <- z$T_return_TS
z$`Volume_(inst)` <- z$Volume_TS

z <- z %>% #Removes the repeated columns, because they were replaced in the respective variables
  select(-Energy_TS, -Flow_TS, -Flow_Tsupply_TS, -Flow_Treturn_TS, -T_supply_TS, -T_return_TS,
       -Volume_TS)


### 5. Calculations -----------------------------------

const <- (985.2*4.183)/3600 #Constant value to be used to calculate Power

z <- z %>%
  arrange(Time_round) %>%
  arrange(Link_1) %>%
  mutate(T_supply_avg = `Flow*T_supply`/Flow, T_return_avg = `Flow*T_return`/Flow, Delta_T_inst =
       `T_supply_(inst)`- `T_return_(inst)`, Delta_T_avg = T_supply_avg - T_return_avg, Power_inst
       = const *`Volume_(inst)`*Delta_T_inst, Power_avg = const *`Volume_(inst)`*Delta_T_avg)


z_group <- split(z,z$Link_1)

Add_Energy_Volume_Hourly <- function(z){

  z %>%
    mutate(Energy_hourly = Energy - lag(Energy, default = first(z$Energy)), Volume_hourly = (Flow -
         lag(Flow, default = first(z$Flow)))*1000)
}  #Calculates from the cumulative variables (Energy and Volume) their hourly values

z_group <- lapply(z_group, Add_Energy_Volume_Hourly)

z <- rbindlist(z_group)

z <- z %>%                        # Data frame organized by date and Building ID
  arrange(Time_round) %>%
  arrange(Link_1)

save(z, file="z_5.Rda") #Saves dataframe
```

Listing F.1. Script 5: Imputation

Script 6: Building type segregation - Data Preprocessing G

```r
### Script to separate the types of buildings

### 1. Setting the main directory -----------------------------------------------------------------
## Setting the data for the folder where is the file with the data.

# Aalborg University
setwd("C:/Users/LAB-PC/OneDrive - Aalborg Universitet/Master Thesis/7. DH
     Analysis/Scripts/S1_Preprocessing")

### 2. Installing important packages
     -----------------------------------------------------------------

if (!require('data.table') & require('magrittr')==FALSE & require('imputeTS')==FALSE &
     require('lubridate')==FALSE & require('dplyr')==FALSE & require('readr')==FALSE &
     require('ggplot2')==FALSE & require('tictoc')==FALSE
    & require('DataExplorer')==FALSE & require('ggpubr')==FALSE & require('lemon')==FALSE &
        require('gghighlight')==FALSE & require('xts')==FALSE
    & require('tidyr')==FALSE & require('tibble')==FALSE & require('purrr')==FALSE &
        require('cluster')==FALSE & require('factoextra')==FALSE & require('ggExtra')==FALSE &
        require('dygraphs')==FALSE){ #If NOT installed then install and check it to be used
  install.packages('data.table')
  install.packages('dplyr')
  install.packages('readr')
  install.packages('ggplot2')
  install.packages('tictoc')
  install.packages('imputeTS')
  install.packages('magrittr')
  install.packages('DataExplorer')
  install.packages('ggpubr')
  install.packages('lemon')
  install.packages('gghighlight')
  install.packages('xts')
  install.packages('tidyr')
  install.packages('tibble')
  install.packages('purrr')
  install.packages('cluster')
  install.packages('ggExtra')
  install.packages("dygraphs")

  library(data.table)
  library(dplyr)
  library(readr)
  library(ggplot2)
  library(tictoc)
  library(lubridate)
  library(imputeTS)
  library(magrittr)
  library(DataExplorer)
  library(ggpubr)
  library(lemon)
  library(gghighlight)
  library(viridis)
  library(hrbrthemes)
  library(xts)
  library(tidyr)
  library(tibble)
  library(purrr)
  library(cluster)
  library(factoextra)
  library(ggExtra)
  library(dygraphs)
  library(plotly)
  library(leaflet)
  library(htmltools)
} else {                        #If already installed then "check it" in the menu "Packages" to be used
  library(data.table)
  library(dplyr)
  library(readr)
  library(ggplot2)
  library(tictoc)
  library(lubridate)
  library(imputeTS)
  library(magrittr)
  library(DataExplorer)
  library(ggpubr)
  library(lemon)
  library(gghighlight)
  library(viridis)
  library(hrbrthemes)
  library(xts)
  library(tidyr)
  library(tibble)
  library(purrr)
  library(cluster)
  library(factoextra)
  library(ggExtra)
  library(dygraphs)
  library(plotly)
  library(leaflet)
  library(htmltools)
}

### 3. Load the *.Rda file from the main directory ------------------------------------------------------------

load("z_5.Rda") #Load dataframe

### 4. Method ----------------------

z_group <- z %>%   # Create dataframe with principal characteristics of each building
  group_by(Link_1, Address) %>%
  summarise(Latitude = mean(Latitude), Longitude = mean(Longitude), Energy_max = max(Energy) - min(Energy), Volume_max = max(Flow) - min(Flow), Power_mean = mean(Power_avg))

## Limits for considering a commercial building

Energy_limit <- quantile(z_group$Energy_max, 0.75, type = 2) + 1.5 * IQR(z_group$Energy_max, type = 2) #[kWh]
Volume_limit <- quantile(z_group$Volume_max, 0.75, type = 2) + 1.5 * IQR(z_group$Volume_max, type = 2) #[m3]
Power_limit <- quantile(z_group$Power_mean, 0.75, type = 2) + 1.5 * IQR(z_group$Power_mean, type = 2) #[kW]
```

```

z_residential <- z_group %>%
  mutate(Residential = if(Energy_max > Energy_limit | Volume_max > Volume_limit | Power_mean >
       Power_limit){FALSE} else{TRUE}) %>%
  select(Link_1, Residential)

z_AllBuilding <- z %>%
  inner_join(z_residential )%>%  #All the buildings: Residential + Commercial
  arrange(Time_round) %>%
  arrange(Link_1)

save(z_AllBuilding, file="Data_AllBuildings.Rda")

z <- z_AllBuilding %>%          #Only residential buildings with adjustments in the columns
  filter(Residential == TRUE) %>%
  transmute(Time = Time_round, Link_1 = Link_1, Link_2 = Link_2, Address = Address, Latitude =
       Latitude, Longitude = Longitude, Lifetime = Lifetime,
            Energy_accum = Energy, Volume_accum = Flow, Volume_TSupply = ‘Flow*T_supply‘,
                 Volume_TReturn = ‘Flow*T_return‘,
            T_supply_inst = ‘T_supply_(inst)‘, T_supply_avg = T_supply_avg , T_return_inst =
                 ‘T_return_(inst)‘, T_return_avg = T_return_avg,
            Delta_T_inst = Delta_T_inst, Delta_T_avg = Delta_T_avg, Flow = ‘Volume_(inst)‘*1000,
            Power_inst = Power_inst, Power_avg = Power_avg, Energy = Energy_hourly, Volume =
                 Volume_hourly,
            Time_hour = Time_hour, Day = Day, Month = Month, Year = Year, Week_day = Week_day,
                 Day_type = Day_type, Week_number = Week_number, Season = Season)

save(z, file="z_6.Rda")

z_short <- z %>%      #Small dataframe creation to program algorithms easily
  filter(Link_1==103067| Link_1==103068| Link_1==103069| Link_1==103070| Link_1==103071|
       Link_1==103072| Link_1==103073| Link_1==103074|
           Link_1==103076 | Link_1==103078| Link_1==103079|
           Link_1==103080| Link_1==103081 | Link_1==103436 | Link_1==111835)

save(z_short, file="z_6_short.Rda")

z_AllBuilding <- z_AllBuilding %>% #Residential + Commercial buildings with adjustments in the
     columns
  transmute(Time = Time_round, Link_1 = Link_1, Link_2 = Link_2, Address = Address, Latitude =
       Latitude, Longitude = Longitude, Residential = Residential, Lifetime = Lifetime,
            Energy_accum = Energy, Volume_accum = Flow, Volume_TSupply = ‘Flow*T_supply‘,
                 Volume_TReturn = ‘Flow*T_return‘,
            T_supply_inst = ‘T_supply_(inst)‘, T_supply_avg = T_supply_avg , T_return_inst =
                 ‘T_return_(inst)‘, T_return_avg = T_return_avg,
            Delta_T_inst = Delta_T_inst, Delta_T_avg = Delta_T_avg, Flow = ‘Volume_(inst)‘*1000,
            Power_inst = Power_inst, Power_avg = Power_avg, Energy = Energy_hourly, Volume =
                 Volume_hourly,
            Time_hour = Time_hour, Day = Day, Month = Month, Year = Year, Week_day = Week_day,
                 Day_type = Day_type, Week_number = Week_number, Season = Season)

save(z_AllBuilding, file="z_AllBuildings.Rda")
```

Listing G.1. Script 6: Building type segregation

Script 7: Linear Regressions - Data Visualisation H

```r
### Title: Linear regressions in Energy & Outdoor Temperature


### 1. Setting the main directory ------------------------------------------------------------------
## Setting the data for the folder where is the file with the data.

# Aalborg University - Full year
setwd("C:/Users/LAB-PC/OneDrive - Aalborg Universitet/Master Thesis/7. DH
    Analysis/Scripts/S2_Data_Visualisation/Energy")

### 2. Installing important packages
    -----------------------------------------------------------------

if (!require('data.table') & require('magrittr')==FALSE & require('imputeTS')==FALSE &
    require('lubridate')==FALSE & require('dplyr')==FALSE & require('readr')==FALSE &
    require('ggplot2')==FALSE & require('tictoc')==FALSE
  & require('DataExplorer')==FALSE & require('ggpubr')==FALSE & require('lemon')==FALSE &
      require('gghighlight')==FALSE & require('xts')==FALSE
  & require('tidyr')==FALSE & require('tibble')==FALSE & require('purrr')==FALSE &
      require('cluster')==FALSE & require('factoextra')==FALSE & require('TSstudio')==FALSE &
      require('dygraphs')==FALSE & require('forecast')==FALSE & require('plotly')==FALSE){ #If
      NOT installed then install and check it to be used
  install.packages('data.table')
  install.packages('dplyr')
  install.packages('readr')
  install.packages('ggplot2')
  install.packages('tictoc')
  install.packages('imputeTS')
  install.packages('magrittr')
  install.packages('DataExplorer')
  install.packages('ggpubr')
  install.packages('lemon')
  install.packages('gghighlight')
  install.packages('xts')
  install.packages('tidyr')
  install.packages('tibble')
  install.packages('purrr')
  install.packages('cluster')
  install.packages('factoextra')
  install.packages("NbClust")
  install.packages("clustree")
  install.packages("clValid")
  install.packages("gridExtra")
  install.packages("amap")
  install.packages("fpc")
  install.packages("TSstudio")
  install.packages("dygraphs")
  install.packages("forecast")
  install.packages("plotly")

  library(data.table)
  library(dplyr)
  library(readr)
  library(ggplot2)
  library(tictoc)
  library(lubridate)
  library(imputeTS)
  library(magrittr)
  library(DataExplorer)
  library(ggpubr)
  library(lemon)
  library(gghighlight)
  library(viridis)
  library(hrbrthemes)
  library(xts)
  library(tidyr)
  library(tibble)
  library(purrr)
  library(cluster)
  library(factoextra)
  library(NbClust)
  library(clustree)
  library(clValid)
  library(gridExtra)
  library(amap)
  library(fpc)
  library(TSstudio)
  library(dygraphs)
  library(forecast)
  library(plotly)
} else {                          #If already installed then "check it" in the menu "Packages" to be
   used
  library(data.table)
  library(dplyr)
  library(readr)
  library(ggplot2)
  library(tictoc)
  library(lubridate)
  library(imputeTS)
  library(magrittr)
  library(DataExplorer)
  library(ggpubr)
  library(lemon)
  library(gghighlight)
  library(viridis)
  library(hrbrthemes)
  library(xts)
  library(tidyr)
  library(tibble)
  library(purrr)
  library(cluster)
  library(factoextra)
  library(NbClust)
  library(clustree)
  library(clValid)
  library(gridExtra)
  library(amap)
  library(fpc)
  library(TSstudio)
  library(dygraphs)
  library(forecast)
  library(plotly)
}

```

```r

### 3. Load the *.Rda file from the main directory
    ------------------------------------------------------------

load("z_6.Rda") #Load the data

z_Heating <- filter(z, Season != "Summer") #Filter dataset according to season
z_No_Heating <- filter(z, Season == "Summer")

z_Inf <- filter(z, Season != "Summer" & Solar_Rad == 0 & Wind_Speed > 1) #Filter by weather
    conditions
z_Rad <- filter(z, Season != "Summer" & Solar_Rad != 0 & Wind_Speed <= 1)

z_Temp <- filter(z, Season != "Summer" & Solar_Rad == 0 & Wind_Speed <= 1)

z_link_Heating <- split(z_Heating, z_Heating$Link_1) #Split dataset by building's ID
z_link_No_Heating <- split(z_No_Heating, z_No_Heating$Link_1)
z_link_Inf <- split(z_Inf, z_Inf$Link_1)
z_link_Rad <- split(z_Rad, z_Rad$Link_1)

z_link_Temp <- split(z_Temp, z_Temp$Link_1)

Conv_Calc_SMA <- function(df){ #Function: Calculate moving average of time-series

  k_roll <- 33 #Range of moving average

  Energy_TS <- xts(x=df$Energy, order.by=df$Time) #Convert to time-series
  Energy_TS <- xts_to_ts(Energy_TS, frequency = 1)

  OutTemp_TS <- xts(x=df$Out_Temp, order.by=df$Time)
  OutTemp_TS <- xts_to_ts(OutTemp_TS, frequency = 1)

  SMA_Energy <- ma(Energy_TS, order=k_roll, centre=FALSE) #Apply moving average
  SMA_OutTemp <- ma(OutTemp_TS, order=k_roll, centre=FALSE)

  df <- transmute(df, Link_1 = Link_1, Out_Temp = as.numeric(SMA_OutTemp), Energy =
      as.numeric(SMA_Energy))
}

z_link_Heating <- lapply(z_link_Heating, Conv_Calc_SMA) #Apply function of moving average
z_link_No_Heating <- lapply(z_link_No_Heating, Conv_Calc_SMA)
z_link_Inf <- lapply(z_link_Inf, Conv_Calc_SMA)
z_link_Rad <- lapply(z_link_Rad, Conv_Calc_SMA)
z_Temp <- lapply(z_link_Temp, Conv_Calc_SMA)

z_Heating <- rbindlist(z_link_Heating) %>% #Bind all the buildings in one dataframe
  mutate(Type = "Heating")
z_No_Heating <- rbindlist(z_link_No_Heating) %>%
  mutate(Type = "No Heating")
z_Inf <- rbindlist(z_link_Inf) %>%
  mutate(Type = "Infiltration")
z_Rad <- rbindlist(z_link_Rad) %>%
  mutate(Type = "Radiation")
z_Temp <- rbindlist(z_link_Temp) %>%
  mutate(Type = "Temperature")

df <- z_Heating %>%
  bind_rows(z_No_Heating, z_Inf, z_Rad, z_Temp) %>%
  transmute(Link_1, Out_Temp, Energy, Type)
df <- split(df, df$Link_1)

calculate_Uvalue <- function(df){ #Create linear regressions and its values

  df_Heating <- filter(df, Type == "Heating")
  linear_Heating <- lm(Energy ~ Out_Temp, data = df_Heating)
  m1 <- coef(linear_Heating)[2]
  b1 <- coef(linear_Heating)[1]

  df_NoHeating <- filter(df, Type == "No Heating")
  linear_NoHeating <- lm(Energy ~ Out_Temp, data = df_NoHeating)
  m2 <- coef(linear_NoHeating)[2]
  b2 <- coef(linear_NoHeating)[1]

  Slope_Heating <- m1
  Slope_No_Heating <- m2
  T_NoInfluence <- (b2-b1)/(m1-m2)
  E_NoHeating <- m1*T_NoInfluence + b1

  df_Infiltration <- filter(df, Type == "Infiltration")
  linear_Infiltration <- lm(Energy ~ Out_Temp, data = df_Infiltration)
  m_Infiltration <- coef(linear_Infiltration)[2]
  b_Infiltration <- coef(linear_Infiltration)[1]

  df_Radiation <- filter(df, Type == "Radiation")
  linear_Radiation <- lm(Energy ~ Out_Temp, data = df_Radiation)
  m_Radiation <- coef(linear_Radiation)[2]
  b_Radiation <- coef(linear_Radiation)[1]

  df_Temperature <- filter(df, Type == "Temperature")
  linear_Temperature <- lm(Energy ~ Out_Temp, data = df_Temperature)
  m_Temperature <- coef(linear_Temperature)[2]
  b_Temperature <- coef(linear_Temperature)[1]

  df <- df %>%
    mutate(Slope_Heating = Slope_Heating, Interc_Heating = b1, Slope_No.Heating = Slope_No_Heating,
         Interc_No.Heating = b2, m_Infiltration = m_Infiltration, b_Infiltration = b_Infiltration,
         m_Radiation = m_Radiation, b_Radiation = b_Radiation, m_Temperature = m_Temperature,
         b_Temperature = b_Temperature, T_No.Influence = T_NoInfluence, E_No.Heating = E_NoHeating)
  }

df <- lapply(df, calculate_Uvalue)
df <- rbindlist(df) %>%
  select(Link_1, Slope_Heating, Interc_Heating, Slope_No.Heating, Interc_No.Heating,
       m_Infiltration, b_Infiltration, m_Radiation, b_Radiation, m_Temperature, b_Temperature,
       T_No.Influence, E_No.Heating) %>%
  group_by(Link_1) %>%
  summarise(Slope_Heating = mean(Slope_Heating), Interc_Heating = mean(Interc_Heating),
       Slope_No.Heating = mean(Slope_No.Heating), Interc_No.Heating = mean(Interc_No.Heating),
            m_Infiltration = mean(m_Infiltration), b_Infiltration = mean(b_Infiltration),
            m_Radiation = mean(m_Radiation), b_Radiation = mean(b_Radiation),
            m_Temperature = mean(m_Temperature), b_Temperature = mean(b_Temperature),
            T_No.Influence = mean(T_No.Influence), E_No.Heating = mean(E_No.Heating))


z <- z %>%
  left_join(df)

save(z, file = "z_7.Rda") #Save dataframe
```

Listing H.1. Script 7: Linear regressions

Script 8: Prepare dataset for clustering - Clustering Analysis

I

```r
### Script to prepare dataframe for clustering

### 1. Setting the main directory -----------------------------------------------------------------
## Setting the data for the folder where is the file with the data.

# Aalborg University - Full year
setwd("C:/Users/LAB-PC/OneDrive - Aalborg Universitet/Master Thesis/7. DH
     Analysis/Scripts/S3_Clustering/Kmeans")

### 2. Installing important packages
     -----------------------------------------------------------------

if (!require('data.table') & require('magrittr')==FALSE & require('imputeTS')==FALSE &
     require('lubridate')==FALSE & require('dplyr')==FALSE & require('readr')==FALSE &
     require('ggplot2')==FALSE & require('tictoc')==FALSE
    & require('DataExplorer')==FALSE & require('ggpubr')==FALSE & require('lemon')==FALSE &
        require('gghighlight')==FALSE & require('xts')==FALSE
    & require('tidyr')==FALSE & require('tibble')==FALSE & require('purrr')==FALSE &
        require('cluster')==FALSE & require('factoextra')==FALSE){ #If NOT installed then install
        and check it to be used
  install.packages('data.table')
  install.packages('dplyr')
  install.packages('readr')
  install.packages('ggplot2')
  install.packages('tictoc')
  install.packages('imputeTS')
  install.packages('magrittr')
  install.packages('DataExplorer')
  install.packages('ggpubr')
  install.packages('lemon')
  install.packages('gghighlight')
  install.packages('xts')
  install.packages('tidyr')
  install.packages('tibble')
  install.packages('purrr')
  install.packages('cluster')
  install.packages('factoextra')

  library(data.table)
  library(dplyr)
  library(readr)
  library(ggplot2)
  library(tictoc)
  library(lubridate)
  library(imputeTS)
  library(magrittr)
  library(DataExplorer)
  library(ggpubr)
  library(lemon)
  library(gghighlight)
  library(viridis)
  library(hrbrthemes)
  library(xts)
  library(tidyr)
  library(tibble)
  library(purrr)
  library(cluster)
  library(factoextra)
} else {                        #If already installed then "check it" in the menu "Packages" to be used
  library(data.table)
  library(dplyr)
  library(readr)
  library(ggplot2)
  library(tictoc)
  library(lubridate)
  library(imputeTS)
  library(magrittr)
  library(DataExplorer)
  library(ggpubr)
  library(lemon)
  library(gghighlight)
  library(viridis)
  library(hrbrthemes)
  library(xts)
  library(tidyr)
  library(tibble)
  library(purrr)
  library(cluster)
  library(factoextra)
}


### 3. Load the *.Rda file from the main directory ------------------------------------------------------------

load("z_7.Rda") #Load dataframe

z_group <- z %>%  #Group buildings and calculate mean DT per building
  group_by(Link_1) %>%
  summarise(DT_group = mean(Delta_T_avg))

z <- z %>%    #Join mean DT values with initial dataframe
  left_join(z_group)

### Temperature difference level ------------------------

#In this case it was performed the preparation for dataset:
#Variable: Flow
#Season: Winter
#Day type: Week working day (Always)
#Mean Delta T group: 50 - 60C

#For other cases, just change the conditions and names of the files below!

z_Flow <- z %>% #Filter and create daily profile
  filter(Season == "Winter", Day_type == "Weekday", DT_group > 50 & DT_group <= 60) %>%
  group_by(Link_1, Time_hour) %>%
  summarise(Flow = mean(Flow))

z_Flow <- split(z_Flow, z_Flow$Link_1)

Time_date <- function(z){ #Hours are converted to date variable type
```

```

  z$Time_hour <- as.POSIXct("2019-07-16 00:00") + as.difftime(z$Time_hour, units="hours")

  return(z)

}

z_Flow <- lapply(z_Flow, Time_date)

z_Flow <- rbindlist(z_Flow) #Dataframe with daily flow profiles of each building


### Variables per season ------------------

z_Flow <- z_Flow %>% #Standarize flow variable
  mutate(Flow_Scale = (Flow - mean(Flow))/sd(Flow)) %>%
  select(-Flow)

z_group <- split(z_Flow, z_Flow$Link_1)

Invert_table <- function(z){ #Transpose table function

  z %>%
    group_by(Link_1, Time_hour) %>%
    mutate(rn = row_number()) %>%
    pivot_wider(names_from = Time_hour, values_from = Flow_Scale) %>%
    filter(rn != 2) %>%
    select(-rn)
}

z_group <- lapply(z_group, Invert_table)

z_Flow_final_Winter_5060 <- rbindlist(z_group) ##Dataframe with daily flow profiles of each
     building (Standarized and tranposed)


z_Flow_final_Winter_5060 <- subset(z_Flow_final_Winter_5060,
     !duplicated(z_Flow_final_Winter_5060[,Link_1]))
save(z_Flow_final_Winter_5060, file="z_Flow_Winter_5060.Rda") #Save dataframe as R file
```

Listing I.1. Script 8: Prepare dataset for clustering

Script 9: Elbow method - Clustering Analysis J

```r
### Script to find optimal number of clusters

### 1. Setting the main directory -----------------------------------------------------------------
## Setting the data for the folder where is the file with the data.

# Aalborg University - Full year
setwd("C:/Users/LAB-PC/OneDrive - Aalborg Universitet/Master Thesis/7. DH
    Analysis/Scripts/S3_Clustering/Kmeans")

### 2. Installing important packages
    -----------------------------------------------------------------

if (!require('data.table') & require('magrittr')==FALSE & require('imputeTS')==FALSE &
    require('lubridate')==FALSE & require('dplyr')==FALSE & require('readr')==FALSE &
    require('ggplot2')==FALSE & require('tictoc')==FALSE
   & require('DataExplorer')==FALSE & require('ggpubr')==FALSE & require('lemon')==FALSE &
       require('gghighlight')==FALSE & require('xts')==FALSE
   & require('tidyr')==FALSE & require('tibble')==FALSE & require('purrr')==FALSE &
       require('cluster')==FALSE & require('factoextra')==FALSE){ #If NOT installed then install
       and check it to be used
  install.packages('data.table')
  install.packages('dplyr')
  install.packages('readr')
  install.packages('ggplot2')
  install.packages('tictoc')
  install.packages('imputeTS')
  install.packages('magrittr')
  install.packages('DataExplorer')
  install.packages('ggpubr')
  install.packages('lemon')
  install.packages('gghighlight')
  install.packages('xts')
  install.packages('tidyr')
  install.packages('tibble')
  install.packages('purrr')
  install.packages('cluster')
  install.packages('factoextra')

  library(data.table)
  library(dplyr)
  library(readr)
  library(ggplot2)
  library(tictoc)
  library(lubridate)
  library(imputeTS)
  library(magrittr)
  library(DataExplorer)
  library(ggpubr)
  library(lemon)
  library(gghighlight)
  library(viridis)
  library(hrbrthemes)
  library(xts)
  library(tidyr)
  library(tibble)
  library(purrr)
  library(cluster)
  library(factoextra)
} else {                        #If already installed then "check it" in the menu "Packages" to be
    used
  library(data.table)
  library(dplyr)
  library(readr)
  library(ggplot2)
  library(tictoc)
  library(lubridate)
  library(imputeTS)
  library(magrittr)
  library(DataExplorer)
  library(ggpubr)
  library(lemon)
  library(gghighlight)
  library(viridis)
  library(hrbrthemes)
  library(xts)
  library(tidyr)
  library(tibble)
  library(purrr)
  library(cluster)
  library(factoextra)
}


### 3. Load the *.Rda file from the main directory
    ------------------------------------------------------------

load("z_Flow_Summer_1030.Rda") #Load prepared for cluster files
load("z_Flow_Summer_3040.Rda")
load("z_Flow_Summer_4050.Rda")
load("z_Flow_Summer_5060.Rda")

x_Flow_Summer_1030 <- z_Flow_final_Summer_1030 %>% select(-Link_1)
x_Flow_Summer_3040 <- z_Flow_final_Summer_3040 %>% select(-Link_1)
x_Flow_Summer_4050 <- z_Flow_final_Summer_4050 %>% select(-Link_1)
x_Flow_Summer_5060 <- z_Flow_final_Summer_5060 %>% select(-Link_1)

### 4. Elbow method -------------------------

#CASEs:
#Variable: Flow
#Season: Summer
#Day type: Week working day (Always)
#Mean Delta T group: All


set.seed(123)

# function to compute total within-cluster sum of square
wss.1030 <- function(k) {
  kmeans(x_Flow_Summer_1030, k, nstart = 10, iter.max = 50)$tot.withinss
}

wss.3040 <- function(k) {
  kmeans(x_Flow_Summer_3040, k, nstart = 10, iter.max = 50)$tot.withinss
}

wss.4050 <- function(k) {
```

```r
  kmeans(x_Flow_Summer_4050, k, nstart = 10, iter.max = 50)$tot.withinss
}

wss.5060 <- function(k) {
  kmeans(x_Flow_Summer_5060, k, nstart = 10, iter.max = 50)$tot.withinss
}


# Compute and plot wss for k = 1 to k = 10 (K is number of clusters)
k.values <- 1:10

# extract wss for 2-10 clusters
wss_1030 <- map_dbl(k.values, wss.1030)
wss_3040 <- map_dbl(k.values, wss.3040)
wss_4050 <- map_dbl(k.values, wss.4050)
wss_5060 <- map_dbl(k.values, wss.5060)


### Plot

plot(k.values, wss_1030,
     type="b", pch = 19, lty=1, col = "black",frame = FALSE, main= "Elbow Method: Flow - Summer",
     xlab="Number of clusters K",
     ylab="Total within-clusters sum of squares",
     xlim = c(0,11), ylim = c(0,15000))

# Add second curve to the same plot by calling points() and lines()
# Use symbol '*' for points.
points(k.values, wss_3040, type = "b", pch = 19, col="red")
lines(k.values, wss_3040, col="red",lty=1)

# Add Third curve to the same plot by calling points() and lines()
# Use symbol '+' for points.
points(k.values, wss_4050, type = "b", pch = 19, col="green")
lines(k.values, wss_4050, col="green", lty=1)

# Add Fourth curve to the same plot by calling points() and lines()
# Use symbol '-' for points.
points(k.values, wss_5060, type = "b", pch = 19, col="blue")
lines(k.values, wss_5060, col="blue", lty=1)

# Legend

legend(8,10000,legend=c("10 - 30C","30 - 40C","40 - 50C", "50 - 60C"),
     col=c("black","red","green","blue"),
       pch=c(19,19,19,19),lty=c(1,1,1,1), ncol=1)
```

Listing J.1. Script 9: Elbow method

Script 10: Average silhouette method - Clustering Analysis K

```
### Script to find optimal number of clusters

### 1. Setting the main directory -----------------------------------------------------------------
## Setting the data for the folder where is the file with the data.

# Aalborg University - Full year
setwd("C:/Users/LAB-PC/OneDrive - Aalborg Universitet/Master Thesis/7. DH
     Analysis/Scripts/S3_Clustering/Kmeans")

### 2. Installing important packages
     -----------------------------------------------------------------

if (!require('data.table') & require('magrittr')==FALSE & require('imputeTS')==FALSE &
     require('lubridate')==FALSE & require('dplyr')==FALSE & require('readr')==FALSE &
     require('ggplot2')==FALSE & require('tictoc')==FALSE
    & require('DataExplorer')==FALSE & require('ggpubr')==FALSE & require('lemon')==FALSE &
        require('gghighlight')==FALSE & require('xts')==FALSE
    & require('tidyr')==FALSE & require('tibble')==FALSE & require('purrr')==FALSE &
        require('cluster')==FALSE & require('factoextra')==FALSE){ #If NOT installed then install
        and check it to be used
  install.packages('data.table')
  install.packages('dplyr')
  install.packages('readr')
  install.packages('ggplot2')
  install.packages('tictoc')
  install.packages('imputeTS')
  install.packages('magrittr')
  install.packages('DataExplorer')
  install.packages('ggpubr')
  install.packages('lemon')
  install.packages('gghighlight')
  install.packages('xts')
  install.packages('tidyr')
  install.packages('tibble')
  install.packages('purrr')
  install.packages('cluster')
  install.packages('factoextra')

  library(data.table)
  library(dplyr)
  library(readr)
  library(ggplot2)
  library(tictoc)
  library(lubridate)
  library(imputeTS)
  library(magrittr)
  library(DataExplorer)
  library(ggpubr)
  library(lemon)
  library(gghighlight)
  library(viridis)
  library(hrbrthemes)
  library(xts)
  library(tidyr)
  library(tibble)
  library(purrr)
  library(cluster)
  library(factoextra)
} else {                        #If already installed then "check it" in the menu "Packages" to be
    used
  library(data.table)
  library(dplyr)
  library(readr)
  library(ggplot2)
  library(tictoc)
  library(lubridate)
  library(imputeTS)
  library(magrittr)
  library(DataExplorer)
  library(ggpubr)
  library(lemon)
  library(gghighlight)
  library(viridis)
  library(hrbrthemes)
  library(xts)
  library(tidyr)
  library(tibble)
  library(purrr)
  library(cluster)
  library(factoextra)
}


### 3. Load the *.Rda file from the main directory
    ------------------------------------------------------------

load("z_Flow_Summer_1030.Rda") #Load prepared for cluster files
load("z_Flow_Summer_3040.Rda")
load("z_Flow_Summer_4050.Rda")
load("z_Flow_Summer_5060.Rda")

x_Flow_Summer_1030 <- z_Flow_final_Summer_1030 %>% select(-Link_1)
x_Flow_Summer_3040 <- z_Flow_final_Summer_3040 %>% select(-Link_1)
x_Flow_Summer_4050 <- z_Flow_final_Summer_4050 %>% select(-Link_1)
x_Flow_Summer_5060 <- z_Flow_final_Summer_5060 %>% select(-Link_1)

### 4. Silhouette method -------------------------

#CASE:
#Variable: Flow
#Season: Summer
#Day type: Week working day (Always)
#Mean Delta T group: All

set.seed(123)

# function to compute total within-cluster sum of square
silh.1030 <- function(k) {
  km.res <- kmeans(x_Flow_Summer_1030, centers = k, nstart = 25, iter.max = 50)
  ss <- silhouette(km.res$cluster, dist(x_Flow_Summer_1030))
  mean(ss[, 3])
}

silh.3040 <- function(k) {
  km.res <- kmeans(x_Flow_Summer_3040, centers = k, nstart = 25, iter.max = 50)
  ss <- silhouette(km.res$cluster, dist(x_Flow_Summer_3040))
  mean(ss[, 3])
}

silh.4050 <- function(k) {
  km.res <- kmeans(x_Flow_Summer_4050, centers = k, nstart = 25, iter.max = 50)
  ss <- silhouette(km.res$cluster, dist(x_Flow_Summer_4050))
  mean(ss[, 3])
}

silh.5060 <- function(k) {
  km.res <- kmeans(x_Flow_Summer_5060, centers = k, nstart = 25, iter.max = 50)
  ss <- silhouette(km.res$cluster, dist(x_Flow_Summer_5060))
  mean(ss[, 3])
}


# Compute and plot wss for k = 2 to k = 10 (K is number of clusters)
k.values <- 2:10

# extract wss for 2-10 clusters
avg_sil_1030 <- map_dbl(k.values, silh.1030)
avg_sil_3040 <- map_dbl(k.values, silh.3040)
avg_sil_4050 <- map_dbl(k.values, silh.4050)
avg_sil_5060 <- map_dbl(k.values, silh.5060)

### Plot

plot(k.values, avg_sil_1030,
     type="b", pch = 19, lty=1, col = "black",frame = FALSE, main="Average Silhouette Method: Flow
         - Summer",
     xlab="Number of clusters K",
     ylab="Average Silhouette Coeff.",
     xlim = c(2,11), ylim = c(0,0.8))

# Add second curve to the same plot by calling points() and lines()
# Use symbol '*' for points.
points(k.values, avg_sil_3040, type = "b", pch = 19, col="red")
lines(k.values, avg_sil_3040, col="red",lty=1)

# Add Third curve to the same plot by calling points() and lines()
# Use symbol '+' for points.
points(k.values, avg_sil_4050, type = "b", pch = 19, col="green")
lines(k.values, avg_sil_4050, col="green", lty=1)

# Add Fourth curve to the same plot by calling points() and lines()
# Use symbol '-' for points.
points(k.values, avg_sil_5060, type = "b", pch = 19, col="blue")
lines(k.values, avg_sil_5060, col="blue", lty=1)

# Legend

legend(8,0.75,legend=c("10 - 30C","30 - 40C","40 - 50C", "50 - 60 C"),
     col=c("black","red","green","blue"),
       pch=c(19,19,19,19),lty=c(1,1,1,1), ncol=1)
```

Listing K.1. Script 10: Average silhouette method

Script 11: Kmeans method - Clustering Analysis L

```r
### Script to clustering: K-means

### 1. Setting the main directory -----------------------------------------------------------------
## Setting the data for the folder where is the file with the data.

# Aalborg University - Full year
setwd("C:/Users/LAB-PC/OneDrive - Aalborg Universitet/Master Thesis/7. DH
    Analysis/Scripts/S3_Clustering/Kmeans")

# Home
#setwd("C:/Users/Utilizador/Desktop/Indoor Environmental and Energy Engineering/Master Thesis/NASB
    Conference")

### 2. Installing important packages
    -----------------------------------------------------------------

if (!require('data.table') & require('magrittr')==FALSE & require('imputeTS')==FALSE &
    require('lubridate')==FALSE & require('dplyr')==FALSE & require('readr')==FALSE &
    require('ggplot2')==FALSE & require('tictoc')==FALSE
   & require('DataExplorer')==FALSE & require('ggpubr')==FALSE & require('lemon')==FALSE &
       require('gghighlight')==FALSE & require('xts')==FALSE
   & require('tidyr')==FALSE & require('tibble')==FALSE & require('purrr')==FALSE &
       require('cluster')==FALSE & require('factoextra')==FALSE & require('factoextra')==FALSE){
       #If NOT installed then install and check it to be used
  install.packages('data.table')
  install.packages('dplyr')
  install.packages('readr')
  install.packages('ggplot2')
  install.packages('tictoc')
  install.packages('imputeTS')
  install.packages('magrittr')
  install.packages('DataExplorer')
  install.packages('ggpubr')
  install.packages('lemon')
  install.packages('gghighlight')
  install.packages('xts')
  install.packages('tidyr')
  install.packages('tibble')
  install.packages('purrr')
  install.packages('cluster')
  install.packages('factoextra')
  install.packages("NbClust")

  library(data.table)
  library(dplyr)
  library(readr)
  library(ggplot2)
  library(tictoc)
  library(lubridate)
  library(imputeTS)
  library(magrittr)
  library(DataExplorer)
  library(ggpubr)
  library(lemon)
  library(gghighlight)
  library(viridis)
  library(hrbrthemes)
  library(xts)
  library(tidyr)
  library(tibble)
  library(purrr)
  library(cluster)
  library(factoextra)
  library(NbClust)
} else {                        #If already installed then "check it" in the menu "Packages" to be
     used
  library(data.table)
  library(dplyr)
  library(readr)
  library(ggplot2)
  library(tictoc)
  library(lubridate)
  library(imputeTS)
  library(magrittr)
  library(DataExplorer)
  library(ggpubr)
  library(lemon)
  library(gghighlight)
  library(viridis)
  library(hrbrthemes)
  library(xts)
  library(tidyr)
  library(tibble)
  library(purrr)
  library(cluster)
  library(factoextra)
  library(NbClust)

}

### 3. Load the *.Rda file from the main directory
     ------------------------------------------------------------

load("z_Flow_Winter_5060.Rda") #Load dataframes
z_Flow_final_Winter <- z_Flow_final_Winter_5060
load("z_7.Rda")

#CASE:
#Variable: Flow
#Season: Winter
#Day type: Week working day (Always)
#Mean Delta T group: 50 - 60C

z_group <- z %>%
  group_by(Link_1) %>%
  summarise(DT_group = mean(Delta_T_avg))

z <- z %>%
  left_join(z_group)

z_Flow <- z %>%
  filter(Season == "Winter", Day_type == "Weekday", DT_group > 50 & DT_group <= 60) %>%
  group_by(Link_1, Time_hour) %>%
  summarise(Flow = mean(Flow))

Mean_x <- mean(z_Flow$Flow)
Sd_x <- sd(z_Flow$Flow)
```

```r

### 4. Kmeans -------------------------------------------


x_Winter <- z_Flow_final_Winter %>% select(-Link_1)

set.seed(123)

kmeans_Winter <- kmeans(x_Winter, centers = 3, iter.max = 50) #Apply Kmeans
avg_centers_wide <- rownames_to_column(as.data.frame(kmeans_Winter$centers), "cluster") #Dataframe
     of the average time series of the clusters

Cluster_Winter <- z_Flow_final_Winter %>% #Dataframe with IF number of the cluster per building ID
  transmute(Link_1 = Link_1, cluster = kmeans_Winter$cluster)

x_long <- z_Flow_final_Winter %>% #Converts wide initial data frame to long version
  pivot_longer(cols = -Link_1, names_to = "Time", values_to = "Flow") %>%
  inner_join(Cluster_Winter)


x_long$Flow <- x_long$Flow*Sd_x + Mean_x #Non-standard value
x_long$Link_1 <- as.numeric(x_long$Link_1)
x_long$Time <- as.POSIXct(x_long$Time)



centers_long <- avg_centers_wide %>% #Converts wide average per cluster data frame to long version
  pivot_longer(cols = -cluster, names_to = "Time", values_to = "Flow")
centers_long$Flow <- centers_long$Flow*Sd_x + Mean_x #Non-standard value
centers_long$cluster <- as.numeric(centers_long$cluster)
centers_long$Time <- as.POSIXct(centers_long$Time)


Number_per_cluster <- Cluster_Winter %>% #Calculate percentage of buildings per cluster
  group_by(cluster) %>%
  summarise(N_buildings_per_cluster = n()) %>%
  mutate(Percentate_of_quantity = round(N_buildings_per_cluster/nrow(z_Flow_final_Winter)*100,
       digits = 0))


plt_quant_cluster <- ggplot(data = Number_per_cluster, aes(x=cluster, y=Percentate_of_quantity)) +
     #plot of quantity of buildings per cluster
  geom_col(fill = Number_per_cluster$cluster) +
  labs(title="Percentage of buildings per cluster vs Cluster ID", y="Percentage of buildings per
       cluster [%]", x="Cluster ID") +
  geom_text(aes(label = Percentate_of_quantity), vjust = -0.5)


Percentage_cluster <- paste("Cluster ", Number_per_cluster$cluster, ": ",
     Number_per_cluster$Percentate_of_quantity, "%")
names(Percentage_cluster) <- as.character(Number_per_cluster$cluster) #Title of cluster in the plot

### Plot

plt_kmeans_Winter <- ggplot() + #plot of kmeans per cluster
  geom_line(data = x_long, aes(y = Flow, x = Time, group = Link_1), colour = "grey") +
  facet_wrap(~cluster, nrow = 1, labeller = labeller(cluster = Percentage_cluster)) +
  geom_line(data = centers_long, aes(y = Flow, x = Time, group = cluster), col = "red") +
  labs(title = "K-means Clustering: Flow (50 - 60C)", subtitle = "Season: Winter", x = "Time
       [hours]", y = "Flow [liters/h]") +
  scale_x_datetime(date_labels = "%H:%M", date_breaks = "6 hour")
```

Listing L.1. Script 11: Kmeans

Script Extra: Imputation Benchmark M

```r
### Script Extra: Imputation Benchmark

### 1. Setting the main directory ------------------------------------------------------------------
## Setting the data for the folder where is the file with the data.

# Aalborg University - Full year
setwd("C:/Users/LAB-PC/OneDrive - Aalborg Universitet/Master Thesis/7. Klarup
    Analysis/Scripts/S1_Preprocessing")

### 2. Installing important packages
    ------------------------------------------------------------------

if (!require('data.table') & require('magrittr')==FALSE & require('imputeTS')==FALSE &
    require('lubridate')==FALSE & require('dplyr')==FALSE & require('readr')==FALSE &
    require('ggplot2')==FALSE & require('tictoc')==FALSE
  & require('DataExplorer')==FALSE & require('ggpubr')==FALSE & require('lemon')==FALSE &
      require('gghighlight')==FALSE & require('xts')==FALSE
  & require('tidyr')==FALSE & require('tibble')==FALSE & require('purrr')==FALSE &
      require('cluster')==FALSE & require('factoextra')==FALSE & require('Metrics')==FALSE){ #If
      NOT installed then install and check it to be used
  install.packages('data.table')
  install.packages('dplyr')
  install.packages('readr')
  install.packages('ggplot2')
  install.packages('tictoc')
  install.packages('imputeTS')
  install.packages('magrittr')
  install.packages('DataExplorer')
  install.packages('ggpubr')
  install.packages('lemon')
  install.packages('gghighlight')
  install.packages('xts')
  install.packages('tidyr')
  install.packages('tibble')
  install.packages('purrr')
  install.packages('cluster')
  install.packages('factoextra')
  install.packages("Metrics")

  library(data.table)
  library(dplyr)
  library(readr)
  library(ggplot2)
  library(tictoc)
  library(lubridate)
  library(imputeTS)
  library(magrittr)
  library(DataExplorer)
  library(ggpubr)
  library(lemon)
  library(gghighlight)
  library(viridis)
  library(hrbrthemes)
  library(xts)
  library(tidyr)
  library(tibble)
  library(purrr)
  library(cluster)
  library(factoextra)
  library(Metrics)
  } else {                        #If already installed then "check it" in the menu "Packages" to
      be used
  library(data.table)
  library(dplyr)
  library(readr)
  library(ggplot2)
  library(tictoc)
  library(lubridate)
  library(imputeTS)
  library(magrittr)
  library(DataExplorer)
  library(ggpubr)
  library(lemon)
  library(gghighlight)
  library(viridis)
  library(hrbrthemes)
  library(xts)
  library(tidyr)
  library(tibble)
  library(purrr)
  library(cluster)
  library(factoextra)
  library(Metrics)
}


### Load the *.Rda file from the main directory
     ------------------------------------------------------------

# Full year
load("z_Imputation.Rda") #Load dataframe
z <- z_impute

## List of IDs that can be selected:
# 130863
# 130877
# 134828
# 138560
# 175576
# 176391
# 300119
# 301916
# 306643
# 312376


z <- z %>%  #Adjusts dataframe and filters per building Link_1 - The buildings must be selected
     individually
  select(Link_1, Time_round, Energy, Flow, `Flow*T_supply`, `Flow*T_return`, `T_supply_(inst)`,
      `T_return_(inst)`, `Volume_(inst)`, NA_value = NA_value) %>%
  filter(Link_1 == 312376 & NA_value == FALSE) %>%
  transmute(Link_1 = Link_1, Time =Time_round, Energy = Energy, Volume_accum = Flow, Volume_TS =
      `Flow*T_supply`, Volume_TR = `Flow*T_return`, T_supply = `T_supply_(inst)`, T_return =
      `T_return_(inst)`, Flow = `Volume_(inst)`)

zna <- z # Dataframe to have the empty data

```

```r
### Values extraction

N <- 500                                    # the number of random values to replace
inds <- round (runif(N, 1, length(zna$Energy))) # draw random values from [1, length(vector)]
zna$Energy[inds] <- NA
zna$Energy[inds +1] <- NA
zna$Volume_accum[inds] <- NA
zna$Volume_TS[inds] <- NA
zna$Volume_TR[inds] <- NA
zna$T_supply[inds] <- NA
zna$T_return[inds] <- NA
zna$Flow[inds] <- NA

### Convert into Time Series ---------------------------------------------------------
TS_Energy <- xts(z$Energy, z$Time)  #Real data to be compared with
TS_Volume <- xts(z$Volume_accum, z$Time)
TS_Volume_Tsupply <- xts(z$Volume_TS, z$Time)
TS_Volume_Treturn <- xts(z$Volume_TR, z$Time)
TS_Tsupply <- xts(z$T_supply, z$Time)
TS_Treturn <- xts(z$T_return, z$Time)
TS_Flow <- xts(z$Flow, z$Time)

TS_NA_Energy <- xts(zna$Energy, zna$Time) #Data with NA to be solved
TS_NA_Volume <- xts(zna$Volume_accum, zna$Time)
TS_NA_Volume_Tsupply <- xts(zna$Volume_TS, zna$Time)
TS_NA_Volume_Treturn <- xts(zna$Volume_TR, zna$Time)
TS_NA_Tsupply <- xts(zna$T_supply, zna$Time)
TS_NA_Treturn <- xts(zna$T_return, zna$Time)
TS_NA_Flow <- xts(zna$Flow, zna$Time)



Results <- data.frame("Method"=character(), # Dataframe with RMSE and MAE results from the
     comparison between values
                "Variable"= character(),
                "RMSE"= numeric(),
                "MAE" = numeric(),
                stringsAsFactors = FALSE)

### Calculations ----------------------------------------

## Linear Interpolation -----------------------------

  method_name <- "linear"

  TS_Energy_final <- na_interpolation(TS_NA_Energy, option = method_name)
  TS_Volume_final <- na_interpolation(TS_NA_Volume, option = method_name)
  TS_Volume_Tsupply_final <- na_interpolation(TS_NA_Volume_Tsupply, option = method_name)
  TS_Volume_Treturn_final <- na_interpolation(TS_NA_Volume_Treturn, option = method_name)
  TS_Tsupply_final <- na_interpolation(TS_NA_Tsupply, option = method_name)
  TS_Treturn_final <- na_interpolation(TS_NA_Treturn, option = method_name)
  TS_Flow_final <- na_interpolation(TS_NA_Flow, option = method_name)

  # Root Mean Squared Error
  RMSE_Energy <- rmse(TS_Energy_final, TS_Energy)
  RMSE_Volume <- rmse(TS_Volume_final, TS_Volume)
  RMSE_Volume_Tsupply <- rmse(TS_Volume_Tsupply_final, TS_Volume_Tsupply)
  RMSE_Volume_Treturn <- rmse(TS_Volume_Treturn_final, TS_Volume_Treturn)
  RMSE_Tsupply <- rmse(TS_Tsupply_final, TS_Tsupply)
  RMSE_Treturn <- rmse(TS_Treturn_final, TS_Treturn)
  RMSE_Flow <- rmse(TS_Flow_final, TS_Flow)

  # Mean Absolute Error
  MAE_Energy <- mae(TS_Energy_final, TS_Energy)
  MAE_Volume <- mae(TS_Volume_final, TS_Volume)
  MAE_Volume_Tsupply <- mae(TS_Volume_Tsupply_final, TS_Volume_Tsupply)
  MAE_Volume_Treturn <- mae(TS_Volume_Treturn_final, TS_Volume_Treturn)
  MAE_Tsupply <- mae(TS_Tsupply_final, TS_Tsupply)
  MAE_Treturn <- mae(TS_Treturn_final, TS_Treturn)
  MAE_Flow <- mae(TS_Flow_final, TS_Flow)

  Results_Energy <- data.frame('Method' = "Linear Interpolation", "Variable" = "Energy", "RMSE" =
       RMSE_Energy, "MAE" = MAE_Energy)
  Results_Volume <- data.frame('Method' = "Linear Interpolation", "Variable" = "Volume", "RMSE" =
       RMSE_Volume, "MAE" = MAE_Volume)
  Results_Volume_Tsupply <- data.frame('Method' = "Linear Interpolation", "Variable" =
       "Volume_Tsupply", "RMSE" = RMSE_Volume_Tsupply, "MAE" = MAE_Volume_Tsupply)
  Results_Volume_Treturn <- data.frame('Method' = "Linear Interpolation", "Variable" =
       "Volume_Treturn", "RMSE" = RMSE_Volume_Treturn, "MAE" = MAE_Volume_Treturn)
  Results_Tsupply <- data.frame('Method' = "Linear Interpolation", "Variable" = "Tsupply", "RMSE"
       = RMSE_Tsupply, "MAE" = MAE_Tsupply)
  Results_Treturn <- data.frame('Method' = "Linear Interpolation", "Variable" = "Treturn", "RMSE"
       = RMSE_Treturn, "MAE" = MAE_Treturn)
  Results_Flow <- data.frame('Method' = "Linear Interpolation", "Variable" = "Flow", "RMSE" =
       RMSE_Flow, "MAE" = MAE_Flow)

  Results_provisory <- bind_rows(Results_Energy, Results_Volume, Results_Volume_Tsupply,
       Results_Volume_Treturn, Results_Tsupply, Results_Treturn, Results_Flow)

  Results <- bind_rows(Results, Results_provisory)

## Spline Interpolation -------------------------------

  method_name <- "spline"

  TS_Energy_final <- na_interpolation(TS_NA_Energy, option = method_name)
  TS_Volume_final <- na_interpolation(TS_NA_Volume, option = method_name)
  TS_Volume_Tsupply_final <- na_interpolation(TS_NA_Volume_Tsupply, option = method_name)
  TS_Volume_Treturn_final <- na_interpolation(TS_NA_Volume_Treturn, option = method_name)
  TS_Tsupply_final <- na_interpolation(TS_NA_Tsupply, option = method_name)
  TS_Treturn_final <- na_interpolation(TS_NA_Treturn, option = method_name)
  TS_Flow_final <- na_interpolation(TS_NA_Flow, option = method_name)

  # Root Mean Squared Error
  RMSE_Energy <- rmse(TS_Energy_final, TS_Energy)
  RMSE_Volume <- rmse(TS_Volume_final, TS_Volume)
  RMSE_Volume_Tsupply <- rmse(TS_Volume_Tsupply_final, TS_Volume_Tsupply)
  RMSE_Volume_Treturn <- rmse(TS_Volume_Treturn_final, TS_Volume_Treturn)
  RMSE_Tsupply <- rmse(TS_Tsupply_final, TS_Tsupply)
  RMSE_Treturn <- rmse(TS_Treturn_final, TS_Treturn)
  RMSE_Flow <- rmse(TS_Flow_final, TS_Flow)

  # Mean Absolute Error
  MAE_Energy <- mae(TS_Energy_final, TS_Energy)
  MAE_Volume <- mae(TS_Volume_final, TS_Volume)
  MAE_Volume_Tsupply <- mae(TS_Volume_Tsupply_final, TS_Volume_Tsupply)
  MAE_Volume_Treturn <- mae(TS_Volume_Treturn_final, TS_Volume_Treturn)
  MAE_Tsupply <- mae(TS_Tsupply_final, TS_Tsupply)
  MAE_Treturn <- mae(TS_Treturn_final, TS_Treturn)
  MAE_Flow <- mae(TS_Flow_final, TS_Flow)

  Results_Energy <- data.frame('Method' = "Spline Interpolation", "Variable" = "Energy", "RMSE" =
       RMSE_Energy, "MAE" = MAE_Energy)
  Results_Volume <- data.frame('Method' = "Spline Interpolation", "Variable" = "Volume", "RMSE" =
       RMSE_Volume, "MAE" = MAE_Volume)
  Results_Volume_Tsupply <- data.frame('Method' = "Spline Interpolation", "Variable" =
       "Volume_Tsupply", "RMSE" = RMSE_Volume_Tsupply, "MAE" = MAE_Volume_Tsupply)
  Results_Volume_Treturn <- data.frame('Method' = "Spline Interpolation", "Variable" =
       "Volume_Treturn", "RMSE" = RMSE_Volume_Treturn, "MAE" = MAE_Volume_Treturn)
  Results_Tsupply <- data.frame('Method' = "Spline Interpolation", "Variable" = "Tsupply", "RMSE"
       = RMSE_Tsupply, "MAE" = MAE_Tsupply)
  Results_Treturn <- data.frame('Method' = "Spline Interpolation", "Variable" = "Treturn", "RMSE"
    = RMSE_Treturn, "MAE" = MAE_Treturn)
  Results_Flow <- data.frame('Method' = "Spline Interpolation", "Variable" = "Flow", "RMSE" =
    RMSE_Flow, "MAE" = MAE_Flow)

  Results_provisory <- bind_rows(Results_Energy, Results_Volume, Results_Volume_Tsupply,
    Results_Volume_Treturn, Results_Tsupply, Results_Treturn, Results_Flow)

  Results <- bind_rows(Results, Results_provisory)

## Stineman Interpolation ------------------------------

  method_name <- "stine"

  TS_Energy_final <- na_interpolation(TS_NA_Energy, option = method_name)
  TS_Volume_final <- na_interpolation(TS_NA_Volume, option = method_name)
  TS_Volume_Tsupply_final <- na_interpolation(TS_NA_Volume_Tsupply, option = method_name)
  TS_Volume_Treturn_final <- na_interpolation(TS_NA_Volume_Treturn, option = method_name)
  TS_Tsupply_final <- na_interpolation(TS_NA_Tsupply, option = method_name)
  TS_Treturn_final <- na_interpolation(TS_NA_Treturn, option = method_name)
  TS_Flow_final <- na_interpolation(TS_NA_Flow, option = method_name)

  # Root Mean Squared Error
  RMSE_Energy <- rmse(TS_Energy_final, TS_Energy)
  RMSE_Volume <- rmse(TS_Volume_final, TS_Volume)
  RMSE_Volume_Tsupply <- rmse(TS_Volume_Tsupply_final, TS_Volume_Tsupply)
  RMSE_Volume_Treturn <- rmse(TS_Volume_Treturn_final, TS_Volume_Treturn)
  RMSE_Tsupply <- rmse(TS_Tsupply_final, TS_Tsupply)
  RMSE_Treturn <- rmse(TS_Treturn_final, TS_Treturn)
  RMSE_Flow <- rmse(TS_Flow_final, TS_Flow)

  # Mean Absolute Error
  MAE_Energy <- mae(TS_Energy_final, TS_Energy)
  MAE_Volume <- mae(TS_Volume_final, TS_Volume)
  MAE_Volume_Tsupply <- mae(TS_Volume_Tsupply_final, TS_Volume_Tsupply)
  MAE_Volume_Treturn <- mae(TS_Volume_Treturn_final, TS_Volume_Treturn)
  MAE_Tsupply <- mae(TS_Tsupply_final, TS_Tsupply)
  MAE_Treturn <- mae(TS_Treturn_final, TS_Treturn)
  MAE_Flow <- mae(TS_Flow_final, TS_Flow)

  Results_Energy <- data.frame('Method' = "Stineman Interpolation", "Variable" = "Energy", "RMSE"
    = RMSE_Energy, "MAE" = MAE_Energy)
  Results_Volume <- data.frame('Method' = "Stineman Interpolation", "Variable" = "Volume", "RMSE"
    = RMSE_Volume, "MAE" = MAE_Volume)
  Results_Volume_Tsupply <- data.frame('Method' = "Stineman Interpolation", "Variable" =
    "Volume_Tsupply", "RMSE" = RMSE_Volume_Tsupply, "MAE" = MAE_Volume_Tsupply)
  Results_Volume_Treturn <- data.frame('Method' = "Stineman Interpolation", "Variable" =
    "Volume_Treturn", "RMSE" = RMSE_Volume_Treturn, "MAE" = MAE_Volume_Treturn)
  Results_Tsupply <- data.frame('Method' = "Stineman Interpolation", "Variable" = "Tsupply",
    "RMSE" = RMSE_Tsupply, "MAE" = MAE_Tsupply)
  Results_Treturn <- data.frame('Method' = "Stineman Interpolation", "Variable" = "Treturn",
    "RMSE" = RMSE_Treturn, "MAE" = MAE_Treturn)
  Results_Flow <- data.frame('Method' = "Stineman Interpolation", "Variable" = "Flow", "RMSE" =
    RMSE_Flow, "MAE" = MAE_Flow)

  Results_provisory <- bind_rows(Results_Energy, Results_Volume, Results_Volume_Tsupply,
    Results_Volume_Treturn, Results_Tsupply, Results_Treturn, Results_Flow)

  Results <- bind_rows(Results, Results_provisory)

## Last Observation Carried Forward ------------------------------

  method_name <- "locf"

  TS_Energy_final <- na_locf(TS_NA_Energy, option = method_name, na_remaining = "rev")
  TS_Volume_final <- na_locf(TS_NA_Volume, option = method_name, na_remaining = "rev")
  TS_Volume_Tsupply_final <- na_locf(TS_NA_Volume_Tsupply, option = method_name, na_remaining =
```

```r
        "rev")
    TS_Volume_Treturn_final <- na_locf(TS_NA_Volume_Treturn, option = method_name, na_remaining =
        "rev")
    TS_Tsupply_final <- na_locf(TS_NA_Tsupply, option = method_name, na_remaining = "rev")
    TS_Treturn_final <- na_locf(TS_NA_Treturn, option = method_name, na_remaining = "rev")
    TS_Flow_final <- na_locf(TS_NA_Flow, option = method_name, na_remaining = "rev")

    # Root Mean Squared Error
    RMSE_Energy <- rmse(TS_Energy_final, TS_Energy)
    RMSE_Volume <- rmse(TS_Volume_final, TS_Volume)
    RMSE_Volume_Tsupply <- rmse(TS_Volume_Tsupply_final, TS_Volume_Tsupply)
    RMSE_Volume_Treturn <- rmse(TS_Volume_Treturn_final, TS_Volume_Treturn)
    RMSE_Tsupply <- rmse(TS_Tsupply_final, TS_Tsupply)
    RMSE_Treturn <- rmse(TS_Treturn_final, TS_Treturn)
    RMSE_Flow <- rmse(TS_Flow_final, TS_Flow)

    # Mean Absolute Error
    MAE_Energy <- mae(TS_Energy_final, TS_Energy)
    MAE_Volume <- mae(TS_Volume_final, TS_Volume)
    MAE_Volume_Tsupply <- mae(TS_Volume_Tsupply_final, TS_Volume_Tsupply)
    MAE_Volume_Treturn <- mae(TS_Volume_Treturn_final, TS_Volume_Treturn)
    MAE_Tsupply <- mae(TS_Tsupply_final, TS_Tsupply)
    MAE_Treturn <- mae(TS_Treturn_final, TS_Treturn)
    MAE_Flow <- mae(TS_Flow_final, TS_Flow)

    Results_Energy <- data.frame('Method' = "LOCF", "Variable" = "Energy", "RMSE" = RMSE_Energy,
        "MAE" = MAE_Energy)
    Results_Volume <- data.frame('Method' = "LOCF", "Variable" = "Volume", "RMSE" = RMSE_Volume,
        "MAE" = MAE_Volume)
    Results_Volume_Tsupply <- data.frame('Method' = "LOCF", "Variable" = "Volume_Tsupply", "RMSE" =
        RMSE_Volume_Tsupply, "MAE" = MAE_Volume_Tsupply)
    Results_Volume_Treturn <- data.frame('Method' = "LOCF", "Variable" = "Volume_Treturn", "RMSE" =
        RMSE_Volume_Treturn, "MAE" = MAE_Volume_Treturn)
    Results_Tsupply <- data.frame('Method' = "LOCF", "Variable" = "Tsupply", "RMSE" = RMSE_Tsupply,
        "MAE" = MAE_Tsupply)
    Results_Treturn <- data.frame('Method' = "LOCF", "Variable" = "Treturn", "RMSE" = RMSE_Treturn,
        "MAE" = MAE_Treturn)
    Results_Flow <- data.frame('Method' = "LOCF", "Variable" = "Flow", "RMSE" = RMSE_Flow, "MAE" =
        MAE_Flow)

    Results_provisory <- bind_rows(Results_Energy, Results_Volume, Results_Volume_Tsupply,
        Results_Volume_Treturn, Results_Tsupply, Results_Treturn, Results_Flow)

    Results <- bind_rows(Results, Results_provisory)

## Next Observation Carried Backward -------------------------------

    method_name <- "nocb"

    TS_Energy_final <- na_locf(TS_NA_Energy, option = method_name, na_remaining = "rev")
    TS_Volume_final <- na_locf(TS_NA_Volume, option = method_name, na_remaining = "rev")
    TS_Volume_Tsupply_final <- na_locf(TS_NA_Volume_Tsupply, option = method_name, na_remaining =
        "rev")
    TS_Volume_Treturn_final <- na_locf(TS_NA_Volume_Treturn, option = method_name, na_remaining =
        "rev")
    TS_Tsupply_final <- na_locf(TS_NA_Tsupply, option = method_name, na_remaining = "rev")
    TS_Treturn_final <- na_locf(TS_NA_Treturn, option = method_name, na_remaining = "rev")
    TS_Flow_final <- na_locf(TS_NA_Flow, option = method_name, na_remaining = "rev")

    # Root Mean Squared Error
    RMSE_Energy <- rmse(TS_Energy_final, TS_Energy)
    RMSE_Volume <- rmse(TS_Volume_final, TS_Volume)
    RMSE_Volume_Tsupply <- rmse(TS_Volume_Tsupply_final, TS_Volume_Tsupply)
    RMSE_Volume_Treturn <- rmse(TS_Volume_Treturn_final, TS_Volume_Treturn)
    RMSE_Tsupply <- rmse(TS_Tsupply_final, TS_Tsupply)
    RMSE_Treturn <- rmse(TS_Treturn_final, TS_Treturn)
```

```
  RMSE_Flow <- rmse(TS_Flow_final, TS_Flow)

  # Mean Absolute Error
  MAE_Energy <- mae(TS_Energy_final, TS_Energy)
  MAE_Volume <- mae(TS_Volume_final, TS_Volume)
  MAE_Volume_Tsupply <- mae(TS_Volume_Tsupply_final, TS_Volume_Tsupply)
  MAE_Volume_Treturn <- mae(TS_Volume_Treturn_final, TS_Volume_Treturn)
  MAE_Tsupply <- mae(TS_Tsupply_final, TS_Tsupply)
  MAE_Treturn <- mae(TS_Treturn_final, TS_Treturn)
  MAE_Flow <- mae(TS_Flow_final, TS_Flow)

  Results_Energy <- data.frame('Method' = "NOCB", "Variable" = "Energy", "RMSE" = RMSE_Energy,
      "MAE" = MAE_Energy)
  Results_Volume <- data.frame('Method' = "NOCB", "Variable" = "Volume", "RMSE" = RMSE_Volume,
      "MAE" = MAE_Volume)
  Results_Volume_Tsupply <- data.frame('Method' = "NOCB", "Variable" = "Volume_Tsupply", "RMSE" =
      RMSE_Volume_Tsupply, "MAE" = MAE_Volume_Tsupply)
  Results_Volume_Treturn <- data.frame('Method' = "NOCB", "Variable" = "Volume_Treturn", "RMSE" =
      RMSE_Volume_Treturn, "MAE" = MAE_Volume_Treturn)
  Results_Tsupply <- data.frame('Method' = "NOCB", "Variable" = "Tsupply", "RMSE" = RMSE_Tsupply,
      "MAE" = MAE_Tsupply)
  Results_Treturn <- data.frame('Method' = "NOCB", "Variable" = "Treturn", "RMSE" = RMSE_Treturn,
      "MAE" = MAE_Treturn)
  Results_Flow <- data.frame('Method' = "NOCB", "Variable" = "Flow", "RMSE" = RMSE_Flow, "MAE" =
      MAE_Flow)

  Results_provisory <- bind_rows(Results_Energy, Results_Volume, Results_Volume_Tsupply,
      Results_Volume_Treturn, Results_Tsupply, Results_Treturn, Results_Flow)

  Results <- bind_rows(Results, Results_provisory)

## Weighted Moving Average: Simple ------------------------------

  method_name <- "simple"

  TS_Energy_final_1 <- na_ma(TS_NA_Energy, k = 2, weighting = method_name)
  TS_Energy_final_2 <- na_ma(TS_NA_Energy, k = 4, weighting = method_name)
  TS_Energy_final_3 <- na_ma(TS_NA_Energy, k = 6, weighting = method_name)
  TS_Energy_final_4 <- na_ma(TS_NA_Energy, k = 8, weighting = method_name)

  TS_Volume_final_1 <- na_ma(TS_NA_Volume, k = 2, weighting = method_name)
  TS_Volume_final_2 <- na_ma(TS_NA_Volume, k = 4, weighting = method_name)
  TS_Volume_final_3 <- na_ma(TS_NA_Volume, k = 6, weighting = method_name)
  TS_Volume_final_4 <- na_ma(TS_NA_Volume, k = 8, weighting = method_name)

  TS_Volume_Tsupply_final_1 <- na_ma(TS_NA_Volume_Tsupply, k = 2, weighting = method_name)
  TS_Volume_Tsupply_final_2 <- na_ma(TS_NA_Volume_Tsupply, k = 4, weighting = method_name)
  TS_Volume_Tsupply_final_3 <- na_ma(TS_NA_Volume_Tsupply, k = 6, weighting = method_name)
  TS_Volume_Tsupply_final_4 <- na_ma(TS_NA_Volume_Tsupply, k = 8, weighting = method_name)

  TS_Volume_Treturn_final_1 <- na_ma(TS_NA_Volume_Treturn, k = 2, weighting = method_name)
  TS_Volume_Treturn_final_2 <- na_ma(TS_NA_Volume_Treturn, k = 4, weighting = method_name)
  TS_Volume_Treturn_final_3 <- na_ma(TS_NA_Volume_Treturn, k = 6, weighting = method_name)
  TS_Volume_Treturn_final_4 <- na_ma(TS_NA_Volume_Treturn, k = 8, weighting = method_name)

  TS_Tsupply_final_1 <- na_ma(TS_NA_Tsupply, k = 2, weighting = method_name)
  TS_Tsupply_final_2 <- na_ma(TS_NA_Tsupply, k = 4, weighting = method_name)
  TS_Tsupply_final_3 <- na_ma(TS_NA_Tsupply, k = 6, weighting = method_name)
  TS_Tsupply_final_4 <- na_ma(TS_NA_Tsupply, k = 8, weighting = method_name)

  TS_Treturn_final_1 <- na_ma(TS_NA_Treturn, k = 2, weighting = method_name)
  TS_Treturn_final_2 <- na_ma(TS_NA_Treturn, k = 4, weighting = method_name)
  TS_Treturn_final_3 <- na_ma(TS_NA_Treturn, k = 6, weighting = method_name)
  TS_Treturn_final_4 <- na_ma(TS_NA_Treturn, k = 8, weighting = method_name)

  TS_Flow_final_1 <- na_ma(TS_NA_Flow, k = 2, weighting = method_name)
```

```r
TS_Flow_final_2 <- na_ma(TS_NA_Flow, k = 4, weighting = method_name)
TS_Flow_final_3 <- na_ma(TS_NA_Flow, k = 6, weighting = method_name)
TS_Flow_final_4 <- na_ma(TS_NA_Flow, k = 8, weighting = method_name)

# Root Mean Squared Error
RMSE_Energy_1 <- rmse(TS_Energy_final_1, TS_Energy)
RMSE_Energy_2 <- rmse(TS_Energy_final_2, TS_Energy)
RMSE_Energy_3 <- rmse(TS_Energy_final_3, TS_Energy)
RMSE_Energy_4 <- rmse(TS_Energy_final_4, TS_Energy)

RMSE_Volume_1 <- rmse(TS_Volume_final_1, TS_Volume)
RMSE_Volume_2 <- rmse(TS_Volume_final_2, TS_Volume)
RMSE_Volume_3 <- rmse(TS_Volume_final_3, TS_Volume)
RMSE_Volume_4 <- rmse(TS_Volume_final_4, TS_Volume)

RMSE_Volume_Tsupply_1 <- rmse(TS_Volume_Tsupply_final_1, TS_Volume_Tsupply)
RMSE_Volume_Tsupply_2 <- rmse(TS_Volume_Tsupply_final_2, TS_Volume_Tsupply)
RMSE_Volume_Tsupply_3 <- rmse(TS_Volume_Tsupply_final_3, TS_Volume_Tsupply)
RMSE_Volume_Tsupply_4 <- rmse(TS_Volume_Tsupply_final_4, TS_Volume_Tsupply)

RMSE_Volume_Treturn_1 <- rmse(TS_Volume_Treturn_final_1, TS_Volume_Treturn)
RMSE_Volume_Treturn_2 <- rmse(TS_Volume_Treturn_final_2, TS_Volume_Treturn)
RMSE_Volume_Treturn_3 <- rmse(TS_Volume_Treturn_final_3, TS_Volume_Treturn)
RMSE_Volume_Treturn_4 <- rmse(TS_Volume_Treturn_final_4, TS_Volume_Treturn)

RMSE_Tsupply_1 <- rmse(TS_Tsupply_final_1, TS_Tsupply)
RMSE_Tsupply_2 <- rmse(TS_Tsupply_final_2, TS_Tsupply)
RMSE_Tsupply_3 <- rmse(TS_Tsupply_final_3, TS_Tsupply)
RMSE_Tsupply_4 <- rmse(TS_Tsupply_final_4, TS_Tsupply)

RMSE_Treturn_1 <- rmse(TS_Treturn_final_1, TS_Treturn)
RMSE_Treturn_2 <- rmse(TS_Treturn_final_2, TS_Treturn)
RMSE_Treturn_3 <- rmse(TS_Treturn_final_3, TS_Treturn)
RMSE_Treturn_4 <- rmse(TS_Treturn_final_4, TS_Treturn)

RMSE_Flow_1 <- rmse(TS_Flow_final_1, TS_Flow)
RMSE_Flow_2 <- rmse(TS_Flow_final_2, TS_Flow)
RMSE_Flow_3 <- rmse(TS_Flow_final_3, TS_Flow)
RMSE_Flow_4 <- rmse(TS_Flow_final_4, TS_Flow)


# Mean Absolute Error
MAE_Energy_1 <- mae(TS_Energy_final_1, TS_Energy)
MAE_Energy_2 <- mae(TS_Energy_final_2, TS_Energy)
MAE_Energy_3 <- mae(TS_Energy_final_3, TS_Energy)
MAE_Energy_4 <- mae(TS_Energy_final_4, TS_Energy)

MAE_Volume_1 <- mae(TS_Volume_final_1, TS_Volume)
MAE_Volume_2 <- mae(TS_Volume_final_2, TS_Volume)
MAE_Volume_3 <- mae(TS_Volume_final_3, TS_Volume)
MAE_Volume_4 <- mae(TS_Volume_final_4, TS_Volume)

MAE_Volume_Tsupply_1 <- mae(TS_Volume_Tsupply_final_1, TS_Volume_Tsupply)
MAE_Volume_Tsupply_2 <- mae(TS_Volume_Tsupply_final_2, TS_Volume_Tsupply)
MAE_Volume_Tsupply_3 <- mae(TS_Volume_Tsupply_final_3, TS_Volume_Tsupply)
MAE_Volume_Tsupply_4 <- mae(TS_Volume_Tsupply_final_4, TS_Volume_Tsupply)

MAE_Volume_Treturn_1 <- mae(TS_Volume_Treturn_final_1, TS_Volume_Treturn)
MAE_Volume_Treturn_2 <- mae(TS_Volume_Treturn_final_2, TS_Volume_Treturn)
MAE_Volume_Treturn_3 <- mae(TS_Volume_Treturn_final_3, TS_Volume_Treturn)
MAE_Volume_Treturn_4 <- mae(TS_Volume_Treturn_final_4, TS_Volume_Treturn)

MAE_Tsupply_1 <- mae(TS_Tsupply_final_1, TS_Tsupply)
MAE_Tsupply_2 <- mae(TS_Tsupply_final_2, TS_Tsupply)
MAE_Tsupply_3 <- mae(TS_Tsupply_final_3, TS_Tsupply)
MAE_Tsupply_4 <- mae(TS_Tsupply_final_4, TS_Tsupply)
```

```
MAE_Treturn_1 <- mae(TS_Treturn_final_1, TS_Treturn)
MAE_Treturn_2 <- mae(TS_Treturn_final_2, TS_Treturn)
MAE_Treturn_3 <- mae(TS_Treturn_final_3, TS_Treturn)
MAE_Treturn_4 <- mae(TS_Treturn_final_4, TS_Treturn)

MAE_Flow_1 <- mae(TS_Flow_final_1, TS_Flow)
MAE_Flow_2 <- mae(TS_Flow_final_2, TS_Flow)
MAE_Flow_3 <- mae(TS_Flow_final_3, TS_Flow)
MAE_Flow_4 <- mae(TS_Flow_final_4, TS_Flow)

Results_Energy_1 <- data.frame('Method' = "WMA Simple, k=2", "Variable" = "Energy", "RMSE" =
     RMSE_Energy_1, "MAE" = MAE_Energy_1)
Results_Energy_2 <- data.frame('Method' = "WMA Simple, k=4", "Variable" = "Energy", "RMSE" =
     RMSE_Energy_2, "MAE" = MAE_Energy_2)
Results_Energy_3 <- data.frame('Method' = "WMA Simple, k=6", "Variable" = "Energy", "RMSE" =
     RMSE_Energy_3, "MAE" = MAE_Energy_3)
Results_Energy_4 <- data.frame('Method' = "WMA Simple, k=8", "Variable" = "Energy", "RMSE" =
     RMSE_Energy_4, "MAE" = MAE_Energy_4)

Results_Volume_1 <- data.frame('Method' = "WMA Simple, k=2", "Variable" = "Volume", "RMSE" =
     RMSE_Volume_1, "MAE" = MAE_Volume_1)
Results_Volume_2 <- data.frame('Method' = "WMA Simple, k=4", "Variable" = "Volume", "RMSE" =
     RMSE_Volume_2, "MAE" = MAE_Volume_2)
Results_Volume_3 <- data.frame('Method' = "WMA Simple, k=6", "Variable" = "Volume", "RMSE" =
     RMSE_Volume_3, "MAE" = MAE_Volume_3)
Results_Volume_4 <- data.frame('Method' = "WMA Simple, k=8", "Variable" = "Volume", "RMSE" =
     RMSE_Volume_4, "MAE" = MAE_Volume_4)

Results_Volume_Tsupply_1 <- data.frame('Method' = "WMA Simple, k=2", "Variable" =
     "Volume_Tsupply", "RMSE" = RMSE_Volume_Tsupply_1, "MAE" = MAE_Volume_Tsupply_1)
Results_Volume_Tsupply_2 <- data.frame('Method' = "WMA Simple, k=4", "Variable" =
     "Volume_Tsupply", "RMSE" = RMSE_Volume_Tsupply_2, "MAE" = MAE_Volume_Tsupply_2)
Results_Volume_Tsupply_3 <- data.frame('Method' = "WMA Simple, k=6", "Variable" =
     "Volume_Tsupply", "RMSE" = RMSE_Volume_Tsupply_3, "MAE" = MAE_Volume_Tsupply_3)
Results_Volume_Tsupply_4 <- data.frame('Method' = "WMA Simple, k=8", "Variable" =
     "Volume_Tsupply", "RMSE" = RMSE_Volume_Tsupply_4, "MAE" = MAE_Volume_Tsupply_4)

Results_Volume_Treturn_1 <- data.frame('Method' = "WMA Simple, k=2", "Variable" =
     "Volume_Treturn", "RMSE" = RMSE_Volume_Treturn_1, "MAE" = MAE_Volume_Treturn_1)
Results_Volume_Treturn_2 <- data.frame('Method' = "WMA Simple, k=4", "Variable" =
     "Volume_Treturn", "RMSE" = RMSE_Volume_Treturn_2, "MAE" = MAE_Volume_Treturn_2)
Results_Volume_Treturn_3 <- data.frame('Method' = "WMA Simple, k=6", "Variable" =
     "Volume_Treturn", "RMSE" = RMSE_Volume_Treturn_3, "MAE" = MAE_Volume_Treturn_3)
Results_Volume_Treturn_4 <- data.frame('Method' = "WMA Simple, k=8", "Variable" =
     "Volume_Treturn", "RMSE" = RMSE_Volume_Treturn_4, "MAE" = MAE_Volume_Treturn_4)

Results_Tsupply_1 <- data.frame('Method' = "WMA Simple, k=2", "Variable" = "Tsupply", "RMSE" =
     RMSE_Tsupply_1, "MAE" = MAE_Tsupply_1)
Results_Tsupply_2 <- data.frame('Method' = "WMA Simple, k=4", "Variable" = "Tsupply", "RMSE" =
     RMSE_Tsupply_2, "MAE" = MAE_Tsupply_2)
Results_Tsupply_3 <- data.frame('Method' = "WMA Simple, k=6", "Variable" = "Tsupply", "RMSE" =
     RMSE_Tsupply_3, "MAE" = MAE_Tsupply_3)
Results_Tsupply_4 <- data.frame('Method' = "WMA Simple, k=8", "Variable" = "Tsupply", "RMSE" =
     RMSE_Tsupply_4, "MAE" = MAE_Tsupply_4)

Results_Treturn_1 <- data.frame('Method' = "WMA Simple, k=2", "Variable" = "Treturn", "RMSE" =
     RMSE_Treturn_1, "MAE" = MAE_Treturn_1)
Results_Treturn_2 <- data.frame('Method' = "WMA Simple, k=4", "Variable" = "Treturn", "RMSE" =
     RMSE_Treturn_2, "MAE" = MAE_Treturn_2)
Results_Treturn_3 <- data.frame('Method' = "WMA Simple, k=6", "Variable" = "Treturn", "RMSE" =
     RMSE_Treturn_3, "MAE" = MAE_Treturn_3)
Results_Treturn_4 <- data.frame('Method' = "WMA Simple, k=8", "Variable" = "Treturn", "RMSE" =
     RMSE_Treturn_4, "MAE" = MAE_Treturn_4)

Results_Flow_1 <- data.frame('Method' = "WMA Simple, k=2", "Variable" = "Flow", "RMSE" =
        RMSE_Flow_1, "MAE" = MAE_Flow_1)
  Results_Flow_2 <- data.frame('Method' = "WMA Simple, k=4", "Variable" = "Flow", "RMSE" =
        RMSE_Flow_2, "MAE" = MAE_Flow_2)
  Results_Flow_3 <- data.frame('Method' = "WMA Simple, k=6", "Variable" = "Flow", "RMSE" =
        RMSE_Flow_3, "MAE" = MAE_Flow_3)
  Results_Flow_4 <- data.frame('Method' = "WMA Simple, k=8", "Variable" = "Flow", "RMSE" =
        RMSE_Flow_4, "MAE" = MAE_Flow_4)

  Results_provisory <- bind_rows(Results_Energy_1, Results_Energy_2, Results_Energy_3,
        Results_Energy_4,
                              Results_Volume_1, Results_Volume_2, Results_Volume_3,
                                    Results_Volume_4,
                              Results_Volume_Tsupply_1, Results_Volume_Tsupply_2,
                                    Results_Volume_Tsupply_3, Results_Volume_Tsupply_4,
                              Results_Volume_Treturn_1, Results_Volume_Treturn_2,
                                    Results_Volume_Treturn_3, Results_Volume_Treturn_4,
                              Results_Tsupply_1, Results_Tsupply_2, Results_Tsupply_3,
                                    Results_Tsupply_4,
                              Results_Treturn_1, Results_Treturn_2, Results_Treturn_3,
                                    Results_Treturn_4,
                              Results_Flow_1, Results_Flow_2, Results_Flow_3, Results_Flow_4)

  Results <- bind_rows(Results, Results_provisory)

## Weighted Moving Average: Linear ------------------------------

  method_name <- "linear"

  TS_Energy_final_1 <- na_ma(TS_NA_Energy, k = 2, weighting = method_name)
  TS_Energy_final_2 <- na_ma(TS_NA_Energy, k = 4, weighting = method_name)
  TS_Energy_final_3 <- na_ma(TS_NA_Energy, k = 6, weighting = method_name)
  TS_Energy_final_4 <- na_ma(TS_NA_Energy, k = 8, weighting = method_name)

  TS_Volume_final_1 <- na_ma(TS_NA_Volume, k = 2, weighting = method_name)
  TS_Volume_final_2 <- na_ma(TS_NA_Volume, k = 4, weighting = method_name)
  TS_Volume_final_3 <- na_ma(TS_NA_Volume, k = 6, weighting = method_name)
  TS_Volume_final_4 <- na_ma(TS_NA_Volume, k = 8, weighting = method_name)

  TS_Volume_Tsupply_final_1 <- na_ma(TS_NA_Volume_Tsupply, k = 2, weighting = method_name)
  TS_Volume_Tsupply_final_2 <- na_ma(TS_NA_Volume_Tsupply, k = 4, weighting = method_name)
  TS_Volume_Tsupply_final_3 <- na_ma(TS_NA_Volume_Tsupply, k = 6, weighting = method_name)
  TS_Volume_Tsupply_final_4 <- na_ma(TS_NA_Volume_Tsupply, k = 8, weighting = method_name)

  TS_Volume_Treturn_final_1 <- na_ma(TS_NA_Volume_Treturn, k = 2, weighting = method_name)
  TS_Volume_Treturn_final_2 <- na_ma(TS_NA_Volume_Treturn, k = 4, weighting = method_name)
  TS_Volume_Treturn_final_3 <- na_ma(TS_NA_Volume_Treturn, k = 6, weighting = method_name)
  TS_Volume_Treturn_final_4 <- na_ma(TS_NA_Volume_Treturn, k = 8, weighting = method_name)

  TS_Tsupply_final_1 <- na_ma(TS_NA_Tsupply, k = 2, weighting = method_name)
  TS_Tsupply_final_2 <- na_ma(TS_NA_Tsupply, k = 4, weighting = method_name)
  TS_Tsupply_final_3 <- na_ma(TS_NA_Tsupply, k = 6, weighting = method_name)
  TS_Tsupply_final_4 <- na_ma(TS_NA_Tsupply, k = 8, weighting = method_name)

  TS_Treturn_final_1 <- na_ma(TS_NA_Treturn, k = 2, weighting = method_name)
  TS_Treturn_final_2 <- na_ma(TS_NA_Treturn, k = 4, weighting = method_name)
  TS_Treturn_final_3 <- na_ma(TS_NA_Treturn, k = 6, weighting = method_name)
  TS_Treturn_final_4 <- na_ma(TS_NA_Treturn, k = 8, weighting = method_name)

  TS_Flow_final_1 <- na_ma(TS_NA_Flow, k = 2, weighting = method_name)
  TS_Flow_final_2 <- na_ma(TS_NA_Flow, k = 4, weighting = method_name)
  TS_Flow_final_3 <- na_ma(TS_NA_Flow, k = 6, weighting = method_name)
  TS_Flow_final_4 <- na_ma(TS_NA_Flow, k = 8, weighting = method_name)

  # Root Mean Squared Error
  RMSE_Energy_1 <- rmse(TS_Energy_final_1, TS_Energy)
  RMSE_Energy_2 <- rmse(TS_Energy_final_2, TS_Energy)
```

```r
RMSE_Energy_3 <- rmse(TS_Energy_final_3, TS_Energy)
RMSE_Energy_4 <- rmse(TS_Energy_final_4, TS_Energy)

RMSE_Volume_1 <- rmse(TS_Volume_final_1, TS_Volume)
RMSE_Volume_2 <- rmse(TS_Volume_final_2, TS_Volume)
RMSE_Volume_3 <- rmse(TS_Volume_final_3, TS_Volume)
RMSE_Volume_4 <- rmse(TS_Volume_final_4, TS_Volume)

RMSE_Volume_Tsupply_1 <- rmse(TS_Volume_Tsupply_final_1, TS_Volume_Tsupply)
RMSE_Volume_Tsupply_2 <- rmse(TS_Volume_Tsupply_final_2, TS_Volume_Tsupply)
RMSE_Volume_Tsupply_3 <- rmse(TS_Volume_Tsupply_final_3, TS_Volume_Tsupply)
RMSE_Volume_Tsupply_4 <- rmse(TS_Volume_Tsupply_final_4, TS_Volume_Tsupply)

RMSE_Volume_Treturn_1 <- rmse(TS_Volume_Treturn_final_1, TS_Volume_Treturn)
RMSE_Volume_Treturn_2 <- rmse(TS_Volume_Treturn_final_2, TS_Volume_Treturn)
RMSE_Volume_Treturn_3 <- rmse(TS_Volume_Treturn_final_3, TS_Volume_Treturn)
RMSE_Volume_Treturn_4 <- rmse(TS_Volume_Treturn_final_4, TS_Volume_Treturn)

RMSE_Tsupply_1 <- rmse(TS_Tsupply_final_1, TS_Tsupply)
RMSE_Tsupply_2 <- rmse(TS_Tsupply_final_2, TS_Tsupply)
RMSE_Tsupply_3 <- rmse(TS_Tsupply_final_3, TS_Tsupply)
RMSE_Tsupply_4 <- rmse(TS_Tsupply_final_4, TS_Tsupply)

RMSE_Treturn_1 <- rmse(TS_Treturn_final_1, TS_Treturn)
RMSE_Treturn_2 <- rmse(TS_Treturn_final_2, TS_Treturn)
RMSE_Treturn_3 <- rmse(TS_Treturn_final_3, TS_Treturn)
RMSE_Treturn_4 <- rmse(TS_Treturn_final_4, TS_Treturn)

RMSE_Flow_1 <- rmse(TS_Flow_final_1, TS_Flow)
RMSE_Flow_2 <- rmse(TS_Flow_final_2, TS_Flow)
RMSE_Flow_3 <- rmse(TS_Flow_final_3, TS_Flow)
RMSE_Flow_4 <- rmse(TS_Flow_final_4, TS_Flow)


# Mean Absolute Error
MAE_Energy_1 <- mae(TS_Energy_final_1, TS_Energy)
MAE_Energy_2 <- mae(TS_Energy_final_2, TS_Energy)
MAE_Energy_3 <- mae(TS_Energy_final_3, TS_Energy)
MAE_Energy_4 <- mae(TS_Energy_final_4, TS_Energy)

MAE_Volume_1 <- mae(TS_Volume_final_1, TS_Volume)
MAE_Volume_2 <- mae(TS_Volume_final_2, TS_Volume)
MAE_Volume_3 <- mae(TS_Volume_final_3, TS_Volume)
MAE_Volume_4 <- mae(TS_Volume_final_4, TS_Volume)

MAE_Volume_Tsupply_1 <- mae(TS_Volume_Tsupply_final_1, TS_Volume_Tsupply)
MAE_Volume_Tsupply_2 <- mae(TS_Volume_Tsupply_final_2, TS_Volume_Tsupply)
MAE_Volume_Tsupply_3 <- mae(TS_Volume_Tsupply_final_3, TS_Volume_Tsupply)
MAE_Volume_Tsupply_4 <- mae(TS_Volume_Tsupply_final_4, TS_Volume_Tsupply)

MAE_Volume_Treturn_1 <- mae(TS_Volume_Treturn_final_1, TS_Volume_Treturn)
MAE_Volume_Treturn_2 <- mae(TS_Volume_Treturn_final_2, TS_Volume_Treturn)
MAE_Volume_Treturn_3 <- mae(TS_Volume_Treturn_final_3, TS_Volume_Treturn)
MAE_Volume_Treturn_4 <- mae(TS_Volume_Treturn_final_4, TS_Volume_Treturn)

MAE_Tsupply_1 <- mae(TS_Tsupply_final_1, TS_Tsupply)
MAE_Tsupply_2 <- mae(TS_Tsupply_final_2, TS_Tsupply)
MAE_Tsupply_3 <- mae(TS_Tsupply_final_3, TS_Tsupply)
MAE_Tsupply_4 <- mae(TS_Tsupply_final_4, TS_Tsupply)

MAE_Treturn_1 <- mae(TS_Treturn_final_1, TS_Treturn)
MAE_Treturn_2 <- mae(TS_Treturn_final_2, TS_Treturn)
MAE_Treturn_3 <- mae(TS_Treturn_final_3, TS_Treturn)
MAE_Treturn_4 <- mae(TS_Treturn_final_4, TS_Treturn)

MAE_Flow_1 <- mae(TS_Flow_final_1, TS_Flow)
```

```
MAE_Flow_2 <- mae(TS_Flow_final_2, TS_Flow)
MAE_Flow_3 <- mae(TS_Flow_final_3, TS_Flow)
MAE_Flow_4 <- mae(TS_Flow_final_4, TS_Flow)

Results_Energy_1 <- data.frame('Method' = "WMA Linear, k=2", "Variable" = "Energy", "RMSE" =
     RMSE_Energy_1, "MAE" = MAE_Energy_1)
Results_Energy_2 <- data.frame('Method' = "WMA Linear, k=4", "Variable" = "Energy", "RMSE" =
     RMSE_Energy_2, "MAE" = MAE_Energy_2)
Results_Energy_3 <- data.frame('Method' = "WMA Linear, k=6", "Variable" = "Energy", "RMSE" =
     RMSE_Energy_3, "MAE" = MAE_Energy_3)
Results_Energy_4 <- data.frame('Method' = "WMA Linear, k=8", "Variable" = "Energy", "RMSE" =
     RMSE_Energy_4, "MAE" = MAE_Energy_4)

Results_Volume_1 <- data.frame('Method' = "WMA Linear, k=2", "Variable" = "Volume", "RMSE" =
     RMSE_Volume_1, "MAE" = MAE_Volume_1)
Results_Volume_2 <- data.frame('Method' = "WMA Linear, k=4", "Variable" = "Volume", "RMSE" =
     RMSE_Volume_2, "MAE" = MAE_Volume_2)
Results_Volume_3 <- data.frame('Method' = "WMA Linear, k=6", "Variable" = "Volume", "RMSE" =
     RMSE_Volume_3, "MAE" = MAE_Volume_3)
Results_Volume_4 <- data.frame('Method' = "WMA Linear, k=8", "Variable" = "Volume", "RMSE" =
     RMSE_Volume_4, "MAE" = MAE_Volume_4)

Results_Volume_Tsupply_1 <- data.frame('Method' = "WMA Linear, k=2", "Variable" =
     "Volume_Tsupply", "RMSE" = RMSE_Volume_Tsupply_1, "MAE" = MAE_Volume_Tsupply_1)
Results_Volume_Tsupply_2 <- data.frame('Method' = "WMA Linear, k=4", "Variable" =
     "Volume_Tsupply", "RMSE" = RMSE_Volume_Tsupply_2, "MAE" = MAE_Volume_Tsupply_2)
Results_Volume_Tsupply_3 <- data.frame('Method' = "WMA Linear, k=6", "Variable" =
     "Volume_Tsupply", "RMSE" = RMSE_Volume_Tsupply_3, "MAE" = MAE_Volume_Tsupply_3)
Results_Volume_Tsupply_4 <- data.frame('Method' = "WMA Linear, k=8", "Variable" =
     "Volume_Tsupply", "RMSE" = RMSE_Volume_Tsupply_4, "MAE" = MAE_Volume_Tsupply_4)

Results_Volume_Treturn_1 <- data.frame('Method' = "WMA Linear, k=2", "Variable" =
     "Volume_Treturn", "RMSE" = RMSE_Volume_Treturn_1, "MAE" = MAE_Volume_Treturn_1)
Results_Volume_Treturn_2 <- data.frame('Method' = "WMA Linear, k=4", "Variable" =
     "Volume_Treturn", "RMSE" = RMSE_Volume_Treturn_2, "MAE" = MAE_Volume_Treturn_2)
Results_Volume_Treturn_3 <- data.frame('Method' = "WMA Linear, k=6", "Variable" =
     "Volume_Treturn", "RMSE" = RMSE_Volume_Treturn_3, "MAE" = MAE_Volume_Treturn_3)
Results_Volume_Treturn_4 <- data.frame('Method' = "WMA Linear, k=8", "Variable" =
     "Volume_Treturn", "RMSE" = RMSE_Volume_Treturn_4, "MAE" = MAE_Volume_Treturn_4)

Results_Tsupply_1 <- data.frame('Method' = "WMA Linear, k=2", "Variable" = "Tsupply", "RMSE" =
     RMSE_Tsupply_1, "MAE" = MAE_Tsupply_1)
Results_Tsupply_2 <- data.frame('Method' = "WMA Linear, k=4", "Variable" = "Tsupply", "RMSE" =
     RMSE_Tsupply_2, "MAE" = MAE_Tsupply_2)
Results_Tsupply_3 <- data.frame('Method' = "WMA Linear, k=6", "Variable" = "Tsupply", "RMSE" =
     RMSE_Tsupply_3, "MAE" = MAE_Tsupply_3)
Results_Tsupply_4 <- data.frame('Method' = "WMA Linear, k=8", "Variable" = "Tsupply", "RMSE" =
     RMSE_Tsupply_4, "MAE" = MAE_Tsupply_4)

Results_Treturn_1 <- data.frame('Method' = "WMA Linear, k=2", "Variable" = "Treturn", "RMSE" =
     RMSE_Treturn_1, "MAE" = MAE_Treturn_1)
Results_Treturn_2 <- data.frame('Method' = "WMA Linear, k=4", "Variable" = "Treturn", "RMSE" =
     RMSE_Treturn_2, "MAE" = MAE_Treturn_2)
Results_Treturn_3 <- data.frame('Method' = "WMA Linear, k=6", "Variable" = "Treturn", "RMSE" =
     RMSE_Treturn_3, "MAE" = MAE_Treturn_3)
Results_Treturn_4 <- data.frame('Method' = "WMA Linear, k=8", "Variable" = "Treturn", "RMSE" =
     RMSE_Treturn_4, "MAE" = MAE_Treturn_4)

Results_Flow_1 <- data.frame('Method' = "WMA Linear, k=2", "Variable" = "Flow", "RMSE" =
     RMSE_Flow_1, "MAE" = MAE_Flow_1)
Results_Flow_2 <- data.frame('Method' = "WMA Linear, k=4", "Variable" = "Flow", "RMSE" =
     RMSE_Flow_2, "MAE" = MAE_Flow_2)
Results_Flow_3 <- data.frame('Method' = "WMA Linear, k=6", "Variable" = "Flow", "RMSE" =
     RMSE_Flow_3, "MAE" = MAE_Flow_3)
Results_Flow_4 <- data.frame('Method' = "WMA Linear, k=8", "Variable" = "Flow", "RMSE" =
     RMSE_Flow_4, "MAE" = MAE_Flow_4)
```

```

  Results_provisory <- bind_rows(Results_Energy_1, Results_Energy_2, Results_Energy_3,
       Results_Energy_4,
                                 Results_Volume_1, Results_Volume_2, Results_Volume_3,
                                      Results_Volume_4,
                                 Results_Volume_Tsupply_1, Results_Volume_Tsupply_2,
                                      Results_Volume_Tsupply_3, Results_Volume_Tsupply_4,
                                 Results_Volume_Treturn_1, Results_Volume_Treturn_2,
                                      Results_Volume_Treturn_3, Results_Volume_Treturn_4,
                                 Results_Tsupply_1, Results_Tsupply_2, Results_Tsupply_3,
                                      Results_Tsupply_4,
                                 Results_Treturn_1, Results_Treturn_2, Results_Treturn_3,
                                      Results_Treturn_4,
                                 Results_Flow_1, Results_Flow_2, Results_Flow_3, Results_Flow_4)

  Results <- bind_rows(Results, Results_provisory)

## Weighted Moving Average: Exponential ------------------------------

  method_name <- "exponential"

  TS_Energy_final_1 <- na_ma(TS_NA_Energy, k = 2, weighting = method_name)
  TS_Energy_final_2 <- na_ma(TS_NA_Energy, k = 4, weighting = method_name)
  TS_Energy_final_3 <- na_ma(TS_NA_Energy, k = 6, weighting = method_name)
  TS_Energy_final_4 <- na_ma(TS_NA_Energy, k = 8, weighting = method_name)

  TS_Volume_final_1 <- na_ma(TS_NA_Volume, k = 2, weighting = method_name)
  TS_Volume_final_2 <- na_ma(TS_NA_Volume, k = 4, weighting = method_name)
  TS_Volume_final_3 <- na_ma(TS_NA_Volume, k = 6, weighting = method_name)
  TS_Volume_final_4 <- na_ma(TS_NA_Volume, k = 8, weighting = method_name)

  TS_Volume_Tsupply_final_1 <- na_ma(TS_NA_Volume_Tsupply, k = 2, weighting = method_name)
  TS_Volume_Tsupply_final_2 <- na_ma(TS_NA_Volume_Tsupply, k = 4, weighting = method_name)
  TS_Volume_Tsupply_final_3 <- na_ma(TS_NA_Volume_Tsupply, k = 6, weighting = method_name)
  TS_Volume_Tsupply_final_4 <- na_ma(TS_NA_Volume_Tsupply, k = 8, weighting = method_name)

  TS_Volume_Treturn_final_1 <- na_ma(TS_NA_Volume_Treturn, k = 2, weighting = method_name)
  TS_Volume_Treturn_final_2 <- na_ma(TS_NA_Volume_Treturn, k = 4, weighting = method_name)
  TS_Volume_Treturn_final_3 <- na_ma(TS_NA_Volume_Treturn, k = 6, weighting = method_name)
  TS_Volume_Treturn_final_4 <- na_ma(TS_NA_Volume_Treturn, k = 8, weighting = method_name)

  TS_Tsupply_final_1 <- na_ma(TS_NA_Tsupply, k = 2, weighting = method_name)
  TS_Tsupply_final_2 <- na_ma(TS_NA_Tsupply, k = 4, weighting = method_name)
  TS_Tsupply_final_3 <- na_ma(TS_NA_Tsupply, k = 6, weighting = method_name)
  TS_Tsupply_final_4 <- na_ma(TS_NA_Tsupply, k = 8, weighting = method_name)

  TS_Treturn_final_1 <- na_ma(TS_NA_Treturn, k = 2, weighting = method_name)
  TS_Treturn_final_2 <- na_ma(TS_NA_Treturn, k = 4, weighting = method_name)
  TS_Treturn_final_3 <- na_ma(TS_NA_Treturn, k = 6, weighting = method_name)
  TS_Treturn_final_4 <- na_ma(TS_NA_Treturn, k = 8, weighting = method_name)

  TS_Flow_final_1 <- na_ma(TS_NA_Flow, k = 2, weighting = method_name)
  TS_Flow_final_2 <- na_ma(TS_NA_Flow, k = 4, weighting = method_name)
  TS_Flow_final_3 <- na_ma(TS_NA_Flow, k = 6, weighting = method_name)
  TS_Flow_final_4 <- na_ma(TS_NA_Flow, k = 8, weighting = method_name)

  # Root Mean Squared Error
  RMSE_Energy_1 <- rmse(TS_Energy_final_1, TS_Energy)
  RMSE_Energy_2 <- rmse(TS_Energy_final_2, TS_Energy)
  RMSE_Energy_3 <- rmse(TS_Energy_final_3, TS_Energy)
  RMSE_Energy_4 <- rmse(TS_Energy_final_4, TS_Energy)

  RMSE_Volume_1 <- rmse(TS_Volume_final_1, TS_Volume)
  RMSE_Volume_2 <- rmse(TS_Volume_final_2, TS_Volume)
  RMSE_Volume_3 <- rmse(TS_Volume_final_3, TS_Volume)
  RMSE_Volume_4 <- rmse(TS_Volume_final_4, TS_Volume)
```

```r

RMSE_Volume_Tsupply_1 <- rmse(TS_Volume_Tsupply_final_1, TS_Volume_Tsupply)
RMSE_Volume_Tsupply_2 <- rmse(TS_Volume_Tsupply_final_2, TS_Volume_Tsupply)
RMSE_Volume_Tsupply_3 <- rmse(TS_Volume_Tsupply_final_3, TS_Volume_Tsupply)
RMSE_Volume_Tsupply_4 <- rmse(TS_Volume_Tsupply_final_4, TS_Volume_Tsupply)

RMSE_Volume_Treturn_1 <- rmse(TS_Volume_Treturn_final_1, TS_Volume_Treturn)
RMSE_Volume_Treturn_2 <- rmse(TS_Volume_Treturn_final_2, TS_Volume_Treturn)
RMSE_Volume_Treturn_3 <- rmse(TS_Volume_Treturn_final_3, TS_Volume_Treturn)
RMSE_Volume_Treturn_4 <- rmse(TS_Volume_Treturn_final_4, TS_Volume_Treturn)

RMSE_Tsupply_1 <- rmse(TS_Tsupply_final_1, TS_Tsupply)
RMSE_Tsupply_2 <- rmse(TS_Tsupply_final_2, TS_Tsupply)
RMSE_Tsupply_3 <- rmse(TS_Tsupply_final_3, TS_Tsupply)
RMSE_Tsupply_4 <- rmse(TS_Tsupply_final_4, TS_Tsupply)

RMSE_Treturn_1 <- rmse(TS_Treturn_final_1, TS_Treturn)
RMSE_Treturn_2 <- rmse(TS_Treturn_final_2, TS_Treturn)
RMSE_Treturn_3 <- rmse(TS_Treturn_final_3, TS_Treturn)
RMSE_Treturn_4 <- rmse(TS_Treturn_final_4, TS_Treturn)

RMSE_Flow_1 <- rmse(TS_Flow_final_1, TS_Flow)
RMSE_Flow_2 <- rmse(TS_Flow_final_2, TS_Flow)
RMSE_Flow_3 <- rmse(TS_Flow_final_3, TS_Flow)
RMSE_Flow_4 <- rmse(TS_Flow_final_4, TS_Flow)

# Mean Absolute Error
MAE_Energy_1 <- mae(TS_Energy_final_1, TS_Energy)
MAE_Energy_2 <- mae(TS_Energy_final_2, TS_Energy)
MAE_Energy_3 <- mae(TS_Energy_final_3, TS_Energy)
MAE_Energy_4 <- mae(TS_Energy_final_4, TS_Energy)

MAE_Volume_1 <- mae(TS_Volume_final_1, TS_Volume)
MAE_Volume_2 <- mae(TS_Volume_final_2, TS_Volume)
MAE_Volume_3 <- mae(TS_Volume_final_3, TS_Volume)
MAE_Volume_4 <- mae(TS_Volume_final_4, TS_Volume)

MAE_Volume_Tsupply_1 <- mae(TS_Volume_Tsupply_final_1, TS_Volume_Tsupply)
MAE_Volume_Tsupply_2 <- mae(TS_Volume_Tsupply_final_2, TS_Volume_Tsupply)
MAE_Volume_Tsupply_3 <- mae(TS_Volume_Tsupply_final_3, TS_Volume_Tsupply)
MAE_Volume_Tsupply_4 <- mae(TS_Volume_Tsupply_final_4, TS_Volume_Tsupply)

MAE_Volume_Treturn_1 <- mae(TS_Volume_Treturn_final_1, TS_Volume_Treturn)
MAE_Volume_Treturn_2 <- mae(TS_Volume_Treturn_final_2, TS_Volume_Treturn)
MAE_Volume_Treturn_3 <- mae(TS_Volume_Treturn_final_3, TS_Volume_Treturn)
MAE_Volume_Treturn_4 <- mae(TS_Volume_Treturn_final_4, TS_Volume_Treturn)

MAE_Tsupply_1 <- mae(TS_Tsupply_final_1, TS_Tsupply)
MAE_Tsupply_2 <- mae(TS_Tsupply_final_2, TS_Tsupply)
MAE_Tsupply_3 <- mae(TS_Tsupply_final_3, TS_Tsupply)
MAE_Tsupply_4 <- mae(TS_Tsupply_final_4, TS_Tsupply)

MAE_Treturn_1 <- mae(TS_Treturn_final_1, TS_Treturn)
MAE_Treturn_2 <- mae(TS_Treturn_final_2, TS_Treturn)
MAE_Treturn_3 <- mae(TS_Treturn_final_3, TS_Treturn)
MAE_Treturn_4 <- mae(TS_Treturn_final_4, TS_Treturn)

MAE_Flow_1 <- mae(TS_Flow_final_1, TS_Flow)
MAE_Flow_2 <- mae(TS_Flow_final_2, TS_Flow)
MAE_Flow_3 <- mae(TS_Flow_final_3, TS_Flow)
MAE_Flow_4 <- mae(TS_Flow_final_4, TS_Flow)

Results_Energy_1 <- data.frame('Method' = "WMA Exponential, k=2", "Variable" = "Energy", "RMSE"
    = RMSE_Energy_1, "MAE" = MAE_Energy_1)
Results_Energy_2 <- data.frame('Method' = "WMA Exponential, k=4", "Variable" = "Energy", "RMSE"
    = RMSE_Energy_2, "MAE" = MAE_Energy_2)
```

```
Results_Energy_3 <- data.frame('Method' = "WMA Exponential, k=6", "Variable" = "Energy", "RMSE"
     = RMSE_Energy_3, "MAE" = MAE_Energy_3)
Results_Energy_4 <- data.frame('Method' = "WMA Exponential, k=8", "Variable" = "Energy", "RMSE"
     = RMSE_Energy_4, "MAE" = MAE_Energy_4)

Results_Volume_1 <- data.frame('Method' = "WMA Exponential, k=2", "Variable" = "Volume", "RMSE"
     = RMSE_Volume_1, "MAE" = MAE_Volume_1)
Results_Volume_2 <- data.frame('Method' = "WMA Exponential, k=4", "Variable" = "Volume", "RMSE"
     = RMSE_Volume_2, "MAE" = MAE_Volume_2)
Results_Volume_3 <- data.frame('Method' = "WMA Exponential, k=6", "Variable" = "Volume", "RMSE"
     = RMSE_Volume_3, "MAE" = MAE_Volume_3)
Results_Volume_4 <- data.frame('Method' = "WMA Exponential, k=8", "Variable" = "Volume", "RMSE"
     = RMSE_Volume_4, "MAE" = MAE_Volume_4)

Results_Volume_Tsupply_1 <- data.frame('Method' = "WMA Exponential, k=2", "Variable" =
     "Volume_Tsupply", "RMSE" = RMSE_Volume_Tsupply_1, "MAE" = MAE_Volume_Tsupply_1)
Results_Volume_Tsupply_2 <- data.frame('Method' = "WMA Exponential, k=4", "Variable" =
     "Volume_Tsupply", "RMSE" = RMSE_Volume_Tsupply_2, "MAE" = MAE_Volume_Tsupply_2)
Results_Volume_Tsupply_3 <- data.frame('Method' = "WMA Exponential, k=6", "Variable" =
     "Volume_Tsupply", "RMSE" = RMSE_Volume_Tsupply_3, "MAE" = MAE_Volume_Tsupply_3)
Results_Volume_Tsupply_4 <- data.frame('Method' = "WMA Exponential, k=8", "Variable" =
     "Volume_Tsupply", "RMSE" = RMSE_Volume_Tsupply_4, "MAE" = MAE_Volume_Tsupply_4)

Results_Volume_Treturn_1 <- data.frame('Method' = "WMA Exponential, k=2", "Variable" =
     "Volume_Treturn", "RMSE" = RMSE_Volume_Treturn_1, "MAE" = MAE_Volume_Treturn_1)
Results_Volume_Treturn_2 <- data.frame('Method' = "WMA Exponential, k=4", "Variable" =
     "Volume_Treturn", "RMSE" = RMSE_Volume_Treturn_2, "MAE" = MAE_Volume_Treturn_2)
Results_Volume_Treturn_3 <- data.frame('Method' = "WMA Exponential, k=6", "Variable" =
     "Volume_Treturn", "RMSE" = RMSE_Volume_Treturn_3, "MAE" = MAE_Volume_Treturn_3)
Results_Volume_Treturn_4 <- data.frame('Method' = "WMA Exponential, k=8", "Variable" =
     "Volume_Treturn", "RMSE" = RMSE_Volume_Treturn_4, "MAE" = MAE_Volume_Treturn_4)

Results_Tsupply_1 <- data.frame('Method' = "WMA Exponential, k=2", "Variable" = "Tsupply",
     "RMSE" = RMSE_Tsupply_1, "MAE" = MAE_Tsupply_1)
Results_Tsupply_2 <- data.frame('Method' = "WMA Exponential, k=4", "Variable" = "Tsupply",
     "RMSE" = RMSE_Tsupply_2, "MAE" = MAE_Tsupply_2)
Results_Tsupply_3 <- data.frame('Method' = "WMA Exponential, k=6", "Variable" = "Tsupply",
     "RMSE" = RMSE_Tsupply_3, "MAE" = MAE_Tsupply_3)
Results_Tsupply_4 <- data.frame('Method' = "WMA Exponential, k=8", "Variable" = "Tsupply",
     "RMSE" = RMSE_Tsupply_4, "MAE" = MAE_Tsupply_4)

Results_Treturn_1 <- data.frame('Method' = "WMA Exponential, k=2", "Variable" = "Treturn",
     "RMSE" = RMSE_Treturn_1, "MAE" = MAE_Treturn_1)
Results_Treturn_2 <- data.frame('Method' = "WMA Exponential, k=4", "Variable" = "Treturn",
     "RMSE" = RMSE_Treturn_2, "MAE" = MAE_Treturn_2)
Results_Treturn_3 <- data.frame('Method' = "WMA Exponential, k=6", "Variable" = "Treturn",
     "RMSE" = RMSE_Treturn_3, "MAE" = MAE_Treturn_3)
Results_Treturn_4 <- data.frame('Method' = "WMA Exponential, k=8", "Variable" = "Treturn",
     "RMSE" = RMSE_Treturn_4, "MAE" = MAE_Treturn_4)

Results_Flow_1 <- data.frame('Method' = "WMA Exponential, k=2", "Variable" = "Flow", "RMSE" =
     RMSE_Flow_1, "MAE" = MAE_Flow_1)
Results_Flow_2 <- data.frame('Method' = "WMA Exponential, k=4", "Variable" = "Flow", "RMSE" =
     RMSE_Flow_2, "MAE" = MAE_Flow_2)
Results_Flow_3 <- data.frame('Method' = "WMA Exponential, k=6", "Variable" = "Flow", "RMSE" =
     RMSE_Flow_3, "MAE" = MAE_Flow_3)
Results_Flow_4 <- data.frame('Method' = "WMA Exponential, k=8", "Variable" = "Flow", "RMSE" =
     RMSE_Flow_4, "MAE" = MAE_Flow_4)

Results_provisory <- bind_rows(Results_Energy_1, Results_Energy_2, Results_Energy_3,
     Results_Energy_4,
                               Results_Volume_1, Results_Volume_2, Results_Volume_3,
                                    Results_Volume_4,
                               Results_Volume_Tsupply_1, Results_Volume_Tsupply_2,
                                    Results_Volume_Tsupply_3, Results_Volume_Tsupply_4,
                               Results_Volume_Treturn_1, Results_Volume_Treturn_2,
                                     Results_Volume_Treturn_3, Results_Volume_Treturn_4,
                                Results_Tsupply_1, Results_Tsupply_2, Results_Tsupply_3,
                                     Results_Tsupply_4,
                                Results_Treturn_1, Results_Treturn_2, Results_Treturn_3,
                                     Results_Treturn_4,
                                Results_Flow_1, Results_Flow_2, Results_Flow_3, Results_Flow_4)

  Results <- bind_rows(Results, Results_provisory)

## Mean -------------------------------

  method_name <- "mean"

  TS_Energy_final <- na_mean(TS_NA_Energy, option = method_name)
  TS_Volume_final <- na_mean(TS_NA_Volume, option = method_name)
  TS_Volume_Tsupply_final <- na_mean(TS_NA_Volume_Tsupply, option = method_name)
  TS_Volume_Treturn_final <- na_mean(TS_NA_Volume_Treturn, option = method_name)
  TS_Tsupply_final <- na_mean(TS_NA_Tsupply, option = method_name)
  TS_Treturn_final <- na_mean(TS_NA_Treturn, option = method_name)
  TS_Flow_final <- na_mean(TS_NA_Flow, option = method_name)

  # Root Mean Squared Error
  RMSE_Energy <- rmse(TS_Energy_final, TS_Energy)
  RMSE_Volume <- rmse(TS_Volume_final, TS_Volume)
  RMSE_Volume_Tsupply <- rmse(TS_Volume_Tsupply_final, TS_Volume_Tsupply)
  RMSE_Volume_Treturn <- rmse(TS_Volume_Treturn_final, TS_Volume_Treturn)
  RMSE_Tsupply <- rmse(TS_Tsupply_final, TS_Tsupply)
  RMSE_Treturn <- rmse(TS_Treturn_final, TS_Treturn)
  RMSE_Flow <- rmse(TS_Flow_final, TS_Flow)

  # Mean Absolute Error
  MAE_Energy <- mae(TS_Energy_final, TS_Energy)
  MAE_Volume <- mae(TS_Volume_final, TS_Volume)
  MAE_Volume_Tsupply <- mae(TS_Volume_Tsupply_final, TS_Volume_Tsupply)
  MAE_Volume_Treturn <- mae(TS_Volume_Treturn_final, TS_Volume_Treturn)
  MAE_Tsupply <- mae(TS_Tsupply_final, TS_Tsupply)
  MAE_Treturn <- mae(TS_Treturn_final, TS_Treturn)
  MAE_Flow <- mae(TS_Flow_final, TS_Flow)

  Results_Energy <- data.frame('Method' = "Mean", "Variable" = "Energy", "RMSE" = RMSE_Energy,
       "MAE" = MAE_Energy)
  Results_Volume <- data.frame('Method' = "Mean", "Variable" = "Volume", "RMSE" = RMSE_Volume,
       "MAE" = MAE_Volume)
  Results_Volume_Tsupply <- data.frame('Method' = "Mean", "Variable" = "Volume_Tsupply", "RMSE" =
       RMSE_Volume_Tsupply, "MAE" = MAE_Volume_Tsupply)
  Results_Volume_Treturn <- data.frame('Method' = "Mean", "Variable" = "Volume_Treturn", "RMSE" =
       RMSE_Volume_Treturn, "MAE" = MAE_Volume_Treturn)
  Results_Tsupply <- data.frame('Method' = "Mean", "Variable" = "Tsupply", "RMSE" = RMSE_Tsupply,
       "MAE" = MAE_Tsupply)
  Results_Treturn <- data.frame('Method' = "Mean", "Variable" = "Treturn", "RMSE" = RMSE_Treturn,
       "MAE" = MAE_Treturn)
  Results_Flow <- data.frame('Method' = "Mean", "Variable" = "Flow", "RMSE" = RMSE_Flow, "MAE" =
       MAE_Flow)

  Results_provisory <- bind_rows(Results_Energy, Results_Volume, Results_Volume_Tsupply,
       Results_Volume_Treturn, Results_Tsupply, Results_Treturn, Results_Flow)

  Results <- bind_rows(Results, Results_provisory)

## Median -------------------------------

  method_name <- "median"

  TS_Energy_final <- na_mean(TS_NA_Energy, option = method_name)
  TS_Volume_final <- na_mean(TS_NA_Volume, option = method_name)
  TS_Volume_Tsupply_final <- na_mean(TS_NA_Volume_Tsupply, option = method_name)
```

```
    TS_Volume_Treturn_final <- na_mean(TS_NA_Volume_Treturn, option = method_name)
    TS_Tsupply_final <- na_mean(TS_NA_Tsupply, option = method_name)
    TS_Treturn_final <- na_mean(TS_NA_Treturn, option = method_name)
    TS_Flow_final <- na_mean(TS_NA_Flow, option = method_name)

    # Root Mean Squared Error
    RMSE_Energy <- rmse(TS_Energy_final, TS_Energy)
    RMSE_Volume <- rmse(TS_Volume_final, TS_Volume)
    RMSE_Volume_Tsupply <- rmse(TS_Volume_Tsupply_final, TS_Volume_Tsupply)
    RMSE_Volume_Treturn <- rmse(TS_Volume_Treturn_final, TS_Volume_Treturn)
    RMSE_Tsupply <- rmse(TS_Tsupply_final, TS_Tsupply)
    RMSE_Treturn <- rmse(TS_Treturn_final, TS_Treturn)
    RMSE_Flow <- rmse(TS_Flow_final, TS_Flow)

    # Mean Absolute Error
    MAE_Energy <- mae(TS_Energy_final, TS_Energy)
    MAE_Volume <- mae(TS_Volume_final, TS_Volume)
    MAE_Volume_Tsupply <- mae(TS_Volume_Tsupply_final, TS_Volume_Tsupply)
    MAE_Volume_Treturn <- mae(TS_Volume_Treturn_final, TS_Volume_Treturn)
    MAE_Tsupply <- mae(TS_Tsupply_final, TS_Tsupply)
    MAE_Treturn <- mae(TS_Treturn_final, TS_Treturn)
    MAE_Flow <- mae(TS_Flow_final, TS_Flow)

    Results_Energy <- data.frame('Method' = "Median", "Variable" = "Energy", "RMSE" = RMSE_Energy,
        "MAE" = MAE_Energy)
    Results_Volume <- data.frame('Method' = "Median", "Variable" = "Volume", "RMSE" = RMSE_Volume,
        "MAE" = MAE_Volume)
    Results_Volume_Tsupply <- data.frame('Method' = "Median", "Variable" = "Volume_Tsupply", "RMSE"
        = RMSE_Volume_Tsupply, "MAE" = MAE_Volume_Tsupply)
    Results_Volume_Treturn <- data.frame('Method' = "Median", "Variable" = "Volume_Treturn", "RMSE"
        = RMSE_Volume_Treturn, "MAE" = MAE_Volume_Treturn)
    Results_Tsupply <- data.frame('Method' = "Median", "Variable" = "Tsupply", "RMSE" =
        RMSE_Tsupply, "MAE" = MAE_Tsupply)
    Results_Treturn <- data.frame('Method' = "Median", "Variable" = "Treturn", "RMSE" =
        RMSE_Treturn, "MAE" = MAE_Treturn)
    Results_Flow <- data.frame('Method' = "Median", "Variable" = "Flow", "RMSE" = RMSE_Flow, "MAE" =
        MAE_Flow)

    Results_provisory <- bind_rows(Results_Energy, Results_Volume, Results_Volume_Tsupply,
        Results_Volume_Treturn, Results_Tsupply, Results_Treturn, Results_Flow)

    Results <- bind_rows(Results, Results_provisory)

## Mode -------------------------------

    method_name <- "mode"

    TS_Energy_final <- na_mean(TS_NA_Energy, option = method_name)
    TS_Volume_final <- na_mean(TS_NA_Volume, option = method_name)
    TS_Volume_Tsupply_final <- na_mean(TS_NA_Volume_Tsupply, option = method_name)
    TS_Volume_Treturn_final <- na_mean(TS_NA_Volume_Treturn, option = method_name)
    TS_Tsupply_final <- na_mean(TS_NA_Tsupply, option = method_name)
    TS_Treturn_final <- na_mean(TS_NA_Treturn, option = method_name)
    TS_Flow_final <- na_mean(TS_NA_Flow, option = method_name)

    # Root Mean Squared Error
    RMSE_Energy <- rmse(TS_Energy_final, TS_Energy)
    RMSE_Volume <- rmse(TS_Volume_final, TS_Volume)
    RMSE_Volume_Tsupply <- rmse(TS_Volume_Tsupply_final, TS_Volume_Tsupply)
    RMSE_Volume_Treturn <- rmse(TS_Volume_Treturn_final, TS_Volume_Treturn)
    RMSE_Tsupply <- rmse(TS_Tsupply_final, TS_Tsupply)
    RMSE_Treturn <- rmse(TS_Treturn_final, TS_Treturn)
    RMSE_Flow <- rmse(TS_Flow_final, TS_Flow)

    # Mean Absolute Error
    MAE_Energy <- mae(TS_Energy_final, TS_Energy)
```

```
  MAE_Volume <- mae(TS_Volume_final, TS_Volume)
  MAE_Volume_Tsupply <- mae(TS_Volume_Tsupply_final, TS_Volume_Tsupply)
  MAE_Volume_Treturn <- mae(TS_Volume_Treturn_final, TS_Volume_Treturn)
  MAE_Tsupply <- mae(TS_Tsupply_final, TS_Tsupply)
  MAE_Treturn <- mae(TS_Treturn_final, TS_Treturn)
  MAE_Flow <- mae(TS_Flow_final, TS_Flow)

  Results_Energy <- data.frame('Method' = "Mode", "Variable" = "Energy", "RMSE" = RMSE_Energy,
      "MAE" = MAE_Energy)
  Results_Volume <- data.frame('Method' = "Mode", "Variable" = "Volume", "RMSE" = RMSE_Volume,
      "MAE" = MAE_Volume)
  Results_Volume_Tsupply <- data.frame('Method' = "Mode", "Variable" = "Volume_Tsupply", "RMSE" =
      RMSE_Volume_Tsupply, "MAE" = MAE_Volume_Tsupply)
  Results_Volume_Treturn <- data.frame('Method' = "Mode", "Variable" = "Volume_Treturn", "RMSE" =
      RMSE_Volume_Treturn, "MAE" = MAE_Volume_Treturn)
  Results_Tsupply <- data.frame('Method' = "Mode", "Variable" = "Tsupply", "RMSE" = RMSE_Tsupply,
      "MAE" = MAE_Tsupply)
  Results_Treturn <- data.frame('Method' = "Mode", "Variable" = "Treturn", "RMSE" = RMSE_Treturn,
      "MAE" = MAE_Treturn)
  Results_Flow <- data.frame('Method' = "Mode", "Variable" = "Flow", "RMSE" = RMSE_Flow, "MAE" =
      MAE_Flow)

  Results_provisory <- bind_rows(Results_Energy, Results_Volume, Results_Volume_Tsupply,
      Results_Volume_Treturn, Results_Tsupply, Results_Treturn, Results_Flow)

  Results <- bind_rows(Results, Results_provisory)

## Random ------------------------------

  TS_Energy_final <- na_random(TS_NA_Energy, lower_bound = NULL, upper_bound = NULL)
  TS_Volume_final <- na_random(TS_NA_Volume, lower_bound = NULL, upper_bound = NULL)
  TS_Volume_Tsupply_final <- na_random(TS_NA_Volume_Tsupply, lower_bound = NULL, upper_bound =
      NULL)
  TS_Volume_Treturn_final <- na_random(TS_NA_Volume_Treturn, lower_bound = NULL, upper_bound =
      NULL)
  TS_Tsupply_final <- na_random(TS_NA_Tsupply, lower_bound = NULL, upper_bound = NULL)
  TS_Treturn_final <- na_random(TS_NA_Treturn, lower_bound = NULL, upper_bound = NULL)
  TS_Flow_final <- na_random(TS_NA_Flow, lower_bound = NULL, upper_bound = NULL)

  # Root Mean Squared Error
  RMSE_Energy <- rmse(TS_Energy_final, TS_Energy)
  RMSE_Volume <- rmse(TS_Volume_final, TS_Volume)
  RMSE_Volume_Tsupply <- rmse(TS_Volume_Tsupply_final, TS_Volume_Tsupply)
  RMSE_Volume_Treturn <- rmse(TS_Volume_Treturn_final, TS_Volume_Treturn)
  RMSE_Tsupply <- rmse(TS_Tsupply_final, TS_Tsupply)
  RMSE_Treturn <- rmse(TS_Treturn_final, TS_Treturn)
  RMSE_Flow <- rmse(TS_Flow_final, TS_Flow)

  # Mean Absolute Error
  MAE_Energy <- mae(TS_Energy_final, TS_Energy)
  MAE_Volume <- mae(TS_Volume_final, TS_Volume)
  MAE_Volume_Tsupply <- mae(TS_Volume_Tsupply_final, TS_Volume_Tsupply)
  MAE_Volume_Treturn <- mae(TS_Volume_Treturn_final, TS_Volume_Treturn)
  MAE_Tsupply <- mae(TS_Tsupply_final, TS_Tsupply)
  MAE_Treturn <- mae(TS_Treturn_final, TS_Treturn)
  MAE_Flow <- mae(TS_Flow_final, TS_Flow)

  Results_Energy <- data.frame('Method' = "Random", "Variable" = "Energy", "RMSE" = RMSE_Energy,
      "MAE" = MAE_Energy)
  Results_Volume <- data.frame('Method' = "Random", "Variable" = "Volume", "RMSE" = RMSE_Volume,
      "MAE" = MAE_Volume)
  Results_Volume_Tsupply <- data.frame('Method' = "Random", "Variable" = "Volume_Tsupply", "RMSE"
      = RMSE_Volume_Tsupply, "MAE" = MAE_Volume_Tsupply)
  Results_Volume_Treturn <- data.frame('Method' = "Random", "Variable" = "Volume_Treturn", "RMSE"
      = RMSE_Volume_Treturn, "MAE" = MAE_Volume_Treturn)
  Results_Tsupply <- data.frame('Method' = "Random", "Variable" = "Tsupply", "RMSE" =
      RMSE_Tsupply, "MAE" = MAE_Tsupply)
  Results_Treturn <- data.frame('Method' = "Random", "Variable" = "Treturn", "RMSE" =
      RMSE_Treturn, "MAE" = MAE_Treturn)
  Results_Flow <- data.frame('Method' = "Random", "Variable" = "Flow", "RMSE" = RMSE_Flow, "MAE" =
      MAE_Flow)

  Results_provisory <- bind_rows(Results_Energy, Results_Volume, Results_Volume_Tsupply,
      Results_Volume_Treturn, Results_Tsupply, Results_Treturn, Results_Flow)

  Results <- bind_rows(Results, Results_provisory)

### Final tables per variable ---------------------------------

  BMark_Energy <- Results %>%
    filter(Variable == "Energy")

  BMark_Volume <- Results %>%
    filter(Variable == "Volume")

  BMark_Volume_Tsupply <- Results %>%
    filter(Variable == "Volume_Tsupply")

  BMark_Volume_Treturn <- Results %>%
    filter(Variable == "Volume_Treturn")

  BMark_Tsupply <- Results %>%
    filter(Variable == "Tsupply")

  BMark_Treturn <- Results %>%
    filter(Variable == "Treturn")

  BMark_Flow <- Results %>%
    filter(Variable == "Flow")


### Convert Time Series into Dataframe -------------------------------

  z_new_Energy <- na_interpolation(TS_NA_Energy, option = "linear")
  z_new_Volume <- na_interpolation(TS_NA_Volume, option = "linear")
  z_new_Volume_Tsupply <- na_interpolation(TS_NA_Volume_Tsupply, option = "linear")
  z_new_Volume_Treturn <- na_interpolation(TS_NA_Volume_Treturn, option = "linear")
  z_new_Tsupply <- na_ma(TS_NA_Tsupply, k = 8, weighting = "exponential")
  z_new_Treturn <- na_ma(TS_NA_Treturn, k = 8, weighting = "exponential")
  z_new_Flow <- na_ma(TS_NA_Flow, k = 8, weighting = "exponential")

  z$Energy <- as.numeric(z_new_Energy)
  z$Volume_accum <- as.numeric(z_new_Volume)
  z$Volume_TS <- as.numeric(z_new_Volume_Tsupply)
  z$Volume_TR <- as.numeric(z_new_Volume_Treturn)
  z$T_supply <- as.numeric(z_new_Tsupply)
  z$T_return <- as.numeric(z_new_Treturn)
  z$Flow <- as.numeric(z_new_Flow)
```

Listing M.1. Script Extra: Imputation Benchmark