README

1 User's guide to the README

This document explains the smart meter data, how to use it and which data processing that was done to create the Clean Cooking Data Release Report.

Three (3) dataframes are available for download:

1. dataframe_raw.csv
    - Raw data with outliers[1] removed.
2. dataframe_info.csv
    - General information about the cooking pilot, e.g. meter location and tariff structure.
3. dataframe_processed.csv
    - Dataframe with numbering of cooking events

2 What does the data look like?

2.1 dataframe_raw.csv

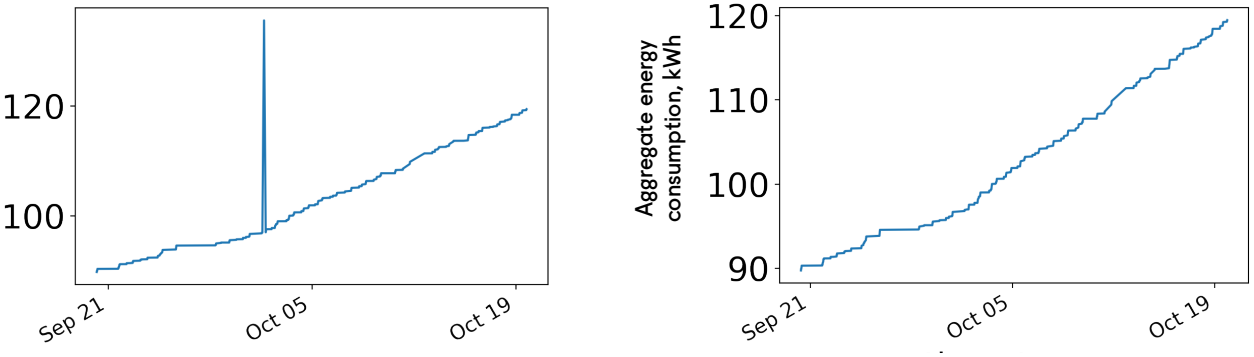
The households were instructed to turn on the smart meters before they started to cook with the EPC and to turn them off when they were finished with their cooking activities. This meant that most of the meters were turned off when the households weren't cooking. The raw data is presented in CSV-format with a recording frequency of 5 minutes.

COLUMN NAME	VALUE	DESCRIPTION
meter_number	ID	unique identifier of smart meter in household
timezone	UTC+HH:MM	timezone of timestamp
timestamp	YYYY-MM-DD HH:MM:SS	timestamp at time of measurement
energy	Kilowatt hour, kWh	cumulative energy used by appliance
voltage	Volt, V	active voltage at time of measurement
current	Ampere, A	active current at time of measurement
power	Kilowatt, kW	active real power at time of measurement
power_factor	Ratio, -	apparent power / real power
frequency	Herz, Hz	frequency at time of measurement

[1] corrupt data due to SIM-card issues and grid outages, see section 2.1.1 for more

2.1.1 Excluding Outliers

We have excluded specific outliers from the raw data. These outliers are corrupt data, which appears in the energy column as big data spikes due to SIM-card issues, modem changes and grid outages. Specifically, all outliers that are resulting in a spike above 1 kWh are corrected.



Sept 20 to Oct 20. meter number: 546375

2.3 dataframe_info.csv

This file contains a list of general information about each smart meter. This includes the date of installation, where the meters are located, electricity tariff details and dates of the field trips.

The field trips were conducted to make sure that all stored data is sent to the server. The underlying reason is that the meters are installed in areas with bad networks and are, therefore, often unable to send real-time data.

Column 'tariff_structure_before' contains details about the tariff structure before the tariff reduction. These were not part of the report. However, we have decided to include them here to enable the analysis of possible tariff structure effects. The block tariff is referring to a structure where the households got a reduced electricity price after reaching a certain consumption threshold. The discount tariff signified a relatively low electricity price, while the flat tariff signified a relatively high electricity price.

COLUMN NAME	VALUE	DESCRIPTION
meter number	ID	unique identifier of smart meter
connection_date	YYYY-MM-DD (UTC)	date of meter installation in household
first_online	YYYY-MM-DD (UTC)	date meter first connected and send data to server
location	mainland OR island	the location of the household
site	ID	unique identifier of the site the meter is on (1-6)
tariff_structure_before	block OR discount OR normal	tariff structure of household before tariff change

tariff_after	TZS/kWh	kWh price after tariff change
date_tariff_change	YYYY-MM-DD (UTC)	date of tariff change
date_field_trip_one_start	YYYY-MM-DD (UTC)	date for taking meter to good network area
date_field_trip_one_end	YYYY-MM-DD (UTC)	date for taking meter to good network area
date_field trip_two	YYYY-MM-DD (UTC)	date for taking meter to good network area, max 24 hr
date_tariff_change	YYYY-MM-DD (UTC)	date of tariff change

2.4 dataframe_processed

This is the processed dataframe, which has these additional columns compared to the raw dataframe:

COLUMN NAME	VALUE	DESCRIPTION
cooking_event_id	ID	numbering of cooking event
cooking_time	Minutes, min	total cooking time
cooking_seq_time	Minutes, min	specific time inside a cooking event
location	island OR mainland	location of household
site	ID	place where the household is located
date_tariff_change	YYYY-MM-DD (UTC)	date of tariff change

3 What processing steps have been done?

3.1 Code Language

We used Python with Pandas for the data processing. Therefore, all code examples are given in Python syntax.

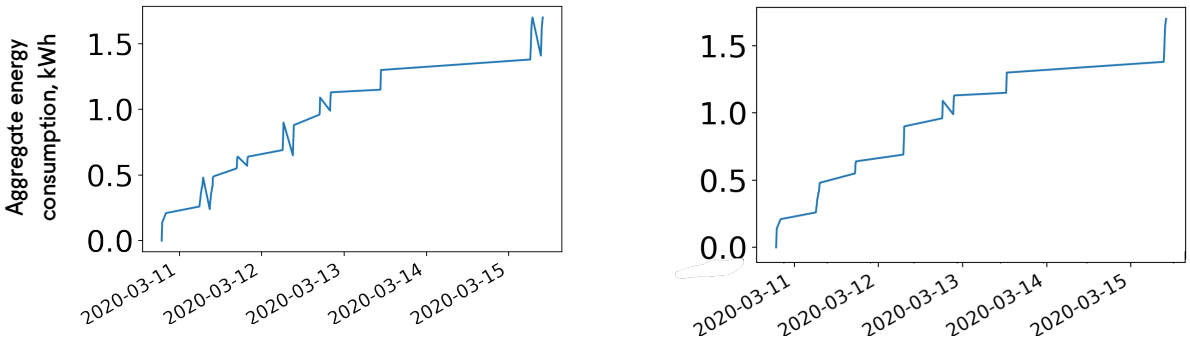
3.2 Defining the Cooking Event

The cooking event is defined as a consistent recording sequence by monitoring the energy consumption and power level. Several conditions are put up to make the cooking event definitions as precise as possible (see attachments). The time between a recording of energy consumption and the end of a cooking session is set to just above 15 minutes, which is equal to 3 recordings at 5 minute recording frequency.

3.3 Handling Cooking Event Duplicates

During the pilot, we have been developing a backfilling functionality. It allows us to send stored data to the server when a meter comes to an area with a good network. Unfortunately, some "baby diseases" appeared at the beginning of using this feature. One such issue was that cooking events appeared twice, because smart meters sent data before they had been synchronized to the timezone settings of the server. Fortunately, this issue is now solved and, thus, only appears in the first half of the pilot.

The processed dataframe is excluding cooking events duplicates. We are, furthermore, expecting that the backfilling functionality to reduce data gaps will become significantly better until the final data release at the end of the pilot.



March 9 to March 17

Attachments - Cooking event algorithm

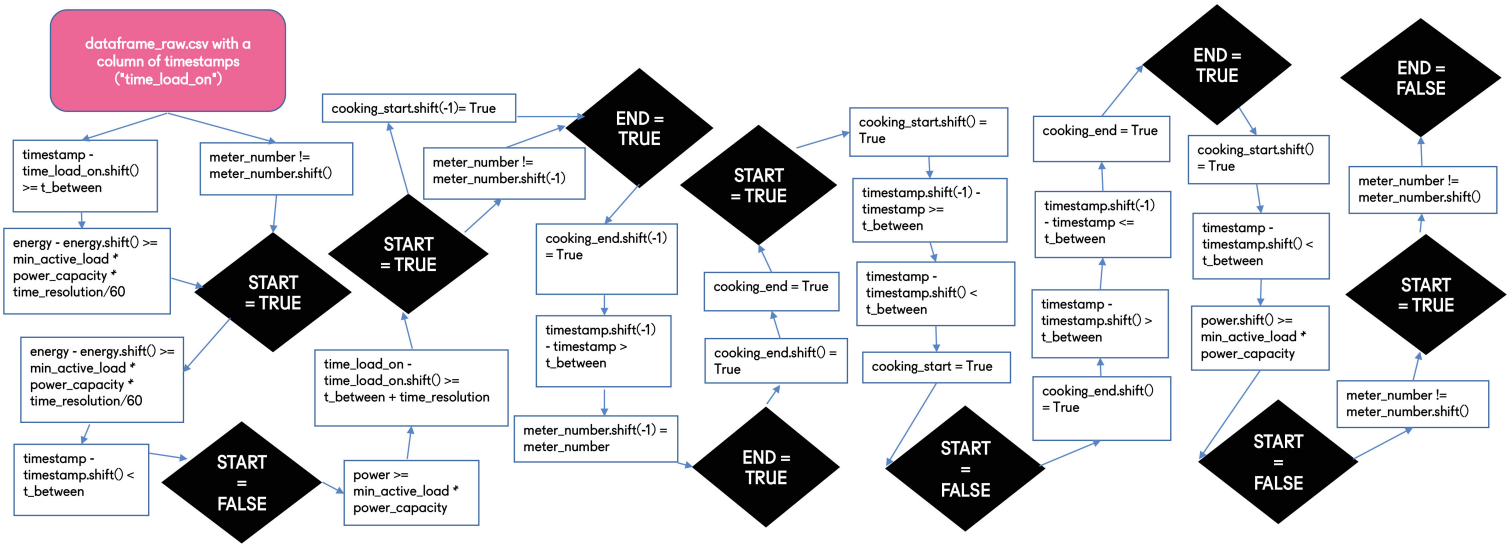
As mentioned before, Python with Pandas was used for data processing. Below is a description of the steps that were taken to define the cooking events:

1) Defining new parameters[2]:
   power = current*voltage
   min_active_load = 15%
   power_capacity = 1 kW
   time_resolution = 5 minutes
   t_between = 15.01 minutes

2) A column called 'load_on' is created to indicate when the EPC is active, i.e. when a power load is applied. Conditions when load_on is TRUE:
   a. energy - energy.shift() > min_active_load * power_capacity * time_resolution/60
   b. power > min_active_load * power_capacity
   c. meter_number = meter_number.shift()

3) Discard cooking events that are both short and consume little energy, i.e. when all these conditions are TRUE:
   a. cooking_event != cooking_event.shift()
   b. cooking_event != cooking_event.shift(-1)
   c. energy - energy.shift() < 0.1

4) A summary of the start and end conditions of the cooking events are found in the illustration below.



[2] The power is calculated again to avoid using mean val