Fuel Data Coding Strategy

Residential wood heating stove loading events measured with fuel consumption sensors

Kiernan Kilkenny[a], Nordica MacCarty[a,b,c]

[a]School of Mechanical, Industrial, and Manufacturing Engineering, 204 Rogers Hall 2000 SW Monroe Avenue Oregon State University, Corvallis, OR, 97331
[b]Aprovecho Research Center, 76132 Blue Mountain School Rd., Cottage Grove, OR 97424
[c]Corresponding author, nordica.maccarty@oregonstate.edu, 541-737-5927

Related Papers

K. Kilkenny, S. Bentson, J. Berger, S. Zhang, N. MacCarty, Integrated Assessment of Residential Wood Heating Practices and Implications for Design and Performance, in review. (2024).

S. Bentson, R. Thompson, J. Berger, J. Wald, N. MacCarty, In-Situ Measurements of Emissions and Fuel Loading of Non-Catalytic Cordwood Stoves in Rural Oregon, in preparation, (2024).

1. Introduction and Goals

User behaviors that can impact stove performance include loading size, frequency, and type and quality of fuels used, as well as whether the stove is operated according to the manufacturer's instructions. In our study we used the Fuel Use Electronic Logger (FUEL) and Extremely Accessible Cookstove Tracker (EXACT) sensors to measure wood heating stove fuel consumption. These sensors are available from Climate Solutions Consulting, but any logging load cell and temperature sensor could be used in their place. The FUEL system was developed at Oregon State University to monitor the impact of improved cookstove projects in low- and middle-income countries, and this marks its first use to measure wood heater fuel consumption [1,2]. With the FUEL sensor, we are now able to measure load sizes and the frequency of loading events without having to rely on direct observation or participant journals. By using the FUEL sensor in conjunction with the EXACT sensor, we are also able to verify that fuel removed from the sensor was placed into the stove. Additionally, knowing the mass of fuel burned in the stove during emissions measurements makes it much easier to perform a carbon balance to calculate emissions factors, eliminating the need to collect velocity measurements in the stack. The code we developed to analyze this data can be found at https://github.com/aprovechodotorg/Data-Processing-Software under PEMS/PEMS_FuelDataCleaning.py and PEMS/PEMS_FuelLoadData.py.

The FUEL sensor consists of a logging load cell that can be used to measure time-resolved fuel mass for cooking and heating activities. It comes in two types: a tensile version and a

compressive version. In this study, we used compressive FUEL sensors (Figure 1) to measure the mass and frequency of loading events for residential wood heating stoves. To do this, study participants were instructed to store their wood on the sensor for a minimum of one minute before placing it into their stove. Participants were provided with a metal rack to keep on the sensor to allow them to store larger quantities of wood.

Figure 1. The FUEL sensor from Climate Solutions Consulting.
https://www.climate-solutions.net/products/fuel

The EXACT sensor is a Stove Use Monitor that uses infrared technology to measure temperature (Figure 2). In this study, we placed EXACT sensors on the outside stove wall or stove pipe of the heating stove used in households where we collected FUEL data. The EXACT sensors were mounted onto the stoves using magnets and required no interaction from the study participants.

Figure 2. The EXACT sensor from Climate Solutions Consulting.
https://www.climate-solutions.net/products/exact-stove-use-monitor

Both sensors can be wirelessly started and stopped using the Unified Wireless Launcher, a handheld device designed to interface with this suite of sensors. With the Launcher it is also possible to wirelessly download sensor data, allowing review of the data while collection is ongoing.

2. Loading in Data

Both sensors record data in .csv format. The data should be stored in a folder titled with the test start date (mm.dd.yy), which is stored in a folder titled with the household number (e.g., GP001). The FUEL and EXACT data file names should follow the mm.dd.yy_FuelData.csv or mm.dd.yy_ExactData.csv format, using the same test start date as the folder title. After locating the data (the files produced by both sensors contain metadata in the first few rows), it is read into dictionaries where the column headers are parsed and used as keys and the data stored in

lists as values. The dictionary entries we are most interested in are shown in Table 1.

Table 1. Dictionary contents.

Sensor	Key	Description
FUEL	‘firewood’	Fuel mass [kg]
EXACT	‘Temperature’	Temperature [C] Note: 0°C is ambient temperature, all other temperatures are relative to ambient
Both	‘time’	Timestamp of log event in datetime format [m(m)/d(d)/yyyy hh:mm:ss (AM/PM)]
	‘seconds’	Seconds elapsed since sensor start, incrementing by log rate (this array is created while loading in the data)

3. Cleaning up Outliers

After being loaded in, the EXACT data is left as-is, but the FUEL data is passed through a cleaning algorithm. Any unintended application of force such as pressing, jostling, picking up, or moving the sensor could be recorded as a change in mass. These types of events often appear as short spikes in the data, where the sensor reports a steady value for a period of time, then reports an increase or decrease in mass for a very short period of time and returns to reporting the same steady value as before the spike. An example of these spikes can be seen in the raw data plot in Figure 3.

To remove these spikes before analyzing the FUEL data, we applied a centered moving median filter. The centered moving median algorithm takes a window with an equal number of data points on either side of the current point, calculates the median of the values in the window, and saves the median as the new current data point. The default window size in this algorithm is 30. Because the window size is even, we chose to take 14 points before and 15 points after the current data point to ensure the window corresponded to a sensing time of two minutes (for a log rate of four seconds). We chose a window size of two minutes because participants are asked to keep wood on the sensor for a minimum of one minute before removing it and placing it in the stove, so two minutes provides a buffer for the filter window size. As there is likely no activity in the first and final minutes of the sensing session, to keep the code simple we do not replace the first or final 15 data points.

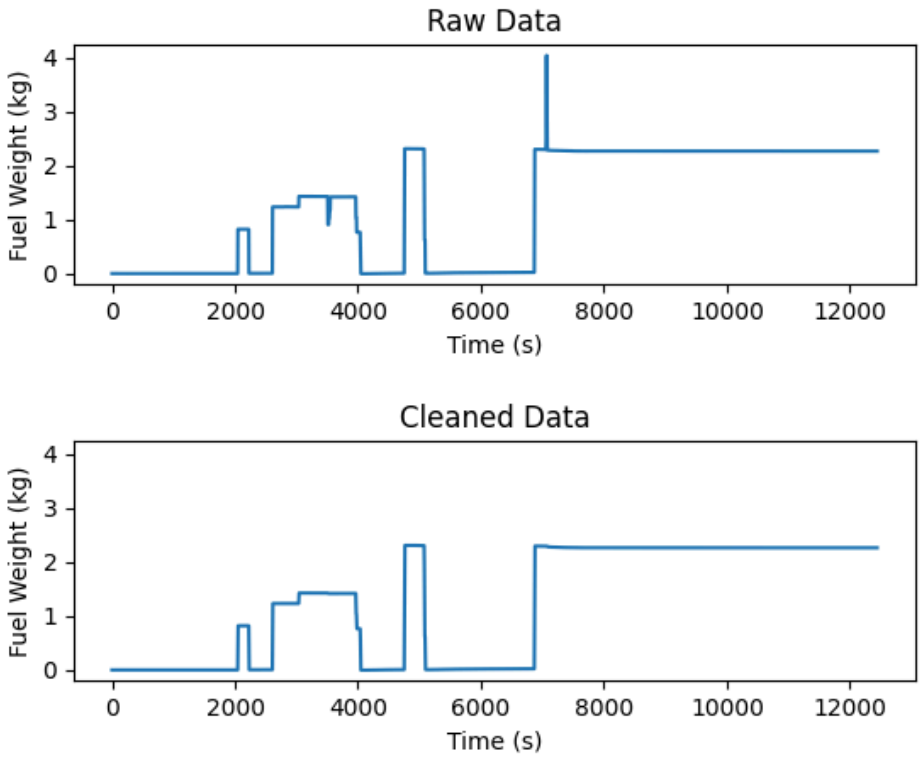


Figure 3. FUEL data before and after cleaning algorithm is applied.

4. Identifying Loading Events

The removal algorithm sorts through the data in multiple passes, refining its definition for removal and loading events with each pass.

4.1 First Pass

The first pass finds all possible removal events and then checks them against the EXACT data to determine if they are also potential loading events. To find a possible removal event, the algorithm loops through each element of the cleaned FUEL mass data and compares the difference between the current data point and the following data point to a set threshold value. The threshold value defaults to 0.125 kg and can be set by the user as a function input. If the difference is greater than the threshold, the current point is a potential removal event. We chose 0.125 kg as the threshold value by iteratively plotting the data with different threshold values. This value is the lowest value that accurately captures all removal events in our datasets.

Next, we look at the EXACT temperature data to determine if the potential removal event is also a potential loading event. First, we find the timestamp of the potential removal event. Then, we find the same timestamp in the EXACT data (if the FUEL and EXACT sensors have the same log rate), or we find an EXACT data point within 28 seconds of the FUEL timestamp (if the sensors use different log rates, e.g., four seconds for FUEL and one minute for EXACT).

Through testing different values, we found 28 seconds to be the largest time difference that kept the output arrays the same size. If a time difference larger than 28 seconds is used, the algorithm will find multiple EXACT data points that correspond to a single FUEL data point. Finally, we look at the 25 minutes of temperature data following the current data point. We use the measured temperature to define an “on” and an “off” condition for the stove. If the measured temperature is 0°C (ambient temperature), the stove is off. If the measured temperature is greater than 0°C, the stove is on. If the stove is on in the 25 minutes following the removal event, it is also a loading event. We save the index of removal events that are also loading events for later use in the algorithm.

4.2 Second Pass

The second pass is necessary because the first pass tends to classify single removal events as multiple events. An example is shown in Figure 4, where each red circle is a removal event classified by the first pass only.

This piece of code iterates through the cleaned fuel data array and checks if there was a removal event at each element. If there was a removal event, we then iterate from the current fuel data point forward through the array until we find a section where the slope of the fuel data is steady (changes by less than 0.05 kg) for a period of slope_window, which defaults to one minute for a log rate of four seconds and can be set by the user. Then, all removal events between the event that started this iteration and the period of steady slope are removed. In this way, we can find the start of a removal event, and later find the end of a removal event to calculate its magnitude.

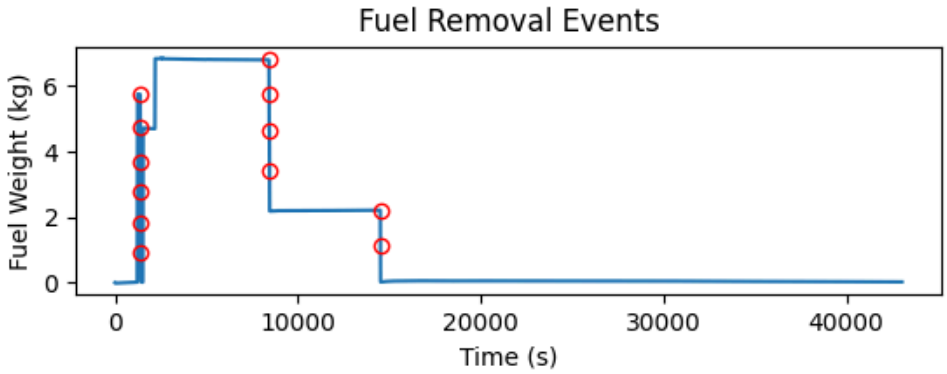


Figure 4. Removal events as classified by the first pass.

4.4 Intermediate Outputs

Now that we have our final array of removal indices, we can select and calculate our desired intermediate outputs. The removal index array allows us to select removal start points (the mass measured by the sensor as removal begins to occur) and timestamps for the removal events. With this information, we can then calculate the frequency of removal events on a desired time basis (e.g., per day, per sensing session, etc.). We chose to record loading frequency as time in

hours passed since the previous load. This means that the loading frequency for the first removal event in each dataset is always 0, because there was no previous removal event. Using the removal index array, we can also create an array of stove temperatures during removal events.

To find removal endpoints we need to first look at the beginning of a removal event, then iterate through the FUEL mass data to find when its slope becomes steady for a long enough period. We chose a slope window of 15 (one minute for a four second log rate) as the default value, and this can be changed by the user as a function input. Starting with the index of the removal event, we iterate through the rest of the data. To determine if the slope is steady, at each data point we look at the difference between the current point and a point a number of values ahead equal to the slope window. If this difference has a magnitude of less than 0.05, the slope is steady. We chose this value by looking at the raw data and observing that when the sensor reading is steady, the values tend to fluctuate by less than 0.01 kg.

Once we have removal endpoints, we can calculate the mass of each removal event and sum them to find the total mass removed over the sensing period. If we know the size of the firebox of the participant's stove, we can calculate a loading density by multiplying the mass of the fuel by 2.2 to convert from kilograms to pounds, then dividing by the firebox volume in cubic feet.

In addition to these outputs, we also wanted to be able to detect cold starts and second, third, and final loads. If the stove temperature is 0°C (ambient) during the first removal event, it is a cold start. Otherwise, if 7.5 hours or more have passed since the last removal event and the stove temperature is 0°C, the event is a cold start. Once a cold start event is defined, we define the previous event (if there is one) as a final load, the event after a cold start as a second load, and the event after a second load as a third load. We save the magnitude (kg), frequency (hours since last event), and loading density (lb/ft^3) of each of these events to individual arrays.

5. Final Plots

The final set of removal events are plotted over the cleaned data to give a visual for what the algorithm is doing, as well as how the participants are using their stoves over the course of the sensing period (Figure 5). The EXACT temperature data is shown below the FUEL data to make it easy to visually correlate fuel removal events with temperature changes in the stove.

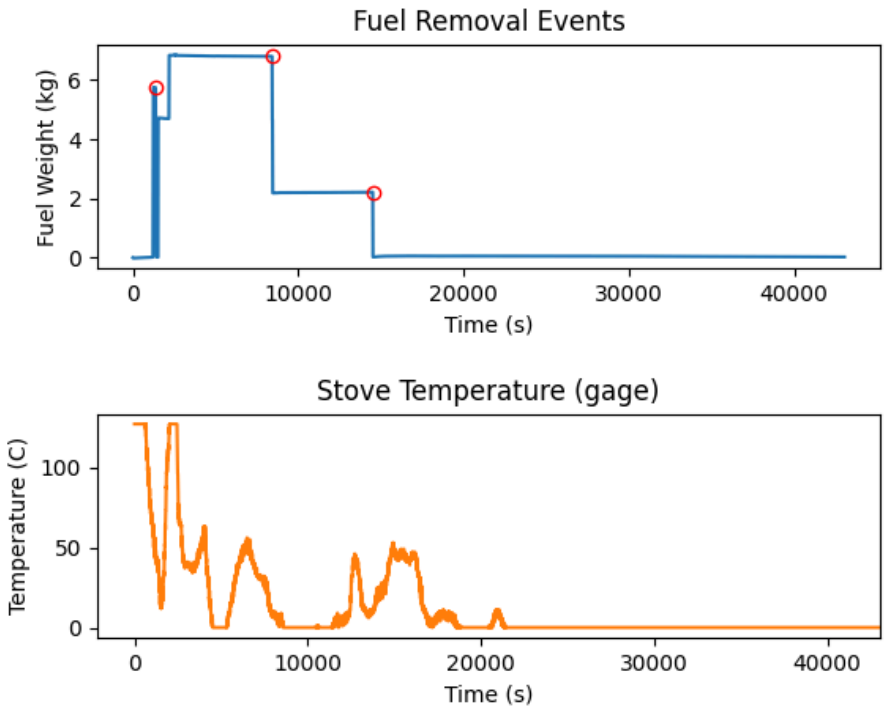


Figure 5. Final plots. Fuel removal events are highlighted with red circles and temperature data is shown below for visual comparison.

6. Verification and Validation

To verify the removal algorithm, we tested it against datasets from multiple use cases. The algorithm was used to identify fuel removal events in data collected from 13 participant households, from 33 researcher tests performed at home, and from laboratory testing. While it is not possible to independently validate the results generated for the participant data, the algorithm was able to handle all the use cases we tested and produce outputs that aligned with the plots of raw and cleaned FUEL data as well as the EXACT data. The datasets performed by a researcher at home are reliable because the researcher was trained in the proper use of the FUEL sensor. Each drop in the FUEL data could be trusted to correspond to a removal event, and the algorithm correctly identified all removal events in these datasets. This cross-sensor analysis has also been shown to be effective in monitoring cookstove use [3].

To validate the algorithm, it was tested on a dataset collected during a laboratory test. During this test, the FUEL and EXACT sensors were used to measure fuel mass and stove temperature. In addition, each piece of fuel removed from the FUEL sensor and placed in the stove was weighed on a scale and its mass was recorded manually. The data from this laboratory test was run through the algorithm, which correctly identified all three loading events as shown in Figure 6. The magnitudes of each loading event as recorded by the scale and as calculated by the algorithm are shown in Table 2.

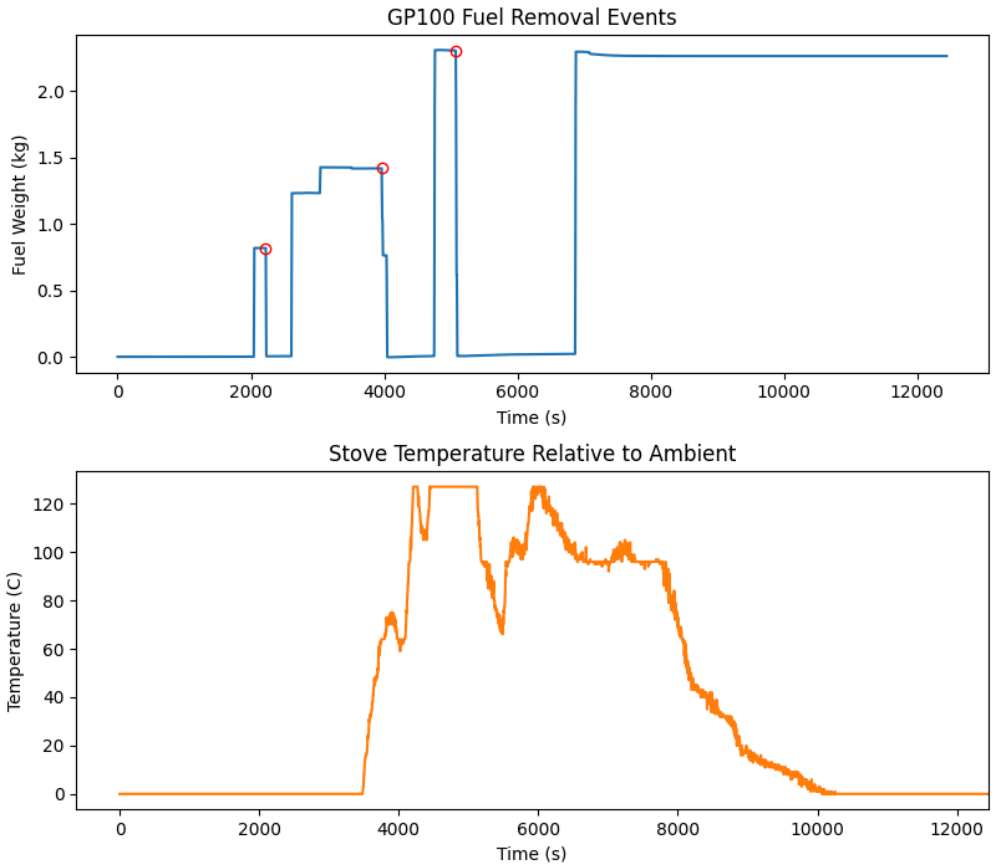


Figure 6. Fuel removal events from laboratory validation test.

Table 2. Differences between recorded and calculated size of loading events.

Measurement Source	Kindling (kg)	Small Pieces (kg)	Large Pieces (kg)	Large Pieces (kg)
Weight on Scale	0.795	1.432	2.308	2.278 (not used)
Algorithm	0.81	1.42	2.3	-

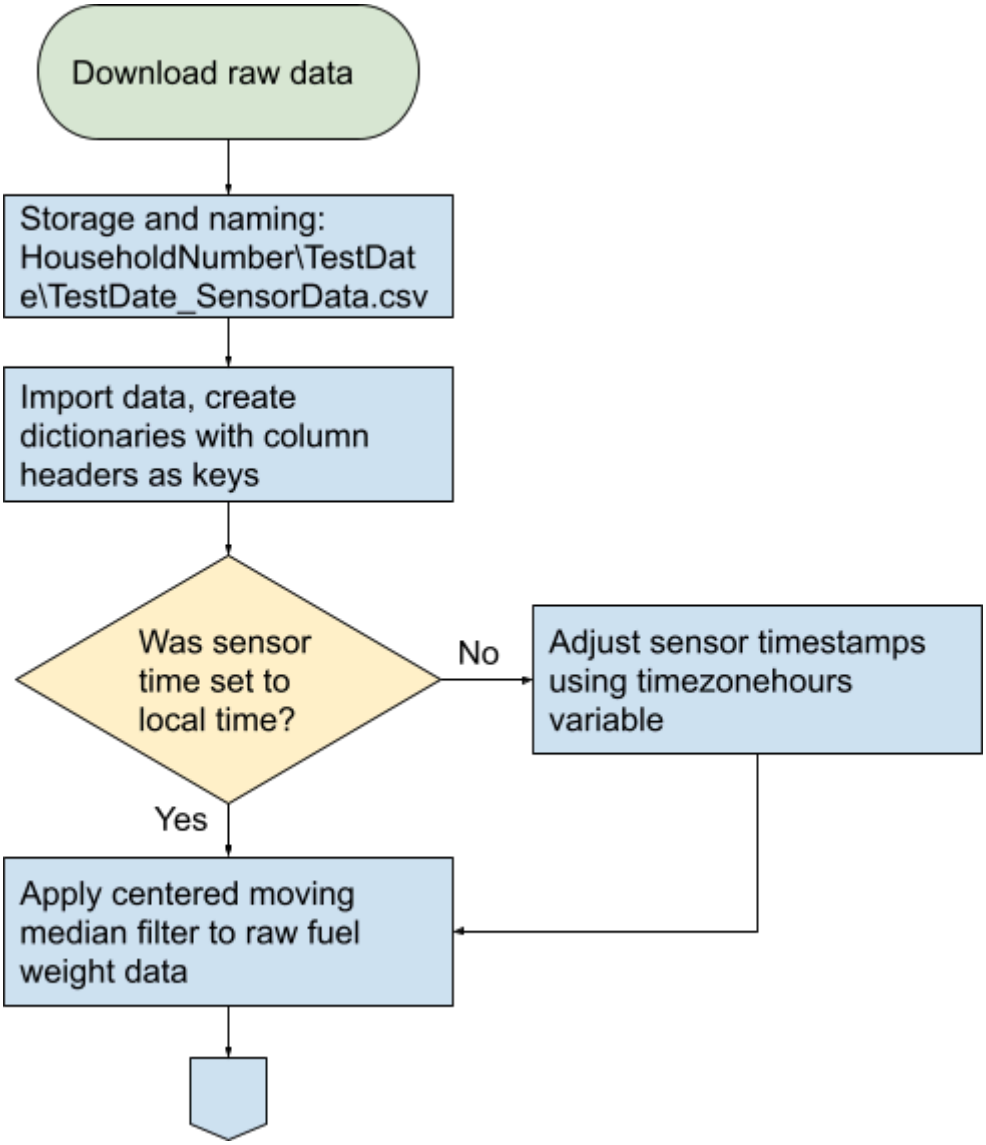
References

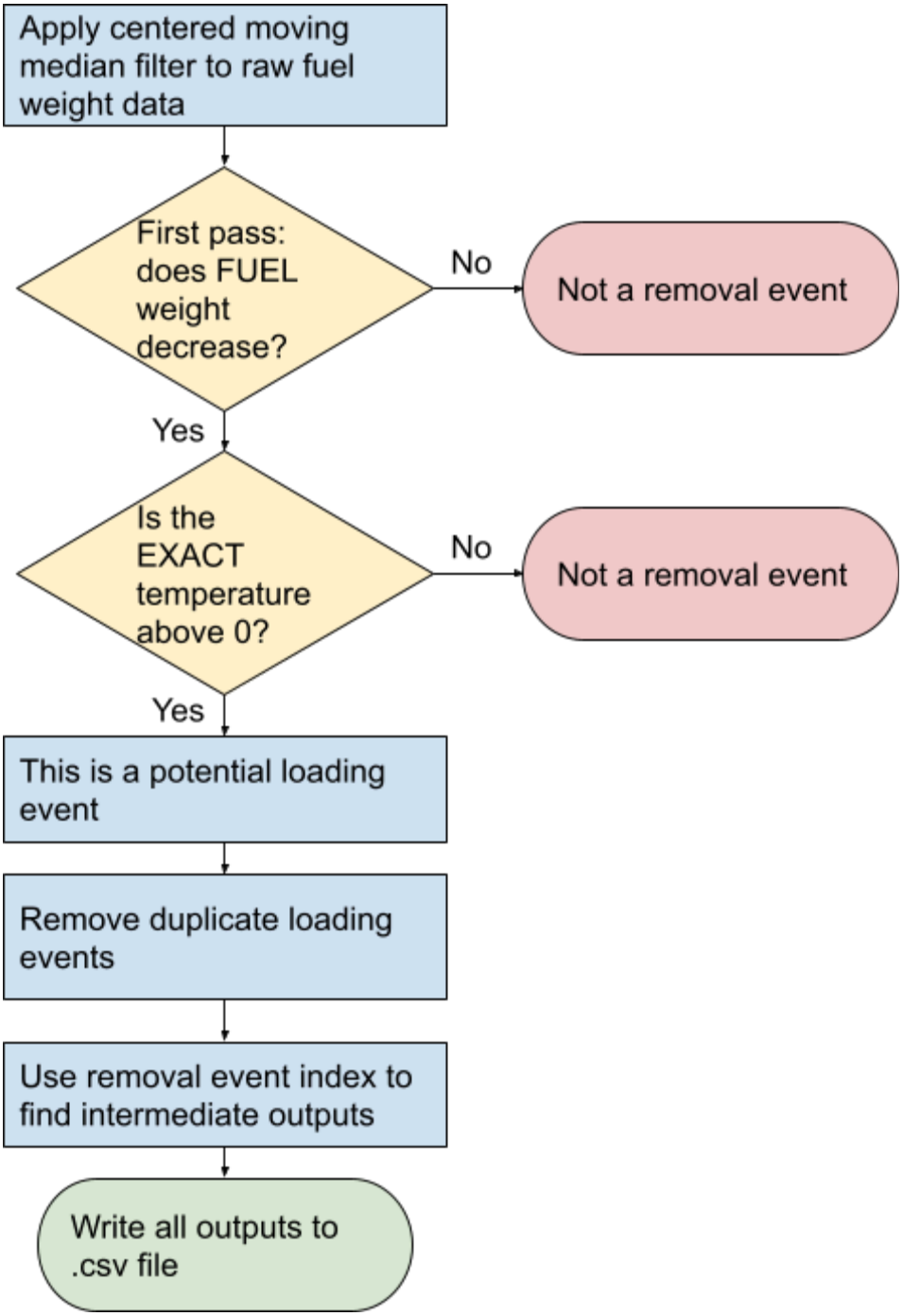
[1] J. Ventrella, N. MacCarty, Monitoring impacts of clean cookstoves and fuels with the Fuel Use Electronic Logger (FUEL): Results of pilot testing, Energy Sustain. Dev. 52 (2019) 82–95. https://doi.org/10.1016/j.esd.2019.06.004.

[2] J. Ventrella, O. Lefebvre, N. MacCarty, Techno-Economic Comparison Of The Fuel Sensor And Kitchen Performance Test To Quantify Household Fuel Consumption With Multiple Cookstoves And Fuels, Dev. Eng. 5 (2020) 100047. https://doi.org/10.1016/j.deveng.2020.100047.

[3] H. Miller, J. Shrestha, O. Lefebvre, N. MacCarty, Use of an integrated suite of sensors to simultaneously monitor fuel consumption, air quality, and adoption provides important insights and validates impact metrics for household stoves, Dev. Eng. 7 (2022) 100099. https://doi.org/10.1016/j.deveng.2022.100099.

Algorithm Flowchart




Apply centered moving median filter to raw fuel weight data
First pass: does FUEL weight decrease?
No
Not a removal event
Yes
Is the EXACT temperature above 0?
No
Not a removal event
Yes
This is a potential loading event
Remove duplicate loading events
Use removal event index to find intermediate outputs
Write all outputs to .csv file