We would like to thank the reviewers and the associate editor for their comments. We have submitted a new version of our paper, with new text in blue font. The main revisions are in section 4 (data generator). We included new experiments showing that the generator produces data that reflect real-world characteristics and extended the functionality of the data generator with the ability to adjust consumption peaks.

Our responses are summarized below:

Associate Editor's review:

(a) Strengthen the practical aspects of the contribution by highlighting new challenges introduced by smart meter data

Response: we revised the introduction – added a new citation on page 1 showing that smart meter analytics is an industry pain point, added an explanation of what's new in smart meter analytics on page 2 (they include a mix of operators and workloads that has not yet been studied), and explicitly listed our novel contributions on page 3.

(b) Strengthen the technical contribution by providing a detailed description of the data generator and highlighting the innovative aspects of the process

Response: we added more details about the data generator, and extended its functionality with the new peak factor parameter (section 4.1). We also included new performance experiments, showing that our Spark implementation of the data generator scales linearly (section 4.2). The innovation lies in the observation that electricity consumption is determined by air conditioning/heating and other temperature-sensitive loads, and activity load. By independently generating temperature-sensitive and insensitive loads using real data, we obtain realistic new time series whose electricity usage combines the characteristics of multiple existing consumers (section 4.1, 2nd paragraph). Also, this allows us to simulate different climates by keeping activity load the same and producing new temperature-sensitive consumption based on new temperature time series.

(c) Experimentally show that the generator provides data that reflects real world smart-meter data characteristics

Response: done – see the new section 4.2

Referee 1:

Is there anything specifically new in smart-meter data?

Response: we added a sentence to the first paragraph of section 1.1 to clarify this. Smart meter analytics include a mix of operators that have been studied in isolation, but not in a single benchmark.

It seems to me that the regression algorithm for the third task “daily profile” also works for the second task “thermal sensitivity”. Please clarify the major differences in these two tasks and the rationale of selecting the described algorithms.

Response: Daily profiles actually remove thermal sensitivity and focus only on activity load (see first paragraph of section 3.3). So, the second and third tasks zoom in on two independent contributions to household electricity consumption: temperature-sensitive load like A/C and heating, and activity load (laundry, cooking) which is temperature-insensitive.

The anomaly detection algorithm detects anomalies when a day finishes. Having a moving window of 24 hours or 3 hours may make more sense for online anomaly detection because online detection algorithm had better provide prompt warning messages (instead of reporting anomalies after one day).

Response: We added a clarification about this to footnote 9 on page 9. Our anomaly detection algorithm can run at higher frequencies or using a sliding rather than “daily jumping” window. In this paper, we check for anomalies daily using the past 24 measurements because this is how smart meter data collection currently works in our jurisdiction. Also, our performance results in sections 5.5 through 5.7 generalize to higher anomaly detection frequencies. For example, on page 34, we observe that it takes 6-7 minutes to detect anomalies in 2.5 million time series with 24 hourly measurements each. This suggests that if we had minutely data, we could run anomaly detection every 24 minutes and it would only take 6-7 minutes.

Minor:
Page 6, Paragraph 2, Line 1, “a recently algorithm” => “a recent algorithm”.
Page 17, last line, “point of time time series”, one “time” is redundant.

Response: fixed

Referee 2:

The paper would benefit from generating datasets that contain missing values for your benchmarking process, as well as lagged reporting of meter data to your streaming application benchmarking. I would then build a few analytics on top of that to deal with these issues.

Response: We added a paragraph at the end of section 3.5.2, which explains that our anomaly detection framework can be used to impute missing data. In anomaly detection, we are predicting consumption based on the past three days. The predicted consumption over the past three days can be used to fill in missing data at various hours of the day. For lagged reporting, we added a clarification on page 18 that we replay one day’s worth of data at a time, so there is no lagged reporting. To deal with it, the simplest approach is to over-provision the system and make sure that all the data that could potentially arrive in one batch, including late-arriving data, can be processed before the next batch arrives. With this approach, our experimental results can still be helpful to practitioners since they show how long it takes to process and detect anomalies in data batches of different sizes.

Include the resampling of data and linear transformations of time series

Response: we didn't include resampling in the benchmark because it is relatively simple to implement: make a single scan over the data and aggregate individual data points into the desired granularity. Since we already include the histogram, which also makes a single scan over the data and performs aggregation (not over time but over the frequency domain), we chose to include other more complex types of tasks in the benchmark. We didn't include transformations such as FFT because, based on our review of the literature, tasks like histograms (to convert into the frequency domain) and temperature sensitivity (to convert into the temperature domain) appeared far more frequently.

Include a benchmark of the PANDAS library

Response: For conciseness, we chose one system from each class. From the class of scientific computing platforms, we picked Matlab, but we also considered PANDAS and SAS. Matlab was the most mature platform, dating back to the 1980s compared to 2008 for PANDAS. We decided against SAS because its free version must run in a virtual machine, where performance necessarily drops.

Referee 3:

Paper has little contribution in terms of algorithm design

Response: We clarified our contributions at the top of page 3. Of the four contributions we make, two are algorithmic: the data generator and the anomaly detection framework.

5 tasks for testing the benchmark are also solved in a naive way

Response: We did our best to optimize the benchmark implementation and we chose state-of-the art data processing platforms. We used as many native operators as possible (e.g., for regression or matrix/vector operations), as we explain in section 5.2. For System C, we had to implement everything in the low-level procedural language that it supports; however, we did not find any significant sources of inefficiency in our code, and System C ended up being the best performer anyway. Furthermore, we tested different ways of storing data in Postrgres and different file formats/distributed processing strategies in Hive/Spark. Overall, we believe that each task in our benchmark was implemented in a non-trivial way, and that testing different ways of implementing the tasks in different platforms is one of the strengths of this paper.