Reproducibility Review of:

Towards national-scale ecological applications: harmonized framework for LiDAR point cloud processing for vegetation metrics calculation

Item	Value
Title	Towards national-scale ecological applications: harmonized framework for LiDAR point cloud processing for vegetation metrics calculation
Authors	Eveli Sisas (ORCID 0009-0002-5032-249X), Holger Virro, Wai Tik Chan, Alexander Kmoch and Evelyn Uuemaa
Full paper reference	Sisas, E., Virro, H., Chan, W. T., Kmoch, A., Helm, A., and Uuemaa, E.: Towards National-Scale Ecological Applications: Harmonised Framework for LiDAR Point Cloud Processing for Vegetation Metrics Calculation, AGILE GIScience Ser., 7, 18, https://doi.org/10.5194/agile-giss-7-18-2026, 2026.
Codechecker(s)	Frank O. Ostermann (ORCID 0000-0002-9317-8291)
Date of Check	31-03-2026
Summary	Full reproduction of sample data
Repository	NA
Ref. certificate	Ostermann, FO (2026): Reproducibility Review of: “Towards national-scale ecological applications: harmonized framework for LiDAR point cloud processing for vegetation metrics calculation”, https://doi.org/10.17605/OSF.IO/75pwn

Table 1: Reproduction metadata

Output	Comment
repro_rev_2026_17_notebook.html	HTML-export of Jupyter notebook containing the comparison of output files

Table 2: Summary of output files generated

Summary

The paper describes a study processing massive amounts of LIDAR point data in a HPC environment, which makes a complete reproduction as part of this review infeasible. However, the analysis workflow is deterministic, which means that a successful reproduction of a sample data set demonstrates the overall reproducibility of the study.

The authors provided such sample data set, consisting of a single input file with a point cloud data set, as well the intermediary and final output files. Following the instructions, this review created another set of output files and compared them to the original ones. Initially, there were minimal deviations between the provided and reproduced files. After correspondence with the authors, this could be attributed to a different buffer size used to create the ancillary input files between runs; modified input files were provided that could be reproduced flawlessly.

Thus, the reproduction of the data sample was successful.

Reproducibility reviewer notes

The study comes with full code repository, sample data repository, and documentation how to use both. However, the instructions were not always clear about which steps were meant for full reproduction (reimplementation) of the workflow, and which steps referred to reproducing the sample workflow. These were minor and could easily be overcome. A bit more information on which criteria constitute a successful reproduction would have been helpful. The comparison was developed as part of this review.

Installation prerequisites and computational environment

The repository contains all code and instructions to set it up using the micromamba package manager. This proved to be less straightforward than hoped in the case of this review: several problems with getting micromamba integrated and running in the shell, followed by (Py)proj and other dependencies issues, both on Windows and Ubuntu environments, had to be solved first. However, except for a minor documentation issue around poetry, none of that is the responsibility of the authors, and this reviewer is willing to take the blame for having a history of messed up Python environments across different operating systems.

Data preparation

Here the documentation was not entirely clear on how to integrate the sample data into the workflow. The reviewer decided to place all the sample data into a new \repro_data\ folder.

Running the code

The code was run as instructed, although the suggested method of calling the code did not work (see comments on computational environment). For this partial reproduction, only two Python scripts need to be called. The output files were distinguished from the provided files with a _repro suffix. The full command line prompts (for Windows) were (run from the project root):

```
python lidar_processor\model\processing_script\fix_laz_file.py
repro_data\568539_2020_tava.laz
repro_data\568539_2020_tava_fixed_repro.laz
```

```
python
lidar_processor\model\processing_script\reclassify_laz_file.py
repro_data\568539_2020_tava_fixed_repro.laz
repro_data\568539_2020_tava_reclassified_repro.laz
repro_data\568539_dem_1m_2017-2020.tif
repro_data\568539_ETAK_EESTI_GPKG_2021_01_02.gpkg
repro_data\568539_est_s2_ndvi_2020-04-01_2020-05-31.tif
```

On a Windows 11 / Ubuntu 24.04 laptop with 16 GB RAM and an Intel i5-8265U CPU, code execution took just a few moments.

Outputs and results

The compressed output files vary slightly in size from the provided ones, which could be attributed to different computational environments. For a more in-depth comparison, the attached Jupyter notebook was used. The results of that comparison initially showed miniscule differences in classification (e.g., a few out of almost 35 million points classified differently for the WithinPowerline class).

Correspondence with the authors and further testing by them revealed that the cause was a too small buffer used for clipping the sample data files. Revised files were provided. The result was now zero difference between provided and reproduced files.

Recommendations

The authors did a commendable job at providing information about reproducing the analysis workflow. It would be helpful if there was a clearer distinction between documentation for the general (HPC) workflow, and the specific steps for reproducing the sample output.

As a more general observation, it seems almost impossible to provide instructions on how to run code on a different rig out-of-the-box without issues, even with virtual environments. My recommendation is thus to provide a clear, high-level instruction on what computational environment is needed to run the code, followed by an example that worked for the authors.

Acknowledgements

NA