

January 28, 2026

Journal of Statistical Software

Dear Editor,

We are thankful for the feedback on our previous submission and apologize for the issues encountered in the submission. Below, we provide a point-by-point response to the comments raised and outline the changes made to address them. We hope that you will find our revised submission suitable for publication in the Journal of Statistical Software.

Here are the detailed comments and our answers:

Journal of Statistical Software requires that the software usage is illustrated on simple problems run step-by-step in the manuscript, showing command lines and their results. This is currently done for the R package but we would find beneficial to have a similar code for the other two packages.

We agree that this would be useful and have added a section in the appendix with versions of the simple example from the R package in Python, Julia, and C++.

The introduction must include a discussion on related software implementations available in all languages (e.g., software implementations related to  $\ell_1$  penalized linear regression for instance, or even slope problem estimation), highlighting the specific contribution of slope.

We think there might be a misunderstanding here, as SLOPE is not related to slopes in the geometric sense, but rather a type of regularized regression problem. It is also distinct from standard  $\ell_1$ -penalized regression (the lasso), which means that it would not be meaningful to include a full background on packages that implement the lasso. We have, however, added a discussion of existing SLOPE implementations in the introduction to clarify this point.

None of the [package files] can be uncompressed (or used) using gunzip.

Thanks for pointing this out! We have now packaged the package files correctly and verified that they can be decompressed without issues.

It means that, at least a summary method is missing.

We appreciate the suggestion! We have added `summary()` methods for both SLOPE and TrainedSLOPE in the R package.

Other methods might be missing that would avoid exposing the internal structure of objects in the replication material [...]

We are not sure we agree that the fields exposed in the objects should be considered internal structure (private) nor that it is idiomatic in R to hide them behind accessor methods, so we have not made any changes in this regard. In the case of the SLOPE

patterns, this object is computed as part of the fitting procedure and can therefore not easily be externalized to a separate function.

Using `example("SLOPE"); plot(fit)` results in an empty plot. It is probably not expected: Can the authors look into this?

Thank you! This was indeed a bug. In the new version of the package, we have added an entirely different (Cleveland dot plot) visualization of the solution for single fits.

Replication material should be cleaned and simplified a bit, in particular to remove hidden configuration files coming from github. Make sure to upload only files required to reproduce results of the manuscript and to adapt README files to avoid having to use the github repository in the instructions (you can of course mention that these files are duplicated in a github repository but explain how to run the replication code from the local directory).

We have cleaned up the replication material and removed unnecessary files. The README file has also been updated to ensure that it describes both how to run the code from the local directory.

Following the instructions to run the benchmark given in the README file of `benchmark_slope` (after activating a virtual environment for Python), we obtained: [...]

This is related to not having the conda environment activated. We have updated the README file to clarify this.

The README file of the replication material indicates that this is the case: please make the two files consistent or remove the README files of subdirectories.

We have also removed redundant README files from the subdirectories.

Also provide a more detailed information about setting the conda environment for user not familiar with it.

We have added detailed instructions on setting up the conda environment in the README file.

Running `benchopt` run in `benchmark_slope` results in errors due to missing dependencies (`skglm` for instance). It would be preferable to provide a `requirements.txt` file allowing users to install all required dependencies without the need to search what they are.

`benchopt` has built-in support for managing dependencies via the benchmark definition files, through which the required dependencies are declared and then automatically installed when `benchopt install` is run. We have updated the README file to clarify this.

Trying to run the same benchmark, we also had: [...] Can you instruct

how this should be solved?

This too is related to not having the proper dependencies installed. The new instructions in the README file should help mitigate this issue.

Could you please also provide scripts with minimal command lines to run the equivalent of `example.R` and/or `real-data.R` with Python and Julia?

We have supplied `example.py`, `example.cpp`, and `example.jl` scripts that replicate the functionality of `example.R` in Python, C++, and Julia, respectively.

`help(package = "SLOPE")` shows that man page titles are not in title style.

We have converted all manual pages to use title case in their titles.

Probably, `plot.TrainedSLOPE()` should be removed from "Other model-tuning".

Thank you for catching this! We have removed the duplicate reference.

It seems that the output of `cvSLOPE` only gives a way to obtain the optimal hyperparameters. In R programming, it is standard to also return the best model fit with these parameters (see e.g., `?e1071::tune`). Can the authors think of an handy way for the user to either obtain this trained model from `cvSLOPE` or use the output of `cvSLOPE` in a method that would directly train the optimal model?

Thanks for the suggestion! You're right that this was an omission. We have now updated `cvSLOPE()` to (optionally and by default), return the fitted model on the full data set. We've also introduced a new function (and generic) `refit()` that can be used to fit SLOPE onto new data using the parameters selected by `cvSLOPE()`.

Yours sincerely,

Johan Larsson, Małgorzata Bogdan,  
Krystyna Grzesiak, Mathurin Massias, and  
Jonas Wallin

encl: manuscript, software, replication materials, original cover letter