Journal of
open research software

Gonzalez-Perez, V, et al. 2020 A Python Package to Preprocess the Data Produced by Novonix High-Precision Battery-Testers. *Journal of Open Research Software,* 8: 3. DOI: https://doi.org/10.5334/jors.281

SOFTWARE METAPAPER

A Python Package to Preprocess the Data Produced by Novonix High-Precision Battery-Testers

V. Gonzalez-Perez, P. Keil, Y. Li, A. Zülke, R. Burrel, D. Csala and H. Hoster

Energy Lancaster, Lancaster University, Lancaster, UK

Corresponding author: V. Gonzalez-Perez (violetagp@protonmail.com)

We present `preparenovonix`, a Python package that handles common issues encountered in data files generated with a range of software versions from the Novonix battery-testers.[1] This package can also add extra information that makes easier coulombic counting and relating a measurement to the experimental protocol. The package provides a master function that can run at once the cleaning and adding derived information, with flexibility to choose only some features. There is a separate function to simply read a column by its given name. The usage of all the functions is documented in the code including examples. The code presented here can be installed either as a python package[2] or from a GitHub repository.[3]

Keywords: Batteries; Battery-testers; Novonix; Data; Clean data; Python

Funding statement: This work has been conducted within the Multi Scale Modelling collaboration from the Faraday Institution: ALC7052 and ALC 7082.

(1) Overview

Introduction

The growth exploiting renewable energies is only possible thanks to the development of adequate energy storage systems [2]. Li-ion batteries have become one of the fastest growing electric energy storage systems in the automotive market [4]. Cycling these batteries through charging and discharging is one of the cornerstone experiments to understand their working performance and ageing behaviour, which are essential to help improving their design [5] and the lifespan prediction [3]. This cycling can be done in different types of testers, using a range of batteries not limited to the Li-ion ones. Novonix is a relatively new company in the market of battery-testing systems, catering to high-precision coulometry [1]. An accurate coulombic efficiency tracking can provide insights for battery ageing mechanism and lifetime prediction at early experimental stages. The `preparenovonix` package prepares the raw data exported from Novonix battery-testers so it can be later analysed with ease. Traditionally, this type of code is not widely shared among different groups working on battery research. However, opening this code to the community has the potential to benefit all users of the Novonix battery-testers and to promote further collaboration developing code relevant for the battery research field.

The `preparenovonix` package prepares exported data files produced by Novonix battery-testers[4] by (i) cleaning them and (ii) adding derived information to the file. The package also allows reading an individual column given its name. The derived information includes:

1. A *State* column with explicit information of the start and end of a given type of measurement. Novonix provides a *Step number* with a different value for each type of measurement, for example, 0 corresponds to an open circuit. However, it is possible to have two consecutive measurements of the same type but with different experimental conditions, for example charging at different currents. These can now be set appart using the *State* value.
2. A *reduced protocol* summarising the experimental protocol into having each command and corresponding experimental conditions in a single line. This is needed to directly relate a measurement with the experimental protocol. The *reduced protocol* is output as a string of arrays and it is stored as part of the header when using the `prepare_novonix` function (see **Figure 2** and the text below).
3. A *Protocol line* column with values that assign a measurement to a particular line of the *reduced protocol*.
4. A *Loop number* column with a counter of the repetitions of a given measurement within a loop in the experimental protocol.

Combining the *state* and *step number* values makes it possible to select the capacities from a charge or discharge experimental step. These are needed for estimating the coulombic efficiency. This combination of *state* with *step number* also allows the computation of resistances based on current experimental steps or pulses. Specific cycles or individual sections of the experiment can be selected

combining the *loop number* with either the *state* and *step number* values or the *protocol line* values.

The example data provided within the repository for this code is shown in **Figure 1**. This figure compares the raw Novonix data with the data after being processed by the `preparenovonix` package. The example raw data contains individual measurements for which the experimental run time decreases. As it can be seen in **Figure 1**, these measurements are removed by the `preparenovonix` package. The example raw data file also includes a failed test. The `preparenovonix` package takes the capacity from the failed test and adds it to the capacities from the completed experiment. This shifts the result capacity curve by a constant value, as it can be seen in **Figure 1**. This figure also shows the increasing *loop number* when the measurements are within a repeat loop and the *protocol line* each measurement corresponds to.

Implementation and architecture

The main functions available in the `preparenovonix` package[5] are listed below in alphabetical order. The list contains the module name followed by the function name with the expected input parameters in brackets.

- `novonix_add.create_reduced_protocol(infile,verbose=False)`: Given a cleaned Novonix data file, `infile`, generate a *reduced protocol*.
- `novonix_add.novonix_add_loopnr(infile,verbose=False)`: Given a cleaned Novonix data file, `infile`, add a *reduced protocol* to the header and the columns *Protocol line* and *Loop number*.
- `novonix_add.novonix_add_state(infile,verbose=False)`: Given a cleaned Novonix data file, `infile`, add the *State* column.

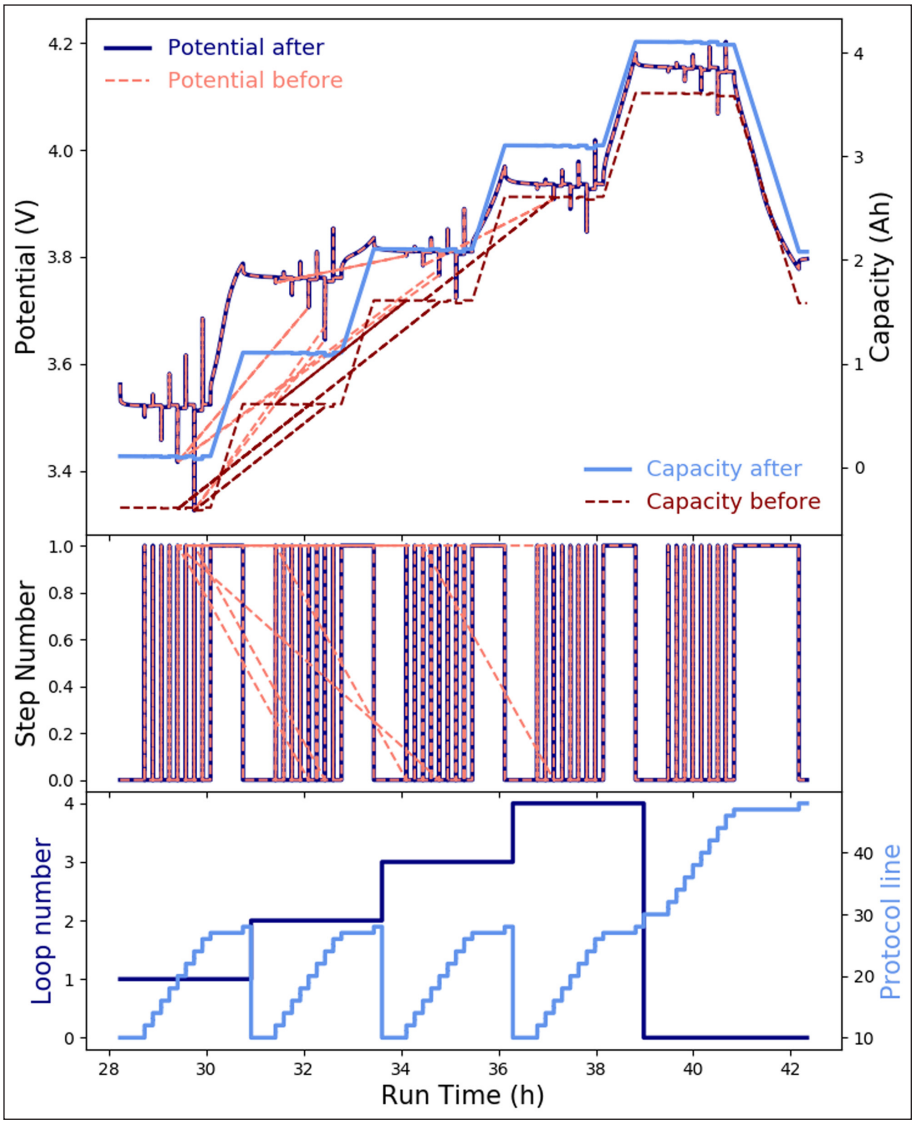

Figure 1: Comparisson of the raw battery testing data (thin dashed lines) and the data after being processed by the `preparenovonix` package (thick solid lines), as a function of the experimental run time. The top panel shows the potential and capacity of the battery. The middle panel shows the *step number*, which indicates the type of measurement being done. The bottom panel shows the *loop number* and *protocol line*, which are only available after processing the raw data with the `preparenovonix` package.

- `novonix_clean.cleannovonix(infile)`: Given a Novonix data file, `infile`, clean it as it is described below.
- `novonix_io.isnovonix(infile)`: Given a file, `infile`, check if it is or not a Novonix data file.
- `novonix_io.read_column(infile,column_name,outtype='float')`: Given a column name, `column_name`, read it from a cleaned Novonix data file, `infile`, as a numpy array of the type given in `outtype`.
- `novonix_prep.prepare_novonix(infile,addstate=False,lprotocol=False,overwrite=False,verbose=False)`: Master function of the `preparenovonix` package that prepares a Novonix data file by cleaning it and adding to it derived information. This function follows the flow chart presented in **Figure 2**. Running all the available features from the `preparenovonix` package through this function can take form few seconds to up to few minutes depending on the size of the input file.

In what follows, the above functions will be referred by simply their name, without stating the modules they belong to.

As it is shown in **Figure 2**, the `preparenovonix` package only cleans data files that are consider to be exported from the Novonix battery-testers and it only derives information for cleaned Novonix files. The master function `prepare_novonix` allows the user to call either the cleaning process or the addition of extra columns ensuring that these dependencies are taken into account. The input parameters for this function are the path to a file and four boolean optional parameters: `addstate`, `lprotocol`, `overwrite` and `verbose`. The last parameter provides the option to output more information about the run. If the `overwrite` parameter is set to `False`, a new file will be generated with a name similar to the input one, except for the addition of `_prep` before the extension of the file.

The function `isnovonix` decides if a file has the expected structure (including a full header) for an

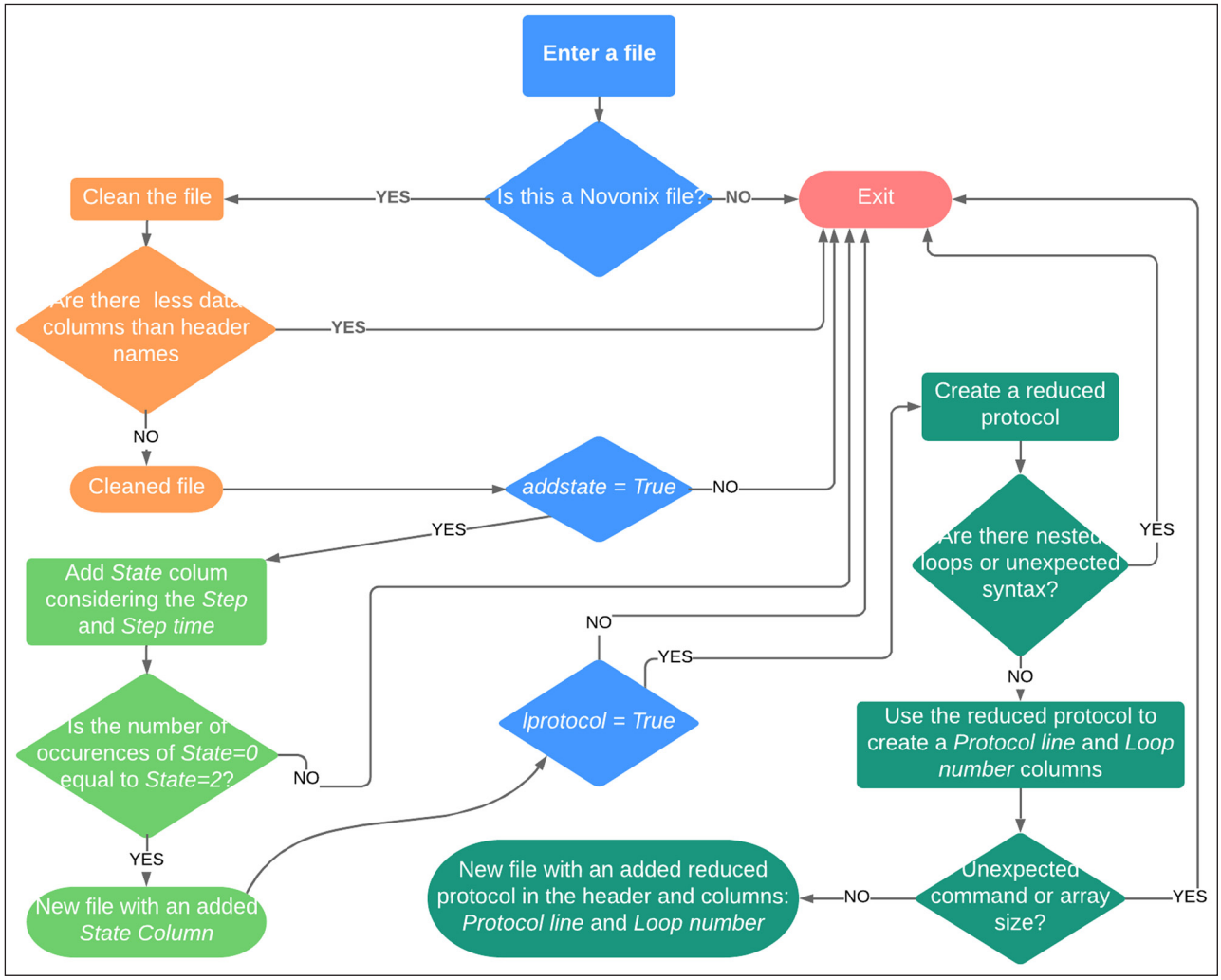

Figure 2: Flow chart for the `prepare_novonix` function (within the `novonix_prep` module) which contains all the functionality of the `preparenovonix` package presented here. Besides the name of the input file, this function has four optional boolean input parameters: `addstate`, `lprotocol`, `overwrite` and `verbose`. The last two parameters are not included in the flow chart, but they are described in the text. In this chart rectangle shapes indicate processes, rounder rectangles end of processes and diamonds decisions. Note that for simplicity not all the decisions made in the code are shown here.

exported file produced by the Novonix battery-testers. If the file is lacking the header or if it has not been exported with a Novonix battery-tester using the covered software,[6] the code will exit with an error message and without generating a new file.

The function `cleannovonix` produces a new Novonix type file after performing the following tasks:

- Delete failed tests within a single file, adding the final capacity of all failed tests to the capacity column of the finished test.
- Remove individual measurements for which the run time goes backwards.
- Remove blank lines from the header.
- Remove any trailing characters from the header produced when the file has been previously open with certain programs, such as Excel.
- Add a dummy header name (`dum[number]`) for each data column lacking a header name.

A *State* column can be added to a cleaned Novonix file by calling the function `novonix_add_state` or setting to `True` the parameter `addstate` when calling the function `prepare_novonix`. This *State* column can have the following values:

0 for the first measurement of a given type (for example, a constant current charge).
1 for measurements between the first and last of a given type.
2 for the last measurement of a given type.
−1 for single measurements. This can happen under different circumstances. A type of measurement can end after a single measurement when some experimental conditions are met, this usually happens while the time resolution is coarse. At times, the current can overshoot from negative to positive values at the beginning of a measurement. A bug in the Novonix software that locks certain values, etc. If two single measurements happen together, the two lines are discarded in the new file containing the additional *State* column.

The *State* column is generated based on the following quantities provided in the raw Novonix data files: *Step number* (integer indicating the type of measurement) and *Step time* (this time is assumed to reset to 0 each time a new type of measurement starts).

The function `create_reduced_protocol` reads the complete header from the input file and generates (or reads) the *reduced protocol*. This function returns the *reduce protocol* itself and a boolean flag, `viable_prot`. The *reduced protocol* consist of an array of strings. Each string contains a line number, a command from the experimental protocol and the corresponding experimental conditions (if aplicable); for example: `[4 : Repeat 49 times :]`. Only commands referring to the following processes will appear in the *reduced protocol*:[7]

- Open circuit storage (or rest)
- Constant current (dis)charge
- Constant current – Constant Voltage (dis)charge
- (End) Repeat

The *reduced protocol* is tested against the number of unique measurements in the file, determined using the column *State*. If the number of measurements expected from the protocol is less than the actual number of measurements, the flag `viable_prot` is set to `False`, indicating that the construction of the *reduced protocol* was not viable.

The *Protocol line* and *Loop number* columns can be generated by either calling directly the function `novonix_add_loopnr` or by setting to `True` the parameter `lprotocol` when calling the function `prepare_novonix`. The column *Protocol line* associates a measurment with its corresponding line in the *reduced protocol*. The *Loop number* column has a value of 0 if a measurement does not correspond to any repetition statement in the protocol and otherwise it grows monotonically with each repetition (see **Figure 1**).

If the flag `viable_prot` was set to `False` by the `reduced_protocol` function, the *Protocol line* and *Loop number* columns are populated with the value –999.

Quality control

Each function in the `preparenovonix` package is tested with internal checks and with `pytest` both locally and through the Travis Continuous Integration service.[8] The tests have been performed in different platforms and using different Python versions. The tests use an example data file. This file is automatically retrieved when the dedicated GitHub repository is either cloned or downloaded (see the 'Software location' section for the relevant urls).

Each function is documented with an example of usage. The expected result when used on the example data is also provided. Moreover, an example script, `example.py`, is provided at the root directory of the dedicated GitHub repository. This script also produces **Figure 1**.

The complete documentation for the `preparenovonix` package can be found at: https://prepare-novonix-data.readthedocs.io/.

(2) Availability

Operating system

Windows, OSX, Linux

Programming language

Python 3.5 and above.

Additional system requirements

The code presented here uses as input the data files exported directly from the Novonix battery-testers. The on-line documentation described in the 'Quality control' section, provides an updated list of the Novonix software versions that the code presented here has been tested against.

Dependencies

This software requires the `numpy` Python library. `Matplotlib` is also required for using the plotting routine `compare.plot_vct.py`. Further details on how to

install these libraries or how to install the software using 'pip' can be found in the 'Readthedocs' documentation mentioned in the 'Quality control' section.

List of contributors

The list of contributors comprises the author list and the contributors reported in the dedicated GitHub repository (see the url below).

Software location

Archive

Name: Zenodo
Persistent identifier: http://doi.org/10.5281/zenodo.3081471
Licence: MIT License
Publisher: Andrew Dawson
Version published: 0.0.1
Date published: 21/05/2019

Code repository

Name: GitHub
Persistent identifier: https://github.com/BatLabLancaster/preparenovonix
Licence: MIT License
Date published: 16/05/19

Language

All documentation is provided in English. For a translation into an other language, contact the corresponding author.

(3) Reuse potential

The software presented here can clean, enhance and facilitate the use of data produced by Novonix battery-testers. The potential for reusing this software is large among users of these testers, both in academic research and industry. Two aspects that are particularly fundamental are the cleaning of raw files, as described above, and the possibility to read a specific column for a range of formats from different software versions from Novonix. The software presented here can be modified and enhanced by contributing to the dedicated GitHub repository. Support can be provided by raising issues in the same repository.

Notes

[1] http://www.novonix.ca/.
[2] https://pypi.org/project/preparenovonix/.
[3] https://github.com/BatLabLancaster/preparenovonix.
[4] Through out the text we will also refer to these as: 'Novonix data files'.
[5] The full list of functions, including those auxiliary of the ones presented here, can be found in https://prepare-novonix-data.readthedocs.io/.
[6] See an up-to-date list in https://github.com/BatLabLancaster/preparenovonix.
[7] Note that the commands corresponding to incrementing the cycle counter and global emergency limits are ignored in the *reduced protocol* as there are no measurements associated with those.
[8] https://travis-ci.org/BatLabLancaster/preparenovonix.
[9] http://energysuperstore.org/esrn/multiscale-modelling/.
[10] https://faraday.ac.uk/.

Acknowledgements

The authors acknowledge all the formative workshops provided by the Multi Scale Modelling collaboration[9] within the Faraday Institution.[10]

Competing Interests

The authors have no competing interests to declare.

References

1. **Burns, J C, Stevens, D E, Dahn, J R** 2015 In-Situ Detection of Lithium Plating Using High Precision Coulometry. *J. Electrochem. Soc.*, 162(6): A959–A964. DOI: https://doi.org/10.1149/2.0621506jes
2. **Csala, D, Hoster, H E** 2017 Emissions: Step on the natural gas for German cars. *Nature*, 541: 157. DOI: https://doi.org/10.1038/541157b
3. **Li, Y, Zou, C, Berecibar, M, Nanini-Maury, E, Chan, J C W, van den Bossche, P, Van Mierlo, J, Omar, N** 2018 Random forest regression for online capacity estimation of lithium-ion batteries. *Applied Energy*, 232: 197–210. DOI: https://doi.org/10.1016/j.apenergy.2018.09.182
4. **Yang, Z** 2011 Electrochemical Energy Storage for Green Grid. *Chemical reviews*, 111(5): 3577–3613. DOI: https://doi.org/10.1021/cr100290v
5. **Wu, B** 2015 Differential thermal voltammetry for tracking of degradation in lithium-ion batteries. *Journal of power sources*, 273: 495–501. DOI: https://doi.org/10.1016/j.jpowsour.2014.09.127

How to cite this article: Gonzalez-Perez, V, Keil, P, Li, Y, Zülke, A, Burrel, R, Csala, D and Hoster, H 2020 A Python Package to Preprocess the Data Produced by Novonix High-Precision Battery-Testers. *Journal of Open Research Software,* 8: 3. DOI: https://doi.org/10.5334/jors.281

Submitted: 11 June 2019 **Accepted:** 07 February 2020 **Published:** 04 March 2020

]u[*Journal of Open Research Software* is a peer-reviewed open access journal published by Ubiquity Press.

OPEN ACCESS