Arbin Electrochemical Tools Documentation

Release 0.1.0

Vincent Wu

Sep 02, 2020

CONTENTS:

CHAPTER

ONE

ARBIN ELECTROCHEMICAL TOOLS

Reads raw electrochemical cycling data and generates useful plots and tables for battery scientists.

- Free software: MIT license
- Documentation: https://electrochem.readthedocs.io.

1.1 Features

- Extracts Arbin electrochemical cycling data files (.res or .xlxs) into csv files or Pandas dataframes for easy manipulation
- Plots publication quality voltage-capacity curves
- Calculates electrochemical properties such as operating voltage, average discharge capacity, ect. for a set of cycling data and generates summary tables

1.2 Credits

This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

CHAPTER

TWO

INSTALLATION

2.1 Stable release

To install Arbin Electrochemical Tools, run this command in your terminal:

```
$ pip install electrochem
```

This is the preferred method to install Arbin Electrochemical Tools, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

2.2 From sources

The sources for Arbin Electrochemical Tools can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/vince-wu/electrochem
```

Or download the tarball:

```
$ curl -OJL https://github.com/vince-wu/electrochem/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

CHAPTER

THREE

QUICK START

First, import the `electrochem` module, along the the `pandas` module (we need `pandas` to read and manipulate dataframe objects):

```
>>> import electrochem as echem
>>> import pandas as pd
```

To open an Arbin .res file and output a readable .csv file, define a raw data path and a path for the .csv file to be saved to:

```
>>> file_path  = "d:/data/cycling_data.res"
>>> save_path = "d:/data/cycling_data.csv"
```

Then, simply use the `parseArbin` function to convert your data into a .csv file.:

```
>>> echem.parseArbin(file_path, save_path)
```

In this example, the file will be saved into **cycling_data.csv**. To further manipulate your data, you can create an easily workable data object from the .csv file by using the `toDataFrame` function. This function takes in a .csv or .xlsx file containing raw cycling data along with your electrode material's mass, in mg:

```
>>> mass = 10 # electrode mass is in mg units
>>> partitioned_data = echem.toDataFrame(save_path, mass)
```

Here, `partitioned_data` is a data object where dataframes are sorted by both cycle index and charge/ discharge cycles. It is meant to facilitate data extraction and analysis. For example, to get the data for the first charge and discharge, simply do:

```
>>> first_charge_df = partitioned_data[1]['charge'] # data table for first charge
>>> first_discharge_df = partitioned_data[1]['discharge'] # data table for first␣
↪discharge
```

To extract column data such as voltage and capacity, simply call the corresponding keys, which are the column names in the .csv file. Note that specific capacity was automatically calculated and added to the table by the `toDataFrame` function based off the mass you inputted.

```
>>> first_charge_voltage = first_charge_df['Voltage'].tolist()
>>> first_charge_capacity = first_charge_df['Charge_Capacity'].tolist()
>>> first_discharge_voltage = first_discharge_df['Voltage'].tolist()
>>> first_discharge_capacity = first_discharge_df['Discharge_Capacity'].tolist()
```

Then, you can plot the voltage-capacity curves:

```
>>> import matplotlib.pyplot as plt
>>> plt.plot(first_charge_capacity, first_charge_voltage, '-', label='First Charge')
>>> plt.plot(first_discharge_capacity, first_discharge_voltage, '-', label='First␣
↪Discharge')
>>> plt.xlabel('Capacity [mAh/g]')
>>> plt.ylabel('Voltage [V]')
>>> plt.legend()
>>> plt.show()
```

CHAPTER

FOUR

USAGE

To use Arbin Electrochemical Tools in a project:

```
import electrochem
```

CHAPTER

FIVE

CONTRIBUTING

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

5.1 Types of Contributions

5.1.1 Report Bugs

Report bugs at https://github.com/vince-wu/electrochem/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

5.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

5.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

5.1.4 Write Documentation

Arbin Electrochemical Tools could always use more documentation, whether as part of the official Arbin Electrochemical Tools docs, in docstrings, or even on the web in blog posts, articles, and such.

5.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/vince-wu/electrochem/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

5.2 Get Started!

Ready to contribute? Here's how to set up *electrochem* for local development.

1. Fork the *electrochem* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/electrochem.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv electrochem
$ cd electrochem/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 electrochem tests
$ python setup.py test or pytest
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

5.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 3.5, 3.6, 3.7 and 3.8, and for PyPy. Check https://travis-ci.com/vince-wu/electrochem/pull_requests and make sure that the tests pass for all supported Python versions.

5.4 Tips

To run a subset of tests:

```
$ pytest tests.test_electrochem
```

5.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ bump2version patch # possible: major / minor / patch
$ git push
$ git push --tags
```

Travis will then deploy to PyPI if tests pass.

CHAPTER

SIX

CREDITS

6.1 Development Lead

- Vincent Wu <vincentwu@ucsb.edu>

6.2 Contributors

None yet. Why not be the first?

CHAPTER

SEVEN

HISTORY

7.1 0.1.0 (2020-07-28)

- First release on PyPI.

CHAPTER

EIGHT

INDICES AND TABLES

- genindex
- modindex
- search