syncing Documentation

Release 1.0.9

Vincenzo Arcidiacono

Nov 15, 2023

TABLE OF CONTENTS

2023-11-15 12:10:00

https://github.com/vinci1it2000/syncing

https://pypi.org/project/syncing/

http://syncing.readthedocs.io/

https://github.com/vinci1it2000/syncing/wiki/

http://github.com/vinci1it2000/syncing/releases/

https://donorbox.org/syncing

data, synchronisation, re-sampling

- Vincenzo Arcidiacono <vinci1it2000@gmail.com>

CHAPTER

ONE

WHAT IS SYNCING?

syncing is an useful library to synchronise and re-sample time series.

synchronisation is based on the **fourier transform** and the **re-sampling** is performed with a specific interpolation method.

CHAPTER

TWO

INSTALLATION

To install it use (with root privileges):

```
$ pip install syncing
```

Or download the last git version and use (with root privileges):

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

2.1 Install extras

Some additional functionality is enabled installing the following extras:

- cli: enables the command line interface.
- plot: enables to plot the model process and its workflow.
- dev: installs all libraries plus the development libraries.

To install syncing and all extras (except development libraries), do:

```
$ pip install syncing[all]
```

2.1.1 What is syncing?

syncing is an useful library to synchronise and re-sample time series.

synchronisation is based on the **fourier transform** and the **re-sampling** is performed with a specific interpolation method.

2.1.2 Installation

To install it use (with root privileges):

```
$ pip install syncing
```

Or download the last git version and use (with root privileges):

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

2.1.2.1 Install extras

Some additional functionality is enabled installing the following extras:

- cli: enables the command line interface.
- plot: enables to plot the model process and its workflow.
- dev: installs all libraries plus the development libraries.

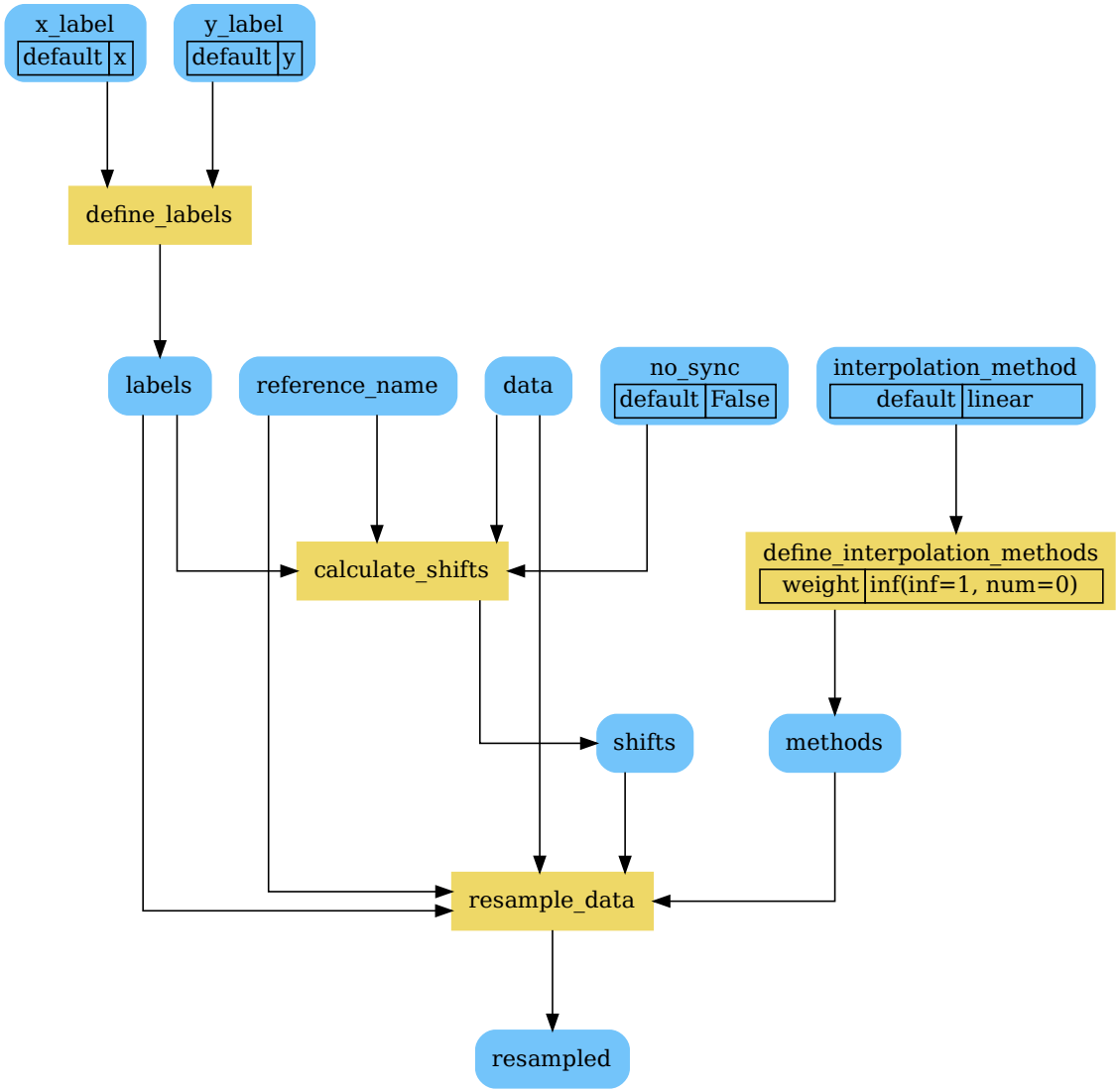
To install syncing and all extras (except development libraries), do:

```
$ pip install syncing[all]
```

2.1.3 Synchronising Laboratory Data

This example shows how to synchronise two data-sets *obd* and *dyno* (respectively they are the On-Board Diagnostics of a vehicle and Chassis dynamometer) with a reference signal *ref*. To achieve this we use the model syncing model to visualize the model:

```
>>> from syncing.model import dsp
>>> model = dsp.register()
>>> model.plot(view=False)
SiteMap(...)
```



Tip: You can explore the diagram by clicking on it.

First of all, we generate synthetically the data-sets to feed the model:

```
>>> import numpy as np
>>> data_sets = {}
>>> time = np.arange(0, 150, .1)
>>> velocity = (1 + np.sin(time / 10)) * 60
>>> data_sets['ref'] = dict(
...     time=time,                                    # [10 Hz]
...     velocity=velocity / 3.6                       # [m/s]
```

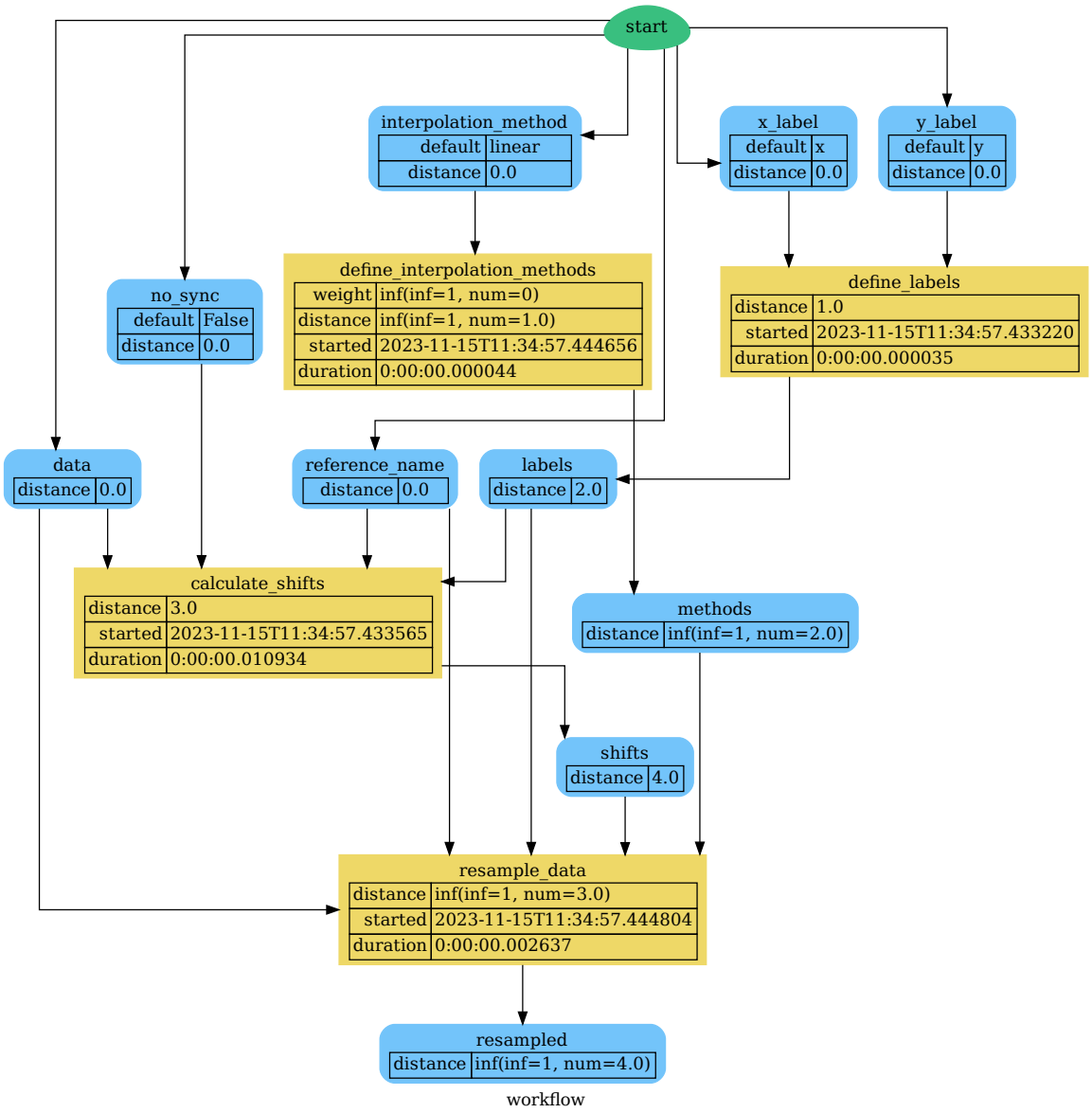
(continues on next page)

(continued from previous page)

```
... )
>>> data_sets['obd'] = dict(
...     time=time[::10] + 12,                                    # 1 Hz
...     velocity=velocity[::10] + np.random.normal(0, 5, 150),   # [km/h]
...     engine_rpm=np.maximum(
...         np.random.normal(velocity[::10] * 3 + 600, 5), 800
...     )                                                        # [RPM]
... )
>>> data_sets['dyno'] = dict(
...     time=time + 6.66,                                        # 10 Hz
...     velocity=velocity + np.random.normal(0, 1, 1500)         # [km/h]
... )
```

To synchronise the data-sets and plot the workflow:

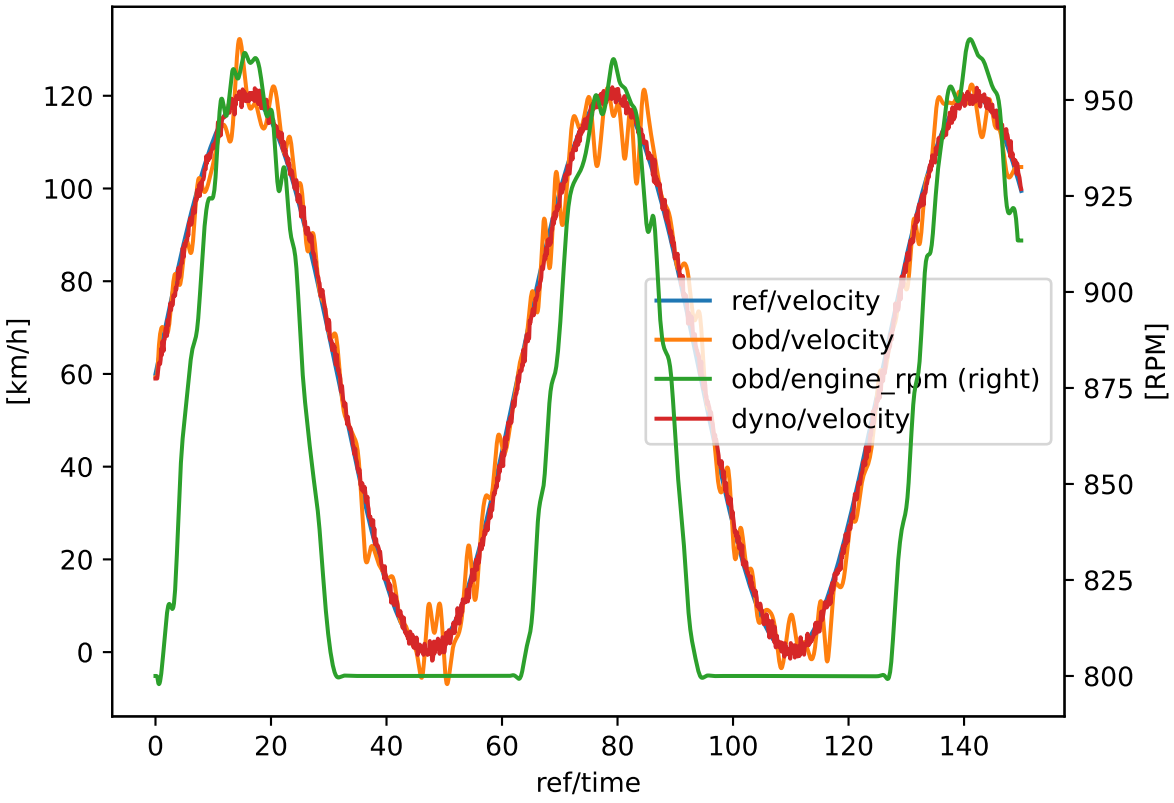
```
>>> from syncing.model import dsp
>>> sol = dsp(dict(
...     data=data_sets, x_label='time', y_label='velocity',
...     reference_name='ref', interpolation_method='cubic'
... ))
>>> sol.plot(view=False)
SiteMap(...)
```



workflow

Finally, we can analyze the time shifts and the synchronised and re-sampled data-sets:

```
>>> import pandas as pd
>>> import schedula as sh
>>> pd.DataFrame(sol['shifts'], index=[0])
     obd  dyno
...
>>> df = pd.DataFrame(dict(sh.stack_nested_keys(sol['resampled'])))
>>> df.columns = df.columns.map('/'.join)
>>> df['ref/velocity'] *= 3.6
>>> ax = df.set_index('ref/time').plot(secondary_y='obd/engine_rpm')
>>> ax.set_ylabel('[km/h]'); ax.right_ax.set_ylabel('[RPM]')
Text(...)
```


120
100
80
60
40
20
0
[km/h]
950
925
900
875
850
825
800
[RPM]
0
20
40
60
80
100
120
140
ref/time
ref/velocity
obd/velocity
obd/engine_rpm (right)
dyno/velocity

2.1.4 Contributing to syncing

If you want to contribute to **syncing** and make it better, your help is very welcome. The contribution should be sent by a *pull request*. Next sections will explain how to implement and submit a new functionality:

- clone the repository
- implement a new functionality
- open a pull request

2.1.4.1 Clone the repository

The first step to contribute to **syncing** is to clone the repository:

- Create a personal fork of the syncing repository on Github.
- Clone the fork on your local machine. Your remote repo on Github is called `origin`.
- Add the original repository as a remote called `upstream`, to maintain updated your fork.
- If you created your fork a while ago be sure to pull `upstream` changes into your local repository.
- Create a new branch to work on! Branch from `dev`.

2.1.4.2 How to implement a new functionality

Test cases are very important. This library uses a data-driven testing approach. To implement a new function I recommend the test-driven development cycle. Hence, when you think that the code is ready, add new test in `test` folder.

When all test cases are ok (`python setup.py test`), open a pull request.

Note: A pull request without new test case will not be taken into consideration.

2.1.4.3 How to open a pull request

Well done! Your contribution is ready to be submitted:

- Squash your commits into a single commit with git's interactive rebase. Create a new branch if necessary. Always write your commit messages in the present tense. Your commit message should describe what the commit, when applied, does to the code – not what you did to the code.
- Push your branch to your fork on Github (i.e., `git push origin dev`).
- From your fork open a *pull request* in the correct branch. Target the project's `dev` branch!
- Once the *pull request* is approved and merged you can pull the changes from `upstream` to your local repo and delete your extra branch(es).

2.1.5 Donate

If you want to support the **syncing** development please donate.

2.1.6 API Reference

The core of the library is composed from the following modules:

`syncing`	Defines the file processing chain model *dsp*.

2.1.7 Changelog

2.1.7.1 v1.0.9 (2023-11-15)

Feat

- (bin): Update publish script.
- (doc): Add Read the Docs configuration file.
- (test): Add github actions.

Fix

- (doc): Correct issues links.
- (read): Correct bug due to pandas version.
- (doc): Remove broken badge.
- (setup): Correct setup config file.

2.1.7.2 v1.0.8 (2023-01-27)

Feat

- (core): Update build scripts.
- (doc): Update Copyright.

Fix

- (core): Correct bug due to schedula version.

2.1.7.3 v1.0.7 (2021-11-08)

Feat

- (core): Add python 3.9.

Fix

- (cli): Correct default options.

2.1.7.4 v1.0.6 (2021-01-04)

Fix

- (rw): Use *openpyxl* to read *xlsx* files.
- (doc): Update copyrights.

2.1.7.5 v1.0.5 (2020-03-31)

Feat

- (core): Update build script.
- (test): Add test cases for the new option commands.
- (setup): Update schedula requirement.
- (cli): Add option to execute only the re-sampling.
- (doc): Add code of conduct.
- (github): Add issues and pull requests templates.
- (core): Add support for python 3.8 and drop python 3.5 and drop *appveyor*.
- (cli): Add option to filter out the *data-names* to work.

Fix

- (setup) #5: Drop setuptools dependency from *setup.py*
- (cli): Correct logging level.
- (travis): Remove graphviz installation in travis.
- (travis): Correct travis test execution.
- (doc): Correct PEP8.
- (rtd): Add missing requirements.
- (rtd): Correct importing issue.
- (doc): Correct contrib.
- (setup): Add missing test requirements.

- (setup): Remove *nose* from *setup_requires*.
- (build): Improve cleaning.

2.1.7.6 v1.0.4 (2019-08-21)

Feat

- (syncing): Remove empty columns.

2.1.7.7 v1.0.3 (2019-08-20)

Feat

- (setup) #3: Add env *ENABLE_SETUP_LONG_DESCRIPTION*.
- (setup) #4: Build as *universal* wheel.
- (write): Add un-synced data to output file.

2.1.7.8 v1.0.2 (2019-07-15)

Fix

- (setup) #2: Correct setup description.

2.1.7.9 v1.0.1 (2019-07-12)

Feat

- (cli): Show options defaults.

Fix

- (setup) #1: Update to canonical pypi name of beautifulsoup4.

2.1.7.10 v1.0.0 (2019-02-23)

Feat

- (doc): Add sphinx documentation.
- (appveyor, travis): Configure continuous integration.
- (test): Add test cases.
- (setup): Add setup script.
- (doc): Add documentation.
- (cli): Add command line interface.

- (core): Add processing chain model.
- (rw): Add *read* and *write* models.
- (model): Add model.

Fix

- (test): Ignore errors when deleting temp folder.
- (setup): Correct requirements.
- (test): Skip doctest of DataFrame.
- (test): Correct test case number approx.

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search