GECo Statistics Documentation

Release 0.2.10

Stefan Countryman

March 17, 2016

Contents:

Introduction

1.1 Purpose

geco_stat can be used to easily generate histograms, statistics, and anomaly reports for timing signals generated by LIGO interferometers. It is meant to simplify the handling of massive amounts of diagnostic data by providing a simple, organized interface to relevant statistics, easy progress tracking for large jobs, effortless data visualization, and convenient ways to combine reports, so that very long timeseries can be efficiently analyzed in parallel and then recombined into single, monolithic reports covering entire eras.

Future versions of *geco_statistics* will also provide tools for automating report generation, reducing the amount of effort that has to go into creating reports on LIGO's functioning.

1.2 Intended Use Cases

This is a python-2.6 compatible Python module intended for use in LIGO production environments. Its python dependencies are minimal, which means it should work out of the box on most python distributions for viewing and manipulating existing reports generated using the package. At time of writing, generating *new* reports requires tools installed in LIGO production environments; it is intended primarily to be run on LIGO servers, though future implementations may integrate remote access tools.

If you are just interested in viewing existing diagnostic report data, you can use Python, though iPython is recommended for interactive use. You can install iPython from the project website. If you are running this module on a LIGO production environment, no additional dependencies should be necessary.

1.3 Getting Started

Install with `pip`:

```
pip install --upgrade geco-stats
```

If you don't have root privileges (as is probably the case on a LIGO server), change to the directory where you would like to work and run:

```
virtualenv env
source env/bin/activate
pip install --upgrade geco-stats
```

When you are done using your `virtualenv`, you can run `deactivate` to go back to using your regular system install of `python` and associated tools.

If you are developing or working from source, visit the github repository.

CHAPTER 2

geco_stat module

CHAPTER 3

Bibliography (Useful Links)

The following links have proved useful in creating, documenting, testing, and deploying this package.

3.1 Anomaly and Outlier Detection

- Some strategies for outlier and anomaly detection (and a good explanation of the differences between the two).

3.2 Packaging and Distribution using PyPI and pip

- How to structure the directories in your python package.
- How to give everything in your project the same version and release numbers, from Read The Docs documentation (via Sphinx) to the module API and the PyPI distribution version.
- Quick and dirty intro to PyPI.
- Comprehensive list of `list_classifiers` used to describe your package.
- Official guide to getting started with PyPI packaging and distribution.
- Setting up a `requirements.txt` file that automatically gets required dependencies from the `setup.py` file.
- More discussion of the `setup.py install_requires` variable.
- Including extra requirements for nonstandard usage.
- Adding a `publish` and autoversion feature to `setup.py`.

3.3 Running a Python Script from the Command Line

- Simple execution of a module as a script (not from a package, but in the current directory).
- Quick explanation of how to use entry points to accomplish this in a package.
- How to make sure that only a single instance of your code is running.
- A stackoverflow question on the topic with some nice alternative approaches.
- Documentation on `sys.excepthook`, used to call cleanup code right before exiting when your program crashes.

3.4 Using Read the Docs for Documentation

- Official getting started guide.
- Astropy is a very good (and thorough) example of advanced Read The Docs support in Python.
- Official solution regarding how to fix Read the Docs build failures resulting from hard c dependencies (like h5py requiring HDF5).
- Example of the above.
- Using the official Read the Docs Theme in your homemade documentation.

3.5 Writing in Restructured Text (reST)

- I'm used to markdown, so this comparison of Restructured Text (reST) was very helpful.

3.6 Python

- A very good guide to python method decorators.
- Detailed description of `super()`.

3.7 Python on Travis CI:

- Getting started with `.travis.yml` for python.
- Deploying to PyPI using Travis.
- Fix build failures due to missing HDF5 dependency (a good hint that this is the problem you are facing is a missing `hdf5.h` header file warning in your logfile).
- Using system site packages (the `system_site_packages` option in `.travis.yml`) to use the `apt` versions of python (NOTE: you *should not* do this for any packages that need to be tested on many versions of Python, since generally only one or two versions will be available via `apt`. See the next link for details.).
- Why `system_site_packages` breaks multi-version tests.
- More info on the `system_site_packages` option breaks multi-version tests, **plus** a very good example of how to test system site package versions of python *without* flagging this option, using the syntax:

```
  - "pypy"
  - 2.7
  - "2.7_with_system_site_packages"
```

  (inclusion of other versions follows the same pattern).
- Test special requirements for each python version.

3.8 Vim Fun

- Negative regex matching.

- Running python code from within vim.
- Saving vim macros.
- Put backspace in a vim macro (make sure to use double-quotes).

3.9 Git Tricks

- Use git show to `cat` the contents of an old revision of a file.
- See the differences in a particular file between revisions.
- Tag your commits with, e.g., version information
- Add hooks to your commits so that `git` automatically performs certain tasks before/after operations like `git commit`. Useful for, e.g., auto-tagging commits when the version updates!

3.10 Python Versions

- `execfile` no longer exists in Python3.x; replace it with this.
- `/` always returns a float in Python3.x; use `//` for integer division in any version of Python.

3.11 Documentation and Testing using Sphinx

- http://thomas-cokelaer.info/tutorials/sphinx/docstring_python.html
- https://docs.python.org/2/library/doctest.html
- http://www.sphinx-doc.org/en/stable/extensions.html
- http://www.sphinx-doc.org/en/stable/ext/doctest.html
- http://www.sphinx-doc.org/en/stable/ext/autosummary.html
- http://www.sphinx-doc.org/en/stable/ext/math.html

3.12 Misc.

- How to schedule jobs on UNIX systems using `chrontab`.
- How to schedule larger cluster jobs using Condor.
- How to determine what OS you are running on.
- How to check if a program or command exists from within a bash script.

Indices and tables

- genindex
- modindex
- search