SnakeTools Documentation

Release 0.0.7

Gus Dunn

Dec 18, 2017

Contents

Contents:

CHAPTER 1

SnakeTools

Small library of helper tools for setting up, graphing, and working with Snakemake rules.

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

1.1 Features

- **`SnakeRun` object to initialize and manage information common to the whole run, such as:**
  - a copy of the config values from the config file provided to `snakemake`.
  - a place to store global variables needed throughout the run.
  - more
- **`SnakeRule` object to manage the initialization and deployment of rule-specific information including:**
  - the rule name
  - a default out directory deduced from the SnakeRun object
  - a default log file path
  - a "pretty name" for the rule to be displayed in the DAG graphs.
  - attributes that store the input, output, and params values for later use.
  - a copy of the values specific to this rule from the original configuration file.
  - more
- `recode_graph` function that cleans up the default output of `snakemake --dag` and allows the use of pretty names stored in the `SnakeRule` objects.

1.2 Credits

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

CHAPTER 2

Installation

2.1 Stable release

To install SnakeTools, run this command in your terminal:

```
$ pip install snaketools
```

This is the preferred method to install SnakeTools, 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 SnakeTools can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/xguse/snaketools
```

Or download the tarball:

```
$ curl  -OL https://github.com/xguse/snaketools/tarball/master
```

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

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

CHAPTER 3

Usage

To use SnakeTools in a project:

```
import snaketools
```

CHAPTER 4

Source Code Documentation

4.1 snaketools package

4.1.1 Submodules

4.1.2 snaketools.errors module

Provide error classes for snaketools.

exception snaketools.errors.**NotImplementedYet**(*msg=None*)
Bases: NotImplementedError, *snaketools.errors.SnaketoolsError*

Raise when a section of code that has been left for another time is asked to execute.

__init__(*msg=None*)
Set up the Exception.

exception snaketools.errors.**SnaketoolsError**
Bases: Exception

Base error class for veoibd-synapse-data-manager.

exception snaketools.errors.**ValidationError**
Bases: *snaketools.errors.SnaketoolsError*

Raise when a validation/sanity check comes back with unexpected value.

4.1.3 snaketools.snaketools module

Provide code supporting the running and automating of Snakemake rules.

snaketools.snaketools.**apply_template**(*template*, *keywords*)
Return a list of strings of form template with values in keywords inserted.

Parameters

- **template** (str) – a string containing keywords ({kw_name}).
- **keywords** (dict-like) – dict with keys of appropriate keyword names and values as equal length ORDERED lists with the correct values to be inserted.

snaketools.snaketools.**pathify_by_key_ends**(*dictionary*)
Return a dict that has had all values with keys containing the suffixes: '_FILE', '_PATH' or '_DIR' converted to Path() instances.

Parameters **dictionary** (*dict-like*) – Usually the loaded, processed config file as a *dict*.

Returns Modified version of the input.

Return type dict-like

class snaketools.snaketools.**SnakeRun**(*cfg*, *snakefile*)
Bases: object

Initialize and manage information common to the whole run.

__init__(*cfg*, *snakefile*)
Initialize common information for a run.

class snaketools.snaketools.**SnakeRule**(*run*, *name*, *pretty_name=None*)
Bases: object

Manage the initialization and deployment of rule-specific information.

__init__(*run*, *name*, *pretty_name=None*)
Initialize logs, inputs, outputs, params, etc for a single rule.

_import_config_dict()
Import configuration values set for this rule so they are directly accessable as attributes.

snaketools.snaketools.**recode_graph**(*dot*, *new_dot*, *pretty_names*, *rules_to_drop*, *color=None*, *use_pretty_names=True*)
Change *dot* label info to pretty_names and alter styling.

snaketools.snaketools.**rewrite_snakefile_no_rules**(*infile*, *outfile*)
Write new file, omitting the snakemake grammar sections.

4.1.4 Module contents

Top-level package for SnakeTools.

CHAPTER 5

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/xguse/snaketools/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

SnakeTools could always use more documentation, whether as part of the official SnakeTools 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/xguse/snaketools/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 *snaketools* for local development.

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

```
$ git clone git@github.com:your_name_here/snaketools.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 snaketools
$ cd snaketools/
$ 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 snaketools tests
$ python setup.py test or py.test
$ 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 2.6, 2.7, 3.3, 3.4 and 3.5, and for PyPy. Check https://travis-ci.org/xguse/snaketools/pull_requests and make sure that the tests pass for all supported Python versions.

5.4 Tips

To run a subset of tests:

```
$ py.test tests.test_snaketools
```

CHAPTER 6

Credits

6.1 Development Lead

- Gus Dunn <w.gus.dunn@gmail.com>

6.2 Contributors

None yet. Why not be the first?

CHAPTER 7

Change Log

7.1 v0.0.7 / 2017-12-18

- change pyup check to monthly
- update reqs from PYUP
- snaketools: SnakeRule now registers with SnakeRun
- snaketools: added attr SnakeRule.extra for more params
- snaketools: added attr SnakeRun.rules
- snaketools: use __all__ for importing from file
- update makefile
- update reqs

7.2 v0.0.6 / 2017-10-26

- added rewrite_snakefile_no_rules()
- flake8
- requirements.txt: removed dev-reqs
- requirements.txt: pinned flake8
- setup.py: upgraded to read from req files
- MANIFEST.in: include req files
- upgraded Makefile
- tox.ini: set line-length etc
- setup.cfg: ignore W292

- setup.cfg: exclude lib & bin from flake8
- updated .gitignore
- added coveralls badge
- HISTORY.rst: replaced header text

7.3 v0.0.5 / 2017-10-10

- requirements_dev.txt: update and pin reqs
- flake8 fixes
- tox.ini: simplified config
- added flake8 to reqs

7.4 v0.0.4 / 2017-10-10

- added preliminary test suite
- Makefile: changed *install* to use *pip install -e .*
- added example files for testing
- requirements.txt: created with *pipreqs*
- snaketools.py: reorder functions
- snaketools.py: formatting
- ignore .vscode/
- pin all reqs since pyup now manages

7.5 v0.0.3 / 2017-09-15

- Configure pyup
- SnakeRun.d -> SnakeRune.interim_dir

7.6 v0.0.2 / 2017-09-06

- fixed bumpversion artifact
- errors.py: pulls metadata from top module
- updated dev reqs for doc building
- activated travis ci
- Set up flake8 configuration
- Docs build corrected

7.7 v0.0.1 / 2017-09-06

- README.rst: added prelim description of features.
- snaketools.py: fixed typo
- Initial commit

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

Python Module Index

S

Index

Symbols

A

N

P

R

S

V