lightcone Documentation

Release 0.1.0

Simon Birrer

May 18, 2023

CONTENTS

Contents

animated lightcone simulation of gravitational lensing using lenstronomy .

- Free software: BSD license
- Documentation: https://lightcone_lenstronomy.readthedocs.io.

CHAPTER

ONE

CONTENTS:

1.1 lightcone

Contents

animated lightcone simulation of gravitational lensing using lenstronomy .

- Free software: BSD license
- Documentation: https://lightcone_lenstronomy.readthedocs.io.

1.2 Installation

1.2.1 Stable release

To install lightcone, run this command in your terminal:

```
$ pip install lightcone
```

This is the preferred method to install lightcone, 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.

1.2.2 From sources

The sources for lightcone can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/sibirrer/lightcone
```

Or download the tarball:

```
$ curl -OJL https://github.com/sibirrer/lightcone/tarball/master
```

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

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

1.2.3 FFMPEG

FFMPEG installation is required to create MP4 files of the animated lightcurves.

- Free download: https://ffmpeg.org/download.html.

After downloading, make sure you include the FFMPEG binary directory (e.g. C:\users\user\ffmpeg\bin) to your PATH.

On Windows you can edit your environment variables and edit your Path to add to it. You may want to restart your PC after doing so too.

1.2.4 Paltas

To install Paltas, run this command in your terminal:

```
$ pip install paltas
```

Lightcone currently uses Paltas version 0.1.1.

If using Paltas with Lightcone, the following catalog should be downloaded:

- COSMOS 23.5 Magnitude Catalog: https://github.com/GalSim-developers/GalSim/wiki/RealGalaxy%20Data

After downloading and unzipping the catalog, make sure you go into the config file (e.g. test_config.py) and assign the cosmos_folder variable to the path name of where you unzipped the catalog.

1.3 Usage

To use lightcone in a project:

```
import lightcone
```

1.4 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:

1.4.1 Types of Contributions

Report Bugs

Report bugs at https://github.com/sibirrer/lightcone/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.

Fix Bugs

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

Implement Features

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

Write Documentation

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

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/sibirrer/lightcone/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 :)

1.4.2 Get Started!

1.4.3 GitHub Workflow

You should only need to do this step once

First *fork* the lightcone repository. A fork is your own remote copy of the repository on GitHub. To create a fork:

1. Go to the lightcone Repository
2. Click the **Fork** button (in the top-right-hand corner)
3. Choose where to create the fork, typically your personal GitHub account

Next *clone* your fork. Cloning creates a local copy of the repository on your computer to work with. To clone your fork:

```
git clone https://github.com/<your-account>/lightcone.git
```

Finally add the `lightcone-project` repository as a *remote*. This will allow you to fetch changes made to the codebase. To add the `lightcone-project` remote:

```
cd lightcone
git remote add lightcone-project https://github.com/sibirrer/lightcone.git
```

Create a *branch* off the `lightcone-project` main branch. Working on unique branches for each new feature simplifies the development, review and merge processes by maintaining logical separation. To create a feature branch:

```
git fetch lightcone-project
git checkout -b <your-branch-name> lightcone-project/main
```

Write the new code you would like to contribute and *commit* it to the feature branch on your local repository. Ideally commit small units of work often with clear and descriptive commit messages describing the changes you made. To commit changes to a file:

```
git add file_containing_your_contribution
git commit -m 'Your clear and descriptive commit message'
```

Push the contributions in your feature branch to your remote fork on GitHub:

```
git push origin <your-branch-name>
```

Note: The first time you *push* a feature branch you will probably need to use *–set-upstream origin* to link to your remote fork:

```
git push --set-upstream origin <your-branch-name>
```

When you feel that work on your new feature is complete, you should create a *Pull Request*. This will propose your work to be merged into the main sim-pipeline repository.

1. Go to lightcone Pull Requests
2. Click the green **New pull request** button
3. Click **compare across forks**
4. Confirm that the base fork is `sibirrer/lightcone` and the base branch is `main`
5. Confirm the head fork is `<your-account>/lightcone` and the compare branch is `<your-branch-name>`
6. Give your pull request a title and fill out the the template for the description
7. Click the green **Create pull request** button

A series of automated checks will be run on your pull request, some of which will be required to pass before it can be merged into the main codebase:

- `Tests` (Required) runs the **`unit tests`_** in four predefined environments; *latest supported versions*, *oldest supported versions*, *macOS latest supported* and *Windows latest supported*. Click "Details" to view the output including any failures.
- `Code Style` (Required) runs flake8 to check that your code conforms to the PEP 8 style guidelines. Click "Details" to view any errors.

- `codecov` reports the test coverage for your pull request; you should aim for *codecov/patch — 100.00%*. Click "Details" to view coverage data.
- `docs` (Required) builds the **`docstrings`_** on readthedocs. Click "Details" to view the documentation or the failed build log.

As you work on your feature, new commits might be made to the `lightcone-project` main branch. You will need to update your branch with these new commits before your pull request can be accepted. You can achieve this in a few different ways:

- If your pull request has no conflicts, click **Update branch**
- If your pull request has conflicts, click **Resolve conflicts**, manually resolve the conflicts and click **Mark as resolved**
- *merge* the `lightcone-project` main branch from the command line:

```
git fetch lightcone-project
git merge lightcone-project/main
```

- *rebase* your feature branch onto the `lightcone-project` main branch from the command line:

```
git fetch lightcone-project
git rebase lightcone-project/main
```

Warning: It is bad practice to *rebase* commits that have already been pushed to a remote such as your fork. Rebasing creates new copies of your commits that can cause the local and remote branches to diverge. `git push --force` will **overwrite** the remote branch with your newly rebased local branch. This is strongly discouraged, particularly when working on a shared branch where you could erase a collaborators commits.

For more information about resolving conflicts see the GitHub guides:

- Resolving a merge conflict on GitHub
- Resolving a merge conflict using the command line
- About Git rebase

More information regarding the usage of GitHub can be found in the GitHub Guides.

1.4.4 Coding Guidelines

Before your pull request can be merged into the codebase, it will be reviewed by one of the sim-pipeline developers and required to pass a number of automated checks. Below are a minimum set of guidelines for developers to follow:

- lightcone is compatible with Python>=3.9 (see setup.cfg). sim-pipeline *does not* support backwards compatibility with Python 2.x; *six*, *__future__* and *2to3* should not be used.
- All contributions should follow the PEP8 Style Guide for Python Code. We recommend using flake8 to check your code for PEP8 compliance.
- Importing lightcone should only depend on having NumPy, SciPy and Astropy installed.
- Code will be grouped into submodules based on broad applications.
- For more information see the Astropy Coding Guidelines.

Pull requests will require existing unit tests to pass before they can be merged. Additionally, new unit tests should be written for all new public methods and functions. Unit tests for each submodule are contained in subdirectories called `tests` and you can run them locally using `pytest`. For more information see the Astropy Testing Guidelines.

If your unit tests check the statistical distribution of a random sample, the test outcome itself is a random variable, and the test will fail from time to time. Please mark such tests with the `@pytest.mark.flaky` decorator, so that they will be automatically tried again on failure. To prevent non-random test failures from being run multiple times, please isolate random statistical tests and deterministic tests in their own test cases.

All public classes, methods and functions require docstrings. You can build documentation locally by installing sphinx-astropy and calling `make html` in the `docs` subdirectory. Docstrings should include the following sections:

- Description
- Parameters
- Notes
- References

For more information see the Astropy guide to Writing Documentation.

1.5 Credits

1.5.1 Developers

- Simon Birrer sibirrer
- Jonathan Benz JonathanBenz

1.6 History

1.6.1 0.1.0 (2022-02-12)

- First release on PyPI.

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search