Planetary Test Data Documentation

Release 0.4.0

PlanetaryPy Developers

Jan 22, 2018

Contents

Contents:

CHAPTER 1

Planetary Test Data

Planetary Test Data contains a list of planetary data for software testing purposes and utilities to retrieve them.

- Free software: BSD license

1.1 Features

- Downloads a core set of sample Planetary test data into `./mission_data/` or `./tests/mission_data/` if `./tests/` exists.

1.2 TODO

- Download to central cache directory and use symbolic links to share data between projects or other locations.
- Find smaller example images to reduce download times.
- Command line usage improvements
  - Include a mode that allows users to somehow specify subsets of data to retrieve. Perhaps selecting by mission or instrument name.
- Improve label testing.
- Include Mission Names with each product.
- Include product type with each product.

See also the Github issues for this project.

1.3 Usage

To download the core set of planetary test data install this package with pip and then run the command `get_mission_data`:

```
pip install planetary_test_data
get_mission_data
```

Additional usage options are shown below:

```
usage: get_mission_data [-h] [--all] [--file FILE] [--dir DIR] [--tags [TAGS]]
                        [--instruments [INSTRUMENTS]] [--missions [MISSIONS]]

optional arguments:
  -h, --help            show this help message and exit
  --all, -a             Download all products.
  --file FILE, -f FILE  Override default data.json by providing path to custom
                        data.json file.
  --dir DIR, -d DIR     Directory to place test data products in.
  --tags [TAGS], -t [TAGS]
                        Retrieve products whose tags match those provided
                        here.
  --instruments [INSTRUMENTS], -i [INSTRUMENTS]
                        Get products by instrument
  --missions [MISSIONS], -m [MISSIONS]
                        Get products by mission
```

To get a copy of a subset of `data.json`:

```
get_mission_json
```

Additional usage options are shown below:

```
usage: get_mission_json [-h] [--all] [--file FILE] [--dir DIR] [--tags [TAGS]]
                        [--instruments [INSTRUMENTS]] [--missions [MISSIONS]]

optional arguments:
  -h, --help            show this help message and exit
  --all, -a             Download all products.
  --file FILE, -f FILE  Override default data.json by providing path to custom
                        data.json file.
  --dir DIR, -d DIR     Directory to place test data products in.
  --tags [TAGS], -t [TAGS]
                        Retrieve products whose tags match those provided
                        here.
  --instruments [INSTRUMENTS], -i [INSTRUMENTS]
                        Get products by instrument
  --missions [MISSIONS], -m [MISSIONS]
                        Get products by mission
```

1.4 Description

Running `get_mission_data` will do the following:

- If `tests` directory exists it will create `tests/mission_data` if necessary. If `tests` does not exist, it will just create `mission_data` in the current directory.

- The data prodcuts tagged to be `core` products will be downloaded into the download directory.
- Use the `-a` or `--all` flag to get all the images. This, however, is NOT recommended for you will download over 200 images and labels.
- The `-t` or `--tags` will retrieve products matched with the supplied tag. To give multiple tags, use the flag multiple times.
- The `-d` or `--dir` flag can be used to save the images in a new custom location.
- If there exists a custom `data.json` locally and you would rather use that file to download images rather than the default, use the `-f` or `--file` flag and then the path to the `data.json`. This is especially useful if there are test images needed that do not exist in or are not part of the `core` in the default `data.json` (see `get_mission_json` below).
- To get products by mission use the `--mission` or `-m` flag. This will download all the products from the given mission, even non `core` products, unless explicity given `core` as a tag. To specify multiple missions, use the flag multiple times. You must spell the mission the same as spelled in *Missions and Instruments* (case matters!).
- To get products by instruments use the `--instruments` or `-i` flag. This will download all the products from the given instrument, even non `core` products, unless explicity given `core` as a tag. To specify multiple instruments, use the flag multiple times. You must spell the instrument the same as spelled in *Missions and Instruments* (case matters!).
- Only products which do not exist in the download directory will be downloaded.

Running `get_mission_json` will do the following:

- Create a copy of `data.json` in the `tests` or `test` directory. This will just be the `core` data by default. The purpose of getting a copy of the `data.json` is so it is easier to include images in respective projects that are not included in the default `data.json`. Then developers can use the `-f` flag on `get_mission_data` (see above) to use this custom `data.json`.
- If `data.json` already exists, an exception is raised.
- The same flags apply to `get_mission_json` as `get_mission_data`.

1.5 Mission Data

The PDS mission data included in the package can be found here.

The following are core products:

- 0047MH0000110010100214C00_DRCL.IMG
- 0047MH0000110010100214C00_DRCL.LBL
- 1p134482118erp0902p2600r8m1.img
- 1p190678905erp64kcp2600l8c1.img
- 2p129641989eth0361p2600r8m1.img
- 2m132591087cfd1800p2977m2f1.img
- h58n3118.img
- r01090al.img

If there are products you think should be included or removed from this dataset please file a Github issue. New images should be images from instruments that are not already included or different file types (i.e. EDR vs RDR). New core images should be distinctly different than the ones that exist and would expose test and/or edge cases for multiple PlanetaryPy projects/affiliates. For example, if there was not an RGB image included in the core products

(which there is), then that would test image would expose an edge case for many projects. However, it is best to use `get_mission_json` to get a copy of `data.json`, add the desired test images to that json file, and then download images using `get_mission_data -f path/to/data.json`. We recommend using a `make test` command to get the proper mission data before testing.

1.6 data.json Format

The `data.json` file contains PDS product names, urls and other metainformation about the product. This structure will be extended to support generic testing, for instance the `label` key will be changed to a dictionary that includes product label keys and the values found at those keys.

Below is a sample snippet of a `data.json` entry:

```
"1m298459885effa312p2956m2m1.img": {
    "instrument": "MICROSCOPIC IMAGER",
    "label": "PDS3",
    "opens": "True",
    "url": "http://pds-imaging.jpl.nasa.gov/data/mer/opportunity/mer1mo_0xxx/data/
↪sol1918/edr/1m298459885effa312p2956m2m1.img"
},
```

1.7 Missions and Instruments

The following missions and their instruments have products available for testing:

- **2001 Mars Odyssey**
  - Thermal Emission Imaging System
- **Cassini**
  - Cassini Radar
  - Imaging Science Subsystem
  - Imaging Science Subsystem Narrow Angle
  - Visual And Infrared Mapping Spectrometer
- **Chandrayaan-1**
  - Context Camera
  - High Resolution Imaging Science Experiment
  - Mars Color Imager
  - Moon Mineralogy Mapper
- **Clementine**
  - High Resolution Camera
  - Long Wave Infrared Camera
  - Nearinfrared Camera
  - Ultraviolet/Visible (Uv/Vis) Camera
  - Ultraviolet/Visible Camera

- **ESA Mars Express**
  - High Resolution Stereo Camera
- **Galileo**
  - Near-Infrared Mapping Spectrometer
  - Solid_State_Imaging
- **Lunar Reconnaissance Orbiter**
  - Lunar Reconnaissance Orbiter Camera
  - Lyman Alpha Mapping Project
  - Mid Infrared Camera 1
  - Mid Infrared Camera 2
  - Near Infrared Camera 1
  - Near Infrared Camera 2
  - Near Infrared Spectrometer 1
  - Near Infrared Spectrometer 2
  - Total Luminance Photometer
  - Visible Camera
  - Visible Spectrometer
- **MESSENGER**
  - Mercury Dual Imaging System Narrow Angle Camera
  - Mercury Dual Imaging System Narrow Angle Camera, Mercury Dual Imaging System Wide Angle Camera
  - Mercury Dual Imaging System Wide Angle Camera
- **Magellan**
  - Global Topography Data Record
  - Radar
  - Radar System
  - Synthetic-Aperture Radar
- **Mariner 10**
  - Mariner 10
- **Mariner 9**
  - Imaging Science Subsystem
- **Mars Exploration Rover**
  - Alpha Particle X-Ray Spectrometer
  - Descent Camera
  - Front Hazard Avoidance Camera Left
  - Front Hazard Avoidance Camera Right

  - Hazard Avoidance Camera
  - Microscopic Imager
  - Moessbauer Spectrometer
  - Navigation Camera
  - Navigation Camera Left
  - Panoramic Camera
  - Panoramic Camera Left
  - Panoramic Camera Right
  - Panoromic Camera
  - Rock Abrasion Tool
- **Mars Global Surveyor**
  - Mars Orbiter Camera - Wide Angle
  - Mars Orbiter Camera Wide Angle
  - Near Infrared Mapping Spectrometer
- **Mars Pathfinder**
  - Alpha X-Ray Spectrometer
  - Alpha X-Ray Spectrometer (Apxs)
  - Atmospheric Structure Instrument / Meteorology Package
  - Imager For Mars Pathfinder
  - Rover Camera Left
- **Mars Science Laboratory**
  - Front Hazard Avoidance Camera Left String B
  - Mars Descent Imager Camera
  - Mars Hand Lens Imager Camera
  - Mast Camera Left
  - Navigation Camera Left String A
- **Phoenix**
  - Optical Microscope
  - Robotic Arm Camera
  - Surface Stereo Imager
- **Viking Lander**
  - Camera_1
  - Camera_2
- **Viking Orbiter**
  - Viking Visual Imaging Subsystem
  - Visual_Imaging_Subsystem_Camera_A, Visual_Imaging_Subsystem_Camera_B

- **Voyager**
  - Imaging Science Subsystem
  - Imaging Science Subsystem - Narrow Angle Camera

CHAPTER 2

Installation

Download using pip:

```
$ pip install planetary_test_data
```

CHAPTER 3

Usage

See README. It is best to use the commands `get_mission_data` and `get_mission_json`.

CHAPTER 4

Planetary Test Data

Download test data for PlanetaryPy packages

4.1 PlanetaryTestDataProducts

class planetary_test_data.planetary_test_data.**PlanetaryTestDataProducts**(*tags=None*, *all_products=None*, *directory=None*, *data_file=None*, *missions=[]*, *instruments=[]*)

products
List of products that match defined tags

4.2 get_mission_data

planetary_test_data.planetary_test_data.**get_mission_data**(*args*)
Downloads products from data.json

Side Effects: The function uses the returned path from *setup_json_file()*. The function will check to see if each product in the data.json file exists in the mission_data directory. If the product does not exist, the function will download the product to the mission_data directory.

Keyword Arguments: None

4.3 get_mission_json

planetary_test_data.planetary_test_data.**get_mission_json**(*args*)

Download the a subset of or the entire data.json

The data.json will download to the tests or test directory by default. If a tests or test directory does not exist and a directory is not given, an exception will be raised

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/planetarypy/planetary_test_data/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" is open to whoever wants to implement it.

5.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "feature" is open to whoever wants to implement it.

5.1.4 Write Documentation

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

5.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/planetarypy/planetary_test_data/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 *planetary_test_data* for local development.

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

```
$ git clone git@github.com:your_name_here/planetary_test_data.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 planetary_test_data
$ cd planetary_test_data/
$ pip install -r requirements.txt
```

4. Create a branch for local development:

```
$ git checkout master
$ git pull origin master
$ 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:

```
$ make lint
$ make test
$ make test-all
```

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

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

```
$ git add file_was_that_changed.ext
$ 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, 3.5, and 3.6, and for PyPy. Check https://travis-ci.org/planetarypy/planetary_test_data/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_planetary_test_data.py
```

CHAPTER 6

Credits

6.1 Development Lead

- PlanetaryPy Developers <contact@planetarypy.com>

6.2 Contributors

None yet. Why not be the first?

CHAPTER 7

History

7.1 0.4.0 (2018-01-21)

- Made new methods/commands to get products by mission and/or instrument
- Made new methods/command to get a copy of the subset of the data.json file
- Added several new products
- Created new documentation
- Made documentation available on RTD
- Create new tests to get %100 coverage

7.2 0.3.3 (2015-07-27)

- Added 0047MH0000110010100214C00_DRCL.IMG to the core dataset.

7.3 0.3.2 (2015-07-21)

- Added 2m132591087cfd1800p2977m2f1.img to the core dataset.

7.4 0.3.1 (2015-07-13)

- Updated data.json to support extended PDS product label tests.

7.5 0.3.0 (2015-07-13)

- Rewritten to be driven by command line options rather than by file system contents. Run `get_mission_data -h` to see options.

7.6 0.2.0 (2015-07-11)

- Updated `data.json` to contain mission data from many more missions.

7.7 0.1.1 (2015-07-09)

- Fixed Python 3 compativility issue.

7.8 0.1.0 (2015-06-24)

- First release on PyPI.

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

Python Module Index

p

Index

G

P