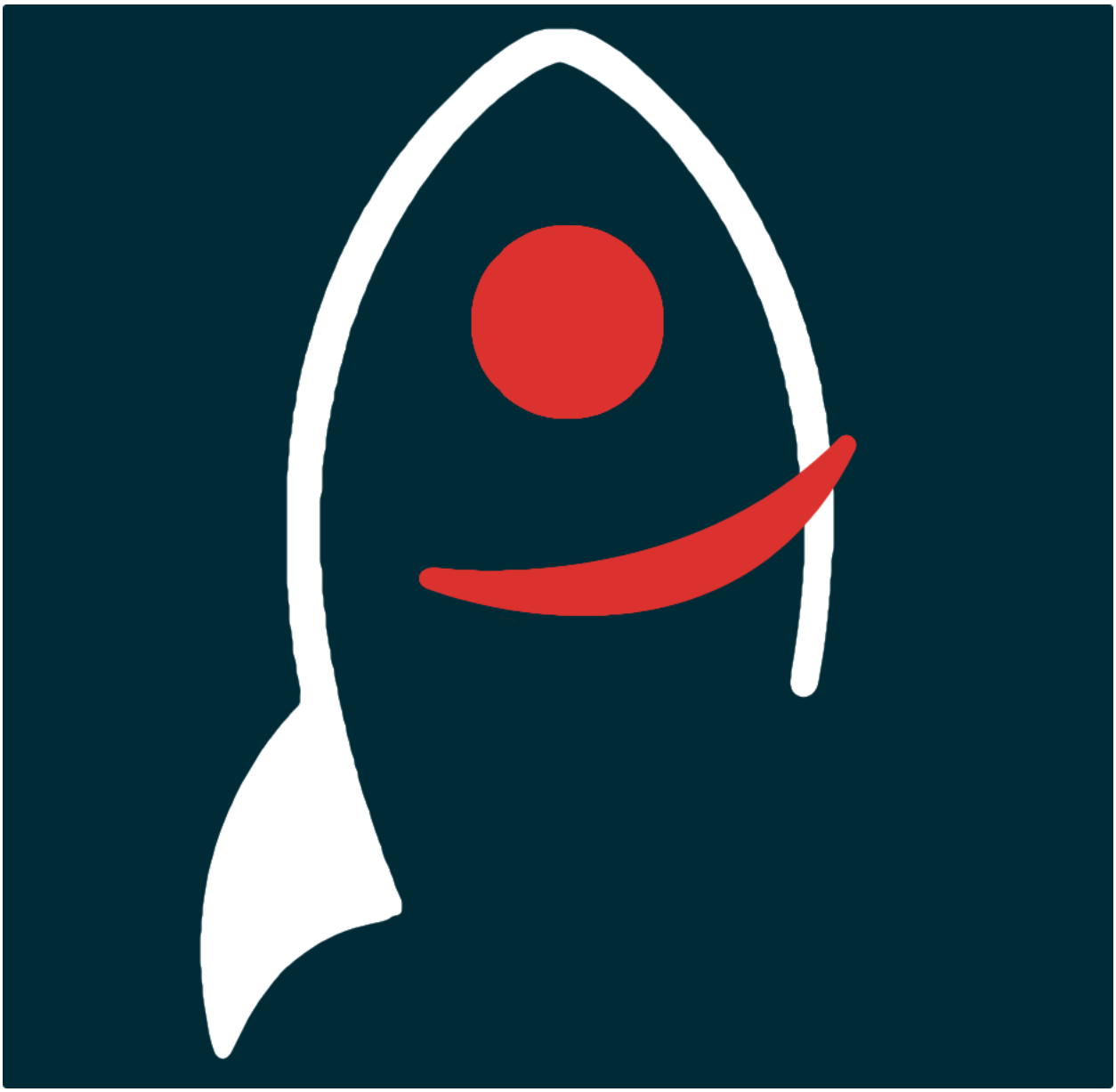


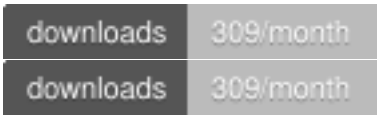
panstamps Documentation

Release v0.6.5

Dave Young

2023

TABLE OF CONTENTS



Download stacked and/or warp image stamps from the STScI PanSTARRS image server via CL or Python API.

Documentation for panstamps is hosted by Read the Docs (development version and master version). The code lives on github. Please report any issues you find here.

Note: If working with warped PS1 images then you need to work off a machine that has an IP address whitelisted by the Pan-STARRS1 data archive, otherwise only stacked images will be available to you. Also *w*-band images are not (yet) accessible from the data archive.

CHAPTER

ONE

FEATURES

-

CHAPTER

TWO

HOW TO CITE PANSTAMPS

If you use `panstamps` in your work, please cite using the following BibTeX entry:

```
@software{Young_panstamps,
    author = {Young, David R.},
    doi = {10.5281/zenodo.8037665},
    license = {GPL-3.0-only},
    title = ,
    url = {https://github.com/thespacedoctor/panstamps}
}
```

2.1 Installation

The easiest way to install panstamps is to use `pip` (here we show the install inside of a conda environment):

```
conda create -n panstamps python=3.7 pip
conda activate panstamps
pip install panstamps
```

Or you can clone the github repo and install from a local version of the code:

```
git clone git@github.com:thespacedoctor/panstamps.git
cd panstamps
python setup.py install
```

To upgrade to the latest version of panstamps use the command:

```
pip install panstamps --upgrade
```

To check installation was successful run `panstamps -v`. This should return the version number of the install.

2.1.1 Troubleshooting on Mac OSX

panstamps uses pillow (a fork of the Python Imaging Library) which requires some external libraries.

If you have issues running panstamps on OSX, try installing Homebrew and running:

```
brew install libtiff libjpeg webp little-cms2
```

2.1.2 Development

If you want to tinker with the code, then install in development mode. This means you can modify the code from your cloned repo:

```
git clone git@github.com:thespacedoctor/panstamps.git
cd panstamps
python setup.py develop
```

Pull requests are welcomed!

2.2 Initialisation

Before using panstamps you need to use the `init` command to generate a user settings file. Running the following creates a yaml settings file in your home folder under `~/.config/panstamps/panstamps.yaml`:

```
panstamps init
```

The file is initially populated with panstamps's default settings which can be adjusted to your preference.

If at any point the user settings file becomes corrupted or you just want to start afresh, simply trash the `panstamps.yaml` file and rerun `panstamps init`.

2.2.1 Modifying the Settings

Once created, open the settings file in any text editor and make any modifications needed.

2.2.2 Basic Python Setup

If you plan to use `panstamps` in your own scripts you will first need to parse your settings file and set up logging etc. One quick way to do this is to use the `fundamentals` package to give you a logger, a settings dictionary and a database connection (if connection details given in settings file):

```
## SOME BASIC SETUP FOR LOGGING, SETTINGS ETC
from fundamentals import tools
from os.path import expanduser
home = expanduser("~")
settingsFile  = home + "/.config/panstamps/panstamps.yaml"
su = tools(
    arguments={"settingsFile": settingsFile},
    docString=__doc__,
)
arguments, settings, log, dbConn = su.setup()
```

2.3 Using Panstamps from the Command-Line

There are 2 ways to use **panstamps**, either via the command-line or import it into your own python code and use it from there.

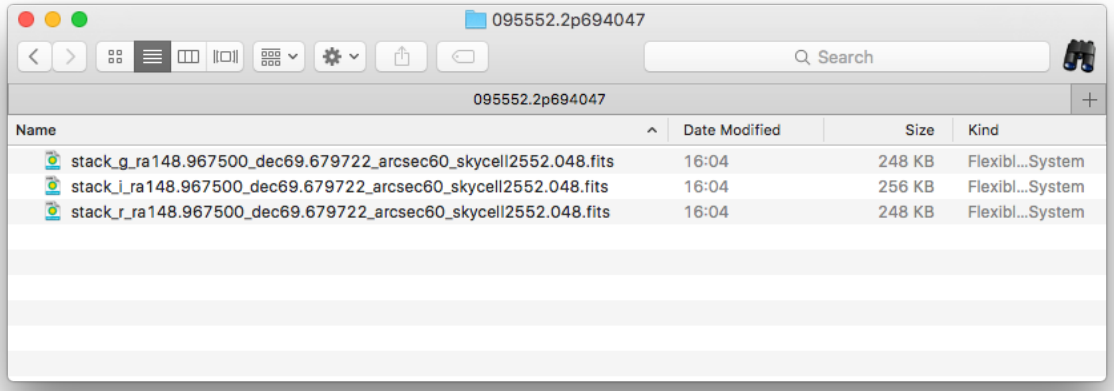
Full usage options can be found by typing:

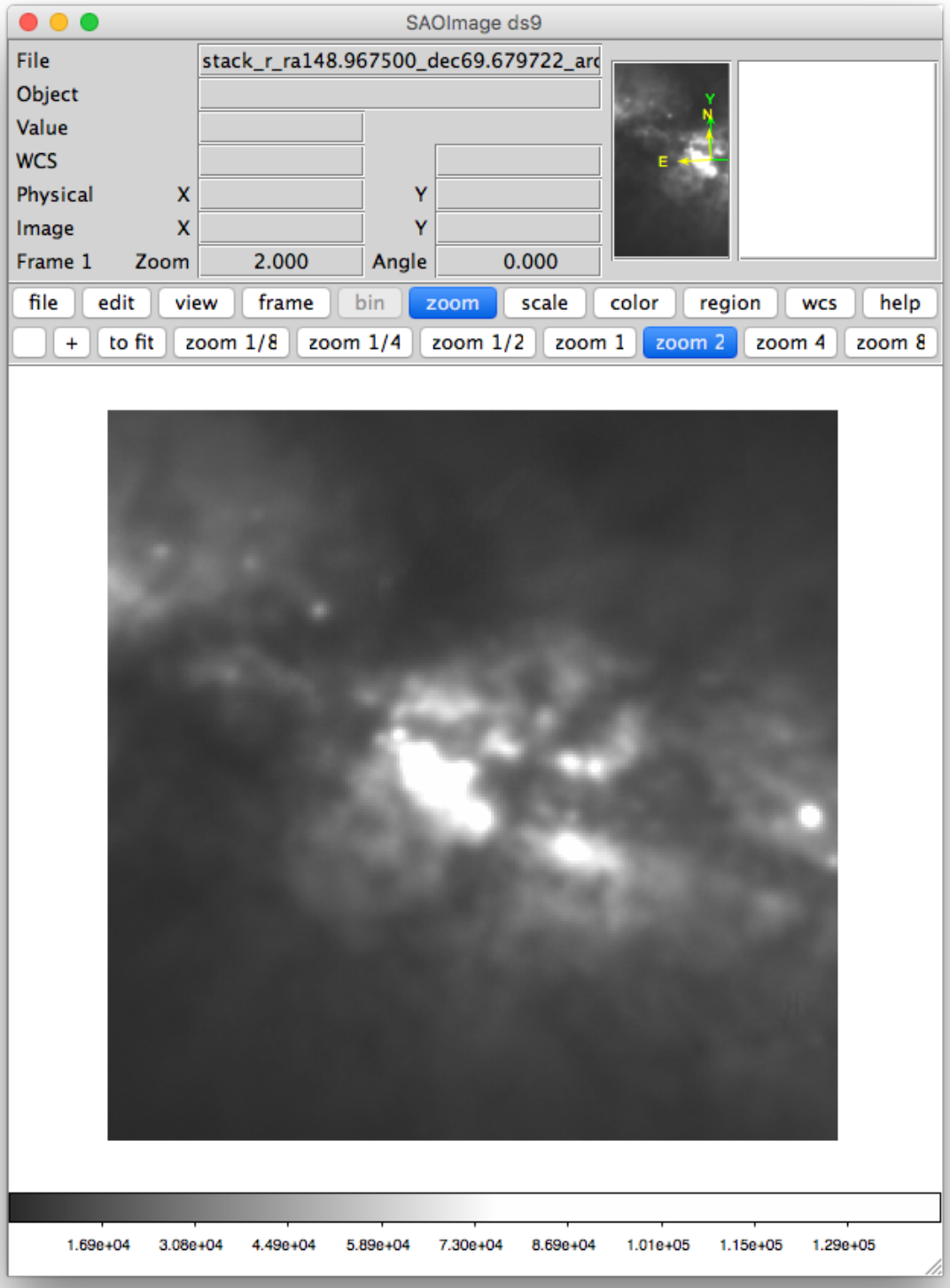
```
panstamps -h
```

Here I'll run through the basics. By default the command will only download the fits files for the location given. To download the stack fits cutouts for M82 run the command:

```
panstamps stack 09:55:52.2 +69:40:47
```

By default the *gri* filter, 1 arcmin fits cutouts are downloaded:

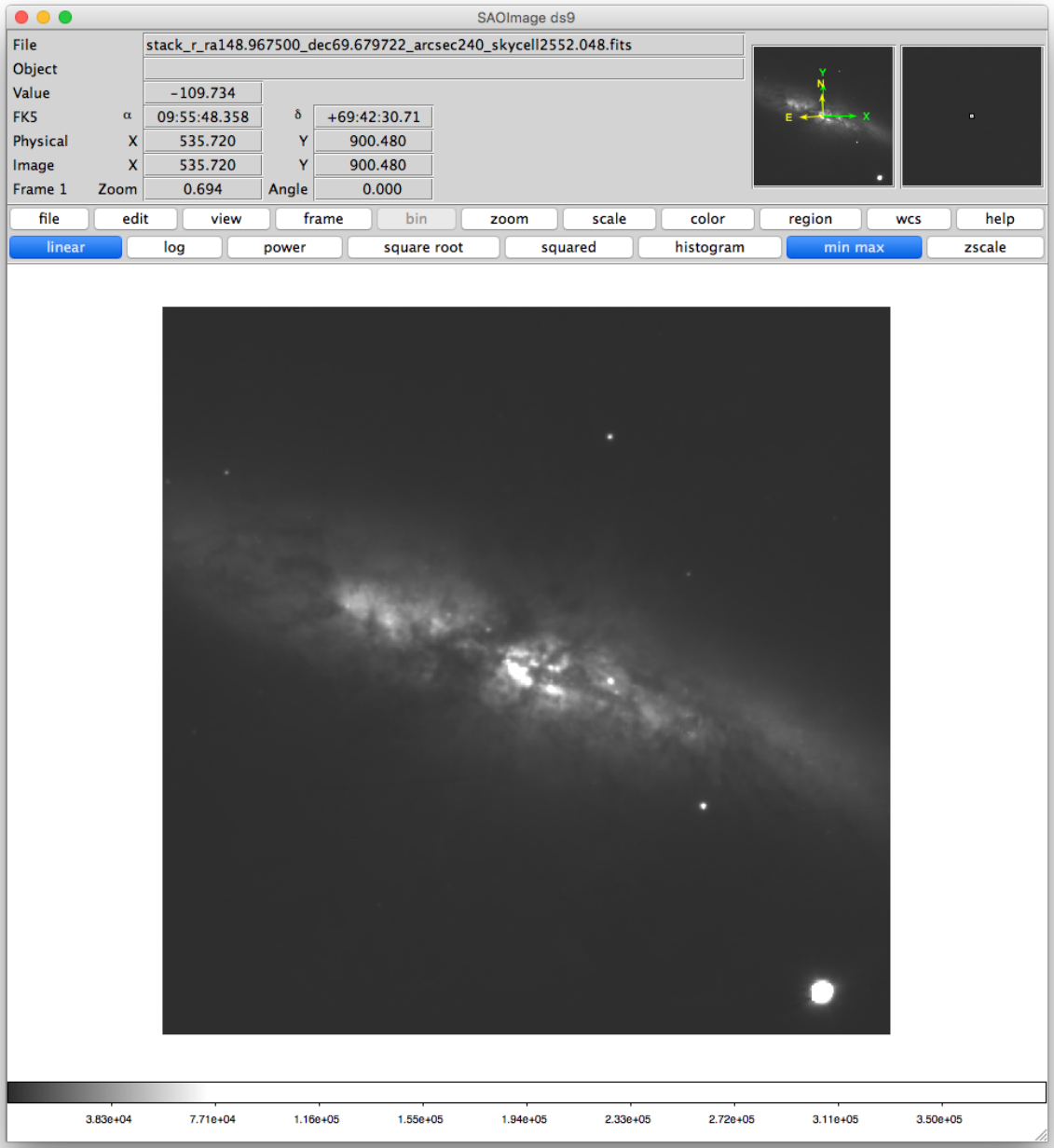




To increase the image width and download all filters, run the command:

```
panstamps --width=4 --filters=griyz stack 09:55:52.2 +69:40:47
```

As you can see we now have a larger cutout:

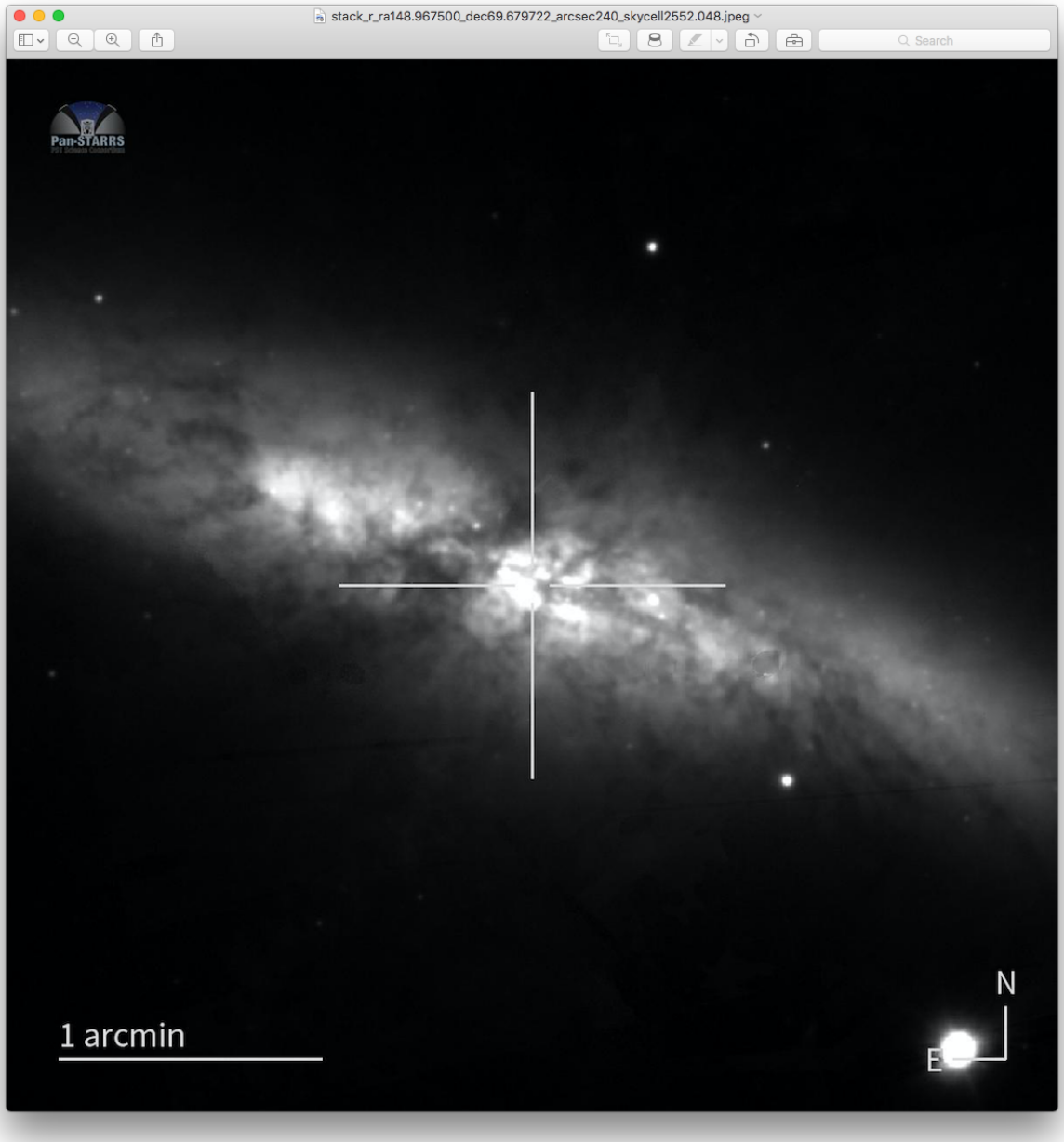


2.3.1 JPEGS

To download the jpegs, and not the fits files rerun the command with the correct flags set. We'll also use the `--downloadFolder` option to assign the download directory.

```
panstamps -Fj --width=4 --filters=gri --downloadFolder=/Users/Dave/Desktop/m81 stack␣
↪09:55:52.2 +69:40:47
```

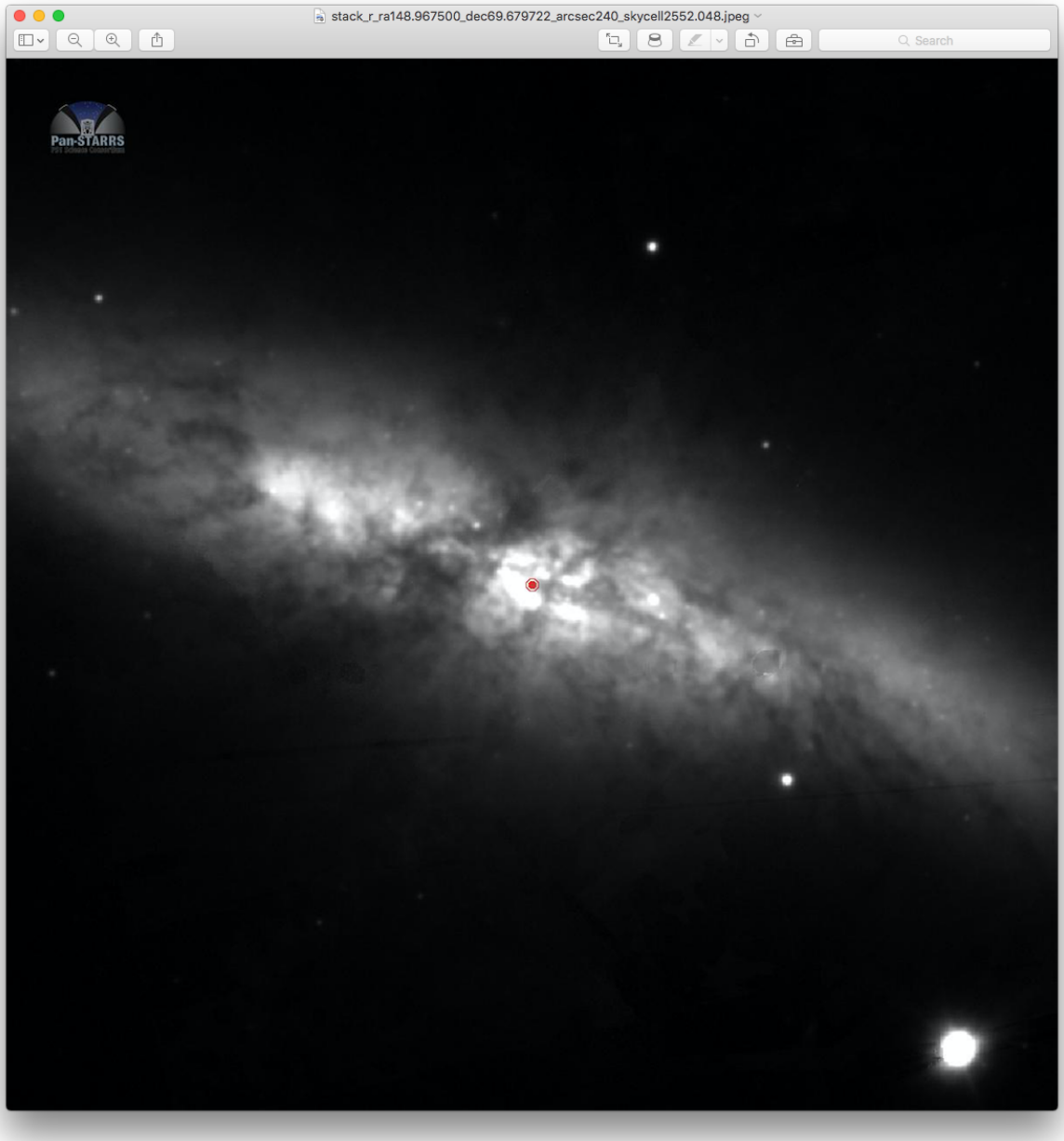
This downloads the jpegs and adds some useful annotation, which can be switched off if required.



Sometimes it maybe useful to add a transient marker at the centre of the

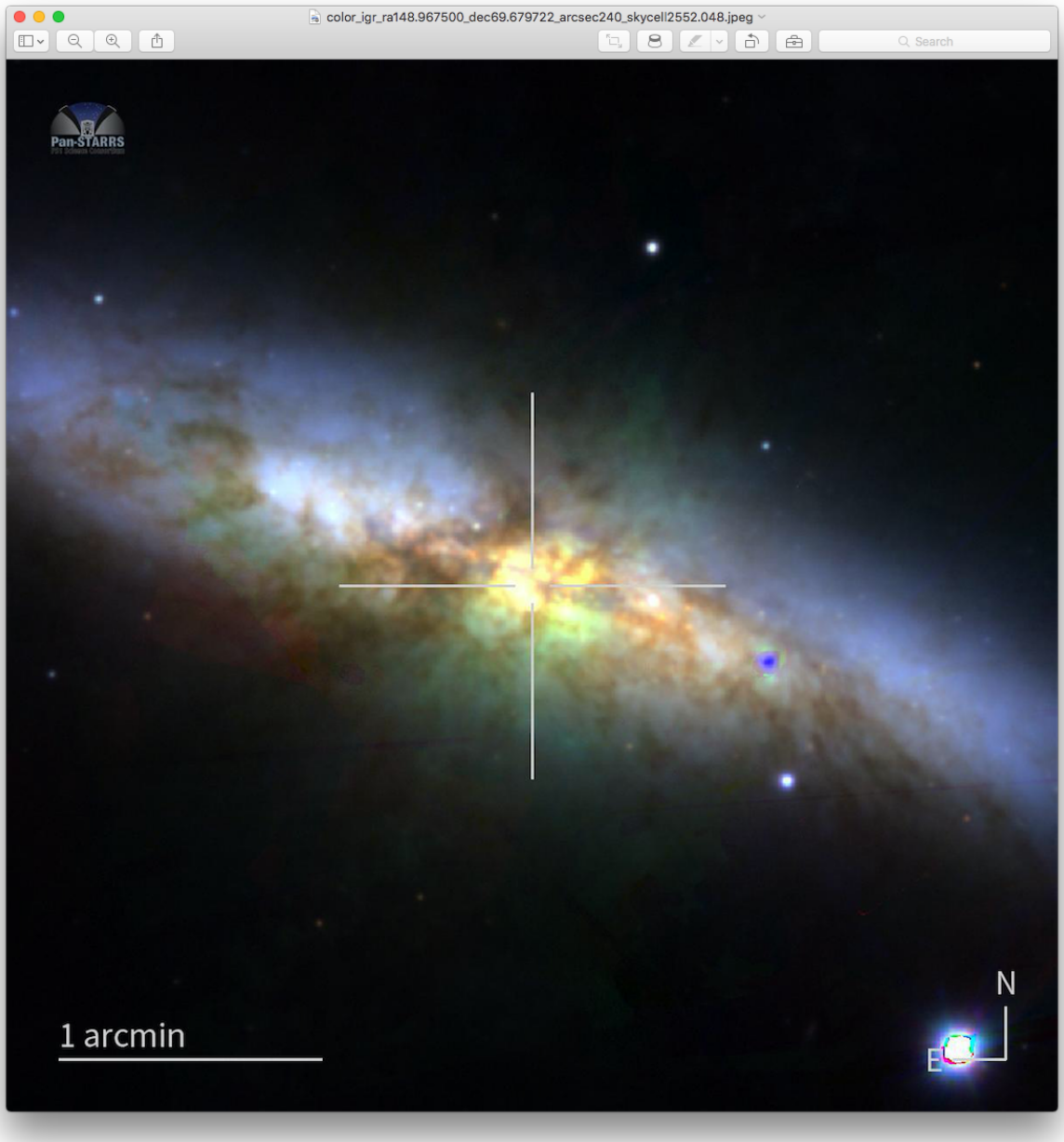
image:

```
panstamps -FjAt --width=4 --filters=gri --downloadFolder=/Users/Dave/Desktop/m81␣
↪stack 09:55:52.2 +69:40:47
```



Or grab the color image as well as/instead of the single filter images:

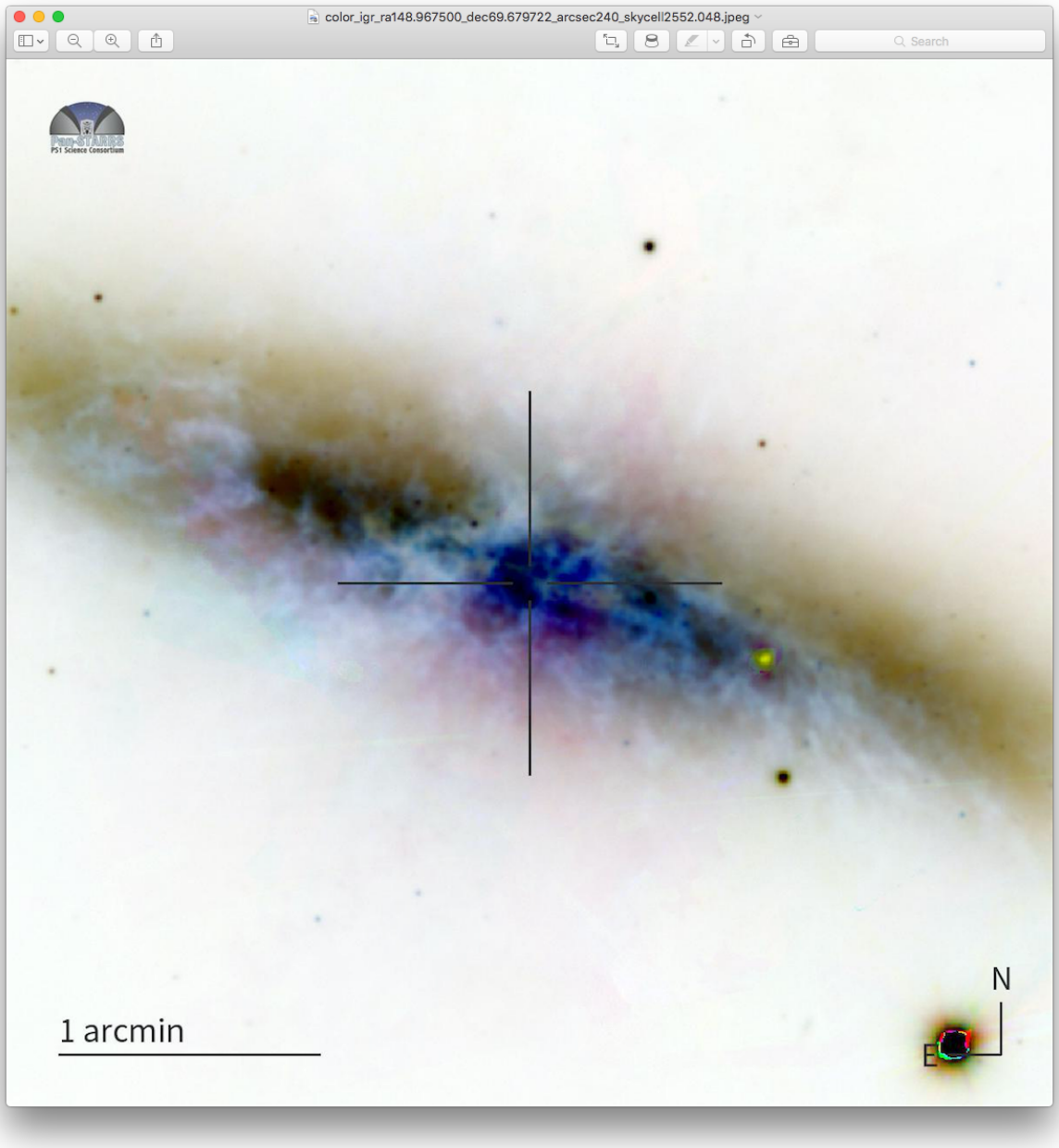
```
panstamps -FJc --width=4 --filters=gri --downloadFolder=/Users/Dave/Desktop/m81 stack␣
↪09:55:52.2 +69:40:47
```



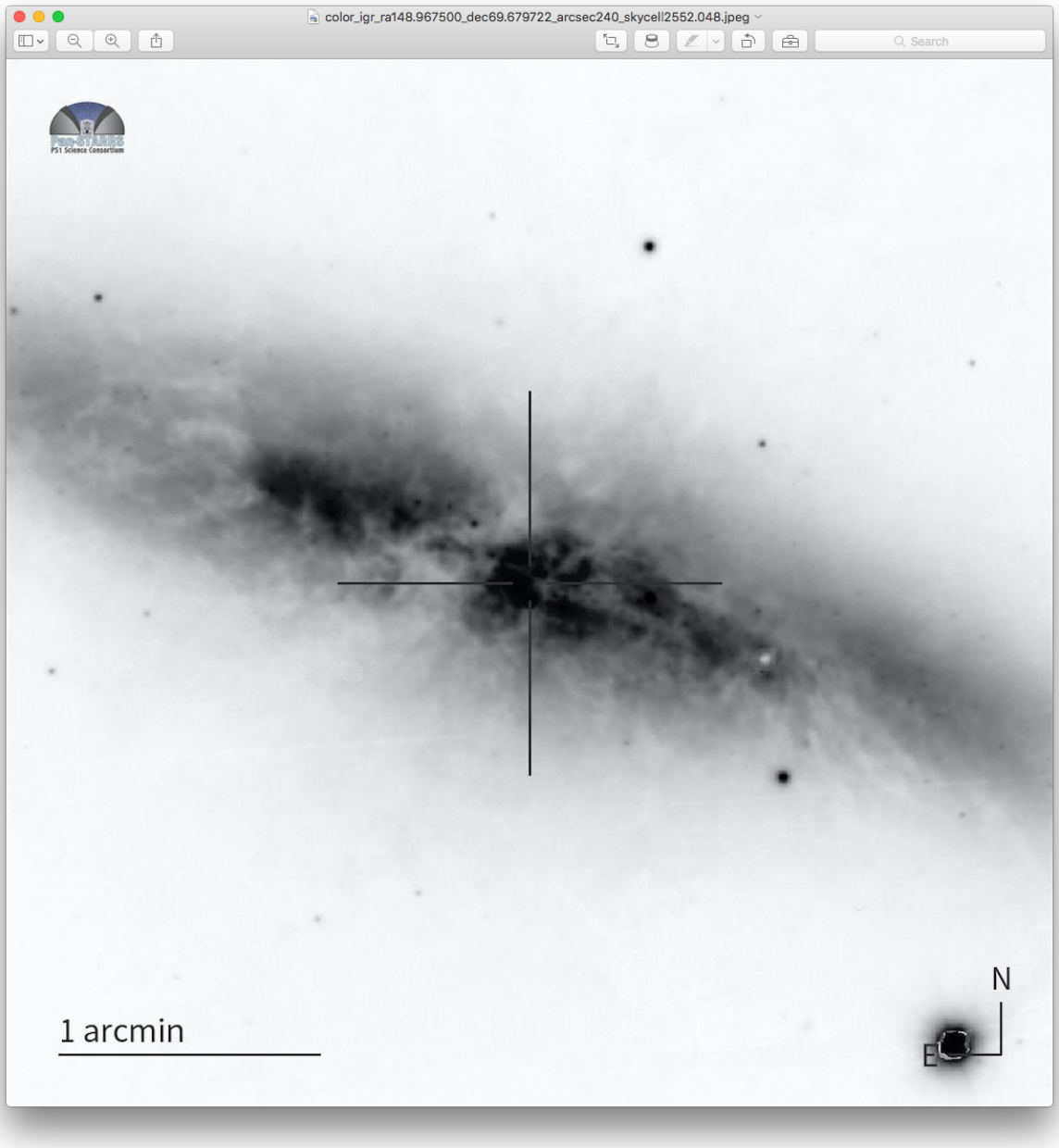
Note the code will try its best to choose a colour for the annotation lines and text to make them contrast well against the background image.

Finally you can invert the image colors or convert the image to greyscale:

```
panstamps -FJci --width=4 --filters=gri --downloadFolder=/Users/Dave/Desktop/m81␣
↪stack 09:55:52.2 +69:40:47
```




```
panstamps -FJcig --width=4 --filters=gri --downloadFolder=/Users/Dave/Desktop/m81␣
↪stack 09:55:52.2 +69:40:47
```



2.3.2 Temporal Constraints (Useful for Moving Objects)

For moving objects, alongside spatially filtering the panstarrs images, we also require a temporal filter. We need to be able to request images at a sky-position that were taken within a given time range. With panstamps we have the option of passing a time-window to filter the images by via the mjdStart and mjdEnd variables:

For example I can run:

```
panstamps -Fj --width=4 --filters=gri --downloadFolder=~/Desktop/movers warp 189.
↪1960991 28.2374845 55246.63 55246.64
```

to only return the 2 images I want within the temporal window at the

location in the sky.

It's also possible to request the closest warp image taken before or

after a requested MJD by using the closest flag. For example, to request the

closest r-band warp taken before MJD=`55246.64` for the location

above, run the command:

```
panstamps -Fj --closest=before --width=4 --filters=gri --downloadFolder=~/Desktop/
↪movers 189.1960991 28.2374845 55246.64
```

To request the closest warp taken after the given MJD then use –closest=after.

Oftentimes it's useful to download the closest warp within a given

time-window, e.g. closest warp in time of the requested MJD taken up to

3 mins before. To do so pass in a postive or negative integer to

represent the time-window in seconds, like so:

```
panstamps -Fj --closest=-120 --width=4 --filters=gri --downloadFolder=~/Desktop/
↪movers 189.1960991 28.2374845 55246.64
```

2.4 Using Panstamps within Python

To use panstamps within your own scripts please read the full

documentation. But for those of you that can't wait, this snippet should

give you the basics:

```
from panstamps.downloader import downloader
from panstamps.image import image
fitsPaths, jpegPaths, colorPath = downloader(
    log=log,
    settings=False,
    downloadDirectory=False,
    fits=False,
    jpeg=True,
    arcsecSize=600,
    filterSet='gri',
    color=True,
    singleFilters=True,
    ra="70.60271",
    dec="-21.72433",
    imageType="stack",  # warp | stack
    mjdStart=False,
    mjdEnd=False,
    window=False
).get()
```

(continues on next page)

(continued from previous page)

```
for j in jpegPaths:

    myimage = image(
        log=log,
        settings=False,
        imagePath=j
        arcsecSize=120,
        crosshairs=True,
        transient=False,
        scale=True,
        invert=False,
        greyscale=False
    ).get()
```

2.5 Todo List

Todo:

- nice!

(The *original entry* is located in /home/docs/checkouts/readthedocs.org/user_builds/panstamps/checkouts/develop/docs/source/_templat
line 1.)

2.6 Release Notes

v0.6.5 - February 23, 2023

- **FIXED**: fixed bug in parsing FITS urls. This was resulting in empty FITS files getting downloaded. (thanks to @fmannucci

  for reporting the issue.)

v0.6.4 - January 19, 2023

- **FIXED**: a change to the JPEG endpoint URLs broke regex looking for images. JPEG images should be downloadable again. (thanks to @fforster

  for reporting the issue.)

v0.6.3 - January 6, 2023

- **refactor**: moving from plpsipp1v.stsci.edu to ps1images.stsci.edu as end point to collect panstars images (thanks to Rick White for the pull-request)

v0.6.2 - May 11, 2022

- **FIXED** doc fixes
- **UNFEATURE** Removing Python 2 support

v0.6.1 - July 10, 2020

- **ENHANCEMENT** Unit tests now added for command-line tools
- **FIXED** Command-line tools now correctly parse arguments

- **FIXED** Regex matching of warp image URLs updated to match filename changes made by STScI PanSTARRS image server

v0.6.0 - May 7, 2020

- Now compatible with Python 3.*

CHAPTER

THREE

API REFERENCE

3.1 Modules

panstamps.commonutils	*common tools used throughout package*
panstamps.image	*The worker class for the image module*
panstamps.commonutils.getpackagepath	*Get common file and folder paths for the host package*
panstamps.utKit	*Unit testing tools*

3.1.1 commonutils *(module)*

common tools used throughout package

Sub-modules

getpackagepath	*Get common file and folder paths for the host package*

3.1.2 image *(class)*

class image(*log*, *imagePath*, *arcsecSize*, *settings=False*, *crosshairs=True*, *transient=False*, *scale=True*, *invert=False*, *greyscale=False*, *colorImage=False*)

Bases: object

The worker class for the image module

Key Arguments

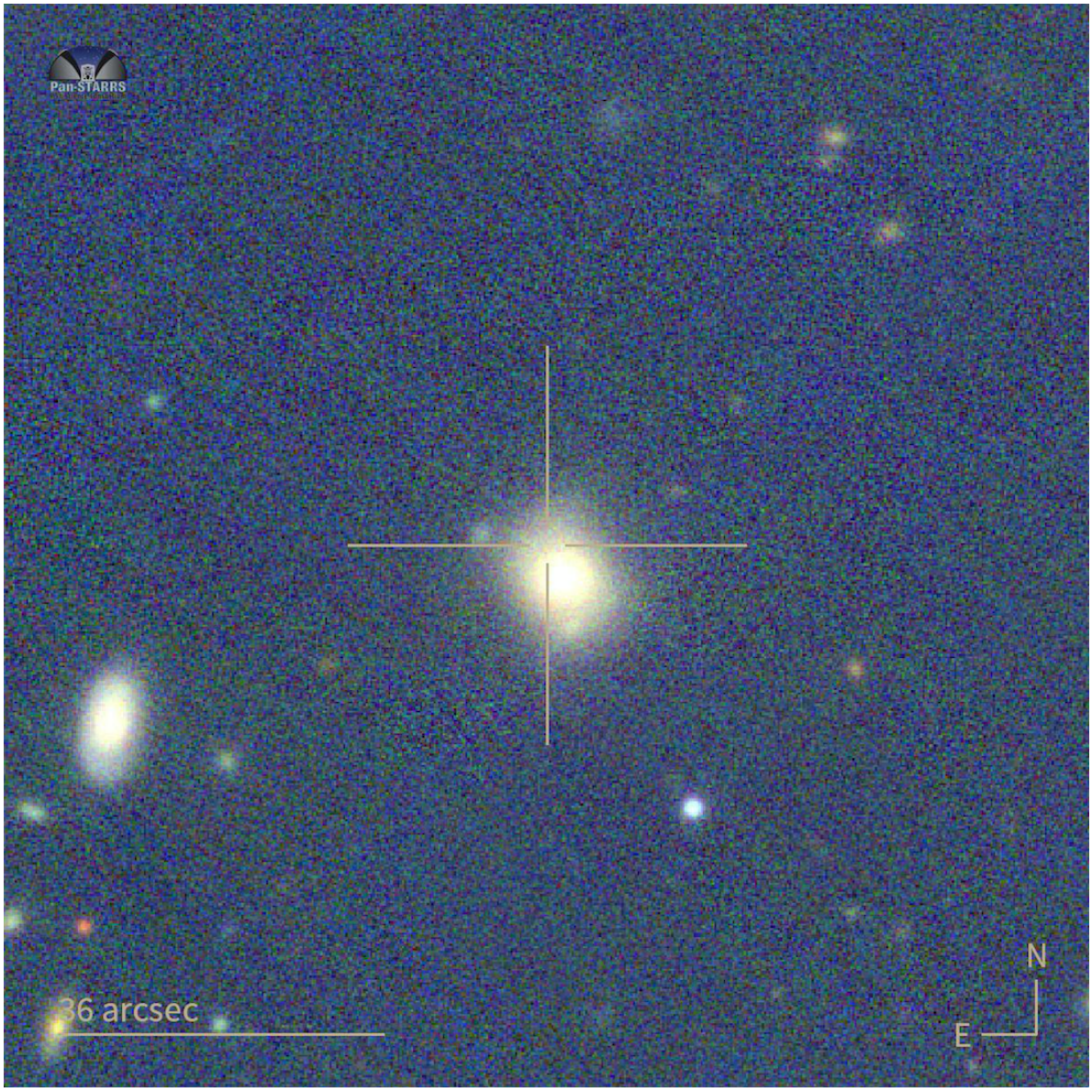
- log – logger
- settings – the settings dictionary
- imagePath – path to the image to manipulate
- arcsecSize – the size of the image stamps to download (1 arcsec == 4 pixels).
- crosshairs – add crosshairs to the image?. Default *True*
- transient – add a small transient marker at the centre of the image. Default *False*
- scale – add scale bar and orientation indicator to the image. Default *True*
- invert – invert the colours of the image. Default *False*

- `greyscale` – convert the image to greyscale. Default *False*
- `colorImage` – is the input image a color image, Default **False*. Note, also assumes a color image if 'color' in filename

Usage

```
from panstamps.image import image
myimage = image(
    log=log,
    settings=False,
    imagePath="70.60271m21.72433/color_igr_ra70.602710_dec-21.724330_arcsec120_
↪skycell0812.050.jpeg",
    arcsecSize=120,
    crosshairs=True,
    transient=False,
    scale=True,
    invert=False,
    greyscale=False,
    colorImage=True
).get()
```

Here's the resulting image from this code:



Methods

—

get()
annotate the PS1 image

Return

- `image` – a PIL image object

3.1.3 getpackagepath *(module)*

Get common file and folder paths for the host package

Author David Young

Functions

getpackagepath()	*getpackagepath*

Sub-modules

getpackagepath()	*getpackagepath*
os	OS routines for NT or Posix depending on what system we're on.

getpackagepath()
getpackagepath

3.1.4 utKit *(module)*

Unit testing tools

Classes

utKit(moduleDirectory[, dbConn])	*Override dryx utKit*

Sub-modules

utKit(moduleDirectory[, dbConn])	*Override dryx utKit*

class utKit(*moduleDirectory*, *dbConn=False*)
Bases: fundamentals.utKit.utKit

Override dryx utKit

get_project_root()
Get the root of the ``python`` package - useful for getting files in the root directory of a project

Return

- rootPath – the root path of a project

refresh_database()
Refresh the unit test database

setupModule()
The setupModule method

Return

- `log` – a logger
- `dbConn` – a database connection to a test database (details from yaml settings file)
- `pathToInputDir` – path to modules own test input directory
- `pathToOutputDir` – path to modules own test output directory

`tearDownModule`()
The tearDownModule method

3.2 Classes

panstamps.downloader	*Tools to download the panstarrs image stamps from STScI PanSTARRS image server*

3.2.1 downloader *(class)*

`class downloader`(*log*, *downloadDirectory=False*, *settings=False*, *fits=True*, *jpeg=False*, *arcsecSize=60*, *filterSet='gri'*, *color=True*, *singleFilters=True*, *ra=False*, *dec=False*, *imageType='stack'*, *mjdStart=False*, *mjdEnd=False*, *window=False*)

Bases: `object`

Tools to download the panstarrs image stamps from STScI PanSTARRS image server

Key Arguments

- `log` – logger
- `settings` – the settings dictionary
- `downloadDirectory` – the path to where you want to download the images to. Downlaods to path command is run from by default.
- `fits` – download the fits files? Default *True*
- `jpeg` – download the jpeg files? Default *False*
- `arcsecSize` – the size of the image stamps to download (1 arcsec == 4 pixels). Default *60*
- `filterSet` – the filter set used to create color and/or download as individual stamps. Default *gri*
- `color` – download the color jpeg? Default *True*
- `singleFilters` – download the single filter stmaps? Default *False*
- `ra` – ra in decimal degrees.
  - `dec` – dec in decimal degrees.
  - `imageType` – warp or stacked images? Default *stack*
  - `mjdStart` – the start of a time-window within which the images required are taken. Default *False* (everything)
  - `mjdEnd` – the end of a time-window within which the images required are taken. Default *False* (everything)

Usage

The following will return 3 lists of paths to local fits, jpeg and color-jpeg files:

```
from panstamps.downloader import downloader
fitsPaths, jpegPaths, colorPath = downloader(
    log=log,
    settings=False,
    fits=False,
    jpeg=True,
    arcsecSize=600,
    filterSet='gri',
    color=True,
    singleFilters=True,
    ra="70.60271",
    dec="-21.72433",
    imageType="stack",
    mjdStart=False,
    mjdEnd=False,
    window=False
).get()
```

Methods

get()	*download the requested jpegs and fits files*
get_html_content()	*Build the URL for the stamp request and extract the HTML content*
parse_html_for_image_urls_and_metadata(content)	*parse the HTML for image urls and metadata*

get()
download the requested jpegs and fits files

Return

- fitsPaths – a list of local paths to downloaded fits files
- jpegPaths – a list of local paths to downloaded jpeg files
- colorPath – a list of local paths to downloaded color jpeg file (just one image)

get_html_content()
Build the URL for the stamp request and extract the HTML content

Return

- content – the HTML content of the requested URL
- status_code – the HTTP status code of the request response
- url – the URL requested from the PS1 stamp server

Usage

```
from panstamps.downloader import downloader
content, status_code, url = downloader(
    log=log,
    settings=False,
    fits=False,
    jpeg=True,
    arcsecSize=600,
    filterSet='gri',
    color=True,
```

(continues on next page)

(continued from previous page)

```
    singleFilters=True,
    ra="70.60271",
    dec="-21.72433",
    imageType="stack",
    mjdStart=False,
    mjdEnd=False,
    window=False
).get_html_content()

print(status_code)
# OUT: 200

print(url)
# OUT: http://ps1images.stsci.edu/cgi-bin/ps1cutouts?filter=gri&filter=color&
↪catlist=&autoscale=99.500000&verbose=0&output_size=2400&filetypes=stack&
↪pos=70.60271+-21.72433&size=2400
```

parse_html_for_image_urls_and_metadata(*content*)

parse html for image urls and metadata

Key Arguments

- `content` – the content of the requested PS1 stamp HTML page

Usage

Note if you want to constrain the images you download with a temporal window then make sure to given values for `mjdStart` and `mjdEnd`.

```
from panstamps.downloader import downloader
mydownloader = downloader(
    log=log,
    settings=False,
    fits=False,
    jpeg=True,
    arcsecSize=600,
    filterSet='gri',
    color=True,
    singleFilters=True,
    ra="70.60271",
    dec="-21.72433",
    imageType="stack",
    mjdStart=False,
    mjdEnd=False,
    window=False
)
content, status_code, url = mydownloader.get_html_content()

allStacks, allWarps, colorImage = mydownloader.parse_html_for_image_urls_and_
↪metadata(content=content)

for k,v in allStacks.items():
    print(k, v)

# OUT:
## jpegs ['http://ps1images.stsci.edu/cgi-bin/fitscut.cgi?red=/data/ps1/
↪node15/stps15.1/nebulous/23/3a/7187453864.gpc1%3ALAP.PV3.20140730%3A2015
↪%3A01%3A29%3ARINGS.V3%3Askycell.0812.050%3ARINGS.V3.skycell.0812.050.stk.
↪4297354.unconv.fits&x=70.602710&y=-21.724330&size=2400&wcs=1&asinh=True&
↪autoscale=99.500000&output_size=2400', 'http://ps1images.stsci.edu/cgi-bin/
↪fitscut.cgi?red=/data/ps1/node08/stps08.1/nebulous/de/fa/5761784572.gpc1
↪%3ALAP.PV3.20140730%3A2014%3A12%3A25%3ARINGS.V3%3Askycell.0812.050%3ARINGS
↪V3.skycell.0812.050.stk.4106421.unconv.fits&x=70.602710&y=-21.724330&
↪size=2400&wcs=1&asinh=True&autoscale=99.500000&output_size=2400', 'http://
↪ps1images.stsci.edu/cgi-bin/fitscut.cgi?red=/data/ps1/node08/stps08.1/
↪nebulous/1b/d7/5756633973.gpc1%3ALAP.PV3.20140730%3A2014%3A12%3A25%3ARINGS
```

(continues on next page)

(continued from previous page)

```
## fits ['http://ps1images.stsci.edu/cgi-bin/fitscut.cgi?red=/data/ps1/node15/
↪stps15.1/nebulous/23/3a/7187453864.gpc1:LAP.PV3.20140730:2015:01:29:RINGS.
↪V3:skycell.0812.050:RINGS.V3.skycell.0812.050.stk.4297354.unconv.fits&
↪format=fits&x=70.602710&y=-21.724330&size=2400&wcs=1&imagename=cutout_rings.
↪v3.skycell.0812.050.stk.g.unconv.fits', 'http://ps1images.stsci.edu/cgi-bin/
↪fitscut.cgi?red=/data/ps1/node08/stps08.1/nebulous/de/fa/5761784572.
↪gpc1:LAP.PV3.20140730:2014:12:25:RINGS.V3:skycell.0812.050:RINGS.V3.skycell.
↪0812.050.stk.4106421.unconv.fits&format=fits&x=70.602710&y=-21.724330&
↪size=2400&wcs=1&imagename=cutout_rings.v3.skycell.0812.050.stk.r.unconv.fits
↪', 'http://ps1images.stsci.edu/cgi-bin/fitscut.cgi?red=/data/ps1/node08/
↪stps08.1/nebulous/1b/d7/5756633973.gpc1:LAP.PV3.20140730:2014:12:25:RINGS.
↪V3:skycell.0812.050:RINGS.V3.skycell.0812.050.stk.4097309.unconv.fits&
↪format=fits&x=70.602710&y=-21.724330&size=2400&wcs=1&imagename=cutout_rings.
↪v3.skycell.0812.050.stk.i.unconv.fits']
## filters ['g', 'r', 'i']
## filenames ['stack_g_ra70.602710_dec-21.724330_arcsec600_skycell0812.050',
↪'stack_r_ra70.602710_dec-21.724330_arcsec600_skycell0812.050', 'stack_i_
↪ra70.602710_dec-21.724330_arcsec600_skycell0812.050']
```

Return

- `allStacks` – dictionary of 4 equal length lists. jpeg remote urls, fits remote urls, filters and filenames.
- `allWarps` – dictionary of 4 equal length lists. jpeg remote urls, fits remote urls, filters and filenames.
- `colorImage` – dictionary of 4 equal length lists. jpeg remote urls, fits remote urls, filters and filenames.

3.3 A-Z Index

Modules

panstamps.commonutils	*common tools used throughout package*
panstamps.image	*The worker class for the image module*
panstamps.commonutils.getpackagepath	*Get common file and folder paths for the host package*
panstamps.utKit	*Unit testing tools*

Classes

panstamps.downloader	*Tools to download the panstarrs image stamps from STScI PanSTARRS image server*

Functions

—

CHAPTER

FOUR

RELEASE NOTES

v0.6.5 - February 23, 2023

- **FIXED**: fixed bug in parsing FITS urls. This was resulting in empty FITS files getting downloaded. (thanks to @fmannucci

  for reporting the issue.)

v0.6.4 - January 19, 2023

- **FIXED**: a change to the JPEG endpoint URLs broke regex looking for images. JPEG images should be downloadable again. (thanks to @fforster

  for reporting the issue.)

v0.6.3 - January 6, 2023

- **refactor**: moving from plpsipp1v.stsci.edu to ps1images.stsci.edu as end point to collect panstars images (thanks to Rick White for the pull-request)

v0.6.2 - May 11, 2022

- **FIXED** doc fixes
- **UNFEATURE** Removing Python 2 support

v0.6.1 - July 10, 2020

- **ENHANCEMENT** Unit tests now added for command-line tools
- **FIXED** Command-line tools now correctly parse arguments
- **FIXED** Regex matching of warp image URLs updated to match filename changes made by STScI PanSTARRS image server

v0.6.0 - May 7, 2020

- Now compatible with Python 3.*

PYTHON MODULE INDEX

C

U

INDEX