AstroConda Documentation

Release 0.0.1

Space Telescope Science Institute

Mar 20, 2017

Contents

1 System Requirements **3**

2 Powered by Conda **5**
2.1 Getting Started . . . 5
2.2 Selecting a Software Stack . . . 7
2.3 Updating a Software Stack . . . 8
2.4 Further Reading . . . 9
2.5 Pipeline Releases . . . 11
2.6 Compatibility Notices . . . 13
2.7 F.A.Q. . . . 14
2.8 Contributing Guide . . . 19
2.9 Packages . . . 26
2.10 Release Notes . . . 46
2.11 Resources . . . 48
2.12 General Disclaimer . . . 49

AstroConda is a free Conda channel maintained by the Space Telescope Science Institute (STScI) in Baltimore, Maryland. This channel provides tools and utilities required to process and analyze data from the Hubble Space Telescope (HST), James Webb Space Telescope (JWST), and others.

CHAPTER 1

System Requirements

- 64-bit Intel/AMD processor (x86_64)
- 64-bit Linux (glibc 2.12) or Mac OS X (10.7)
- BASH or ZSH as your default shell environment (T/CSH is NOT supported)

CHAPTER 2

Powered by Conda

Conda is an open-source software package management system provided and maintained by Continuum Analytics. Many software packages, provided both by Continuum and through third parties, are able to be quickly and easily installed using the Conda utility. AstroConda serves as a third-party add-on channel to provide easy access to STScI's software packages.

- **New to conda**: *Installing Conda - The choice is yours*
- **Familiar with conda**: *Configure Conda to use the Astroconda Channel*

To receive AstroConda announcements, or engage in general discussion, feel free to subscribe to our mailing list.

Getting Started

Installing Conda - The choice is yours

AstroConda is a third-party addon channel designed for use with the Conda package management system, so therefore in order to install software provided by our channel, you must first install a basic Conda environment on your system. This can be achieved in multiple ways (described below). Our channel's software is compatible with both of the `2` and `3` variants of Continuum Analytics, Inc.'s Miniconda and Anaconda distributions (i.e. Miniconda2, Miniconda3, Anaconda2, and Anaconda3).

Miniconda2 and Miniconda3 provide a bare-minimum Conda root environment with Python 2.7 or Python 3.x, respectively. (*Recommended*)

Anaconda2 and Anaconda3 are Continuum Analytics Inc.'s flagship products, and provide a full-featured Conda root environment as well as hundreds of useful tools, libraries, and utilities by default.

Both of Continuum's official distributions support a variety of operating systems and architectures, however the AstroConda channel specifically provides packages for Linux and Apple OS X running on x86_64 Intel/AMD processors. It is important to note Microsoft Windows is not supported at this time.

Now head over to **one** of the following sites and download a copy of the installer of your choice:

- Download Miniconda

- Download Anaconda (OS X users should choose the command-line installer)

The installation method used for Miniconda and Anaconda are identical, however keep in mind the scripts are written in BASH (not SH), so therefore you *must* execute the installer using `bash`:

```
$ cd <download_directory_here>
$ bash <install_script_here>
```

After the installation is complete double-check the bottom of `~/.bash_profile` to ensure Miniconda or Anaconda has been added to your `PATH`. Otherwise, you will be unable to successfully complete this guide.

> **Warning:** Conda requires BASH, or a BASH-compatible shell in order to function correctly. If your default shell environment is not BASH (see also, *System Requirements*), please execute `bash -l` before proceeding.
>
> From this point forward any time you wish to use Conda's environment activation script (i.e. `source activate <env_name>`), you will need to execute `bash -l` prior to doing so.

Verifying your Conda Environment

Execute the command: `which conda`

If the path to `conda` (i.e. `/home/username/miniconda3/bin/conda`), is not returned, continue reading, otherwise skip ahead to *Configure Conda to use the Astroconda Channel*.

If you answered `Y` or `Yes` when prompted during installation to place Miniconda or Anaconda in your `PATH`, and `which conda` still does not return a path leading back to `conda`, go ahead and execute `source ~/.bash_profile`, then re-execute `which conda`. If the path to `conda` appears, skip ahead to *Configure Conda to use the Astroconda Channel*.

However, if you answered `N` or `No` when prompted, you will need to fix your `PATH` manually. If you installed Miniconda or Anaconda using the defaults selected by the installer, but are not sure what the directory is named, use the following command to find out:

```
$ ls -d ~/*conda?
#[example output]
/home/username/miniconda3
```

Now append **one of the following** `export` commands that best matches the output of *ls -d* above to the bottom of `~/.bash_profile` using a plain-text editor:

```
export PATH="~/miniconda/bin:$PATH"
export PATH="~/miniconda3/bin:$PATH"
export PATH="~/anaconda/bin:$PATH"
export PATH="~/anaconda3/bin:$PATH"
```

At this point, to assume the new environment with `conda` in your `PATH`, open a new terminal or execute `source ~/.bash_profile` and continue on to *Configure Conda to use the Astroconda Channel*.

Configure Conda to use the Astroconda Channel

In order to install packages directly from the AstroConda channel you will need to append our URL to Conda's channel search path.

```
$ conda config --add channels http://ssb.stsci.edu/astroconda
# Writes changes to ~/.condarc
```

Be aware that indiscriminately adding channels to your configuration, be it from anaconda.org or via direct-URL can effect the stability of software packages in your run-time environment.

For example, if you add a channel found on anaconda.org because it contains a software package you're interested in, but it too provides the same software found in AstroConda, it's possible you may lose track of where packages are coming from. Or worse, the software you installed from the other channel was built incorrectly or did not account for a special case, so now the packages in your environment relying on this as a dependency could very well cease to function correctly.

If you decide to have multiple channels defined in your configuration and bugs begin to appear, it may be best to check their origin before issuing a support ticket to help@stsci.edu. `conda list` can be used to display such information about the packages installed in your environment.

Selecting a Software Stack

A "stack" is a collection of software designed to target the various use cases of our end-users. The three officially supported stacks are as follows:

- ***Standard Software Stack (without IRAF)*** **provides:**
  - The full compliment of STScI software and utilities
  - Python 2.7 or 3.x
- ***Legacy Software Stack (with IRAF)*** **provides:**
  - The full compliment of STScI software and utilities
  - A IRAF/PyRAF environment
  - Python 2.7 only
- ***Pipeline Software Stack*** **provides:**
  - The data processing environment used by STScI operations and instrument teams
  - Python 3.x only

Standard Software Stack (without IRAF)

The package management system, Conda, is now configured to pull from our repository, so you may go ahead and install the `stsci` package. This package installs nearly all of the software provided by STScI in one shot.

The following example generates a new conda environment named "astroconda", however this naming convention is merely a suggestion. Feel free to use a name that works best for you.

```
$ conda create -n astroconda stsci
```

After the installation is complete go ahead and activate the "astroconda" environment. This command only needs to be executed one time per terminal session.

```
$ source activate astroconda
```

To deactivate the "astroconda" environment, close your terminal window or run:

```
$ source deactivate
```

Legacy Software Stack (with IRAF)

The maintainers of the AstroConda channel have limited resources to support IRAF (Image Reduction and Analysis Facility), but users that require the ability to run IRAF and PyRAF tasks may want to install it via AstroConda. For help with any issues that come up during installation or use, please visit the PyRAF FAQ. **Linux users** please be sure to visit this FAQ entry for a quick guide to installing IRAF's 32-bit dependencies.

The package management system, Conda, is now configured to pull from our repository, so you may go ahead and install the `stsci` package, as well as `pyraf`, and finally `iraf`. The `stsci` package installs nearly all of the software provided by STScI in one shot, however if you prefer a slimmed down IRAF/PyRAF experience, feel free to omit it.

Due to Python 3.x incompatibilities present in several tasks, it is recommended to install IRAF alongside Python 2.7.

The following example generates a new conda environment named "iraf27", however this naming convention is merely a suggestion, so please feel free to apply a name that works best for you.

```
$ conda create -n iraf27 python=2.7 stsci pyraf iraf
```

After the installation is complete go ahead and activate the "iraf27" environment. This command only needs to be executed one time per terminal session.

```
$ source activate iraf27
```

To deactivate the "iraf27" environment, close your terminal window or run:

```
$ source deactivate
```

Pipeline Software Stack

Due to the nature of the pipeline software stack, the installation instructions have been consolidated under a separate section, *Pipeline Releases*.

Updating a Software Stack

Conda, will not automatically update unless a newer version of a package is detected during a routine package installation. Suffice to say, unless you keep your packages up to date with `conda update`, the packages installed in your environment will remain static.

There are few simple ways to update packages obtained from AstroConda:

Updating via Metapackage

```
$ conda update -n astroconda stsci
```

This is best used by individuals that favor software stability over receiving the "bleeding edge". Remember, updating the `stsci` package only effects packages part of the **official release** of our software. Packages provided by the AstroConda channel, but are not controlled by the `stsci` package **will not receive updates**. This is true for other packages as well (e.g. stsci-hst, stsci-data-analysis, etc).

To clarify what this does, if the `stsci` package (used to create your environment) has not been updated by STScI to accommodate recently added packages, or newer versions of those packages, nothing will be updated in your environment on behalf of `stsci`. In general, to receive interim bug fix releases please consider updating all packages, or individual packages of interest.

Updating All Packages

```
$ conda update -n astroconda --all
```

This will apply updates to all packages installed in your `astroconda` environment[1] regardless if they were installed via AstroConda, Continuum Analytics, Inc., or other third party channels defined within your `$HOME/.condarc`.

(ref)

Updating Individual Packages

```
$ conda update -n astroconda <name_of_pkg>
```

If you are interested in receiving updates for a particular package, then this method is for you. Be aware that packages may depend on other packages, so the total list of package updates returned by this command will vary.

Updating Conda

```
$ source deactivate
$ conda update --all
```

Keeping AstroConda packages up to date is important, but not nearly as important as keeping your 'root' (i.e. the base installation) updated as well. Conda is like any other software project and it requires periodic refreshing to stay current with the latest changes. Failing to do this can, over time, cause side-effects such as, the inability to upgrade, install, remove, or search for packages.

Updating extremely old releases of Conda to the latest version have been known to break the 'root' environment due to a variety of API changes in the code. There is not much STScI can do about this as Conda itself is not our product, however if this happens to you, reinstalling the latest release of Miniconda or Anaconda, then regenerating your AstroConda software environment is the fastest way to resolve the problem. Refer to the FAQ for more details.

Further Reading

Fine-tuning the Software

If you are short on hard drive space, have a slow internet connection, or are simply not interested in installing *everything but the kitchen sink*; take a quick look at the package manifest and select a custom mix of packages tailored to your needs.

```
$ conda create -n <name> [package [package ...]]
$ source activate <name>
```

For example, if the work you intend to perform requires `drizzlepac` and nothing else, you can simply create a custom environment that contains *only* `drizzlepac` and its dependencies.

```
$ conda create -n mydriz drizzlepac
$ source activate mydriz
```

[1] (STScI-Specific) "Updating All Packages" now assumes the role of "SSBX" in the AstroConda distribution model.

Additional Software Considerations

While our channel provides a suite of scientific software packages that are known to work well together and are supported by engineers from STScI, by default, Conda already has access to a hundreds of packages provided directly by Continuum Analytics, Inc. The AstroConda channel itself relies heavily on Continuum for its own dependencies, so you may find it beneficial to explore what is available to you.

Conda's full documentation set covers topics ranging from installation to maintaining environments and is available from its creators and maintainers: http://conda.io/docs/using/index.html.

Installing additional software into your AstroConda environment is as simple as:

```
$ source activate astroconda
$ conda install <name_of_pkg>
```

Often, the fastest way to discover if a package exists in the `conda` ecosystem is to try searching for it with `conda search <name_of_pkg>`. A comprehensive list of software available directly from Continuum's default channel can be found here: http://repo.continuum.io/pkgs/.

In addition to `conda install` the Python-standard tool `pip` is also available to install packages distributed through the Python Package Index (PyPI):

```
$ source activate astroconda
$ pip install <name_of_pkg>
```

Downgrading Packages

Did a recent update break your code? For example, if a bug is introduced into `stsci.tools`, you can, for example, easily downgrade it to a known-good version:

```
$ conda search stsci.tools
.  3.4.0                  py35_6  http://ssb.stsci.edu/astroconda/linux-64
*  3.4.1                  py35_0  http://ssb.stsci.edu/astroconda/linux-64
```

The `*` denotes the current version installed locally, while the `.` indicates Conda has cached that version of the package on your hard drive for quick-use. No prefix implies the package is neither cached, nor installed.

Now the only thing left to do, is to tell Conda to install the previous release of the package:

```
$ conda install stsci.tools=3.4.0
```

At this point you will prompted to downgrade the package and hopefully be back in business. However, if you find yourself in this predicament please send an email to help@stsci.edu describing the situtation in detail so that we may either begin working on a bug fix release to accommodate you, or offer alternative solutions the problem.

(ref)

Pinning Packages

> **Caution:** Pinning packages has the potential to break Conda. Only pin packages as a last resort.

Let's take the previous example one step further... Imagine `stsci.tools` is broken, and the hotfix release of `3.4.2` only partially solved the original issue. Now what? You still need to receive updates to other packages, but `stsci.tools` keeps trying to update back to `3.4.2` every time you touch `conda update`.

```
$ echo "stsci.tools <=3.4.0" > ${CONDA_PREFIX}/conda-meta/pinned
```

From now on, future calls to `conda update` will omit `stsci.tools` while performing dependency resolution. However, a clear side-effect of this will also be losing the ability to update packages that depend strictly on version `3.4.2`. Although this is not a permanent solution it can prove useful in a bad situation.

(ref)

Removing a Conda Environment

It is possible to remove a Conda environment by running:

```
$ conda env remove -n <env_name>
```

Let's assume you've created a small test environment, "simple", with only a few packages you are interested in, such as `drizzlepac` and `hstcal`.

```
$ conda create -n simple drizzlepac hstcal
$ source activate simple
```

You've played around for a bit, maybe calibrated some data or checked out a new feature, but at this point you've decided you no longer want the "simple" environment anymore. So you delete it:

```
$ source deactivate
$ conda env remove -n simple
```

And then you quickly verify your "simple" environment no longer exists using `conda env list`:

```
$ conda env list
# ["simple" is not listed]
astroconda               /home/username/miniconda3/envs/astroconda
iraf27                   /home/username/miniconda3/envs/iraf27
2016.2                   /home/username/miniconda3/envs/2016.2
root                  *  /home/username/miniconda3
```

Pipeline Releases

Note:

- A working installation of Miniconda or Anaconda is required.
- Python 2.x.x is not supported (unless noted otherwise).
- 32-bit operating systems are not supported.

Pipeline releases differ from the standard software stack and serve a different purpose. The release files described below are immutable snapshots of STScI operational software, and can be used to replicate the environment used by STScI to perform mission-specific data processing. Be aware that upgrading packages with `conda update [pkg]` or `conda update --all` is not recommended as it will likely introduce unwanted bugs and/or break the environment all together.

If you have any questions, comments, or concerns related to pipeline releases please feel free to contact help@stsci.edu

Installation

Pipeline release installations use the following `conda create` command format:

```
conda create -n [custom_env_name] --file [URL]
source activate [custom_env_name]
```

Example

```
conda create -n demo_2016.1 \
    --file http://ssb.stsci.edu/conda/hstdp-2016.1/hstdp-2016.1-linux-py35.2.txt
source activate demo_2016.1
```

The URL used here will not be updated to reflect the latest iteration available. Please consult the *File URLs* section to ensure you are installing the correct release.

File URLs

Select the URL that matches your intended platform and environment.

HST Data Processing (HSTDP)

HSTDP was previously known as *OPUS*.

2016.1

PLATFORM	Python	URL
Linux	3.5	http://ssb.stsci.edu/conda/hstdp-2016.1/hstdp-2016.1-linux-py35.2.txt
OS X	3.5	http://ssb.stsci.edu/conda/hstdp-2016.1/hstdp-2016.1-osx-py35.2.txt

2016.2

PLATFORM	Python	URL
Linux	3.5	http://ssb.stsci.edu/conda/hstdp-2016.2/hstdp-2016.2-linux-py35.2.txt
OS X	3.5	http://ssb.stsci.edu/conda/hstdp-2016.2/hstdp-2016.2-osx-py35.2.txt

Release Schema

If you wish to write shell scripts to manage your local pipeline installations, this may be of interest to you:

```
RELEASE_HOME=http://ssb.stsci.edu/conda

#               hstdp 2016  1
#               ^     ^     ^
RELEASE_PARENT=$NAME-$YEAR.$BUILD

#                             linux     py35            2
#                             ^         ^               ^
RELEASE_CHILD=$RELEASE_PARENT-$PLATFORM-$PYTHON_VERSION.$ITERATION.txt
```

Compatibility Notices

As the Conda package ecosystem evolves and third-party software updates are released by Continuum and other providers, this may interfere with the stability of other codebases, such as STScI's software. This page will proactively chronicle such events as they occur as well as provide workarounds to these issues.

If you spot a compatibility problem not listed here please let us know by sending an email to *help[at]stsci.edu*.

Note: You may be affected by an issue if you have updated your AstroConda environment on or after the dates listed in each section below.

2017-02-11

NumPy v1.12.0 modified the way array slicing is handled and triggered a regression in the `acstools` and `pysynphot` packages:

- `acstools <= 2.0.6` - 2.0.7 released (Feb 16, 2017)
- `pysynphot <= 0.9.8.5` - 0.9.8.6 released (Feb 21, 2017)

One of the traceback messages to be aware of is as follows (traceback may be worded differently but usually complains about index not being an integer):

```
TypeError('slice indices must be integers or None or have an __index__ method',)
```

Recommended user actions:

- Upgrade `acstools` to version 2.0.7 (i.e., `conda update acstools`)
- Upgrade `pysynphot` to version 0.9.8.6 (i.e., `conda update pysynphot`)

Alternative user action:

- Downgrade `numpy` to version 1.11 (i.e., `conda install numpy=1.11`)

2016-12-23

AstroPy v1.3 fully deprecated calls to `astropy.io.fits.new_table`. The following packages are known to be incompatible with this release:

- `calcos <= 3.1.8` - Bugfix pending
- `costools <= 1.2.1` - Bugfix pending
- `fitsblender <= 0.2.6` - 0.3.0 released (Jan 17, 2017)

Recommended user actions:

- Upgrade `fitsblender` to version 0.3.0 (i.e., `conda update fitsblender`)

Alternative user actions:

- Downgrade `astropy` to version 1.2.1 (i.e., `conda install astropy=1.2.1`)

Future

A list of known deprecation warnings detected in regression tests managed by STScI Science Software Branch is available here. This list is refreshed daily from "dev" and "public" test results.

Drizzlepac

These deprecation warnings have been fixed in `drizzlepac` 2.1.8, which is now available in AstroConda:

- https://github.com/spacetelescope/drizzlepac/issues/14
- https://github.com/spacetelescope/drizzlepac/issues/15
- https://github.com/spacetelescope/drizzlepac/issues/16
- https://github.com/spacetelescope/drizzlepac/issues/17
- https://github.com/spacetelescope/drizzlepac/issues/21
- https://github.com/spacetelescope/drizzlepac/issues/27

F.A.Q.

What is the difference between Conda, Miniconda, Anaconda, and AstroConda?

- **Conda** is a user-land environment and package management utility maintained by Continuum Analytics, Inc.
- **Miniconda** is distributed by Continuum Analytics, Inc. as a standalone installer, and provides a minimal conda-based environment (i.e. `conda` and `python`)
- **Anaconda** is distributed by Continuum Analytics, Inc. as a standalone installer, and provides a fully featured conda-based environment (i.e. `conda`, `python`, and commonly used utilities, libraries, and Python packages).
- **AstroConda** is a collection of Conda packages, also known as a "Conda Channel", to be used with existing Miniconda and Anaconda installations via `conda create`, `conda install`, and `conda update`.

Note: Package recipes are freely available on GitHub and are maintained by the Science Software Branch (SSB) of the Space Telescope Science Institute (STScI).

Why do you recommend installing Miniconda instead of Anaconda?

Miniconda (~100MBs) has a significantly smaller footprint than Anaconda (~1GBs). However, you are free to continue using Anaconda as your base installation if it suits your personal needs.

If I already have Miniconda or Anaconda installed, do I need to reinstall it to use AstroConda?

No. Adding our channel URL to your current configuration with `conda config --add` should suffice.

Does AstroConda provide an installation script similar to Miniconda or Anaconda?

No. AstroConda requires Miniconda or Anaconda to be installed prior to installing packages from the AstroConda channel.

Could not find URL: http://ssb.stsci.edu/astroconda/...

This error may occur if...

- You are not connected to the internet
- Your network requires a proxy (outside the scope of this FAQ)
- STScI is experiencing a network or power outage
- The AstroConda package repository crashed or rebooted due to routine patching

If you have verified your internet connection is functioning properly but our parent site (http://www.stsci.edu) is unresponsive, then feel free to try our mirror.

Currently there are two ways to access it:

Using conda directly

The basic `conda` installation commands accept `-c` or `--channel` argument, which allows you to select a different repository to install packages from.

```
conda {create|install|update} [-n name] -c http://astroconda.org/channel/main [[pkg] .
↪..]

# Examples
conda create -n astroconda -c http://astroconda.org/channel/main stsci
conda install -n astroconda -c http://astroconda.org/channel/main webbpsf
conda update -n astroconda -c http://astroconda.org/channel/main --all
```

Using a modified .condarc

If our primary AstroConda channel appears to be down for longer than ten minutes, and the direct method isn't convenient, you can add the URL to your `~/.condarc` configuration.

```
channels:
  #- http://ssb.stsci.edu/astroconda
  - http://astroconda.org/channel/main
  - defaults
```

We recommend against using the astroconda.org mirror on a regular basis. The bandwidth and hardware are not better or faster than the primary server. If the reason for switching to the mirror has been resolved, you should go back to using the primary.

After "conda create -n ..." why does "source activate astroconda" fail?

Conda does not support CSH (C-Shell). Please switch to a POSIX-compatible shell (e.g. BASH, KSH, ZSH).

Note: STScI will not maintain a separate codebase of `conda` in order to implement CSH support. Feel free to create an issue with the developers.

What do you mean by Conda "root" environment?

The root environment refers to the top-level directory structure created by the Miniconda or Anaconda installer. The package manager, Conda, resides under `/home/username/*conda/bin`. So for example, if you execute `source deactivate` your `PATH` environment variable is reset to point to the root environment. Thus, installing software at this level will be installed in the root `bin` directory too.

Please refrain from installing any software into the root environment with either `pip` or `conda`. It is recommended to always create a new environment with `conda create` and install new software into it. It is far easier to delete a broken environment with `conda env remove` than it is to reinstall Miniconda or Anaconda in its entirety. Custom environments are safely abstracted away from the root and live under `/home/username/*conda/envs/<name>`.

I installed AstroConda packages into my [Mini/Ana]conda 'root' environment. What now?

Please reinstall Miniconda (or Anaconda) from scratch. AstroConda uses `source activate` and `source deactivate` calls to control your shell environment. The Anaconda 'root' environment **does not** use this feature, and thus, the packages you have installed there will not work properly.

To clarify, installing AstroConda packages directly into the 'root' may cause Miniconda (or Anaconda) itself to be come unstable. In addition to this, removing packages from this environment is tedious and will likely break your Anaconda installation if you are not careful. *Reinstalling from scratch is the safest option.*

How do I reinstall Miniconda?

This assumes Miniconda3 has been installed to the default location (`/home/username/miniconda3`). We will refer to your home directory as `~/` from here on.

```
# Make a backup of your existing installation if desired:
$ tar cfz /safe/place/miniconda3_OLD.tar.gz ~/miniconda3

# Remove Miniconda3
$ rm -rf ~/miniconda3

# Download the Miniconda3 installer for your platform from: http://conda.io/miniconda.
↪html

# Execute the installer (where PLATFORM is one of Linux or MacOSX)
$ bash /path/to/Miniconda3-latest-PLATFORM-x86_64.sh
```

The installer may offer to automatically edit `~/.bash_profile` and prepend `~/miniconda3/bin` to your `PATH`. If you already have this entry in your `PATH`, simply respond with 'no'. Responding with 'yes' will add another entry even if it exists resulting in a `PATH` that looks much like: `~/miniconda3/bin:~/miniconda3/bin:[...]`.

How do I reinstall Anaconda?

This assumes Anaconda3 has been installed to the default location (`/home/username/anaconda3`). We will refer to your home directory as `~/` from here on.

```
# Make a backup of your existing installation if desired:
$ tar cfz /safe/place/anaconda3_old.tar.gz ~/anaconda3

# Remove anaconda3
```

```
$ rm -rf ~/anaconda3

# Download the Anaconda3 installer for your platform from: https://www.continuum.io/
↪downloads
# (Note: Use the "COMMAND-LINE INSTALLER")

# Execute the installer (where VERSION is the version you downloaded, and  PLATFORM␣
↪is one of Linux or MacOSX)
$ bash /path/to/Anaconda3-VERSION-PLATFORM-x86_64.sh
```

The installer may offer to automatically edit `~/.bash_profile` and prepend `~/anaconda3/bin` to your `PATH`. If you already have this entry in your `PATH`, simply respond with 'no'. Responding with 'yes' will add another entry even if it exists resulting in a `PATH` that looks much like: `~/anaconda3/bin:~/anaconda3/bin:[...]`.

Why am I being prompted by NumPy/SciPy with a MKL 30-day trial warning?

The `root` environment of your installation is severely outdated (`<=2.4.0`) and suffers from a crippling bug introduced by the `conda-3.19.x` package.

It is possible to verify the version of Anaconda you have by running:

```
$ conda info anaconda
```

The only solution is to update your Anaconda installation to the latest release:

```
$ source deactivate
$ conda update conda
$ conda update anaconda
```

Next, update the `astroconda` environment to realign your packages with the `root` environment:

```
$ conda update -n astroconda --all
$ source activate astroconda
```

After doing this, the `mkl` 30-day trial warning will not be displayed while importing `numpy`, `scipy`, or any other package requiring `mkl`.

How does AstroConda differ from Ureka?

Ureka and AstroConda both provide applications and libraries in binary form, however Ureka was a *one size fits all* distribution which included every package in one giant tarball. If an end-user only really wanted a small set of packages they were forced to install everything *and then some*. AstroConda is a *if the shoe fits wear it* distribution of packages. If an end-user needs `HSTCAL`, for example, they can install `HSTCAL` and omit much of the rest of HST's software suite.

A *major difference* most people will appreciate is the sheer lack of shell scripts. Ureka's environment was controlled by several dozen independent scripts. What makes AstroConda different? For one, it is not controlled by user-executed scripts. Changes to the environment are applied by special (read: embedded) scripts that are executed automatically by `source activate`, and the environment variables are unset with every `source deactivate`. This allows you to switch between several different environments rapidly without needing to keep track of what is defined in the environment.

How often are updates released?

Updates to (STScI) software will be released as bugs are identified and squashed. The `stsci-*` packages, for example, provide "releases" (i.e. a set of software used by our internal pipelines). After installing a release it is then possible to upgrade to the latest out-of-band packages by simply running:

```
conda update -n astroconda --all
```

Non-STScI software will be upgraded on an as-needed basis. See the Contibuting Guide to learn more about asking for updates to existing packages.

What happened to SSBX?

SSBX was a weekly release of STScI's software suite regardless of the stability of the codebase. This release was often times broken and caused issues for external users. SSBX has been rolled into AstroConda as out-of-band package updates. This offers a much better user experience, because as bugs are fixed, people will be able to upgrade without waiting until the following week.

What happened to SSBDEV?

SSBDEV was a nightly snapshot release of STScI's software suite. This release was often times broken and caused issues for external users. Nightly snapshots are ***no longer available*** for use by the public. Updates to AstroConda will be made directly as existing software is improved and/or new software is released.

Why isn't IRAF installed by default?

IRAF is an extremely large software package. Not every developer or scientist requires it.

If you wish to use IRAF, simply install it :

```
conda create -n iraf27 python=2.7 iraf pyraf stsci && source activate iraf27
```

If you are already using AstroConda under a Python 2 environment, you may simply install IRAF/PyRAF into that environment:

```
conda install -n astroconda iraf pyraf
```

Why is IRAF/PyRAF less functional under Python 3?

The Python code in `stsdas`, for example, is targeted specifically for Python 2.7 and earlier. If the demand for Python 3.x support under IRAF is great enough we may be able to pull our resources to accommodate the community. It is recommended to install IRAF into its own environment under Python 2.7:

```
conda create -n iraf27 python=2.7 iraf pyraf stsci && source activate iraf27
```

Why is IRAF 32-bit instead of 64-bit?

Many of the IRAF tasks that we include with AstroConda are so old that they cannot be compiled as 64-bit executables without significant changes to the source code. Because of this restriction, we always build IRAF as a 32-bit program, even for our 64-bit distributions.

In Linux, how do I install IRAF's 32-bit dependencies?

Debian >=7, Ubuntu >=14.04

```
# If on Debian execute this first (not required on Ubuntu):
sudo dpkg --add-architecture i386

sudo apt-get update
sudo apt-get install libc6:i386 libz1:i386 libncurses5:i386 libbz2-1.0:i386␣
↪libuuid1:i386 libxcb1:i386 libxmu6:i386
```

RHEL/CentOS >=6, Fedora >=14

```
sudo yum install glibc.i686 zlib.i686 ncurses-libs.i686 bzip2-libs.i686 uuid.i686␣
↪libxcb.i686
```

Will AstroConda interfere with other scientific distributions (e.g. SciSoft)?

Probably, however unlike Ureka, we do not impose any restrictions on your environment or issue compatibility warnings at run-time. It is your responsibility to maintain a functional shell environment so [insert scientific distribution here] does not conflict with your Conda installation.

Ds9 - Cannot select regions

The default edit mode is now `None` rather than `Region`. To select `Region` as the default editing mode perform the steps listed here:

- **Click `Edit`**
  - **Click `Preferences`**
    - On the left pane, select `Menus and Buttons`
    - On the right pane, click the `Menu --` drop-down menu beneath the `Edit` group
    - Select `Region` (default is `None`)
  - Click the `Save` button at the bottom of the `Preferences` dialog box

Contributing Guide

> **Attention:** A GitHub account is required to begin contributing to AstroConda

Guidelines

> **Attention:** The following packaging guidelines are subject to change at any time.
>
> - Please be respectful when commenting on pull-requests or issues.
> - If your contribution is not accepted into AstroConda, as a general courtesy, you will be given a clear and concise reason.

- As a contributor you may not claim exclusive rights to a particular recipe.
- You are free to maintain a recipe in AstroConda by issuing regular pull requests.
- Everyone is welcome to improve upon recipes so long as the changes do not introduce packaging conflicts.
- Abandoned recipes may be moved into the `deprecated` directory at any time without warning. (i.e. The package no longer compiles, has been obsoleted, or presents a conflict that cannot be resolved, etc).
- Packages derived from `deprecated` recipes will remain available in AstroConda for historical purposes (i.e. to preserve backwards compatibility).

Bugs, questions, and requests

Please open a new issue or send us a pull request for bugs, feedback, questions, or enhancements.

- For documentation issues use the astroconda issue tracker
- For recipe issues, use the astroconda-contrib issue tracker

Adding a recipe to astroconda-contrib

In this example we will be adding a new recipe to the AstroConda repository for sympy, the symbolic mathematics library.

Navigate to the astroconda-contrib repository on GitHub, login, and create a fork (or click here to have your fork created automatically).

Now that you have a fork of astroconda-contrib, go ahead and clone it to your system:

```
git clone https://github.com/[Your_Account]/astroconda-contrib
cd astroconda-contrib
```

To get started adding our recipe, create a new branch and name it `sympy-contrib`:

```
git checkout -t -b sympy-contrib
```

Git will automatically switch your branch from `master` to `sympy-contrib` as denoted by the following output:

```
Branch sympy-contrib set up to track local branch master.
Switched to a new branch 'sympy-contrib'
```

If you have taken the liberty of looking around the astroconda-contrib directory, you will have noticed a bunch of directories are sitting in there all named by-package. So let's keep things simple and straight forward. Go ahead and create a directory and name it `sympy`, and proceed inside:

```
mkdir sympy
cd sympy
```

Note: This is not an full Conda packaging tutorial. For more information about creating recipes from scratch, please refer to the conda-build documentation.

Hint: Investigate the contents of the recipes in astroconda-contrib. For most cases, copying an existing recipe and changing its values will suffice.

Copy the contents of the `astroconda-contrib/template` recipe. Three files `bld.bat`, `build.sh`, and `meta.yaml` will be copied to your working directory:

```
cp ../template/* .
```

Go ahead and open `meta.yaml` with your favorite plain-text editor:

> **Caution:** It is *highly recommended* that you enable "tabs to spaces" for your editor. Tab widths are unpredictable and may cause Conda's YAML parser to fail.

```
vim meta.yaml
```

The general structure of the file is as follows:

```
# These directives are commented here due to Pygments
# failing to parse this section of code.
# ... They are not commented in the real file.

#{% set name = '' %}
#{% set version = '' %}
#{% set number = '0' %}

about:
    # Package homepage
    home:
    # Package license
    license:
    # A brief description
    summary:

package:
    # Define these values above
    name: {{ name }}
    version: {{ version }}

build:
    # Define this value above
    number: {{ number }}

source:
    fn: {{ name }}-{{ version }}.tar.gz
    url: http://example.com/example/{{ name }}-{{ version }}.tar.gz

requirements:
    build:
    # Dependencies required at BUILD-TIME go here
    - setuptools
    - python x.x
    run:
    # Dependencies required at RUN-TIME go here
    # - ...

#test:
#   imports:
#   # - (e.g. a_python_module)
#
```

```
#   commands:
#   # - (e.g. program --help)
```

First, let's fill in the blanks. Modify the JINJA2 template markers for `name`, `version`:

```
{% set name = 'sympy' %}
{% set version = '1.0' %}
```

Fill in the `about` section with relevant information regarding the package:

```
about:
    home: http://sympy.org
    license: GPL
    summary: Python library for symbolic mathematics
```

Next, modify the `source` section's `url` so that it points to `sympy`'s source archive (on the internet):

```
source:
    fn: {{ name }}-{{ version }}.tar.gz
    url: https://github.com/{{ name }}/{{ name }}/releases/download/{{ name }}-{{␣
↪version }}/{{ name }}-{{ version }}.tar.gz
```

What's with the never-ending stream of bracket encapsulated keywords, you ask? Conda uses JINJA2, a basic template system that provides basic string interpolation within recipes. This comes in handy if, let's say, you decide to build a more recent version of `sympy`, you need only modify the first two variable definitions in this file (assuming the URL structure has not changed).

The `requirements` section may be confusing to some, so let's clarify the distinction between `build` and `run` before diving in. The `build` section defines Conda packages required at compile-time (i.e. `python setup.py install`), whereas the `run` section lists Conda packages required at install-time (i.e. `conda install [package]`).

As recipe maintainer the method used to dependency discover is that of trial and error. For many Python packages obtained via PyPi, it is easy enough to visually examine `setup.py` or `requirements.txt` to get a good idea of the recipes you need to depend on. Some package may require several iterations of executing `conda build` and returning to your recipe in the editor to append more packages.

As we can see below, `sympy` requires `mpmath`, `setuptools` and `python` to *build* the recipe, but only `mpmath` and `python` to *run it* successfully after installation:

```
requirements:
    build:
    - mpmath
    - setuptools
    - python x.x
    run:
    - mpmath
    - python x.x
```

What does the `x.x` imply exactly? This instructs `conda build` *not* to proceed unless `python=[version]` has been issued as an argument on the command-line. If `x.x` is omitted here, the recipe will choose the version of Python currently active in your environment. In most cases it is best to be explicit rather than implicit when it comes to defining version requirements in Conda.

The `test` section defines few useful lists, `imports`, `commands`, and `requires`. While these are not *required* to be used in any given recipe, we do use them in AstroConda. The `imports` section is a list of Python module imports, the `commands` are executed in a basic shell environment, and finally `requires` defines any extraneous packages to be installed into the environment before running the tests.

```
test:
    imports:
        - sympy

    #commands:
    #   - no shell commands to execute

    #requires:
    #   - does not require any extra testing-related packages
```

If `sympy` provided a command-line utility named `sympy-show`, you would use the `commands` section to verify the utility's functionality. A simple example of this would be to output a usage statement.

```
test:
    # ...
    commands:
        - sympy-show --help
```

If a program returns greater than zero `conda build` will fail as if it observed an error. Not all programs return zero after issuing `--help` (or an equivalent argument). Due to this, you may need to omit this style of test.

It will not hurt to keep the `commands` section populated but disabled with a note describing why it doesn't work. Others will find this information useful. Given this scenario, the optimal approach would be to contact the developers and plead with them to normalize the exit value.

Below is our `sympy` final recipe. Despite the overwhelming use of JINGA2 in our example, things are looking pretty streamlined.

```
{% set name = 'sympy' %}
{% set version = '1.0' %}
{% set number = '0' %}

about:
    home: http://sympy.org
    license: GPL
    summary: Python library for symbolic mathematics

source:
    fn: {{ name }}-{{ version }}.tar.gz
    url: https://github.com/{{ name }}/{{ name }}/releases/download/{{ name }}-{{␣
↪version }}/{{ name }}-{{ version }}.tar.gz

requirements:
    build:
    - mpmath
    - setuptools
    - python x.x
    run:
    - mpmath
    - python x.x

test:
    imports:
        - sympy
```

The `template` directory copied earlier in this tutorial contains two basic python build scripts for both *Nix (`build.sh`) and Windows (`bld.bat`), and is coincidentally compatible with the example we're using here. Not all Python packages (especially Makefile-based packages) will compile successfully using this "one-liner" template. Always

refer to the INSTALL file or equivalent documentation for the program you are attempting to compile to learn more about what the package expects from the end-user at compile-time.

Before we can issue a pull request on GitHub, we first ensure it builds, tests, and installs properly on our local system. To do this we need to change our directory to one level above the recipe.

```
cd ..
# i.e. /path/to/astroconda-contrib
```

Now run conda build to compile our sympy recipe into a Conda package. In the example below we are building against Python 3.5:

```
conda build -c http://ssb.stsci.edu/astroconda --skip-existing --python=3.5 sympy
```

That's probably a bit more involved than you thought. Let's break it down. We issue -c [URL] which instructs the build to utilize the AstroConda channel while checking for package dependencies (i.e. the recipe's requirements section). Secondly, we issue --skip-existing to prevent conda build from rebuilding dependencies discovered in the local astroconda-contrib directory. That is to say, if a package defined as a requirement exists remotely, it will then download and install it, rather than rebuild it from scratch. --python= is self-explanatory, and the final argument is the name of the recipe(s) we intend to build.

At this point, if the build was successful, our Conda package (a bzipped tarball) called sympy-1.0-py35_0.tar.bz2 is emitted to /path/to/miniconda3/conda-bld/[os-arch]/. This directory is a local Conda package repository.

To install this new sympy package and interact with it ourselves you could run the following:

```
conda create -n sympy_test --use-local sympy
source activate sympy_test
```

Then manually verify the package is working:

```
python
```

And checking it out for yourself:

```
>>> import sympy
>>> sympy.__file__
'/path/to/miniconda3/envs/sympy_test/lib/python3.5/site-packages/sympy/__init__.py'
```

Now that you have verified the recipe is fully functional and are happy with the outcome, it's time to create a pull request against astroconda-contrib main repository.

Push your sympy-contrib branch up to your fork on GitHub:

```
git push origin sympy-contrib
```

It is recommended that you familiarize yourself with GitHub pull requests before proceeding (see: tutorial).

Using GitHub, you will want to browse to your astroconda-contrib fork, select the sympy-contrib branch from the drop-down menu (the default will read: "Branch: master", next to a black downward-pointing caret). Once selected, click the large green button labeled: "New pull request".

From here, you may wish to edit the title of your pull request and add initial comments or notes regarding what you have done, or list any work that may still need to be done. After submitting your pull request, a member of the Science Software Branch at STScI, or fellow contributors will review the requested changes, ask questions, offer feedback or advice.

At this point if everything appears to be in order your recipe will likely be merged, built, and incorporated into AstroConda!

Updating a recipe in astroconda-contrib

Let's assume time has passed and our `sympy` package from the previous example is no longer up to date with what's generally available on GitHub. Updating recipes is a fairly straight forward process.

At the top of the file you will remember we have a few variables defined encapsulated by `{% %}`. These `jinja2` variables control the name, version, and build revision of the recipe. Using variable interpolation saves time, because it's much easier to edit a single variable that effects an entire recipe, than it is to search/replace specific fields. Typos are also much easier to spot.

`{{ name }}`, `{{ version }}` and `{{ number }}` expand to `sympy`, `1.0` and `0` respectively:

```
{% set name = 'sympy' %}
{% set version = '1.0' %}
{% set number = '0' %}

[...]

build:
    number: {{ number }}

[...]
source:
    fn: {{ name }}-{{ version }}.tar.gz
    url: https://github.com/{{ name }}/{{ name }}/releases/download/{{ name }}-{{␣
↪version }}/{{ name }}-{{ version }}.tar.gz
```

So to update `sympy` to version `1.2`, for example, you would perform the following steps in your forked `astroconda-contrib` repository.

Checkout a new branch

```
git checkout -tb update-sympy master
```

Doing this ensures your new branch is based on `master` rather than your current branch, if any. It also keeps your `master` branch pristine, avoiding merge conflicts in the future.

Make your modifications

```
$EDITOR sympy/meta.yaml

[...]
{% set version = '1.2' %}
#                   ^ Was '1.0', but not anymore.
```

Now save and exit your editor.

Review your modifications

As stated earlier, the fastest way to find out whether your recipe works correctly is to try building it for yourself.

```
conda build -c http://ssb.stsci.edu/astroconda --skip-existing --python=2.7 sympy
conda build -c http://ssb.stsci.edu/astroconda --skip-existing --python=3.5 sympy
```

Did it work? If not, review the error message and make changes accordingly.

Commit your modifications

Assuming you are able to build the package locally, then you're ready to push your changes up to your fork.

```
git add sympy/meta.yaml
git commit -m 'Update sympy 1.0 -> 1.2'
git push origin update-sympy
```

Open a pull request

See: Using Pull Requests

1. Using your browser, visit the `update-sympy` branch of your fork: https://github.com/YOUR_ACCOUNT/astroconda-contrib/tree/update-sympy
2. Click the gray "New pull request" button
3. Fill out the pull request form
4. Click the green "Create pull request" button

That's all there is to it. One of our maintainers will review the pull request and get back to you.

Packages

Packaging reference key:

```
[package]-[version]-[glob]_[build_number]
^Name     ^Version  ^      ^Conda package revision
                    |
                    npXXpyYY
                      ^   ^
                      |   |
                      |   Compiled for Python version
                      |
                      Compiled for NumPY version
```

linux-64 metapackages

- **stsci-1.0.1 (np111py35_0)**
  - astropy >=1.1
  - cfitsio >=3.370
  - d2to1 >=0.2.12
  - ds9 >=7.1
  - fftw >=3.3.4
  - htc_utils >=0.1
  - imexam >=0.5.2
  - numpy 1.11*
  - photutils >=0.2.1

  - poppy >=0.4.0
  - purge_path >=1.0.0
  - pyds9 >=1.8.1
  - pyfftw >=0.9.2
  - python 3.5*
  - stsci-data-analysis
  - stsci-hst
  - webbpsf >=0.4.0
  - webbpsf-data >=0.4.0
- **stsci-1.0.1 (np111py27_0)**
  - astropy >=1.1
  - cfitsio >=3.370
  - d2to1 >=0.2.12
  - ds9 >=7.1
  - fftw >=3.3.4
  - htc_utils >=0.1
  - imexam >=0.5.2
  - numpy 1.11*
  - photutils >=0.2.1
  - poppy >=0.4.0
  - purge_path >=1.0.0
  - pyds9 >=1.8.1
  - pyfftw >=0.9.2
  - python 2.7*
  - stsci-data-analysis
  - stsci-hst
  - webbpsf >=0.4.0
  - webbpsf-data >=0.4.0
- **stsci-data-analysis-1.0.0 (np110py35_0)**
  - asdf ==1.0.2
  - astroimtools ==0.1
  - astropy >=1.1
  - asv ==0.1.1
  - cube-tools ==0.0.0
  - numpy 1.10*
  - python 3.5*

  - specview ==0.1
  - stginga ==0.0.0
- **stsci-data-analysis-1.0.0 (np110py34_0)**
  - asdf ==1.0.2
  - astroimtools ==0.1
  - astropy >=1.1
  - asv ==0.1.1
  - cube-tools ==0.0.0
  - numpy 1.10*
  - python 3.4*
  - specview ==0.1
  - stginga ==0.0.0
- **stsci-data-analysis-1.0.0 (np110py27_0)**
  - asdf ==1.0.2
  - astroimtools ==0.1
  - astropy >=1.1
  - asv ==0.1.1
  - cube-tools ==0.0.0
  - numpy 1.10*
  - python 2.7*
  - specview ==0.1
  - stginga ==0.0.0
- **stsci-data-analysis-1.0.0 (np111py35_2)**
  - asdf >=1.0.2
  - astroimtools >=0.1
  - astropy >=1.1
  - asv >=0.1.1
  - cube-tools >=0.0.0
  - numpy 1.11*
  - python 3.5*
  - specview >=0.1
  - stginga >=0.1
- **stsci-data-analysis-1.0.0 (np111py27_2)**
  - asdf >=1.0.2
  - astroimtools >=0.1
  - astropy >=1.1

  - asv >=0.1.1
  - cube-tools >=0.0.0
  - numpy 1.11*
  - python 2.7*
  - specview >=0.1
  - stginga >=0.1
- **stsci-data-analysis-1.0.0 (np111py35_1)**
  - asdf ==1.0.2
  - astroimtools ==0.1
  - astropy >=1.1
  - asv ==0.1.1
  - cube-tools ==0.0.0
  - numpy 1.11*
  - python 3.5*
  - specview ==0.1
  - stginga ==0.0.0
- **stsci-hst-1.0.4 (np111py35_0)**
  - acstools >=2.0.0
  - astropy >=1.1
  - calcos >=3.1.8
  - costools >=1.2.1
  - crds >=7.0.1
  - d2to1 >=0.2.12
  - drizzlepac >=2.1.3
  - fitsblender >=0.2.6
  - hstcal >=1.0.0
  - nictools >=1.1.3
  - numpy 1.11*
  - purge_path >=1.0.0
  - pyregion >=1.1.2
  - pysynphot >=0.9.8.2
  - python 3.5*
  - reftools >=1.7.1
  - stistools >=1.1
  - stsci.convolve >=2.1.3
  - stsci.distutils >=0.3.8

  - stsci.image >=2.2.0
  - stsci.imagemanip >=1.1.2
  - stsci.imagestats >=1.4.1
  - stsci.ndimage >=0.10.1
  - stsci.numdisplay >=1.6.1
  - stsci.skypac >=0.9
  - stsci.sphere >=0.2
  - stsci.sphinxext >=1.2.2
  - stsci.stimage >=0.2.1
  - stsci.tools >=3.4.1
  - stwcs >=1.2.3
  - wfc3tools >=1.3.1
  - wfpc2tools >=1.0.3
- **stsci-hst-1.0.4 (np111py27_0)**
  - acstools >=2.0.0
  - astropy >=1.1
  - calcos >=3.1.8
  - costools >=1.2.1
  - crds >=7.0.1
  - d2to1 >=0.2.12
  - drizzlepac >=2.1.3
  - fitsblender >=0.2.6
  - hstcal >=1.0.0
  - nictools >=1.1.3
  - numpy 1.11*
  - purge_path >=1.0.0
  - pyregion >=1.1.2
  - pysynphot >=0.9.8.2
  - python 2.7*
  - reftools >=1.7.1
  - stistools >=1.1
  - stsci.convolve >=2.1.3
  - stsci.distutils >=0.3.8
  - stsci.image >=2.2.0
  - stsci.imagemanip >=1.1.2
  - stsci.imagestats >=1.4.1

  - stsci.ndimage >=0.10.1
  - stsci.numdisplay >=1.6.1
  - stsci.skypac >=0.9
  - stsci.sphere >=0.2
  - stsci.sphinxext >=1.2.2
  - stsci.stimage >=0.2.1
  - stsci.tools >=3.4.1
  - stwcs >=1.2.3
  - wfc3tools >=1.3.1
  - wfpc2tools >=1.0.3

linux-64 packages

- acstools-2.0.0
- acstools-2.0.2
- acstools-2.0.4
- acstools-2.0.5
- acstools-2.0.6
- acstools-2.0.7
- appdirs-1.4.0
- aprio-1.0.1
- asdf-1.0.2
- asdf-1.0.3
- asdf-1.2.1
- asdf-standard-1.0.0
- astroimtools-0.1
- astroimtools-0.1.1
- astrolib.coords-0.39.6
- asv-0.1.1
- atlas-generic-3.10.2
- calcos-3.1.3
- calcos-3.1.7
- calcos-3.1.8
- cfitsio-3.370
- costools-1.2.1
- crds-0.0.0
- crds-6.0.0

- crds-7.0.1
- crds-7.0.10
- crds-7.0.7
- crds-7.1.0
- cube-tools-0.0.0
- cube-tools-0.0.1
- d2to1-0.2.12
- decorator-4.0.9
- drizzle-1.1
- drizzle-1.5
- drizzlepac-2.1.3
- drizzlepac-2.1.4
- drizzlepac-2.1.5
- drizzlepac-2.1.6
- drizzlepac-2.1.8
- ds9-7.1
- ds9-7.4
- ds9-7.5
- fftw-3.3.4
- fitsblender-0.2.6
- fitsblender-0.3.0
- fitsblender-0.3.1
- fitsverify-4.18
- ginga-2.5.20151215011852
- ginga-2.5.20160627100500
- ginga-2.5.20160706100500
- ginga-2.5.20160801083400
- ginga-2.5.20161005204600
- ginga-2.5.20161108122300
- ginga-2.5.20161108122301
- ginga-2.6.1
- ginga-2.6.2
- ginga-2.6.2.1
- ginga-v2.6.0
- glue-ginga-0.1.1
- glue-vispy-viewers-0.4

- glue-vispy-viewers-0.5
- glue-vispy-viewers-0.6
- glue-vispy-viewers-0.7.1
- glueviz-0.10.0
- glueviz-0.8.2
- glueviz-0.9.0
- glueviz-0.9.1
- gwcs-0.5
- gwcs-0.5.1
- gwcs-0.6rc1
- gwcs-0.7
- hstcal-1.0.0
- hstcal-1.0.1
- hstcal-1.1.1
- htc_utils-0.1
- imexam-0.5.2
- imexam-0.6.2
- imexam-0.6.3
- imexam-0.7.0
- imexam-0.7.1
- iraf-2.16.1
- jwst_coronagraph_visibility-0.1.0
- jwst_gtvt-0.0.1
- jwst_gtvt-0.0.2
- jwxml-0.1.0
- jwxml-0.2.0
- mosviz-0.0.1
- nictools-1.1.3
- opuscoords-1.0.2
- pandokia-1.3.11
- photutils-0.2.1
- photutils-0.2.2
- photutils-0.3
- photutils-0.3.1
- poppy-0.4.0
- poppy-0.5.0

- poppy-0.5.1
- purge_path-1.0.0
- pydrizzle-6.4.4
- pyds9-1.8.1
- pyds9-1.9.0.dev
- pyfftw-0.9.2
- pyqtgraph-0.10.0
- pyqtgraph-0.9.10
- pyqtgraph-0.9.11
- pyraf-2.1.10
- pyraf-2.1.11
- pyregion-1.1.2
- pysynphot-0.9.8.2
- pysynphot-0.9.8.3
- pysynphot-0.9.8.4
- pysynphot-0.9.8.5
- pysynphot-0.9.8.6
- python-daemon-2.0.5
- pytools-2016.1
- pywcs-1.12.1
- recon-1.0.2
- reftools-1.7.1
- reftools-1.7.3
- reftools-1.7.4
- relic-1.0.4
- relic-1.0.5
- selenium-2.49.2
- sextractor-2.19.5
- shunit2-2.0.3
- specutils-0.2.1
- specview-0.1
- specviz-0.1.2rc3
- specviz-0.2.0rc3
- specviz-0.2.1rc4
- specviz-0.2.1rc5
- specviz-0.2.2rc5

- specviz-0.2.3
- sphere-1.0.10
- sphere-1.0.6
- sphere-1.0.7
- sphinx_rtd_theme-0.1.9
- sphinxcontrib-programoutput-0.8
- stginga-0.0.0
- stginga-0.1
- stginga-0.1.1
- stginga-0.1.2
- stginga-0.1.3
- stistools-1.1
- stsci-1.0.0
- stsci-1.0.1
- stsci-1.0.2
- stsci-2.0.0
- stsci-data-analysis-1.0.0
- stsci-data-analysis-1.1.0
- stsci-data-analysis-1.1.1
- stsci-data-analysis-2.0.1
- stsci-hst-1.0.0
- stsci-hst-1.0.1
- stsci-hst-1.0.2
- stsci-hst-1.0.3
- stsci-hst-1.0.4
- stsci-hst-2.0.0
- stsci.convolve-2.1.3
- stsci.convolve-2.2.0
- stsci.convolve-2.2.1
- stsci.distutils-0.3.8
- stsci.image-2.2.0
- stsci.imagemanip-1.1.2
- stsci.imagestats-1.4.1
- stsci.ndimage-0.10.1
- stsci.numdisplay-1.6.1
- stsci.skypac-0.9

- stsci.skypac-0.9.1
- stsci.sphere-0.2
- stsci.sphinxext-1.2.2
- stsci.stimage-0.2.1
- stsci.tools-3.4.1
- stsci.tools-3.4.2.1
- stsci.tools-3.4.3
- stsci.tools-3.4.4
- stsci.tools-3.4.5
- stsci_sphinx_theme-0.1.0
- stsynphot-0.1b2
- stwcs-1.2.3
- stwcs-1.2.4
- stwcs-1.2.5
- synphot-0.1b3
- verhawk-0.0.1
- verhawk-0.0.2
- wcstools-3.9.2
- wcstools-3.9.4
- webbpsf-0.4.0
- webbpsf-0.5.0
- webbpsf-0.5.1
- webbpsf-data-0.4.0
- webbpsf-data-0.5.0
- wfc3tools-1.3.1
- wfc3tools-1.3.3
- wfc3tools-1.3.4
- wfpc2tools-1.0.3
- xpa-2.1.17
- xpa-2.1.18

osx-64 metapackages

- **stsci-1.0.1 (np111py35_0)**
  - astropy >=1.1
  - cfitsio >=3.370
  - d2to1 >=0.2.12

  - ds9 >=7.1
  - fftw >=3.3.4
  - htc_utils >=0.1
  - imexam >=0.5.2
  - numpy 1.11*
  - photutils >=0.2.1
  - poppy >=0.4.0
  - purge_path >=1.0.0
  - pyds9 >=1.8.1
  - pyfftw >=0.9.2
  - python 3.5*
  - stsci-data-analysis
  - stsci-hst
  - webbpsf >=0.4.0
  - webbpsf-data >=0.4.0
- **stsci-1.0.1 (np111py27_0)**
  - astropy >=1.1
  - cfitsio >=3.370
  - d2to1 >=0.2.12
  - ds9 >=7.1
  - fftw >=3.3.4
  - htc_utils >=0.1
  - imexam >=0.5.2
  - numpy 1.11*
  - photutils >=0.2.1
  - poppy >=0.4.0
  - purge_path >=1.0.0
  - pyds9 >=1.8.1
  - pyfftw >=0.9.2
  - python 2.7*
  - stsci-data-analysis
  - stsci-hst
  - webbpsf >=0.4.0
  - webbpsf-data >=0.4.0
- **stsci-data-analysis-1.0.0 (np110py35_0)**
  - asdf ==1.0.2

  - astroimtools ==0.1
  - astropy >=1.1
  - asv ==0.1.1
  - cube-tools ==0.0.0
  - numpy 1.10*
  - python 3.5*
  - specview ==0.1
  - stginga ==0.0.0
- **stsci-data-analysis-1.0.0 (np110py34_0)**
  - asdf ==1.0.2
  - astroimtools ==0.1
  - astropy >=1.1
  - asv ==0.1.1
  - cube-tools ==0.0.0
  - numpy 1.10*
  - python 3.4*
  - specview ==0.1
  - stginga ==0.0.0
- **stsci-data-analysis-1.0.0 (np110py27_0)**
  - asdf ==1.0.2
  - astroimtools ==0.1
  - astropy >=1.1
  - asv ==0.1.1
  - cube-tools ==0.0.0
  - numpy 1.10*
  - python 2.7*
  - specview ==0.1
  - stginga ==0.0.0
- **stsci-data-analysis-1.0.0 (np111py35_2)**
  - asdf >=1.0.2
  - astroimtools >=0.1
  - astropy >=1.1
  - asv >=0.1.1
  - cube-tools >=0.0.0
  - numpy 1.11*
  - python 3.5*

  - specview >=0.1
  - stginga >=0.1
- **stsci-data-analysis-1.0.0 (np111py27_2)**
  - asdf >=1.0.2
  - astroimtools >=0.1
  - astropy >=1.1
  - asv >=0.1.1
  - cube-tools >=0.0.0
  - numpy 1.11*
  - python 2.7*
  - specview >=0.1
  - stginga >=0.1
- **stsci-hst-1.0.4 (np111py35_0)**
  - acstools >=2.0.0
  - astropy >=1.1
  - calcos >=3.1.8
  - costools >=1.2.1
  - crds >=7.0.1
  - d2to1 >=0.2.12
  - drizzlepac >=2.1.3
  - fitsblender >=0.2.6
  - hstcal >=1.0.0
  - nictools >=1.1.3
  - numpy 1.11*
  - purge_path >=1.0.0
  - pyregion >=1.1.2
  - pysynphot >=0.9.8.2
  - python 3.5*
  - reftools >=1.7.1
  - stistools >=1.1
  - stsci.convolve >=2.1.3
  - stsci.distutils >=0.3.8
  - stsci.image >=2.2.0
  - stsci.imagemanip >=1.1.2
  - stsci.imagestats >=1.4.1
  - stsci.ndimage >=0.10.1

  - stsci.numdisplay >=1.6.1
  - stsci.skypac >=0.9
  - stsci.sphere >=0.2
  - stsci.sphinxext >=1.2.2
  - stsci.stimage >=0.2.1
  - stsci.tools >=3.4.1
  - stwcs >=1.2.3
  - wfc3tools >=1.3.1
  - wfpc2tools >=1.0.3
- **stsci-hst-1.0.4 (np111py27_0)**
  - acstools >=2.0.0
  - astropy >=1.1
  - calcos >=3.1.8
  - costools >=1.2.1
  - crds >=7.0.1
  - d2to1 >=0.2.12
  - drizzlepac >=2.1.3
  - fitsblender >=0.2.6
  - hstcal >=1.0.0
  - nictools >=1.1.3
  - numpy 1.11*
  - purge_path >=1.0.0
  - pyregion >=1.1.2
  - pysynphot >=0.9.8.2
  - python 2.7*
  - reftools >=1.7.1
  - stistools >=1.1
  - stsci.convolve >=2.1.3
  - stsci.distutils >=0.3.8
  - stsci.image >=2.2.0
  - stsci.imagemanip >=1.1.2
  - stsci.imagestats >=1.4.1
  - stsci.ndimage >=0.10.1
  - stsci.numdisplay >=1.6.1
  - stsci.skypac >=0.9
  - stsci.sphere >=0.2

  - stsci.sphinxext >=1.2.2
  - stsci.stimage >=0.2.1
  - stsci.tools >=3.4.1
  - stwcs >=1.2.3
  - wfc3tools >=1.3.1
  - wfpc2tools >=1.0.3

osx-64 packages

- acstools-2.0.0
- acstools-2.0.2
- acstools-2.0.4
- acstools-2.0.5
- acstools-2.0.6
- acstools-2.0.7
- appdirs-1.4.0
- aprio-1.0.1
- asdf-1.0.2
- asdf-1.0.3
- asdf-1.2.1
- asdf-standard-1.0.0
- astroimtools-0.1
- astroimtools-0.1.1
- astrolib.coords-0.39.6
- asv-0.1.1
- calcos-3.1.3
- calcos-3.1.7
- calcos-3.1.8
- cfitsio-3.370
- costools-1.2.1
- crds-0.0.0
- crds-6.0.0
- crds-7.0.1
- crds-7.0.10
- crds-7.0.7
- crds-7.1.0
- cube-tools-0.0.0

- cube-tools-0.0.1
- d2to1-0.2.12
- decorator-4.0.9
- drizzle-1.1
- drizzle-1.5
- drizzlepac-2.1.3
- drizzlepac-2.1.4
- drizzlepac-2.1.5
- drizzlepac-2.1.6
- drizzlepac-2.1.8
- ds9-7.1
- ds9-7.4
- ds9-7.5
- fftw-3.3.4
- fitsblender-0.2.6
- fitsblender-0.3.0
- fitsblender-0.3.1
- fitsverify-4.18
- ginga-2.5.20151215011852
- ginga-2.5.20160627100500
- ginga-2.5.20160706100500
- ginga-2.5.20160801083400
- ginga-2.5.20161005204600
- ginga-2.5.20161108122300
- ginga-2.5.20161108122301
- ginga-2.6.1
- ginga-2.6.2
- ginga-2.6.2.1
- ginga-v2.6.0
- glue-ginga-0.1.1
- glue-vispy-viewers-0.4
- glue-vispy-viewers-0.5
- glue-vispy-viewers-0.6
- glue-vispy-viewers-0.7.1
- glueviz-0.10.0
- glueviz-0.8.2

- glueviz-0.9.0
- glueviz-0.9.1
- gwcs-0.5
- gwcs-0.5.1
- gwcs-0.6rc1
- gwcs-0.7
- hstcal-1.0.0
- hstcal-1.0.1
- hstcal-1.1.1
- htc_utils-0.1
- imexam-0.5.2
- imexam-0.6.2
- imexam-0.6.3
- imexam-0.7.0
- imexam-0.7.1
- iraf-2.16.1
- jwst_coronagraph_visibility-0.1.0
- jwst_gtvt-0.0.1
- jwst_gtvt-0.0.2
- jwxml-0.1.0
- jwxml-0.2.0
- mosviz-0.0.1
- nictools-1.1.3
- opuscoords-1.0.2
- pandokia-1.3.11
- photutils-0.2.1
- photutils-0.2.2
- photutils-0.3
- photutils-0.3.1
- poppy-0.4.0
- poppy-0.5.0
- poppy-0.5.1
- purge_path-1.0.0
- pydrizzle-6.4.4
- pyds9-1.8.1
- pyds9-1.9.0.dev

- pyfftw-0.9.2
- pyobjc-core-3.0.4
- pyobjc-core-3.1.1
- pyobjc-framework-cocoa-3.0.4
- pyobjc-framework-cocoa-3.1.1
- pyobjc-framework-quartz-3.0.4
- pyobjc-framework-quartz-3.1.1
- pyqtgraph-0.10.0
- pyqtgraph-0.9.10
- pyqtgraph-0.9.11
- pyraf-2.1.10
- pyraf-2.1.11
- pyregion-1.1.2
- pysynphot-0.9.8.2
- pysynphot-0.9.8.3
- pysynphot-0.9.8.4
- pysynphot-0.9.8.5
- pysynphot-0.9.8.6
- python-daemon-2.0.5
- pytools-2016.1
- pywcs-1.12.1
- recon-1.0.2
- reftools-1.7.1
- reftools-1.7.3
- reftools-1.7.4
- relic-1.0.4
- relic-1.0.5
- selenium-2.49.2
- sextractor-2.19.5
- shunit2-2.0.3
- specutils-0.2.1
- specview-0.1
- specviz-0.1.2rc3
- specviz-0.2.0rc3
- specviz-0.2.1rc5
- specviz-0.2.2rc5

- specviz-0.2.3
- sphere-1.0.10
- sphere-1.0.6
- sphere-1.0.7
- sphinx_rtd_theme-0.1.9
- sphinxcontrib-programoutput-0.8
- stginga-0.0.0
- stginga-0.1
- stginga-0.1.1
- stginga-0.1.2
- stginga-0.1.3
- stistools-1.1
- stsci-1.0.0
- stsci-1.0.1
- stsci-1.0.2
- stsci-2.0.0
- stsci-data-analysis-0.0.0.dev0
- stsci-data-analysis-1.0.0
- stsci-data-analysis-1.1.0
- stsci-data-analysis-1.1.1
- stsci-data-analysis-2.0.1
- stsci-hst-1.0.0
- stsci-hst-1.0.1
- stsci-hst-1.0.2
- stsci-hst-1.0.3
- stsci-hst-1.0.4
- stsci-hst-2.0.0
- stsci.convolve-2.1.3
- stsci.convolve-2.2.0
- stsci.convolve-2.2.1
- stsci.distutils-0.3.8
- stsci.image-2.2.0
- stsci.imagemanip-1.1.2
- stsci.imagestats-1.4.1
- stsci.ndimage-0.10.1
- stsci.numdisplay-1.6.1

- stsci.skypac-0.9
- stsci.skypac-0.9.1
- stsci.sphere-0.2
- stsci.sphinxext-1.2.2
- stsci.stimage-0.2.1
- stsci.tools-3.4.1
- stsci.tools-3.4.2.1
- stsci.tools-3.4.3
- stsci.tools-3.4.4
- stsci.tools-3.4.5
- stsci_sphinx_theme-0.1.0
- stsynphot-0.1b2
- stwcs-1.2.3
- stwcs-1.2.4
- stwcs-1.2.5
- synphot-0.1b3
- verhawk-0.0.1
- verhawk-0.0.2
- wcstools-3.9.2
- wcstools-3.9.4
- webbpsf-0.4.0
- webbpsf-0.5.0
- webbpsf-0.5.1
- webbpsf-data-0.4.0
- webbpsf-data-0.5.0
- wfc3tools-1.3.1
- wfc3tools-1.3.3
- wfc3tools-1.3.4
- wfpc2tools-1.0.3
- xpa-2.1.17
- xpa-2.1.18

Release Notes

For individual package release notes, please follow the links below. Note that not all packages have release notes, and may simply provide numbered releases.

- ACSTOOLS

- ASDF
- ASDF-STANDARD
- ASTROIMTOOLS
- ASTROLIB.COORDS
- ASV
- CALCOS
- COSTOOLS
- CRDS
- CUBE-TOOLS
- D2TO1
- DRIZZLE
- DRIZZLEPAC
- FITSBLENDER
- GWCS
- HSTCAL
- IMEXAM
- JWST_CORONAGRAPH_VISIBILITY
- JWST_GTVT
- MOSVIZ
- NICTOOLS
- OPUSCOORDS
- PANDOKIA
- PYDRIZZLE
- PYRAF
- PYREGION
- PYSYNPHOT
- PYWCS
- REFTOOLS
- SPECVIEW
- SPECVIZ
- SPHERE
- STGINGA
- STISTOOLS
- STSCI.CONVOLVE
- STSCI.DISTUTILS
- STSCI.IMAGE

- STSCI.IMAGEMANIP
- STSCI.IMAGESTATS
- STSCI.NDIMAGE
- STSCI.NUMDISPLAY
- STSCI.SKYPAC
- STSCI.SPHINXEXT
- STSCI.STIMAGE
- STSCI.TOOLS
- STSCI_SPHINX_THEME
- STWCS
- VERHAWK
- WFC3TOOLS
- WFPC2TOOLS

Resources

AstroConda

GitHub Repositories

- https://github.com/astroconda

General Discussion

- https://groups.google.com/forum/#!forum/astroconda

Recipe Support

- Issue tracker

Space Telescope Science Institute

GitHub Repositories

- https://github.com/spacetelescope

Main Portal

- http://www.stsci.edu

Science Software Branch

- http://ssb.stsci.edu

Science Software Support

- help@stsci.edu

PyRAF FAQ

- http://www.stsci.edu/institute/software_hardware/pyraf/pyraf_faq

Gemini Observatory

Data Processing Software

- http://www.gemini.edu/node/10795

Miniconda & Anaconda (Continuum Analytics, Inc.)

Important: AstroConda is not affiliated with Continuum Analytics, Inc. or its partners.

Miniconda

- http://conda.io/miniconda.html

Anaconda

- https://www.continuum.io/why-anaconda

Anaconda Cloud

- http://anaconda.org

Support

- https://groups.google.com/a/continuum.io/forum/#!forum/anaconda

Conda

General Documentation

- http://conda.io/docs/

Build Documentation

- http://conda.io/docs/building/build.html

Environment Management Documentation

- http://conda.io/docs/using/envs.html

Support

- https://groups.google.com/a/continuum.io/forum/#!forum/conda
- https://gitter.im/conda/conda

General Disclaimer

```
THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS''
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED
TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR
CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
```

```
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT
OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
SUCH DAMAGE.
```