Installation guide - Python environments

Master Class Hard X-ray spectroscopy/imaging

September 10, 2024

This tutorial will give a basic introduction to analyzing solar specific data using **PYTHON**. Heliophysics is a very data-rich field given our favored position close to our local star. Most problems in heliophysics involve combining data from multiple ground and space-based instruments to obtain a global picture in understanding how the sun and the heliosphere evolve.

Traditionally, data analysis in heliophysics was done using a custom-made wrapper library called SolarSoftWare(SSW) written in the programming language IDL. Unfortunately, IDL is not free to use and lacks support to many of the modern functionalities that one would need currently - for example, machine-learning libraries. There is growing community support in porting the existing code written in IDL for various instruments to Python. SunPy is currently one of the most active efforts in this direction. In this tutorial, we will use a range of libraries and custom made scripts to get you started in Solar Physics data analysis in Python.

For this purpose we will use data from Solar orbiter: A spacecraft launched in February 2020 with 10 instruments onboard dedicated to study the sun from up close (at distances down to 0.27 AU or 60 $R_{\odot}$). among these instruments there is one dedicated to study the emission of Hard X-rays in the sun: **STIX**, the **S**pectrometer **T**elescope for **I**maging **X**-rays. Since the spacecraft was launched *recently*, the scientific teams of each instrument are still figuring out the behavior of the instruments and the measurements they take therefore, specialized libraries for instruments such as STIX are still under development or revision. For the Master Class, you are going to read and process the STIX data files (FITS format) using some tools from astropy (common library for astronomy and astrophysics) and sunpy (astropy-based library to process solar specific data).

Contents

The creation of the environment and the requirements installation BEFORE the session is important for optimizing the time of the class. If you are still having trouble with the installation/test we encourage you to use the provided communication channels to solve any doubt.

Installing Python

Python comes pre-installed in most machines. Depending on the operating system that you use it can be either Python 2 or Python 3. The code written for this tutorial and the libraries used will only support Python3.

Though it is possible to install stand-alone Python programming IDEs, we highly recommend that you install Anaconda Python on your system. Anaconda Python will work independently of your system Python and this will help resolve lots of conflicts. Anaconda Python is free to use and is available for Linux, Windows and Mac OS. Below we give short instructions on installing Anaconda Python for these three operating systems in section 1. However, if you want to work with the installation in python, you can skip to 2. After succeeding with the installation, proceed to section 3 referring to python notebooks.

1 Installing Anaconda

1.1 Windows

1.1.1 Visit the Anaconda downloads page

Go to the following link: Anaconda.com/downloads

1.1.2 Download the most latest version of Python3

You will be given an option to download Python2 and Python3 once you land on the downloads page and you have to choose Python3. The installer is around 500 MB.

1.1.3 Run the installer

Once the installer is downloaded, install following the on-screen instructions. It should be relatively straightforward.

You should not select 'Add Anaconda to PATH environment variable' option as this might create problems with already existing programs. You should select 'Register Anaconda as my default Python 3.x' option.

1.1.4 Open Anaconda

Once it finishes the installation, search for 'Anaconda Navigator' in the Programmes menu. If you are able to open it, the installation was successful

1.2 Mac OS

1.2.1 Visit the Anaconda downloads page

Go to the following link: Anaconda.com/downloads

1.2.2 Download .pkg installer for Python3

1.2.3 Run the installer

Double-click open the .pkg installer from the Downloads folder(or the folder you downloaded to) and follow the instructions. This installation should be run as the current USER and towards the end of the installation you've to choose to 'Add Anaconda to PATH'

1.2.4 Source your .bash-rc file

Once the installation process is complete, you need to open a MacOS terminal and type the following commands to make the changes take effect

```
cd ~
source .bashrc
```

1.2.5 Run Python

Open terminal and type

```
python
```

If the installation worked correctly, you should see something like this

Python 3.X.X — Anaconda Inc. —

1.3 Linux

1.3.1 Visit the Anaconda downloads page

Go to the following link: Anaconda.com/downloads

1.3.2 Download the bash (.sh file) installer

1.3.3 Run the installer

Open a terminal run the following command. Do note you need to copy the file name of the just downloaded bash file. Also, make sure you are on the right path to install.

```
bash <filename>.sh
```

Accept the license agreement and type Y when prompted to add Anaconda to your PATH

1.3.4 Source the bash file

Open a terminal and type

```
cd ~
source .bashrc
```

to add Anaconda to your path.

1.4 Creating a Virtual Environment

With conda you can create Python virtual environments, inside of which code development can be done. We will create a new environment named 'MC_HardXRay'[1] and we will install the requisite libraries inside of this environment. These libraries will be inaccessible outside of this environment and hence prevents any conflicts with the rest of the system. Environments are also an easy way to port your code to a new system. You just need to clone the environment in the new system and you're good to go.

To create the environment open a terminal and type the following.

```
conda create --name MC_HardXRay
```

Conda will now create a new environment 'MC_HardXRay' with some packages already pre-installed. Now activate the environment by typing:

```
conda activate MC_HardXRay
```

You'll notice that 'MC_HardXRay' appears at the beginning of your bash new line, indicating you're inside this environment. Now all that is remaining to do is install some

[1]Suggested name. You can name the environment as you want.

necessary packages (in the same order as specified below) so that we can run the code. The exact utilities of these packages will be discussed during the tutorial.

```
conda install python=3.6
conda install -c conda-forge jupyter
conda install numpy
conda install -c conda-forge matplotlib
conda install -c anaconda scipy
conda install astropy
pip3 install sunpy
conda install -c conda-forge notebook
pip install --upgrade jupyter_client
```

You can view the installed packages by typing:

```
conda list
```

Check if all packages are installed. If everything is in order, proceed to section 3.

2 Installation in Python

This section is useful if you had problems installing Anaconda or you prefer to use directly python. Assuming you installed python 3 correctly in your system, proceed to type the following commands in a terminal according to your Operative System.

2.1 Installing virtualEnv

2.1.1 Windows

let's first install pip

```
curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
python3 get-pip.py
pip3 --version
```

You should see the pip version printed in the terminal. Then, install virtualEnv

```
pip3 install virtualenv
```

2.1.2 Mac OS

Follow the same steps as for Windows

2.1.3 Linux

```
sudo apt update
sudo apt install python3-pip
sudo pip3 install virtualenv
```

2.2 Creating the environment and installing libraries

Create the virtual environment from a terminal:

```
python3 -m venv PATH
```

where PATH is the path to the desired location of the virtual environment. If, in terminal, you are in the desired directory, then PATH is just the name of the virtual environment

```
python3 -m venv MC_HardXRay
```

if you want to save it in other location then specify the path to the desired directory:

```
python3 -m venv ROOTPATH/MC_HardXRay
```

Once it's created, you can activate it with the command

```
source MC_HardXRay/bin/activate
```

Again, if your virtual environment is in a different directory, you should specify the path.

You'll notice that 'MC_HardXRay' appears at the beginning of your bash new line, indicating you're inside this environment. Now all that is remaining to do is install some necessary packages (in the same order as specified below) so that we can run the code. The exact utilities of these packages will be discussed during the tutorial.

```
pip3 install numpy
pip3 install datetime
pip3 install matplotlib
pip3 install jupyter
pip3 install scipy
pip3 install astropy
pip3 install sunpy
pip3 install --upgrade jupyter_client
```

You can view the installed packages by typing:

```
pip3 list
```

Check if all packages are installed. If everything is in order, proceed to section 3.

3 Notebooks

As jupyter is installed, we can access jupyter notebooks as long as this virtual environment is active. In order to open the jupyter notebook interface:

```
jupyter notebook
```

To execute the different cells of the notebook use ’Ctrl+Enter’. Use the test script (and test file) to diagnose if your installation was correct.

3.1 Common problems

3.1.1 Installation incompatibility numpy-astropy

For Anaconda and python installations can happen that the recent numpy versions ($\geq$ 1.23) have deprecated functions that the current astropy version cannot support. In this case, some solutions are:

Verify that you have the latest Astropy/sunpy versions (try first). Can happen that some of the issues you might have were already solved by the latest versions of these modules. However, it might not be the case.

Downgrading numpy. You can also try to downgrade the numpy version so that the deprecated functions are still included. This is risky as many Python modules are dependent on numpy and downgrading/uninstalling it may affect other modules in the environment.

AttributeError: module ’numpy’ has no attribute ’asscalar’. This is other issue of incompatibility of external libraries with numpy that miht arise. If this happens, a temporary solution might be to include the following snippet of code in each notebook just after the module imports

```
import numpy
def patch_asscalar(a):
    return a.item()
setattr(numpy, "asscalar", patch_asscalar)
```

3.1.2 Notebooks are not running

For Anaconda, the jupyter package may present some issues during installation or when trying to open notebooks. In that case, we recommend creating a new environment in which you use *notebook* instead of *jupyter*

```
conda install -c conda-forge notebook
```

If you're still having problems, try creating a new environment but installing jupyter at last (after installing all the other modules).

If you can open the notebooks correctly, but the plots are not shown when executed, then you might add the following line at the beginning of your notebook.

```
%matplotlib inline
```