Installation guide - SoloLab V0.1

Python Tool for multi-instrument studies with Solar Orbiter

Created by David Paipa

May 26, 2023

Contents

1 Introduction **2**

2 Installing Anaconda **4**
2.1 Windows . . . 4
2.1.1 Visit the Anaconda downloads page . . . 4
2.1.2 Download the most latest version of Python3 . . . 4
2.1.3 Run the installer . . . 4
2.1.4 Open Anaconda . . . 4
2.2 Mac OS . . . 5
2.2.1 Visit the Anaconda downloads page . . . 5
2.2.2 Download .pkg installer for Python3 . . . 5
2.2.3 Run the installer . . . 5
2.2.4 Source your .bash-rc file . . . 5
2.2.5 Run Python . . . 5
2.3 Linux . . . 6
2.3.1 Visit the Anaconda downloads page . . . 6
2.3.2 Download the bash (.sh file) installer . . . 6
2.3.3 Run the installer . . . 6
2.3.4 Source the bash file . . . 6
2.4 Creating a Virtual Environment . . . 6

3 Installation in Python **8**
3.1 Installing virtualEnv . . . 8
3.1.1 Windows . . . 8
3.1.2 Mac OS . . . 8
3.1.3 Linux . . . 8
3.2 Creating the environment and installing libraries . . . 8

4 pyCDF **10**

5 Notebooks **11**
5.1 Common problems . . . 11

1 Introduction

This tutorial will help you in the installation the requirements for using soloLab, a tool to process and visualize multi-instrument 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, the three instruments we are concerned about in the early development of this tool are:

- **STIX**: the **S**pectrometer **T**elescope for **I**maging **X**-rays.
- **RPW**: the **R**adio and **P**lasma **W**aves instrument.
- **EPD**: the **E**nergetic **P**articles **D**etector.

Since the spacecraft was launched *recently*, the scientific teams of each instrument are still figuring out the behavior of the measurements they take therefore, specialized libraries for processing data from the Solar Orbiter instruments are still under development (or revision).

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.

The utility of a pseudo-automatic visualization tool in python comes from the need of measuring the timing properties of different emissions of solar flares, which can provide information on the physical processes of particle acceleration in flares and their properties. Moreover, comparing the data time series from multiple instruments can lead to interesting results in other areas of heliophysics; not only measuring the properties of solar transients, but may also help to get a deeper knowledge of the quiet sun and the heliosphere.

SoloLab V0.1 uses python libraries and custom made scripts to treat data obtained by Solar Orbiter, helping in the following tasks:

- **Data extraction:** Functions to extract STIX FITS files (L1 pixel data and spectrograms, L1 BKG files) and RPW CDF files (HFR and TNR L2 files).
- **Data processing:** STIX background subtraction (from BKG and/or *quiet* time intervals) and manual energy shifts. RPW background subtraction (from quiet time interval). Filtering of polluted frequencies for RPW.

- **Data visualization:** Plots of simultaneous measurements of STIX and RPW (spectrograms and time profiles per energy/frequency channel) with the possibility of introducing simultaneous EPD time profiles (EPT/electrons) and X-ray spectroscopy results (injected electron powerlaws, electron abundances at different energy thresholds).
- **Estimations and fits:** Fit of RPW time profiles (per frequency) to estimate the exciter velocity using Frequency Drift Rate Analysis (FDRA) and regression methods. Estimation of electron abundances as a function of threshold energy once given the powerlaws obtained from X-ray spectroscopy.

Considerations and contact

Created by David Paipa [contact]

This code is **not** an official Solar Orbiter software, moreover is still a work in progress initially developed as a tool for my thesis. If you have any question, suggestion or notice any bug/mistake do not hesitate to contact me.

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 2. However, if you want to work with the installation in python, you can skip to 3. After succeeding with the installation, proceed to section 5 referring to python notebooks.

2 Installing Anaconda

2.1 Windows

2.1.1 Visit the Anaconda downloads page

Go to the following link: Anaconda.com/downloads

2.1.2 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.

2.1.3 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.

2.1.4 Open Anaconda

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

2.2 Mac OS

2.2.1 Visit the Anaconda downloads page

Go to the following link: Anaconda.com/downloads

2.2.2 Download .pkg installer for Python3

2.2.3 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 have to choose to 'Add Anaconda to PATH'

2.2.4 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
```

2.2.5 Run Python

Open terminal and type

```
python
```

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

Python 3.X.X — Anaconda Inc. —

2.3 Linux

2.3.1 Visit the Anaconda downloads page

Go to the following link: Anaconda.com/downloads

2.3.2 Download the bash (.sh file) installer

2.3.3 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

2.3.4 Source the bash file

Open a terminal and type

```
cd ~
source .bashrc
```

to add Anaconda to your path.

2.4 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 'sololab_workenv'[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 sololab_workenv
```

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

```
conda activate sololab_workenv
```

You'll notice that 'sololab_workenv' 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. These packages are essential for SoloLab to work properly. to do that first is necessary to install pip.

```
conda install pip
```

Once this is done, you'll be able to install the required modules from the *requirements.txt* file available in the SoloLab installation pack.

```
pip install -r requirements.txt
```

Check that there are no errors. If everything is in order, proceed to section 5. You can view the installed packages by typing:

```
conda list
```

3 Installation in Python

This section is better recommended 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.

3.1 Installing virtualEnv

3.1.1 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
```

3.1.2 Mac OS

Follow the same steps as for Windows

3.1.3 Linux

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

3.2 Creating the environment and installing libraries

Create the virtual environment from a terminal:

```
python3 -m venv PATH/sololab_workenv
```

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

```
python3 -m venv sololab_workenv
```

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

```
source sololab_workenv/bin/activate
```

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

You'll notice that 'sololab_workenv' 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. These packages are essential for SoloLab to work properly. To do that first is necessary to install pip.

Once this is done, you'll be able to install the required modules from the *requirements.txt* file available in the SoloLab installation pack.

```
pip install -r requirements.txt
```

Check that there are no errors. If everything is in order, proceed to section 5. You can view the installed packages by typing:

```
conda list
```

4 pyCDF

Is important to check that the spacepy/pyCDF module is installed as it is essential for extracting the information on RPW CDF files. The first step is to verify that *spacepy* is in the list of installed modules. After this, the installation of pyCDF might be a bit more *manual* depending on your OS and how you are working (python or conda).

NOTE: The installation of pycdf is very tricky. The CDF C library must be properly installed in order to use this package. The CDF distribution provides scripts meant to be called in a user's login scripts, definitions.B for bash and definitions.C for C-shell derivatives. (See the module specifications for the CDF Library). These will set environment variables specifying the location of the library; pycdf will respect these variables if they are set. Otherwise it will search the standard system library path and the default installation locations for the CDF library.

In case of having complications, you can check the troubleshooting page of pycdf.

5 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.

5.1 Common problems

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
```

The creation of the environment and the correct 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.