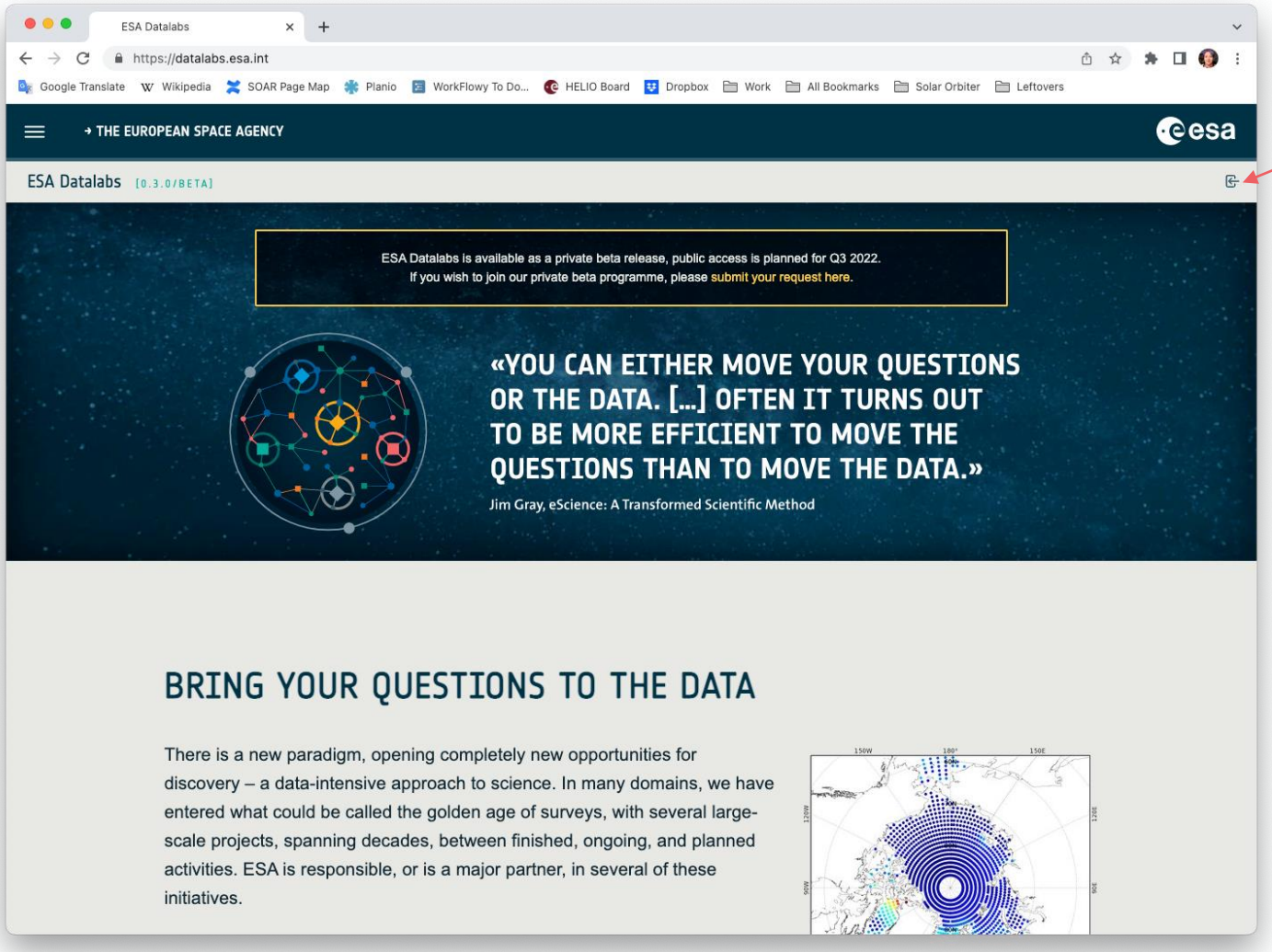
(Starting) Heliophysics in Datalabs

Helen Middleton

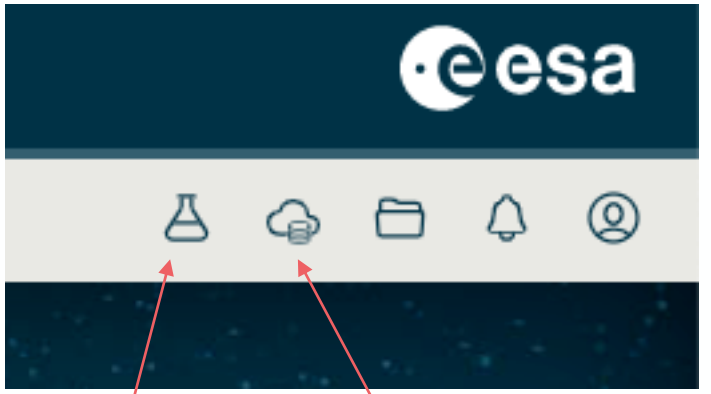
ESA ESAC

22/11/2022

So last week I thought I was a participant…



Log in (LDAP)

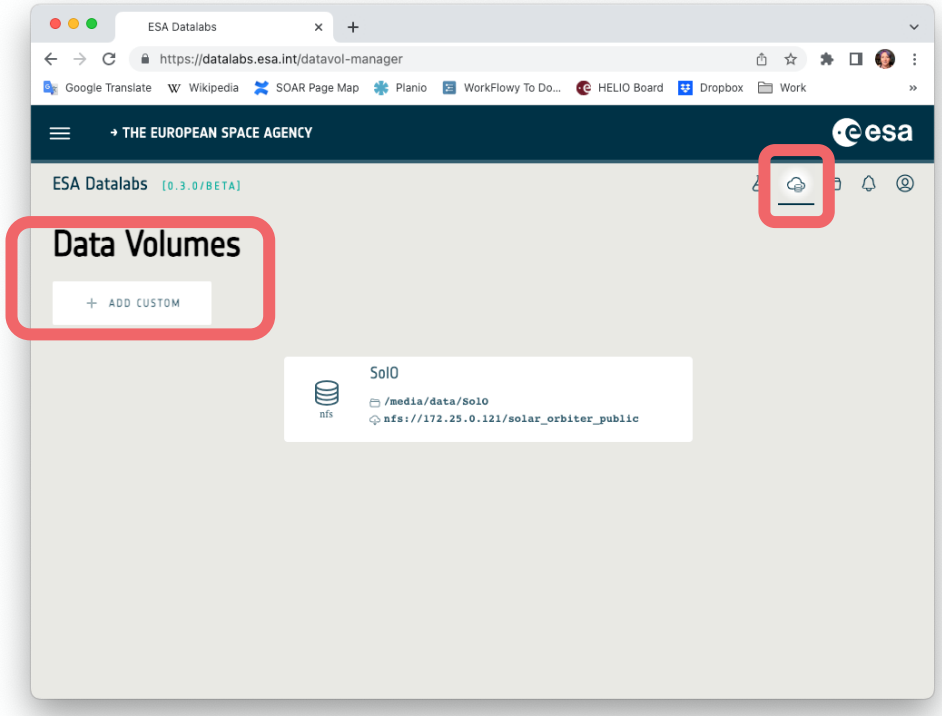


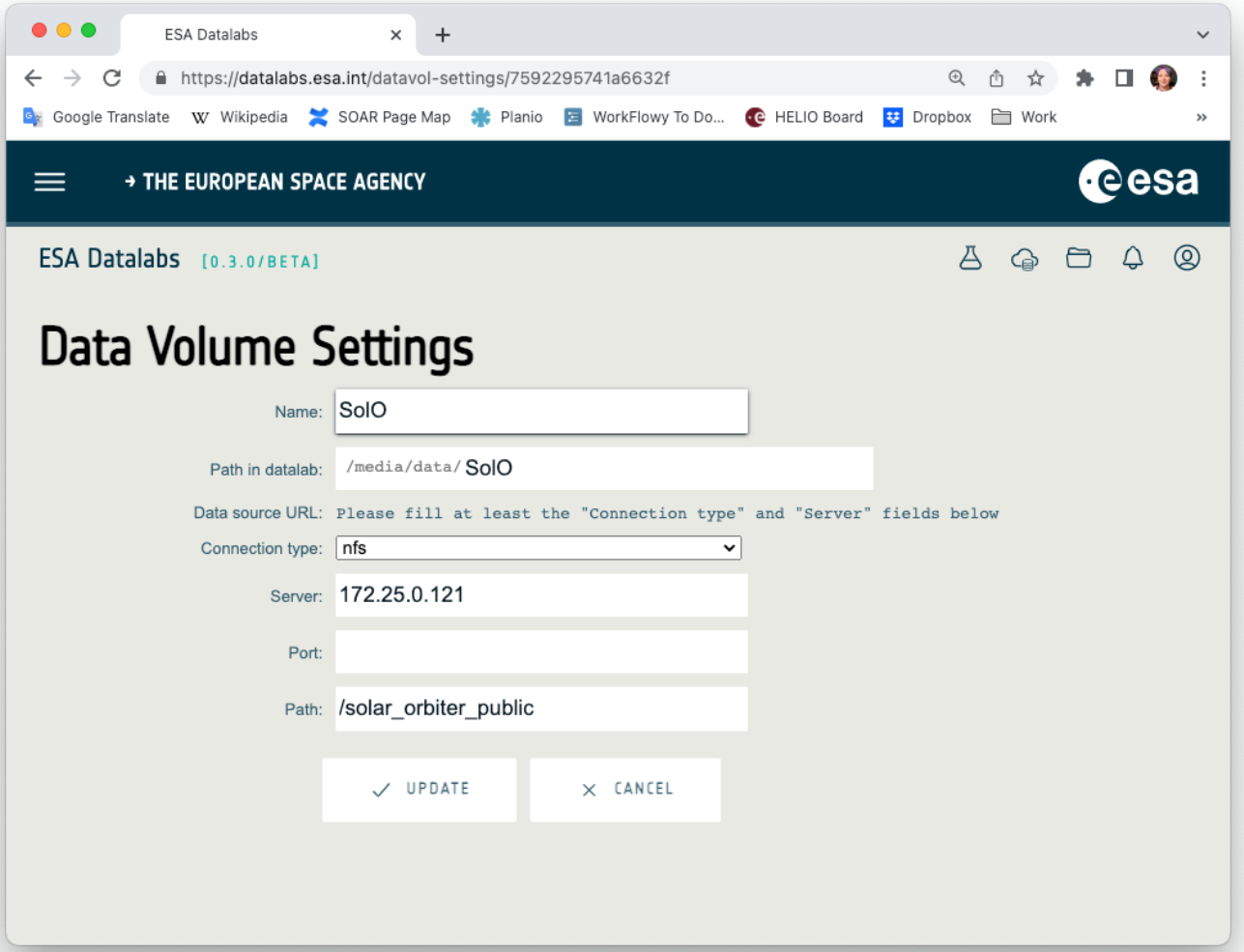
Datalabs (notebooks)

Mount data

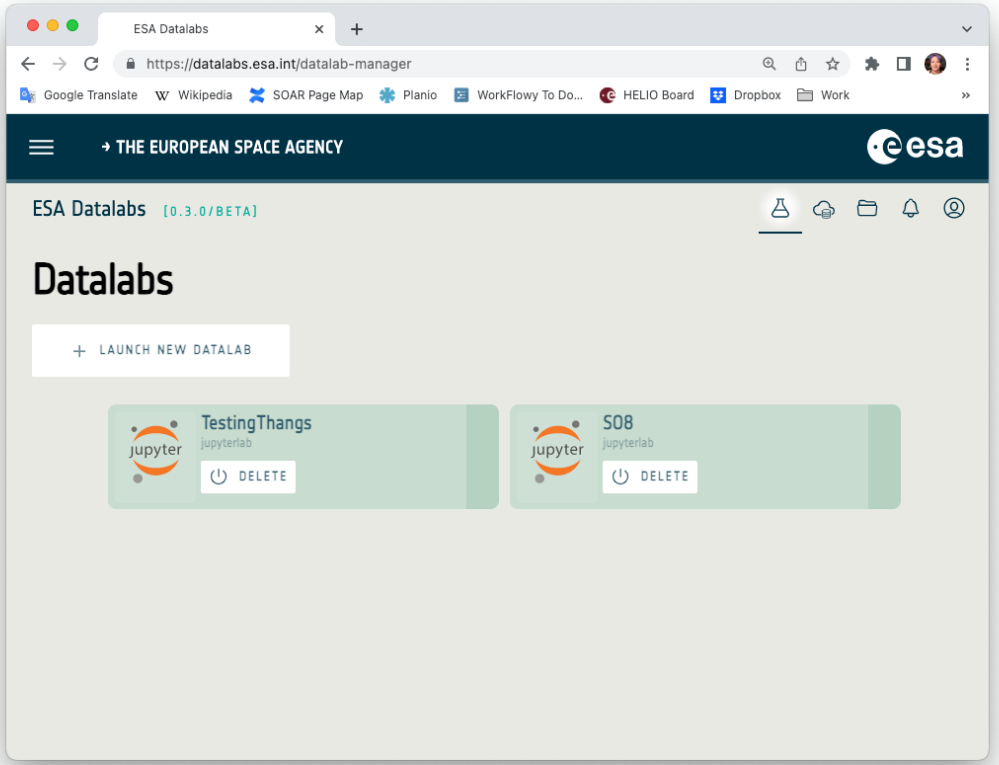
Mounting an archive!

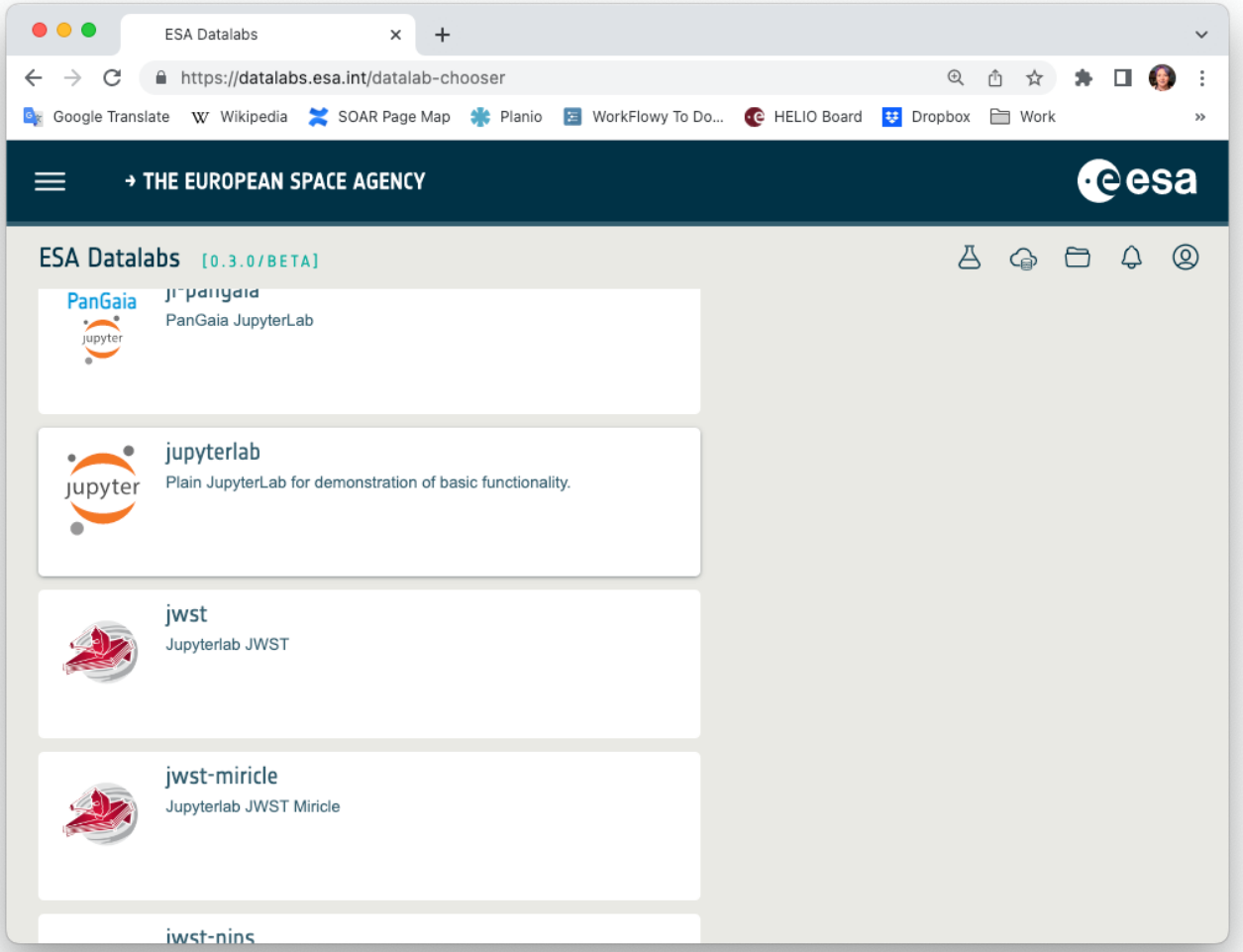
Given this information: SOAR science data: 172.25.0.121:/solar_orbiter_public



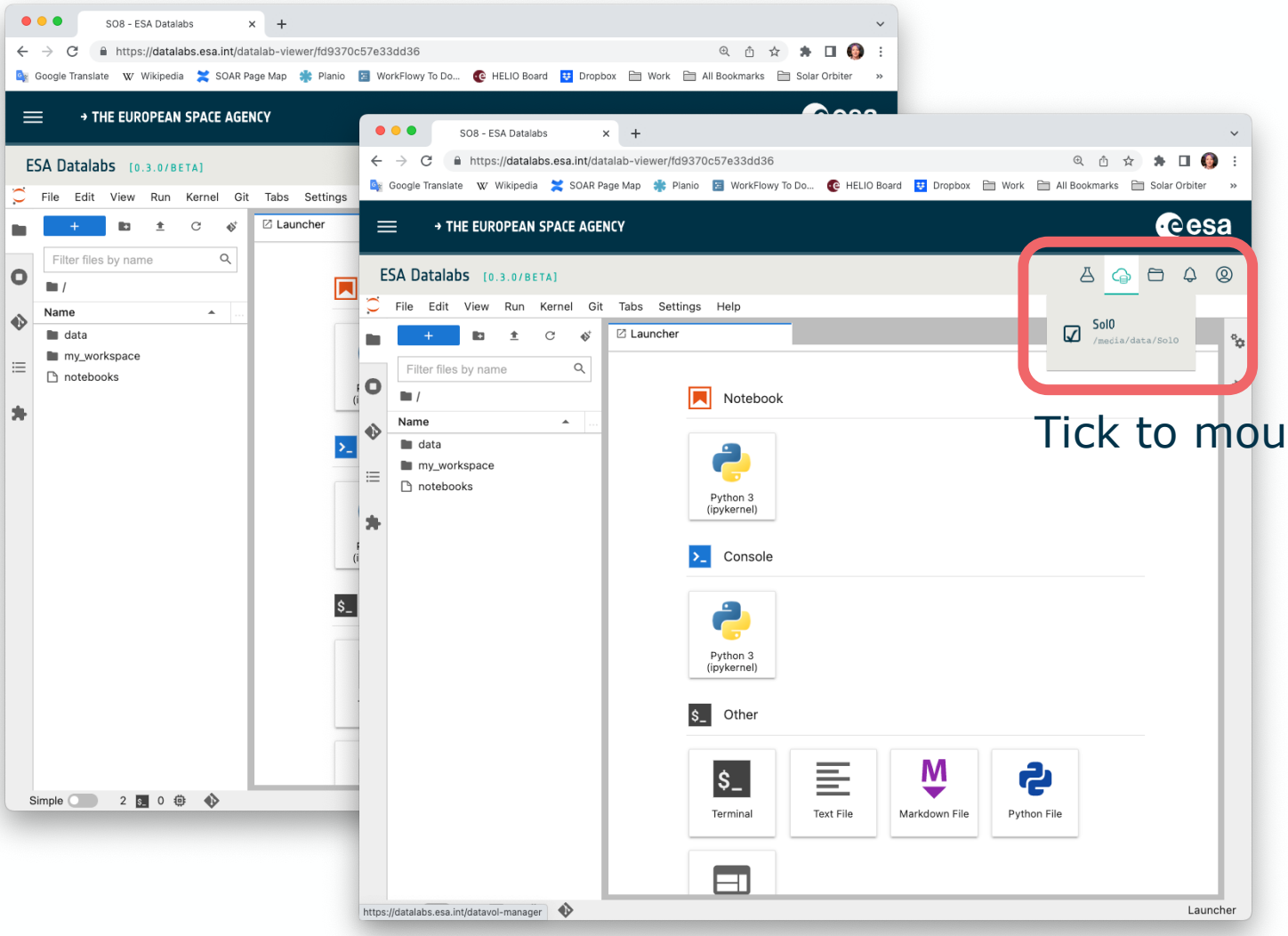


Start a Jupyterlab

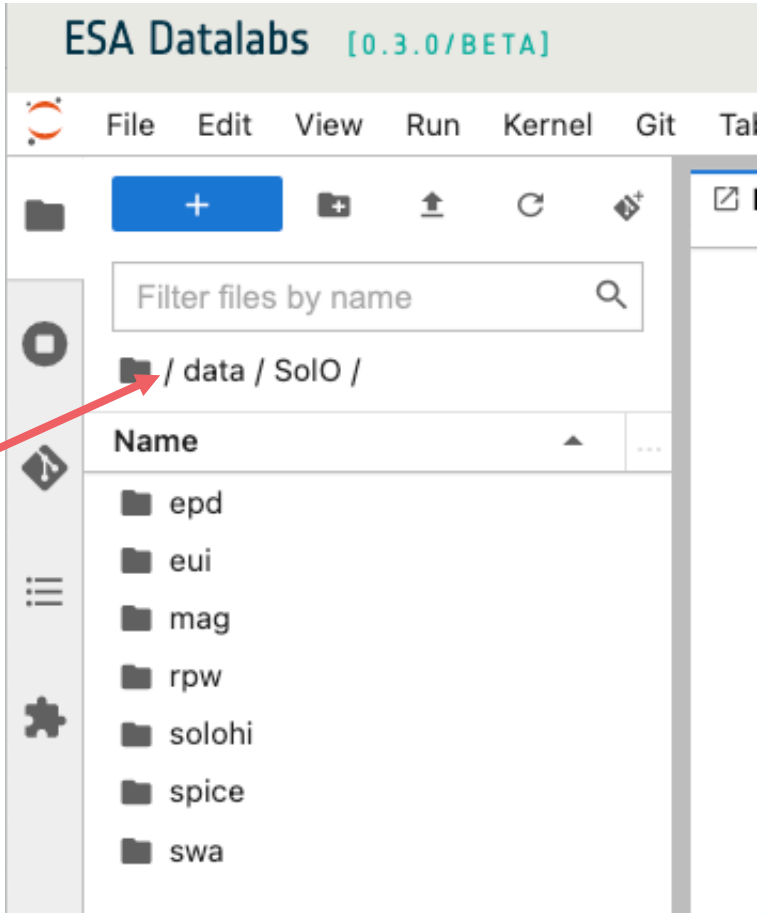




Connect the jupyterlab and the mounted data

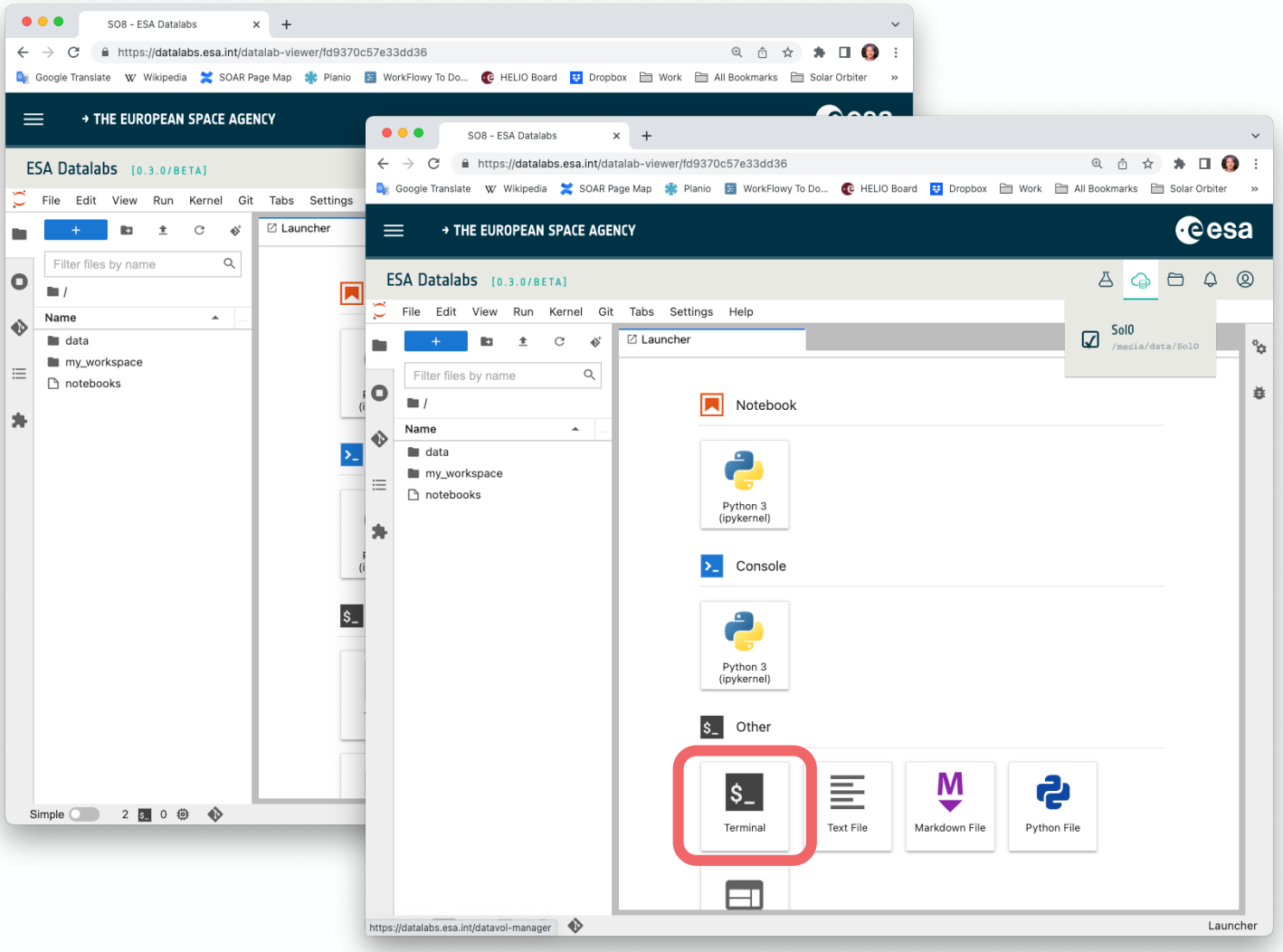


Tick to mount

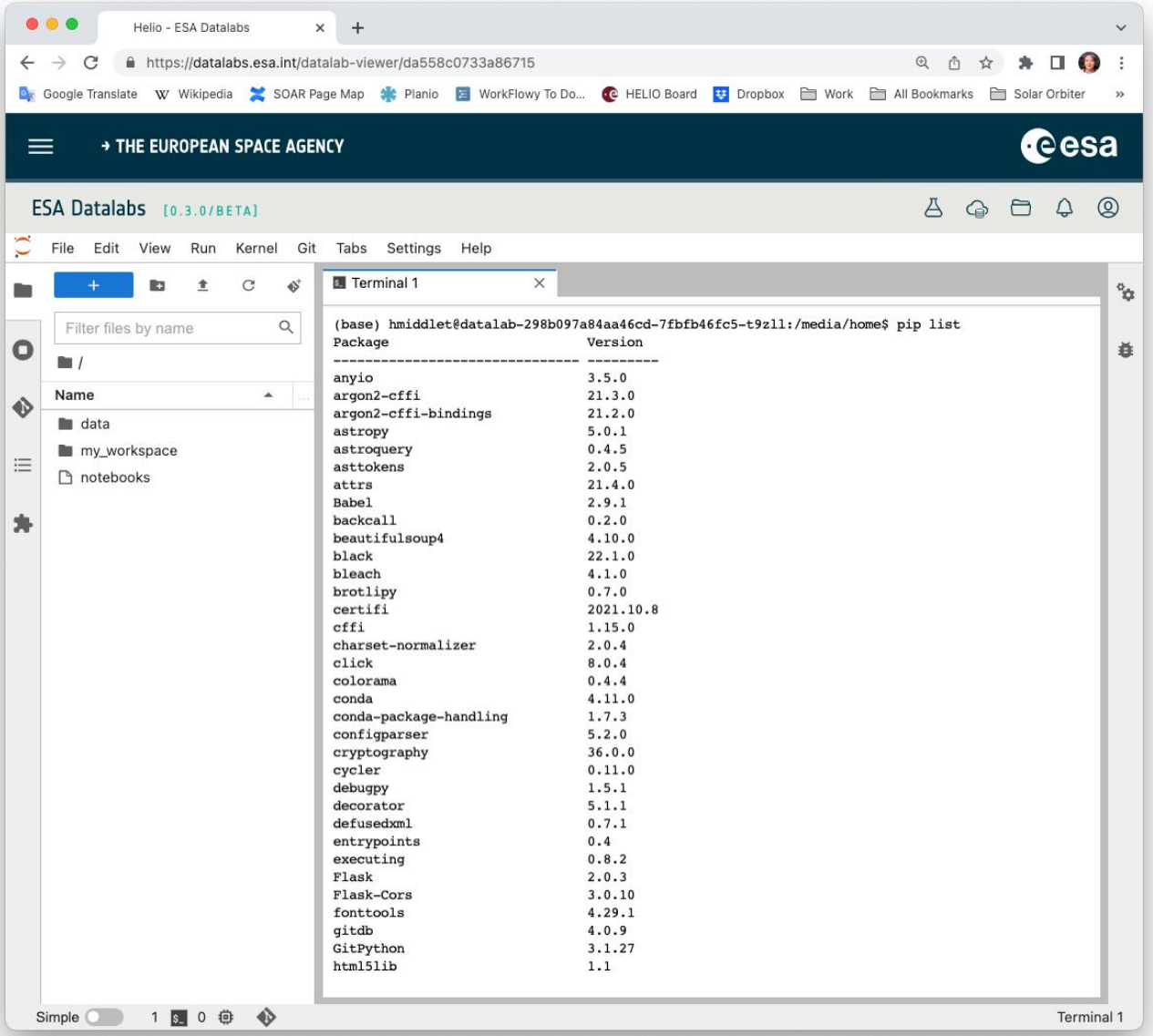


STIX, PHI and Metis data is coming!

Next, find a terminal



What's already there? How to install other stuff



To find out what's there already:
$ pip list

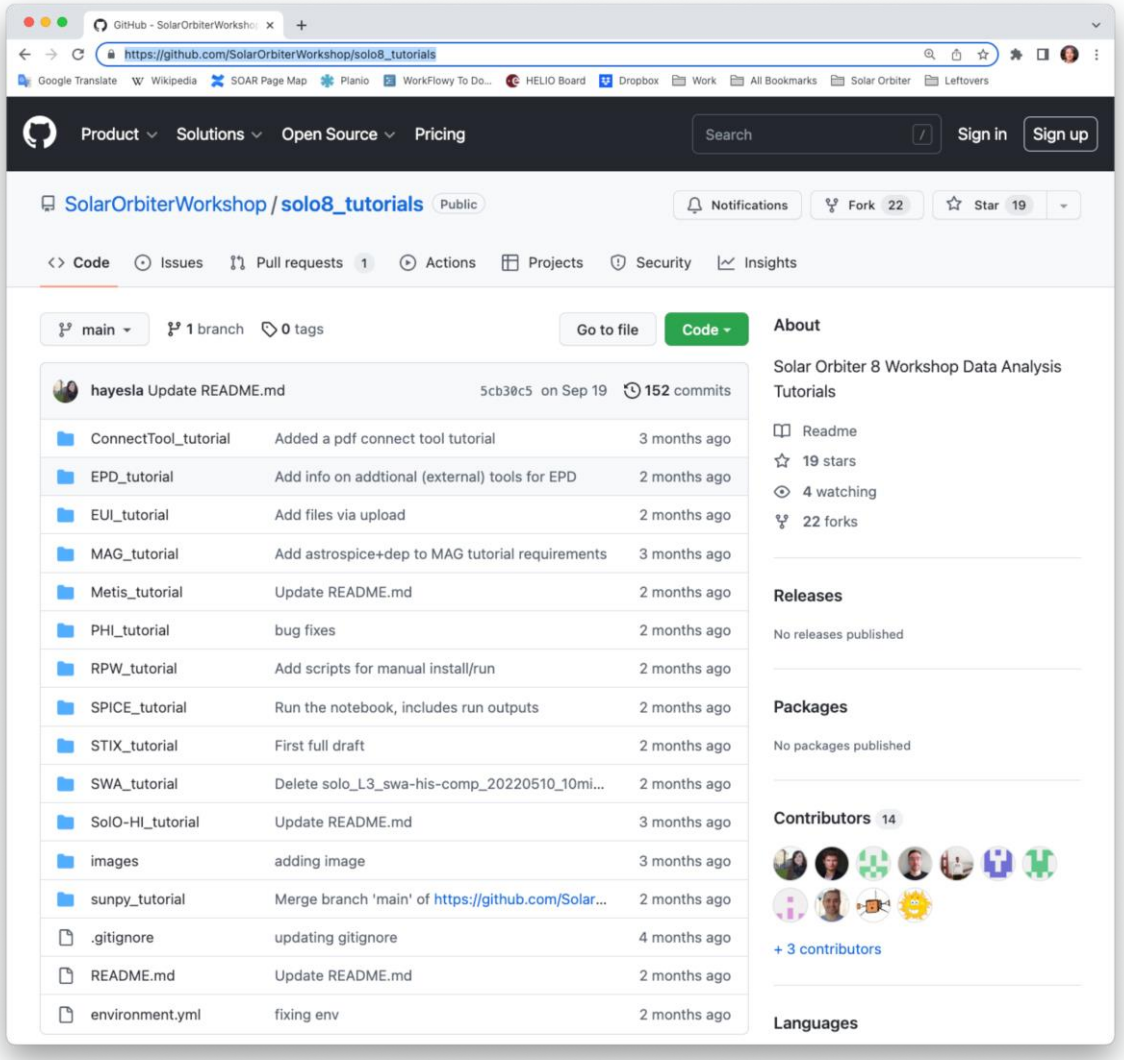
I know I'll need these:
- astropy
- astroquery
- matplotlib
- numpy
- pandas

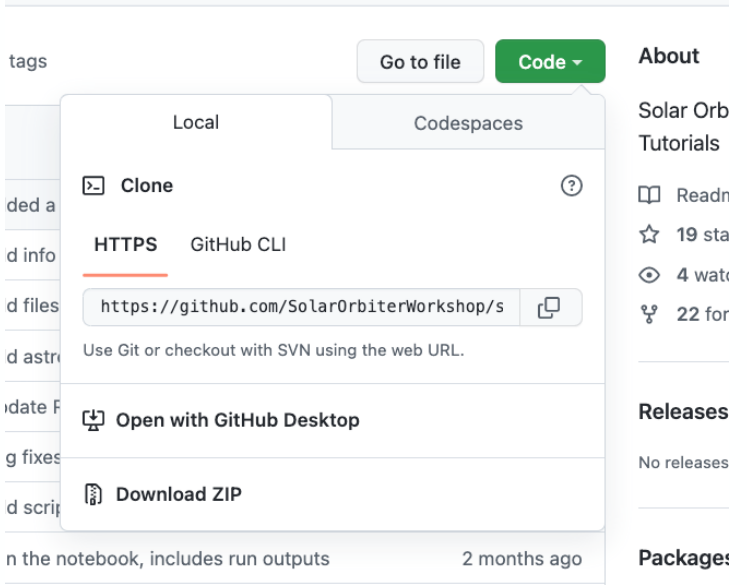
But I'll also need at least:
- sunpy (which needs scipy)
- cdflib

$ pip install sunpy
etc.

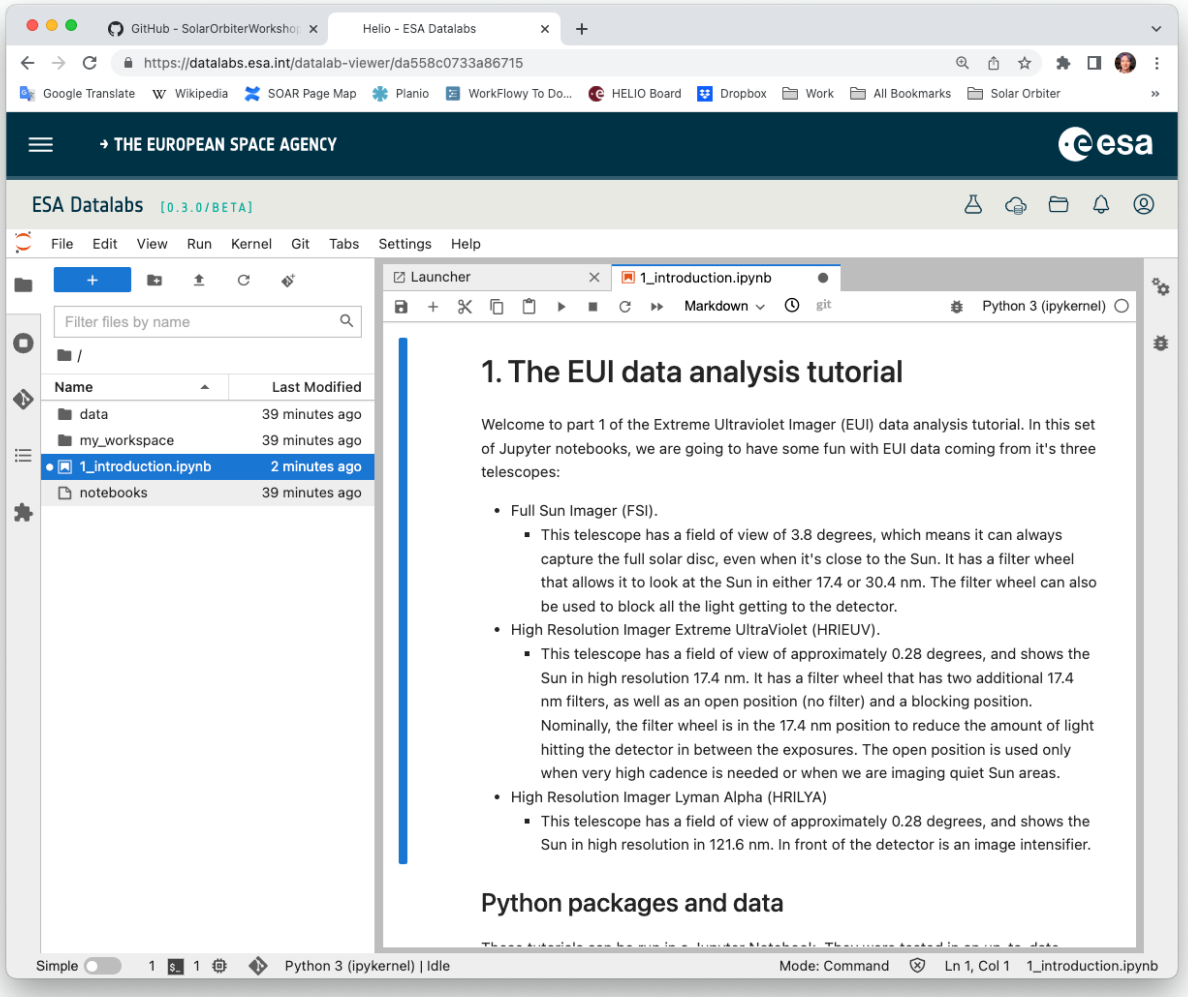
Solar Orbiter Data Tutorial

SO8, Belfast: https://github.com/SolarOrbiterWorkshop/solo8_tutorials





Play with it!

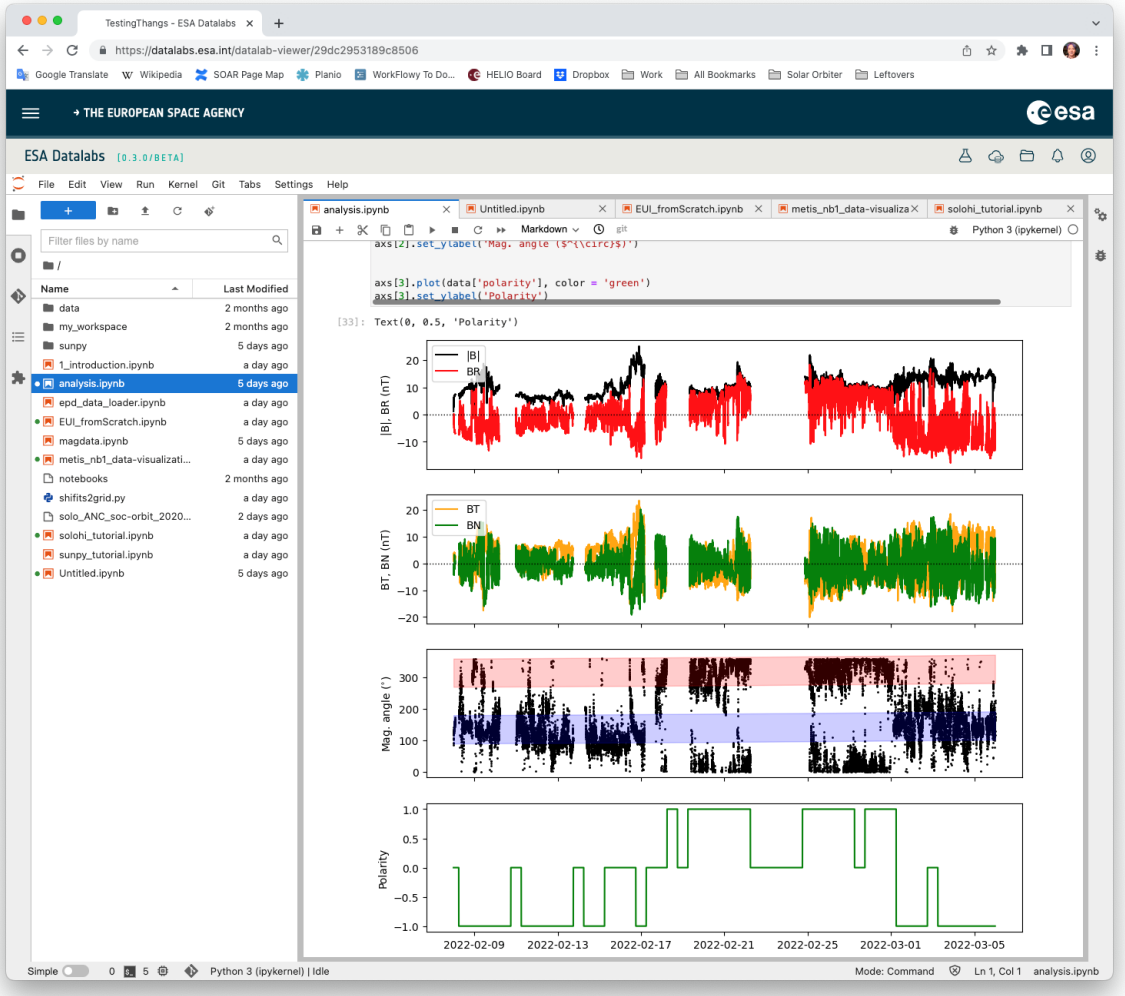


Drag and drop into folder space, double click to open

Contains names of packages to install, and location of a data package for the demonstration…

…but we know we don't need most of those because all the science data is right here!

What next for heliophysics?



Quick and easy

- Inclusion of Proba-2 Jupyter notebooks already tested and available in the Proba-2 archive
- Inclusion of all Jupyter notebooks presented at the September 2022 Solar Orbiter workshop
- Inclusion of some of the Jupyter notebooks detailed at the first Python in Heliophysics summer school (ESAC, June 2022) especially astropy tutorial for heliophysics and sunpy tutorial

What next for heliophysics?

More long term

- Inclusion of JHelioviewer (already planned) and Solar Mach (a python tool developed through the Heliophysics archives user group), both very popular in the Solar Orbiter community
- Develop in-house Jupyter notebooks for science cases, especially on magnetic connectivity for Solar Orbiter, based on interaction with the community
- Inclusion of SOHO data and related Jupyter notebooks
- Inclusion of Cluster data, which is not straightforward (files are concatenated on the fly), e.g., which data format? CEF, CDF, HDF5?
- Inclusion of Jupyter notebooks for Cluster (curlometer already developed by HM) and adapt some of the many MMS notebooks already available on pySPEDAS
- Make Ulysses data available

Launcher | Terminal 3 | Learning.ipynb | time_utils.py | 1_introduction.ipynb

Markdown | git | Python 3 (ipykernel)

EUI (Extreme Ultraviolet Imager)

I've started by picking stuff out of the tutorial bit by bit and trying to get it to work. I don't want to just blindly load packages, but install and import them as I need them. I know I'll need these:

[16]:
```
# For TAP metadata request to SOAR (Solar Orbiter ARchive):
from astroquery.utils.tap.core import TapPlus

# We know we'll be plotting stuff:
import matplotlib.pyplot as plt

# For bigger plots:
plt.rcParams["figure.figsize"] = (10, 10)
```

The EUI tutorial notebook uses sunpy_soar and its search and fetch functions (called Fido), but we can use a TAP metadata search to find the filenames that we already have locally in the mounted data volume. We'll start with some Lyman Alpha images.

Detailed instructions for this are on the lovely SOAR help pages at https://www.cosmos.esa.int/web/soar/home

[17]:
```
# Equivalent to sunpy_soar Fido search:

#instrument = a.Instrument('EUI')
#time = a.Time('2022-03-06 18:0:00', '2022-03-06 18:10:00')
#level = a.Level(2)
#product = a.soar.Product('EUI-HRILYA1216-IMAGE')
#result = Fido.search(instrument & time & level & product)
#print(result)


SOAR = TapPlus(url="http://soar.esac.esa.int/soar-sl-tap/tap/")
results = SOAR.launch_job("SELECT filename FROM v_sc_data_item "
                          "WHERE descriptor='EUI-HRILYA1216-IMAGE' "
                          "AND begin_time>='2022-03-06T18:00:00' "
                          "AND end_time<='2022-03-06T18:10:00' "
                          "AND level='L2' "
                          "ORDER BY filename"
                          )
lya = results.get_results()

lya
```

[17]: *Table length=10*

filename
object
solo_L2_eui-hrilya1216-image_20220306T180030287_V01.fits
solo_L2_eui-hrilya1216-image_20220306T180130287_V01.fits

Launcher | Terminal 3 | Learning.ipynb | time_utils.py | 1_introduction.ipynb

Markdown | git | Python 3 (ipykernel)

```
lya
```

[17]: *Table length=10*

filename
object
solo_L2_eui-hrilya1216-image_20220306T180030287_V01.fits
solo_L2_eui-hrilya1216-image_20220306T180130287_V01.fits
solo_L2_eui-hrilya1216-image_20220306T180230287_V01.fits
solo_L2_eui-hrilya1216-image_20220306T180330287_V01.fits
solo_L2_eui-hrilya1216-image_20220306T180430287_V01.fits
solo_L2_eui-hrilya1216-image_20220306T180530287_V01.fits
solo_L2_eui-hrilya1216-image_20220306T180630288_V01.fits
solo_L2_eui-hrilya1216-image_20220306T180730288_V01.fits
solo_L2_eui-hrilya1216-image_20220306T180830288_V01.fits
solo_L2_eui-hrilya1216-image_20220306T180930288_V01.fits

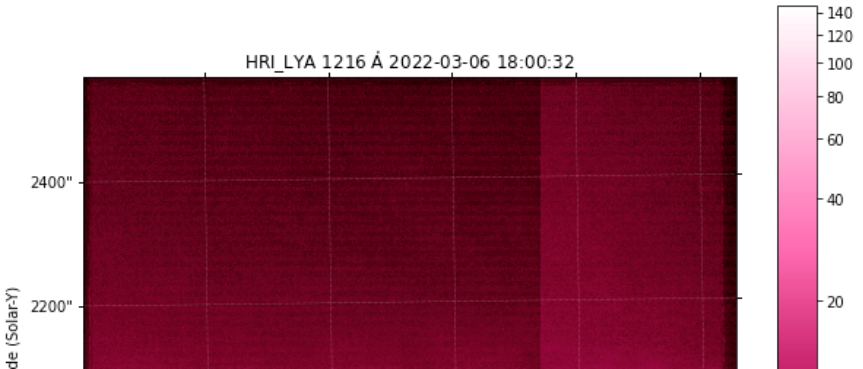
[ ]:
```
# 'data/SolO/' + instrument (lower case) + level (upper case) + year
# Given our request, the data will be here
EUIDATAPATH = 'data/SolO/eui/L2/2022/'

# GIVE WARNING HERE ABOUT OPENING LARGE FOLDERS!!!
```

OK... sunpy needs scipy which isn't already there, so go install it.

[18]:
```
import sunpy.map

mymap = sunpy.map.Map(EUIDATAPATH+lya['filename'][0])
mymap.peek()
```

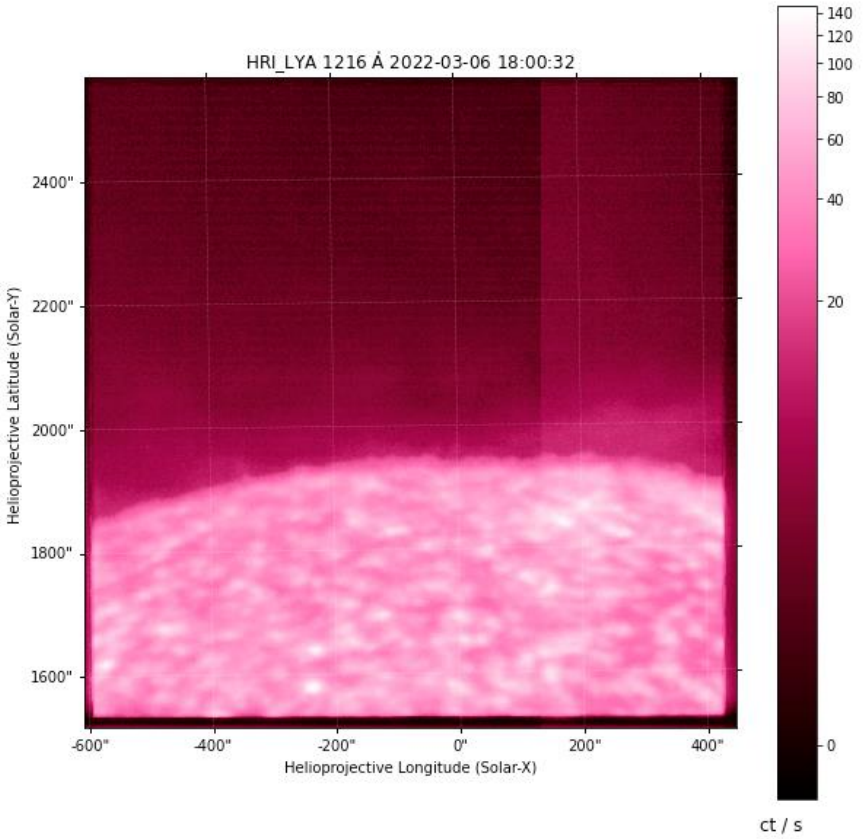


Launcher | Terminal 3 | Learning.ipynb | time_utils.py | 1_introduction.ipynb

Markdown | git | Python 3 (ipykernel)

[18]:
```
import sunpy.map

mymap = sunpy.map.Map(EUIDATAPATH+lya['filename'][0])
mymap.peek()
```



Just the map on its own gives info about the image:

[H! Click!]

[19]:
```
mymap
```

[19]:
```
<sunpy.map.sources.solo.EUIMap object at 0x7f191878b5b0>
```

Observatory	Solar Orbiter
Instrument	EUI
Detector	HRI_LYA
Measurement	1216.0 Angstrom

Image colormap uses histogram equalization
Click on the image to toggle between units

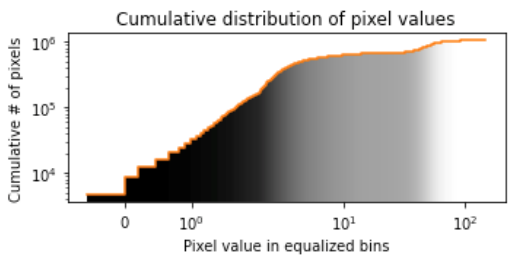
Just the map on its own gives info about the image:

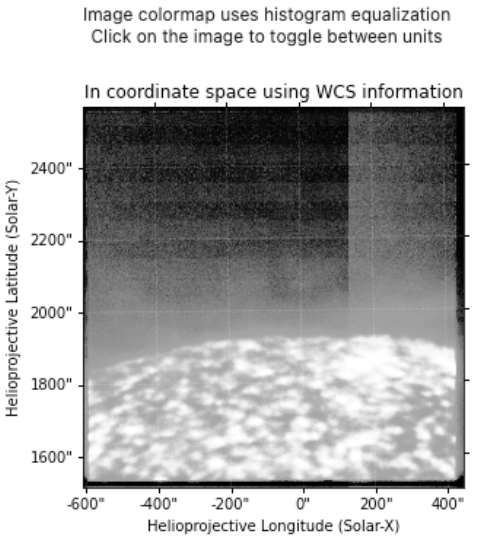
[H! Click!]

[19]:
```
mymap
```

[19]:
```
<sunpy.map.sources.solo.EUIMap object at 0x7f191878b5b0>
```

Observatory	Solar Orbiter
Instrument	EUI
Detector	HRI_LYA
Measurement	1216.0 Angstrom
Wavelength	1216.0 Angstrom
Observation Date	2022-03-06 18:00:32
Exposure Time	5.0 s
Dimension	[1024. 1024.] pix
Coordinate System	helioprojective
Scale	[1.028 1.028] arcsec / pix
Reference Pixel	[511.5 511.5] pix
Reference Coord	[-75.15083347 2037.10438067] arcsec





This following cell is supposed to give a lovely animation, but I couldn't get it working. It uses

`%matplotlib notebook`

or

`%matplotlib widget`

[20]:
```
sequence = sunpy.map.Map([EUIDATAPATH+x for x in lya['filename']], sequence=True)

ani = sequence.plot(interval=1000)

plt.show()
```

Launcher | Terminal 3 | Learning.ipynb | time_utils.py | 1_introduction.ipynb

Code | git | Python 3 (ipykernel)

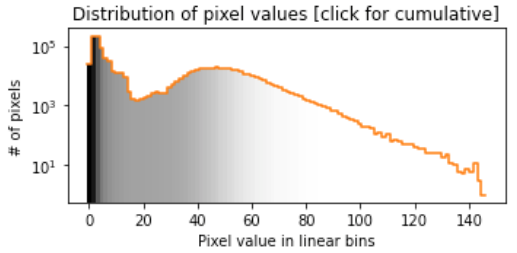
Just the map on its own gives info about the image:

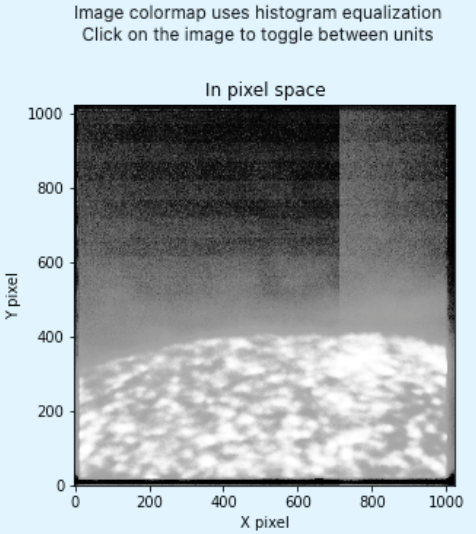
[H! Click!]

[19]:
```
mymap
```

[19]:
```
<sunpy.map.sources.solo.EUIMap object at 0x7f191878b5b0>
```

Observatory	Solar Orbiter
Instrument	EUI
Detector	HRI_LYA
Measurement	1216.0 Angstrom
Wavelength	1216.0 Angstrom
Observation Date	2022-03-06 18:00:32
Exposure Time	5.0 s
Dimension	[1024. 1024.] pix
Coordinate System	helioprojective
Scale	[1.028 1.028] arcsec / pix
Reference Pixel	[511.5 511.5] pix
Reference Coord	[-75.15083347 2037.10438067] arcsec





This following cell is supposed to give a lovely animation, but I couldn't get it working. It uses

`%matplotlib notebook`

or

`%matplotlib widget`

[20]:
```
sequence = sunpy.map.Map([EUIDATAPATH+x for x in lya['filename']], sequence=True)

ani = sequence.plot(interval=1000)

plt.show()
```

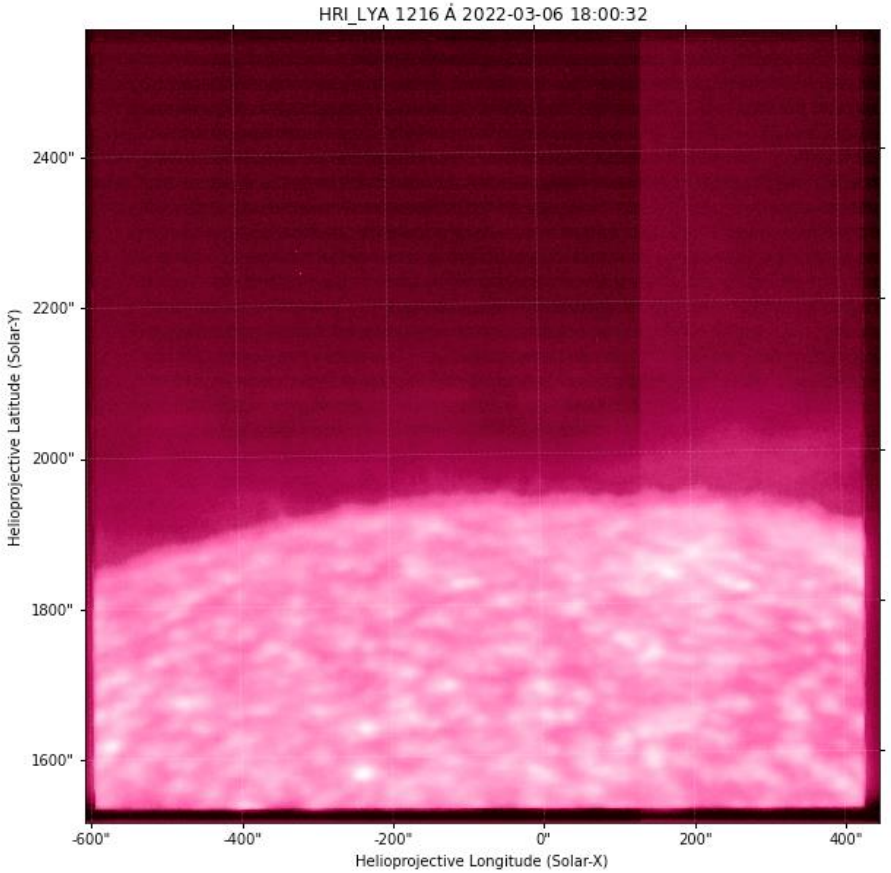
Launcher | Terminal 3 | Learning.ipynb | time_utils.py | 1_introduction.ipynb

Code | git | Python 3 (ipykernel)

[20]:
```
sequence = sunpy.map.Map([EUIDATAPATH+x for x in lya['filename']], sequence=True)

ani = sequence.plot(interval=1000)

plt.show()
```



But I can look through the details of each image:

[H! Remember to click!]

[21]:
```
# Show the map:

sequence
```

```
WARNING: SunpyUserWarning: Rendering the summary for a MapSequence of 10 Maps may take a while. [sunpy.map.mapsequence]
```

[21]:
```
<sunpy.map.mapsequence.MapSequence object at 0x7f195963b370>
```

Launcher | Terminal 3 | Learning.ipynb | time_utils.py | 1_introduction.ipynb

Code | git | Python 3 (ipykernel)

[21]:
```
# Show the map:

sequence
```

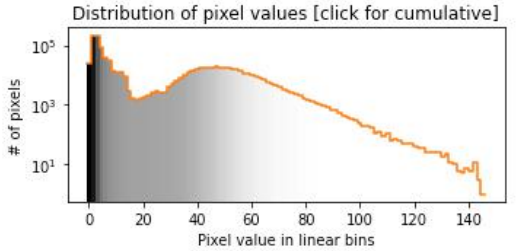
```
WARNING: SunpyUserWarning: Rendering the summary for a MapSequence of 10 Maps may take a while. [sunpy.map.mapsequence]
```

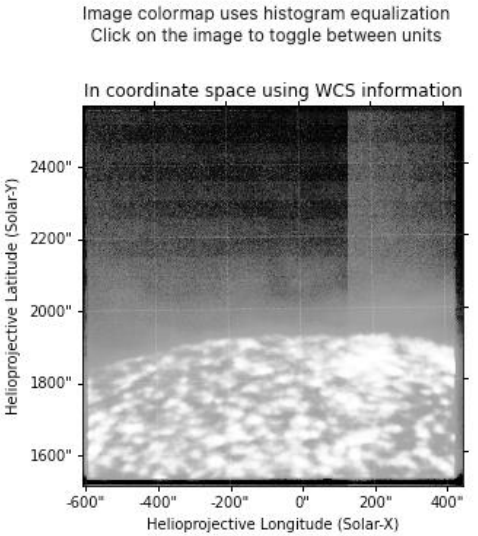
[21]:
```
<sunpy.map.mapsequence.MapSequence object at 0x7f195963b370>
MapSequence of 10 elements, with maps from EUIMap
```

← → Map at index 0

```
<sunpy.map.sources.solo.EUIMap object at 0x7f1959abb370>
```

Observatory	Solar Orbiter
Instrument	EUI
Detector	HRI_LYA
Measurement	1216.0 Angstrom
Wavelength	1216.0 Angstrom
Observation Date	2022-03-06 18:00:32
Exposure Time	5.0 s
Dimension	[1024. 1024.] pix
Coordinate System	helioprojective
Scale	[1.028 1.028] arcsec / pix
Reference Pixel	[511.5 511.5] pix
Reference Coord	[-75.15083347 2037.10438067] arcsec





Now try a different filter: HRI EUV174

[22]:
```
# Fido code:
#instrument = a.Instrument('EUI')
#time = a.Time('2022-03-27 21:24:00', '2022-03-27 21:50:00')
#level = a.Level(2)
#product = a.soar.Product('EUI-HRIEUV174-IMAGE')


results = SOAR.launch_job("SELECT filename FROM v_sc_data_item "
                          "WHERE descriptor='EUI-HRIEUV174-IMAGE' "
```

Launcher | Terminal 3 | Learning.ipynb | time_utils.py | 1_introduction.ipynb

Code | git | Python 3 (ipykernel)

Now try a different filter: HRI EUV174

[22]:
```
# Fido code:
#instrument = a.Instrument('EUI')
#time = a.Time('2022-03-27 21:24:00', '2022-03-27 21:50:00')
#level = a.Level(2)
#product = a.soar.Product('EUI-HRIEUV174-IMAGE')


results = SOAR.launch_job("SELECT filename FROM v_sc_data_item "
                          "WHERE descriptor='EUI-HRIEUV174-IMAGE' "
                          "AND begin_time>='2022-03-27 21:24:00' "
                          "AND end_time<='2022-03-27 21:50:00' "
                          "AND level='L2' "
                          "ORDER BY filename"
                          )
euv174 = results.get_results()
euv174
```

[22]: *Table length=26*

filename
object
solo_L2_eui-hrieuv174-image_20220327T212410251_V01.fits
solo_L2_eui-hrieuv174-image_20220327T212510251_V01.fits
solo_L2_eui-hrieuv174-image_20220327T212610250_V01.fits
solo_L2_eui-hrieuv174-image_20220327T212710431_V01.fits
solo_L2_eui-hrieuv174-image_20220327T212810252_V01.fits
solo_L2_eui-hrieuv174-image_20220327T212910252_V01.fits
solo_L2_eui-hrieuv174-image_20220327T213010284_V01.fits
solo_L2_eui-hrieuv174-image_20220327T213110252_V01.fits
solo_L2_eui-hrieuv174-image_20220327T213210252_V01.fits
solo_L2_eui-hrieuv174-image_20220327T213310252_V01.fits
...
solo_L2_eui-hrieuv174-image_20220327T214010253_V01.fits
solo_L2_eui-hrieuv174-image_20220327T214110277_V01.fits
solo_L2_eui-hrieuv174-image_20220327T214210252_V01.fits
solo_L2_eui-hrieuv174-image_20220327T214310253_V01.fits
solo_L2_eui-hrieuv174-image_20220327T214410253_V01.fits
solo_L2_eui-hrieuv174-image_20220327T214510328_V01.fits
solo_L2_eui-hrieuv174-image_20220327T214610255_V01.fits
solo_L2_eui-hrieuv174-image_20220327T214710254_V01.fits

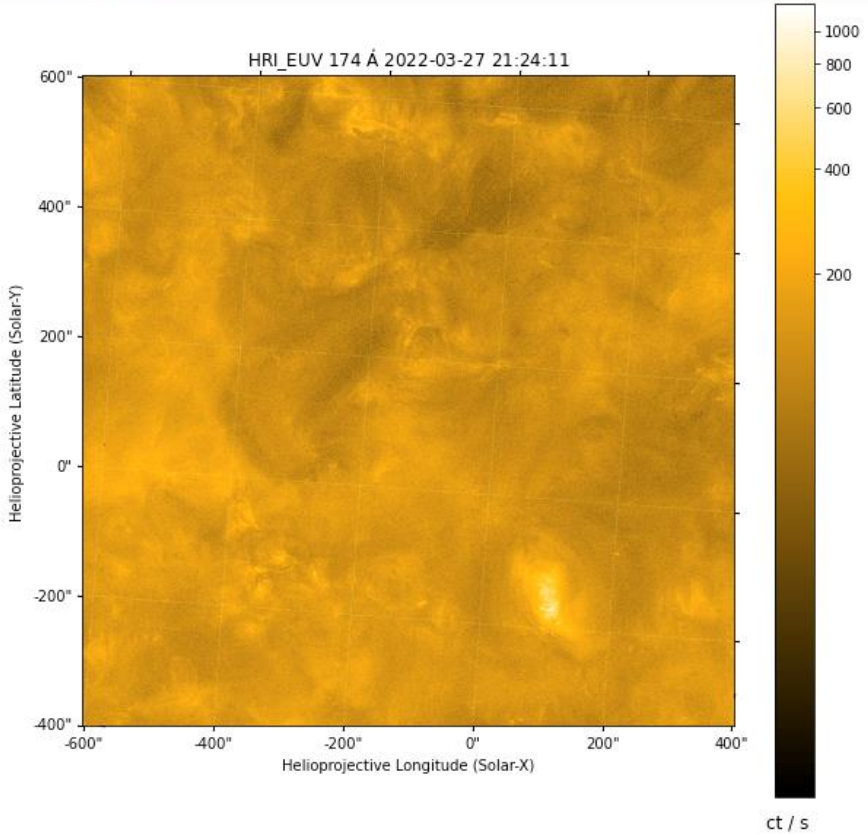
Launcher | Terminal 3 | Learning.ipynb | time_utils.py | 1_introduction.ipynb

Code | git | Python 3 (ipykernel)

Note that EUIDATAPATH is the same for the different descriptor/dataset.

[32]:
```
mymap = sunpy.map.Map(EUIDATAPATH+euv174['filename'][0])
mymap.peek()
```

```
/opt/miniconda/lib/python3.9/site-packages/matplotlib/animation.py:889: UserWarning: Animation was deleted without rendering anything. This is most likely not intended. To prevent deletion, assign the Animation to a variable, e.g. `anim`, that exists until you have outputted the Animation using `plt.show()` or `anim.save()`.
  warnings.warn(
```



Those lines do stuff for stretching and normalising: https://docs.astropy.org/en/stable/visualization/normalization.html

[33]:
```
from astropy.visualization import ImageNormalize, SqrtStretch, PowerStretch

mymap = sunpy.map.Map(EUIDATAPATH+euv174['filename'][0])
mymap.plot_settings['norm'] = ImageNormalize(vmin=0, vmax=2000, stretch=SqrtStretch())
mymap.peek()
```

2000

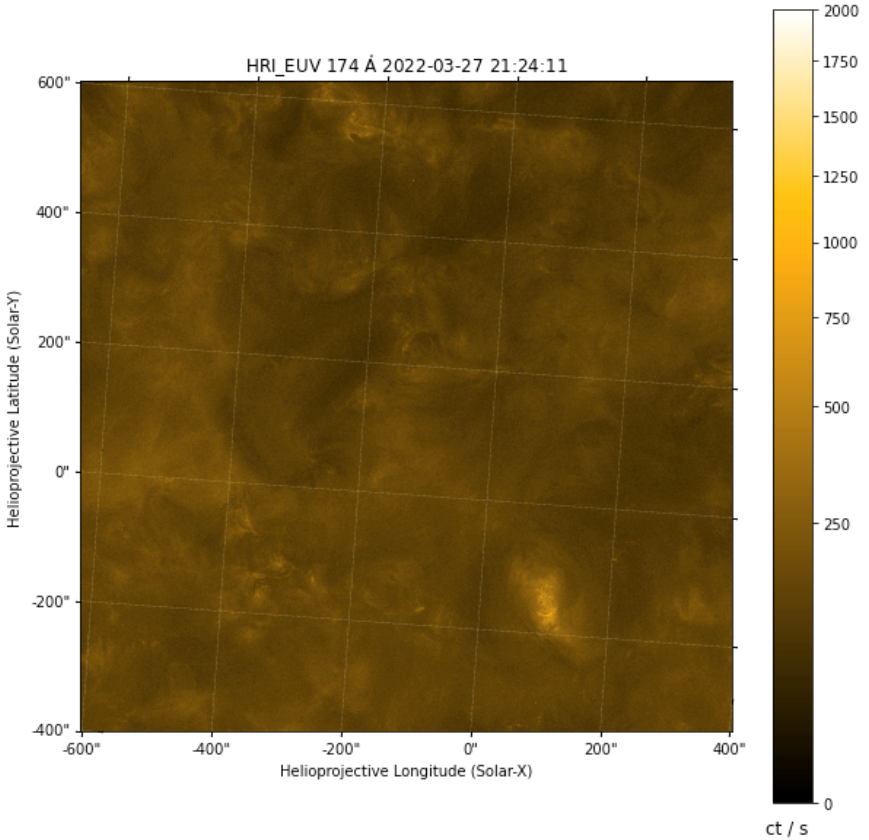
Launcher | Terminal 3 | Learning.ipynb | time_utils.py | 1_introduction.ipynb

Code | git | Python 3 (ipykernel)

Those lines do stuff for stretching and normalising: https://docs.astropy.org/en/stable/visualization/normalization.html

[33]:
```
from astropy.visualization import ImageNormalize, SqrtStretch, PowerStretch

mymap = sunpy.map.Map(EUIDATAPATH+euv174['filename'][0])
mymap.plot_settings['norm'] = ImageNormalize(vmin=0, vmax=2000, stretch=SqrtStretch())
mymap.peek()
```



MAG (Magnetometer)

RTN is the reference frame, and this is 1 minute resolution data:

[25]:
```
results = SOAR.launch_job("SELECT filename FROM v_sc_data_item "
                          "WHERE descriptor='MAG-RTN-NORMAL-1-MINUTE' "
                          "AND begin_time>='2022-02-08' "
```

Launcher | Terminal 3 | Learning.ipynb | time_utils.py | 1_introduction.ipynb

Code | git | Python 3 (ipykernel)

MAG (Magnetometer)

RTN is the reference frame, and this is 1 minute resolution data:

[25]:
```
results = SOAR.launch_job("SELECT filename FROM v_sc_data_item "
                          "WHERE descriptor='MAG-RTN-NORMAL-1-MINUTE' "
                          "AND begin_time>='2022-02-08' "
                          "AND end_time<='2022-03-05' "
                          "AND level='L2'"
                          )
mag_fn = results.get_results()

MAGDATAPATH = 'data/Solo/mag/L2/2022/'

mag_fn
```

[25]: *Table length=24*

filename
object
solo_L2_mag-rtn-normal-1-minute_20220304_V01.cdf
solo_L2_mag-rtn-normal-1-minute_20220303_V01.cdf
solo_L2_mag-rtn-normal-1-minute_20220302_V01.cdf
solo_L2_mag-rtn-normal-1-minute_20220301_V01.cdf
solo_L2_mag-rtn-normal-1-minute_20220228_V01.cdf
solo_L2_mag-rtn-normal-1-minute_20220227_V01.cdf
solo_L2_mag-rtn-normal-1-minute_20220226_V01.cdf
solo_L2_mag-rtn-normal-1-minute_20220225_V01.cdf
solo_L2_mag-rtn-normal-1-minute_20220224_V01.cdf
solo_L2_mag-rtn-normal-1-minute_20220222_V01.cdf
...
solo_L2_mag-rtn-normal-1-minute_20220217_V01.cdf
solo_L2_mag-rtn-normal-1-minute_20220216_V01.cdf
solo_L2_mag-rtn-normal-1-minute_20220215_V01.cdf
solo_L2_mag-rtn-normal-1-minute_20220214_V01.cdf
solo_L2_mag-rtn-normal-1-minute_20220213_V01.cdf
solo_L2_mag-rtn-normal-1-minute_20220212_V01.cdf
solo_L2_mag-rtn-normal-1-minute_20220211_V01.cdf
solo_L2_mag-rtn-normal-1-minute_20220210_V01.cdf
solo_L2_mag-rtn-normal-1-minute_20220209_V01.cdf

Launcher | Terminal 3 | Learning.ipynb | time_utils.py | 1_introduction.ipynb

Code | git | Python 3 (ipykernel)

[5]:
```
import cdflib # needed installing first

def cdf2df(path):

    with cdflib.cdfread.CDF(path) as file:

        # Extract epoch times
        epoch = file.varget(variable='EPOCH', expand=False, to_np=True)

        # Extract epoch times
        CDF_epoch_class = cdflib.epochs.CDFepoch()
        time = CDF_epoch_class.to_datetime(epoch, to_np=True)

        # Extract B vectors times
        B = file.varget(variable='B_RTN', expand=False)
        norm = np.linalg.norm(B, axis=1)

        # Get data attributes
        attributes = file.globalattsget(expand=True)

        df = pd.DataFrame({'BR': B.T[0],
                           'BT': B.T[1],
                           'BN': B.T[2],
                           '|B|': norm}, index = time)

        return df
```

[26]:
```
import pandas as pd
import numpy as np

data = pd.DataFrame()

for file in [MAGDATAPATH + x for x in mag_fn['filename']]:
    temp_df = cdf2df(file)
    data = pd.concat([data, temp_df])
    #data.sort_index(inplace=True) # What happens if I don't sort here?
```

[27]:
```
#just plot the data to check its ok
plt.rcParams["figure.figsize"] = (10, 10)

fig, axs = plt.subplots(4,1, sharex=True)

axs[0].plot(data['|B|'], color = 'black')
axs[0].set_ylabel('|B|')
axs[1].plot(data['BR'], color = 'red')
axs[1].set_ylabel('BR')
axs[2].plot(data['BT'], color = 'green')
axs[2].set_ylabel('BT')
axs[3].plot(data['BN'], color = 'orange')
axs[3].set_ylabel('BN')

fig.autofmt_xdate()
```

Launcher | Terminal 3 | Learning.ipynb | time_utils.py | 1_introduction.ipynb

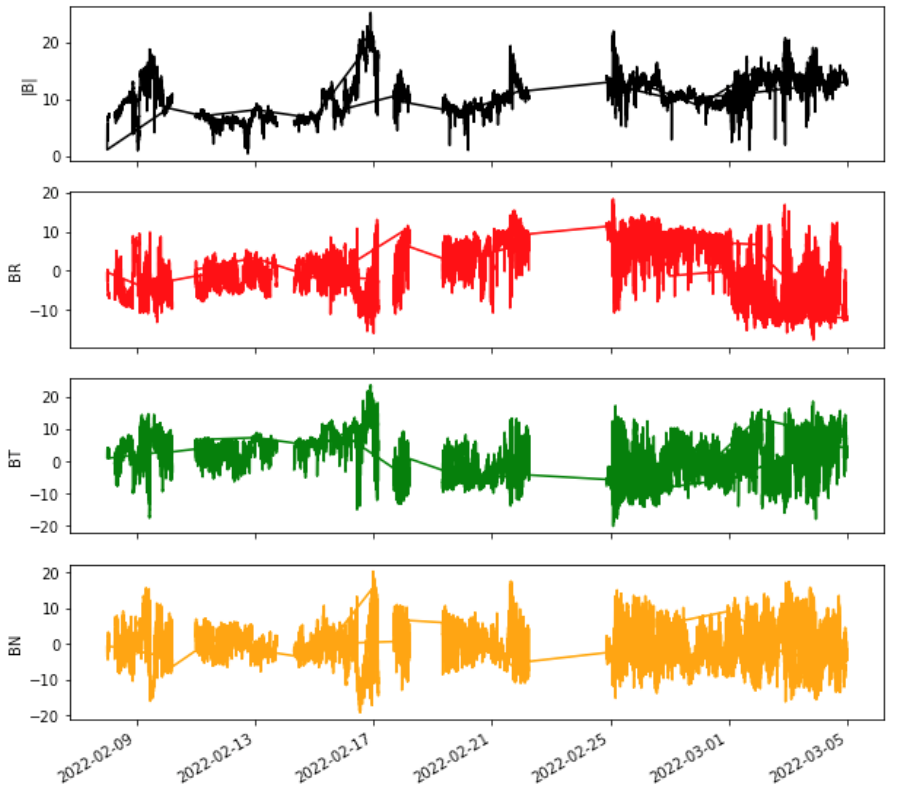
Code | git | Python 3 (ipykernel)

[27]:
```
#just plot the data to check its ok
plt.rcParams["figure.figsize"] = (10, 10)

fig, axs = plt.subplots(4,1, sharex=True)

axs[0].plot(data['|B|'], color = 'black')
axs[0].set_ylabel('|B|')
axs[1].plot(data['BR'], color = 'red')
axs[1].set_ylabel('BR')
axs[2].plot(data['BT'], color = 'green')
axs[2].set_ylabel('BT')
axs[3].plot(data['BN'], color = 'orange')
axs[3].set_ylabel('BN')

fig.autofmt_xdate()
```



[11]:
```
results = SOAR.launch_job("SELECT filename FROM v_sc_data_item "
                          "WHERE descriptor='MAG-RTN-NORMAL-1-MINUTE' "
                          "AND begin_time>='2022-02-08' "
                          "AND end_time<='2022-03-05' "
                          "AND level='L2'"
```

Launcher | Terminal 3 | Learning.ipynb | time_utils.py | 1_introduction.ipynb

Code | git | Python 3 (ipykernel)

[11]:
```
results = SOAR.launch_job("SELECT filename FROM v_sc_data_item "
                          "WHERE descriptor='MAG-RTN-NORMAL-1-MINUTE' "
                          "AND begin_time>='2022-02-08' "
                          "AND end_time<='2022-03-05' "
                          "AND level='L2'"
                          "ORDER BY filename"
                          )
mag_burst_fn = results.get_results()
mag_burst_fn
```

[11]: *Table length=7*

filename	filesize
object	int64
solo_L2_mag-rtn-burst_20220208_V01.cdf	145016554
solo_L2_mag-rtn-burst_20220209_V01.cdf	161503352
solo_L2_mag-rtn-burst_20220210_V01.cdf	75255039
solo_L2_mag-rtn-burst_20220211_V01.cdf	167052440
solo_L2_mag-rtn-burst_20220212_V01.cdf	166567753
solo_L2_mag-rtn-burst_20220213_V01.cdf	134615639
solo_L2_mag-rtn-burst_20220214_V01.cdf	129284630

What other MAG datasets are there? I know we can look in the v_sc_data_item table, but what are the column names? Am I looking for experiment or instrument or something else?

[28]:
```
table = SOAR.load_table('soar.v_sc_data_item')
for column in (table.columns):
    print(column.name)
```

```
Retrieving table 'soar.v_sc_data_item'
begin_time
data_item_id
data_item_oid
data_type
"descriptor"
end_time
file_format
filename
filesize
generation_time
icon
insertion_time
instrument
is_active
is_public
"level"
ll_flag
postcard_item_oid
prop_end
```

Launcher | Terminal 3 | Learning.ipynb | time_utils.py | 1_introduction.ipynb

Code | git | Python 3 (ipykernel)

[29]:
```
results = SOAR.launch_job("SELECT DISTINCT descriptor FROM v_sc_data_item "
                          "WHERE instrument='MAG' "
                          "AND level='L2'"
                          )
mag_ds = results.get_results()
mag_ds
```

[29]: *Table length=11*

descriptor
object
MAG-VSO-NORMAL
MAG-RTN-NORMAL
MAG-RTN-NORMAL-1-MINUTE
MAG-VSO-NORMAL-1-MINUTE
MAG-RTN-LL
MAG-RTN-BURST
MAG-SRF-NORMAL
MAG-SRF-LL
MAG-RTN-LL-1-MINUTE
MAG-VSO-BURST
MAG-SRF-BURST

[30]:
```
results = SOAR.launch_job("SELECT filename,filesize FROM v_sc_data_item "
                          "WHERE descriptor='MAG-RTN-BURST' "
                          "AND begin_time>='2022-02-08' "
                          "AND end_time<='2022-02-15' "
                          "AND level='L2'"
                          )
mag_burst_fn = results.get_results()
mag_burst_fn
```

[30]: *Table length=7*

filename	filesize
object	int64
solo_L2_mag-rtn-burst_20220214_V01.cdf	129284630
solo_L2_mag-rtn-burst_20220213_V01.cdf	134615639
solo_L2_mag-rtn-burst_20220212_V01.cdf	166567753
solo_L2_mag-rtn-burst_20220211_V01.cdf	167052440
solo_L2_mag-rtn-burst_20220210_V01.cdf	75255039
solo_L2_mag-rtn-burst_20220209_V01.cdf	161503352

Launcher | Terminal 3 | Learning.ipynb | time_utils.py | 1_introduction.ipynb

Markdown | git | Python 3 (ipykernel)

solo_L2_mag-rtn-burst_20220211_V01.cdf	167052440
solo_L2_mag-rtn-burst_20220210_V01.cdf	75255039
solo_L2_mag-rtn-burst_20220209_V01.cdf	161503352
solo_L2_mag-rtn-burst_20220208_V01.cdf	145016554

[9]:
```
print(sum(mag_burst_fn['filesize']))
```

```
979295407
```

[31]:
```
%%time
data_burst = pd.DataFrame()

for file in [MAGDATAPATH + x for x in mag_burst_fn['filename']]:
    print(file)
    temp_df = cdf2df(file)
    data_burst = pd.concat([data, temp_df])
    data_burst.sort_index(inplace=True)
```

```
data/Solo/mag/L2/2022/solo_L2_mag-rtn-burst_20220214_V01.cdf
data/Solo/mag/L2/2022/solo_L2_mag-rtn-burst_20220213_V01.cdf
data/Solo/mag/L2/2022/solo_L2_mag-rtn-burst_20220212_V01.cdf
data/Solo/mag/L2/2022/solo_L2_mag-rtn-burst_20220211_V01.cdf
data/Solo/mag/L2/2022/solo_L2_mag-rtn-burst_20220210_V01.cdf
data/Solo/mag/L2/2022/solo_L2_mag-rtn-burst_20220209_V01.cdf
data/Solo/mag/L2/2022/solo_L2_mag-rtn-burst_20220208_V01.cdf
CPU times: user 59.4 s, sys: 27.4 s, total: 1min 26s
Wall time: 1min 26s
```

Get the filenames already sorted, instead of sorting in the dataframe:

My code

Can drop and drag my own utility code into the box and import it!

[15]:
```
import time_utils as tu
```

Solar Orbiter Notebooks

https://github.com/SolarOrbiterWorkshop/solo8_tutorials

Python in Heliophysics (PyHC)

https://heliopython.org/

[ ]: