



Skdaccess: The **Scikit Data Access** Python Package

Quick Start Guide

v1.9.12 for Python 3.6

<https://pypi.python.org/pypi/scikit-dataaccess>

*Created and maintained by
Massachusetts Institute of Technology
Haystack Observatory, Astro-&Geo-Informatics Group*

*Project lead: Victor Pankratius
Contact email: skdaccess@mit.edu*

*Code Contributors: Cody M. Rude, Justin D. Li, Guillaume Rongier, David M. Blair, Michael G. Gowanlock,
Victor Pankratius*

1 Overview

The Scikit Data Access package simplifies the handling of scientific data sets in Python. It provides a common interface across all data sets, based on a data fetcher and iterator pattern, as illustrated in the Figure below.



This paradigm places the requirements for parsing and interpreting the data inside of the data fetcher, which returns a data wrapper that provides a uniform method for accessing the data. In particular, the data wrapper implements an iterator which returns the next segment of data when requested by another function or by the user.

Advantages of Scikit Data Access

- Import scientific data from various sources through one easy Python API.
- Use iterator patterns for each data source (configurable data generators + functions to get next data chunk).
- Skip parser programming and file format handling.
- Enjoy a common namespace for all data and unleash the power of data fusion.
- Handle data distribution in different modes: (1) local download, (2) caching of accessed data, or (3) online stream access.
- Easily pull data on cloud servers through Python scripts and facilitate large-scale parallel processing.

- Build on an extensible platform: Adding access to a new data source only requires addition of its “DataFetcher.py”.
- Open source (MIT License).

2 Supported Data Sets

The package introduces a common namespace and currently supports the following data sets:

Name-space	Data structure	Original Source	Data Size	Description
skdaccess. astro. kepler	Dictionary of Data Frames	Mikulski Archive for Space Telescopes (ftp://archive.stsci.edu/pub/kepler/lightcurves/)	≈ 1TB	Light curves for stars imaged by the <i>Kepler</i> Space Telescope (https://keplerscience.arc.nasa.gov/). This data set uses a cache data fetcher.
skdaccess. astro. voyager	Dictionary of Data Frames	Space Physics Data Facility (https://spdf.gsfc.nasa.gov/pub/data/voyager/)	≈ 0.1GB	Data from the Voyager mission (https://voyager.jpl.nasa.gov/mission/). This data set uses a cache data fetcher.
skdaccess. geo. era_interim	XArray Dataset	The University Corporation for Atmospheric Research (https://rda.ucar.edu/datasets/ds627.0/)	≈ ??TB	Atmospheric weather information from the ERA-Interim project at various pressure levels (https://www.ecmwf.int/en/forecasts/datasets/archive-datasets/reanalysis-datasets/era-interim). This data set uses a cache data fetcher.
skdaccess. geo. gldas	Dictionary of Data Frames	NASA Jet Propulsion Laboratory (ftp://podaac-ftp.jpl.nasa.gov/allData/tellus/L3/gldas_monthly/netcdf)	≈ 0.1GB	Land hydrology model produced by NASA. This version of the data is generated to match the GRACE temporal and spatial characteristics and is available as a complementary data product (https://grace.jpl.nasa.gov/data/get-data/land-water-content/). This data set uses a download data fetcher.
skdaccess. geo. grace	Dictionary of Data Frames	NASA Jet Propulsion Laboratory (ftp://podaac-ftp.jpl.nasa.gov/allData/tellus/L3/land_mass/RL05/netcdf)	≈ 0.1GB	GRACE Tellus Monthly Mass Grids. 30-day measurements of changes in Earth’s gravity field to quantify equivalent water thickness (https://grace.jpl.nasa.gov/data/get-data/monthly-mass-grids-land/). This data set uses a download data fetcher.
skdaccess. geo. grace. mascon	Dictionary of Data Frames	NASA Jet Propulsion Laboratory (ftp://podaac.jpl.nasa.gov/allData/tellus/L3/mascon/RL05/JPL/CRI/netcdf)	≈ 1GB	GRACE Tellus Monthly Mass Grids - Global Mascons. 30-day measurements of changes in Earth’s gravity field to quantify equivalent water thickness (https://grace.jpl.nasa.gov/data/get-data/jpl_global_mascons). This data set uses a download data fetcher.

skdaccess. geo. groundwater	Dictionary of Data Frames	USGS National Water Information System (https://waterservices.usgs.gov/rest/DV-Service.html)	$\approx 1\text{GB}$	United States groundwater monitoring wells measuring the depth to water level (https://waterservices.usgs.gov/). This data set uses a download data fetcher.
skdaccess. geo. magnetometer	Dictionary of Data Frames	USGS National Geomagnetism Program (https://geomag.usgs.gov/products/downloads.php)	$\approx 1\text{GB}$	Measurement of Earth's magnetic field from the USGS geomagnetism program (https://geomag.usgs.gov/). This data set uses a stream data fetcher.
skdaccess. geo. mahali. rinex	List of rinex file paths	MIT-Haystack Observatory (http://apollo.haystack.mit.edu/mahali-data/)	$\approx 10\text{GB}$	Rinex files from the MIT led NSF project studying the Earth's ionosphere with GPS (http://mahali.mit.edu). This data set uses a cache data fetcher.
skdaccess. geo. mahali. tec	Dictionary of Data Frames	MIT-Haystack Observatory (http://apollo.haystack.mit.edu/mahali-data/)	$\approx 1\text{GB}$	TEC measurements from the MIT led NSF project studying the Earth's ionosphere with GPS (http://mahali.mit.edu). This data set uses a cache data fetcher.
skdaccess. geo. mahali. temperature	Dictionary of Data Frames	MIT-Haystack Observatory (http://apollo.haystack.mit.edu/mahali-data/)	$\approx 0.1\text{GB}$	Temperature measurements from the MIT led NSF project studying the Earth's ionosphere with GPS (http://mahali.mit.edu). This data set uses a stream data fetcher.
skdaccess. geo. modis	Dictionary of Numpy arrays	NASA MODIS (https://ladsweb.modaps.eosdis.nasa.gov/tools-and-services/)	$\approx 100\text{MB}$ /image	Spectroradiometer aboard the NASA Terra and Aqua satellites that generates approximately daily images of the Earth's surface (https://modis.gsfc.nasa.gov/). This data set uses a cache and stream data fetcher.
skdaccess. geo. pbo	Dictionary of Data Frames	UNAVCO Plate Boundary Observatory (ftp://data-out.unavco.org/pub/products/position/pbo.nam08.pos.tar.gz and https://www.unavco.org/data/gps-gnss/derived-products/derived-products.html)	$\approx 1\text{GB}$	Daily GPS displacement time series measurements throughout the United States (http://www.unavco.org/projects/major-projects/pbo/pbo.html). This data set uses a download data fetcher.
skdaccess. geo. sentinel1	Dictionary of Numpy arrays	Alaska Satellite Facility (https://www.asf.alaska.edu/sentinel/)	$\approx 1 - 10\text{GB}$ / image	Synthetic Aperture Radar data from the Sentinel 1 satellites operated by the European Space Agency (https://www.esa.int/Our_Activities/Observing_the_Earth/Copernicus/Sentinel-1). This data set uses a cache data fetcher.
skdaccess. geo. srtm	Dictionary of Numpy arrays	United States Geological Survey (https://e4ftl01.cr.usgs.gov/MEASURES/)	$\approx 100\text{GB}$	Digital elevation data from the Shuttle Radar Topography Mission (https://www2.jpl.nasa.gov/srtm/). This data set uses a cache data fetcher.

skdaccess. geo. uavsar	Dictionary of Numpy arrays	NASA Jet Propulsion Laboratory (https://uavsar.jpl.nasa.gov/cgi-bin/data.pl)	Data product depend- ent	Synthetic Aperture Radar Single Look Complex data from the Uninhab- ited Aerial Vehicle Synthetic Aperture Radar (https://uavsar.jpl.nasa.gov/). This data set uses a cache data fetcher.
skdaccess. geo. wyoming_sounding	Dictionary of Data Files	University of Wyoming (http://weather.uwyo.edu/upperair/sounding.html)	≈ 10GB	Sounding data from The University of Wyoming (http://weather.uwyo.edu/upperair/sounding.html). This data set has a cache and stream data fetcher.
skdaccess. planetary. ode	Dictionary of Numpy arrays	Orbital Data Explorer at the University of Washington in St. Louis (http://oderest.rsl.wustl.edu)	Data product depend- ent	Planetary data from PDS Geo- sciences Node's Orbital Data Explorer (http://pds-geosciences.wustl.edu/default.htm). This data set uses a cache data fetcher
skdaccess. solar. sdo	Dictionary of Numpy arrays	Solar Dynamics Ob- servatory (https://sdo.gsfc.nasa.gov/assets/img/browse/) and the Joint Science Op- erations Center (http://jsoc2.stanford.edu/data/aia/synoptic/)	Data product depend- ent	Images from the Solar Dy- namics Observatory (https://sdo.gsfc.nasa.gov). This data set uses a stream data fetcher.

3 Installation and Modes of Operation

The package can easily installed by using the standard Python “pip install” command:

```
> pip install scikit-dataaccess
```

After successful installation, a script called “skdaccess” allows users to specify the data sets that should be downloaded from their original sources to the local machine. The PBO, GRACE and groundwater data sets must be downloaded using this script before they can be used. For example, to download the PBO data use:

```
> skdaccess pbo
```

The script also completes all necessary configurations to make the data access seamlessly available in the Python environment.

3.1 Modes of Operation

There are three modes of operation available for accessing the data through the skdaccess package. The two local options are “Download” and “Cache”. Using the “Download” option, the dataset is downloaded to local disk before use. The “Cache” option allows for data of interest to be downloaded during use and stored in case of future use. The online option is “Stream”, which accesses the data without storing a local copy.

3.2 The Skdaccess Script

This script downloads scientific data sets from preconfigured Web sources, makes them available offline on the users machine, and configures the Python environment for data access.

For the following data sets, the skdaccess script must be used to download and prepare the data.

- GPS data from the Plate Boundary Observatory
- Depth to groundwater for wells in California
- Equivalent water thickness from GRACE Tellus Monthly Land Grids
- Equivalent water thickness from GLDAS

The skdaccess script does not download Kepler data, as the data is downloaded for each star individually the first time the star is accessed by the data fetcher.

To download a dataset, use the command with the dataset name as the argument. For example, to download groundwater data available from California type

```
> skdaccess groundwater
```

The data will be downloaded into the current directory, and the .skdaccess config file located in the users home directory will be updated. Each data set can be downloaded into different directories depending on the user preferences.

To list all supported data sets, call

```
> skdaccess -l
```

This utility can install one of the following data sets:

```
PBO - Plate Boundary Observatory GPS Time Series
GRACE - Monthly Mass Grids
GLDAS - Monthly estimates from GDLAS model in same resolution as GRACE
Groundwater - Ground water daily values from across the US
```

Calling the script without any arguments provides a list of available commands as shown below.

```
> skdaccess
```

```
usage: skdaccess [-h] [-l] [-i LOCAL_DATA] [-c] [data_set]
```

The Sci-kit Data Access (skdaccess) package is a tool for integrating various scientific data sets into the Python environment using a common interface. This script can download different scientific data sets for offline analysis.

positional arguments:

data_set	Name of data set
----------	------------------

optional arguments:

-h, --help	show this help message and exit
-l, --list	List data sets
-i LOCAL_DATA, --input LOCAL_DATA	Use LOCAL_DATA that has already been downloaded
-c, --check	Print data location for data set

4 Scientific Data Access in Python

Data is retrieved in a Python program via a DataFetcher object. Each data set has its own data fetcher. There are two ways of handling the data: (1) directly accessing the data structure created by the DataFetcher, or (2) through an iterator interface provided by a data wrapper.

Data Access Example:

```
# First import the data generator for water
# Note: This assumes the groundwater data has been downloaded
from skdaccess.geo.groundwater import DataFetcher as waterDF

# Create a data fetcher and get the data wrapper:
fullDF = waterDF(start_date='2007-01-01', end_date='2011-01-01')
wdata = fullDF.output().get()
```

5 Usage Examples

The following examples show how to use the data fetcher for the data sets described earlier and displaying / plotting the data. These notebooks can be accessed at <https://github.com/MITHaystack/scikit-dataaccess/tree/master/skdaccess/examples>.

5.1 skdaccess.astro.kepler

Computer-Aided
Discovery

Demo_Kepler

Last Checkpoint: a minute ago (autosaved)

FileEditViewInsertCellKernelWidgetsHelpskdaccessskdiscoveryExamples

TrustedPython 3 O

In [1]:

%matplotlib notebook
import matplotlib.pyplot as plt

In [2]:

Kepler Exoplanet Light Curves Time Series
Source: http://keplerscience.arc.nasa.gov
Light curve in relative flux versus phase

In [3]:

from skdaccess.astro.kepler import DataFetcher as Kepler_DF
from skdaccess.utilities.kepler_util import normalize
from skdaccess.framework.param_class import *
import numpy as np

In [4]:

kepler_fetcher = Kepler_DF([AutoList(['009941662'])])

In [5]:

kepler_data = kepler_fetcher.output().get()

Downloading data for 1 star(s)

In [6]:

normalize(kepler_data['009941662'])

In [7]:

kepler_data['009941662'].head()

Out[7]:

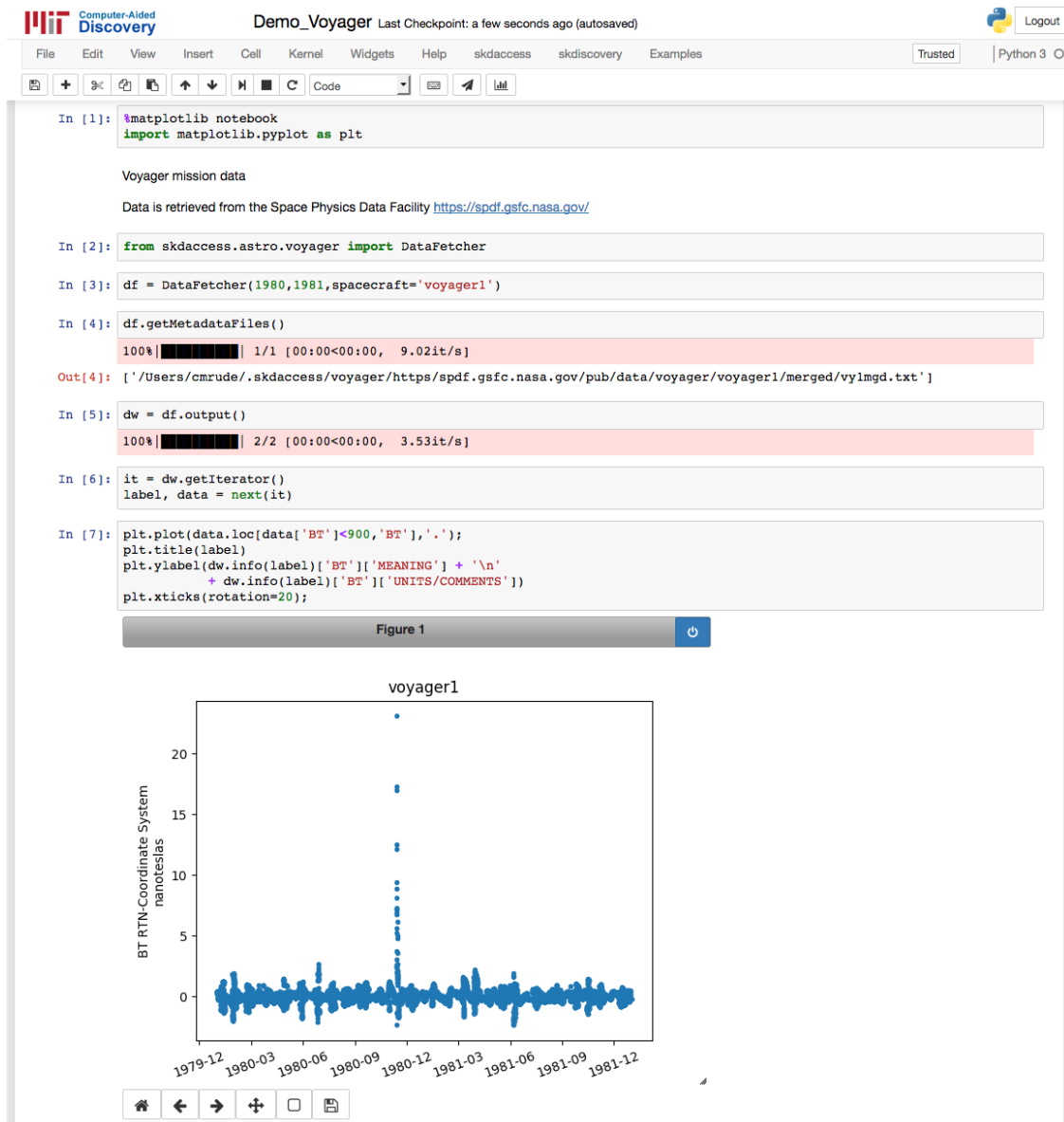
	TIME	TIMECORR	SAP_FLUX	SAP_FLUX_ERR	SAP_BKG	SAP_BKG_ERR	PDCSAP_FLUX	PDCSAP_FLUX_ERR	SAP_QUALITY	PSF_CEN
CADENCENO										
568	120.539195	0.001042	1778533.750	33.049557	4841.642090	1.547178	0.995571	32.609024	0	
569	120.559629	0.001043	1778263.875	33.047188	4846.805664	1.546246	0.995457	32.732418	0	
570	120.580063	0.001044	1778347.750	33.048054	4848.539062	1.549641	0.995509	32.837833	0	
571	120.600498	0.001044	1778901.000	33.052914	4847.870117	1.543734	0.995785	32.684124	0	
572	120.620932	0.001045	1781658.250	33.081059	4852.192871	1.546612	0.997348	32.769455	0	

In [8]:

plt.figure(figsize=(8,4));
data = kepler_data['009941662'].iloc[0:1000]
plt.plot(np.array(data['TIME']) % 1.7636, data['PDCSAP_FLUX'], '.');
plt.tight_layout();

Figure 1

5.2 skdaccess.astro.voyager



5.3 skdaccess.geo.era_interim

Computer-Aided
Discovery

Demo_ERA_Interim Last Checkpoint: 02/07/2018 (unsaved changes)

Logout

File Edit View Insert Cell Kernel Help Not Trusted Python 3

Run

Enter/Exit Live Reveal Slideshow

Data Citation

European Centre for Medium-Range Weather Forecasts (2009): ERA-Interim Project. Research Data Archive at the National Center for Atmospheric Research, Computational and Information Systems Laboratory. <https://doi.org/10.5065/D6CR5RD9>.

```
In [1]: %matplotlib inline
import matplotlib.pyplot as plt
plt.rcParams['figure.dpi'] = 150
from getpass import getpass
import pandas as pd

In [2]: from skdaccess.framework.param_class import *
from skdaccess.geo.era_interim.cache import DataFetcher as EDF

Specify list of dates

In [3]: date_list = pd.date_range('2015-06-06 00:00:00', '2015-06-06 06:00:00', freq='6H')

Enter Research Data Archive (NCAR) credentials

In [4]: username='Enter username'
password = getpass()
.....

Create data fetcher

In [5]: edf = EDF(date_list=date_list, data_names=['Geopotential', 'Temperature'],
username=username, password=password)

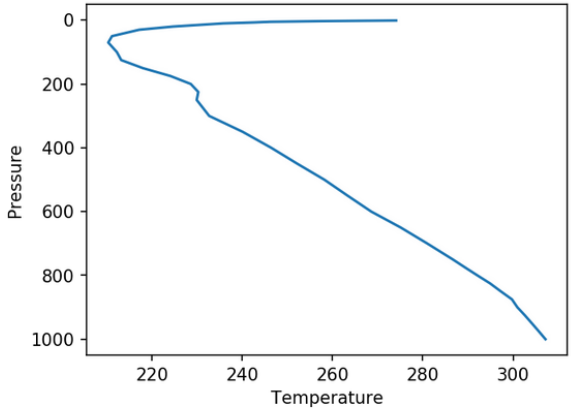
Access data

In [6]: edw = edf.output()

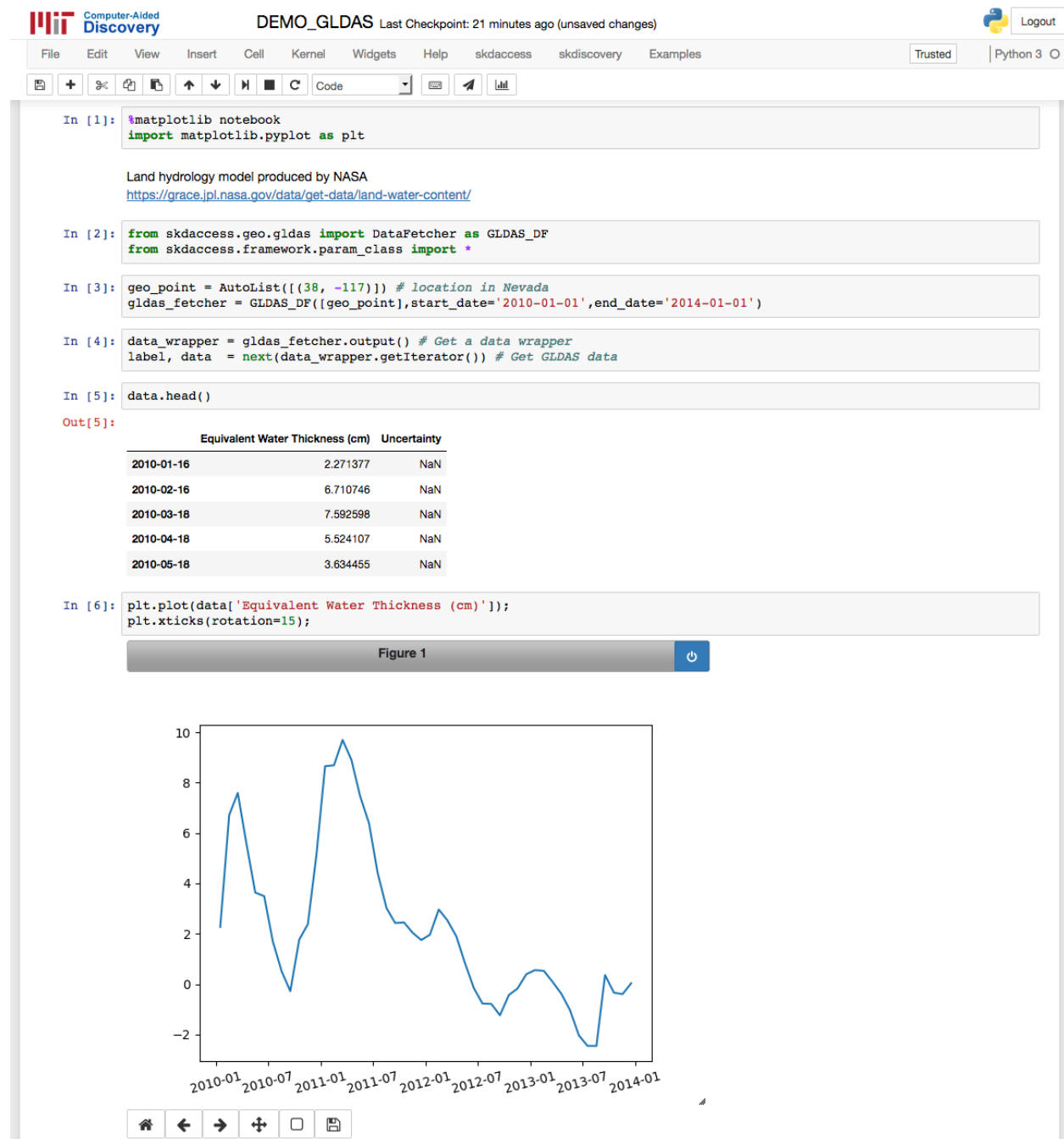
In [7]: iterator = edw.getiterator()
geo_label, geo_data = next(iterator)
temp_label, temp_data = next(iterator)

Plot temperature data

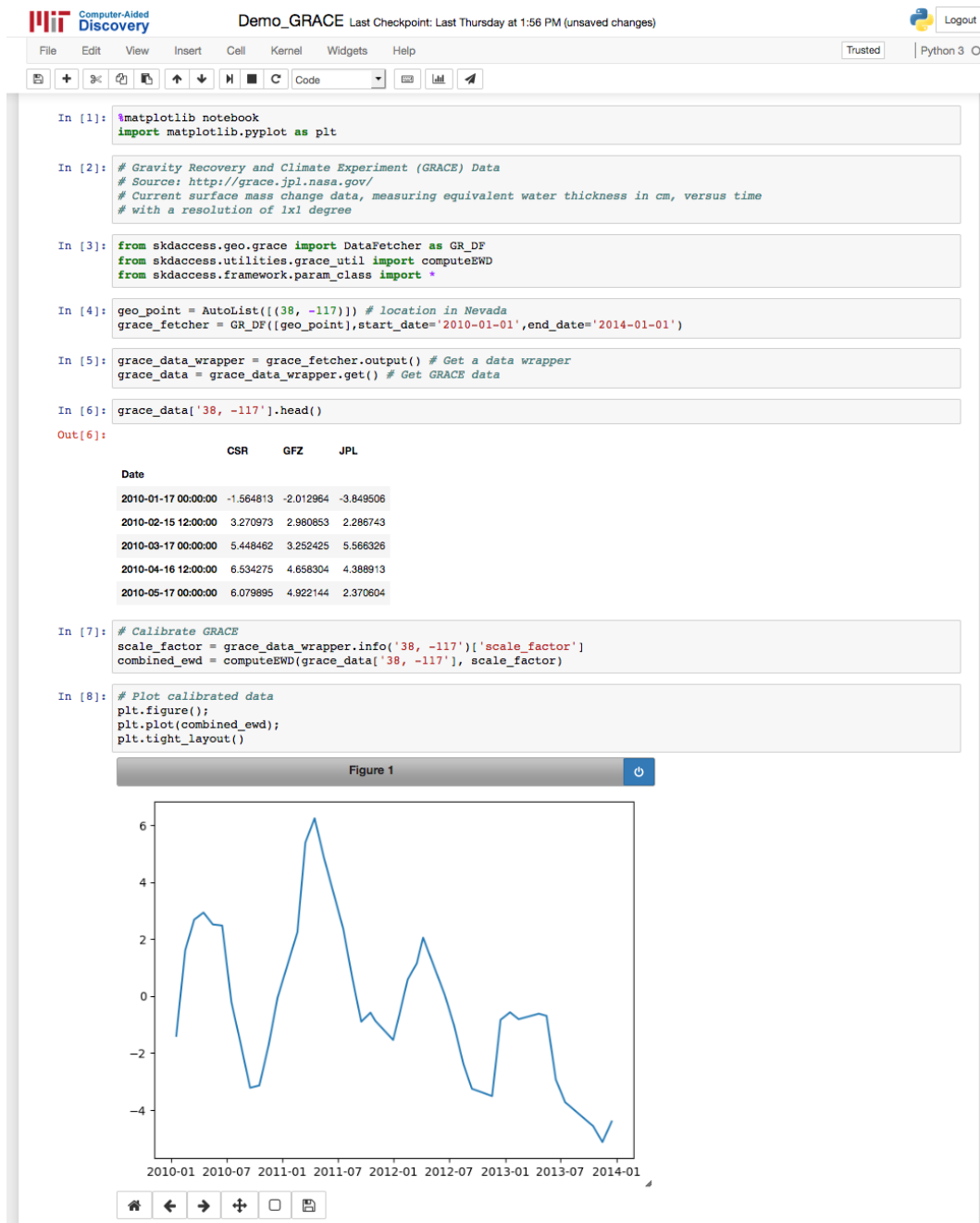
In [9]: plt.figure(figsize=(5,3.75));
plt.plot(temp_data[0,:,75,350], temp_data['pressure']);
plt.gca().invert_yaxis();
plt.ylabel('Pressure');
plt.xlabel('Temperature');
```



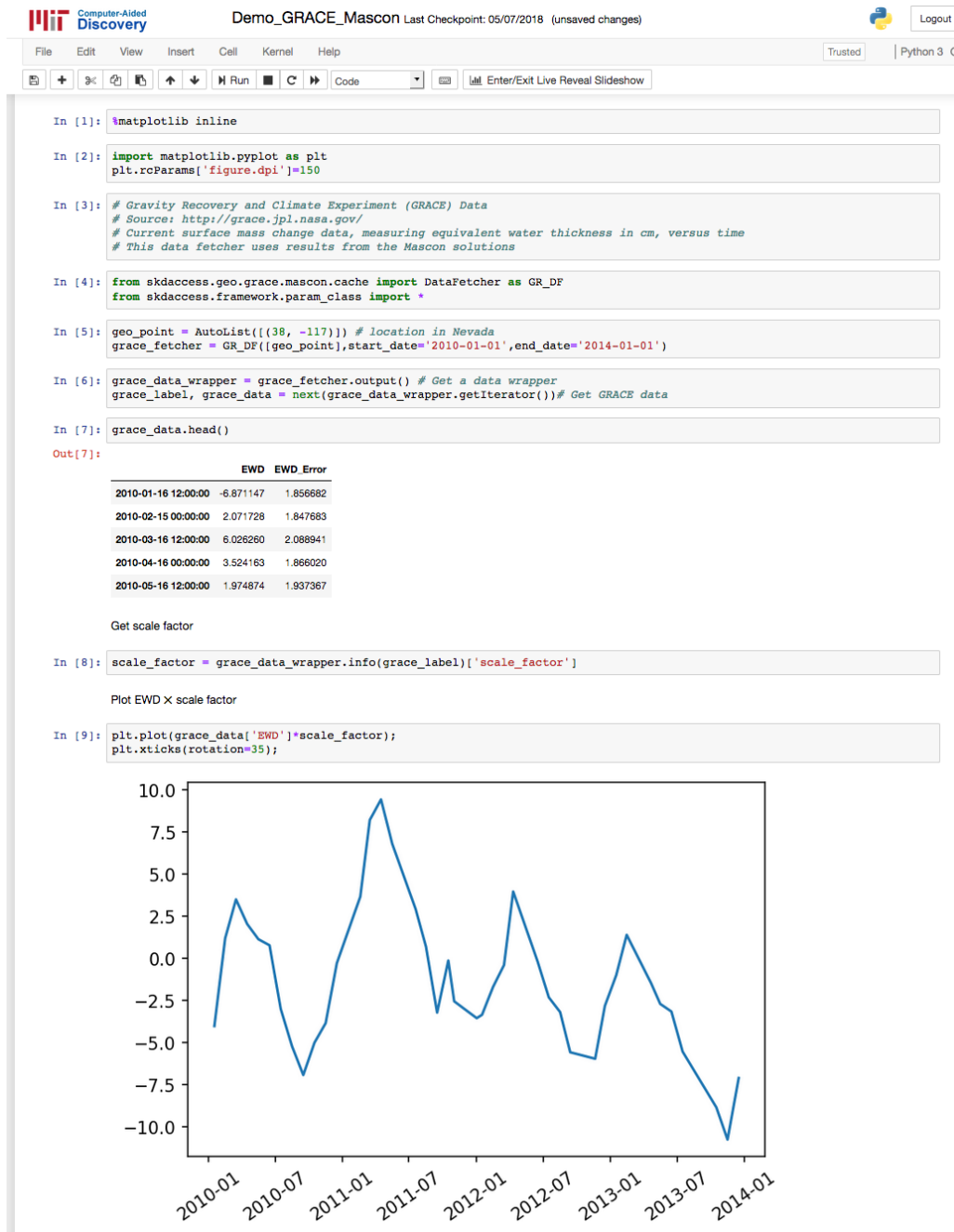
5.4 skdaccess.geo.gldas



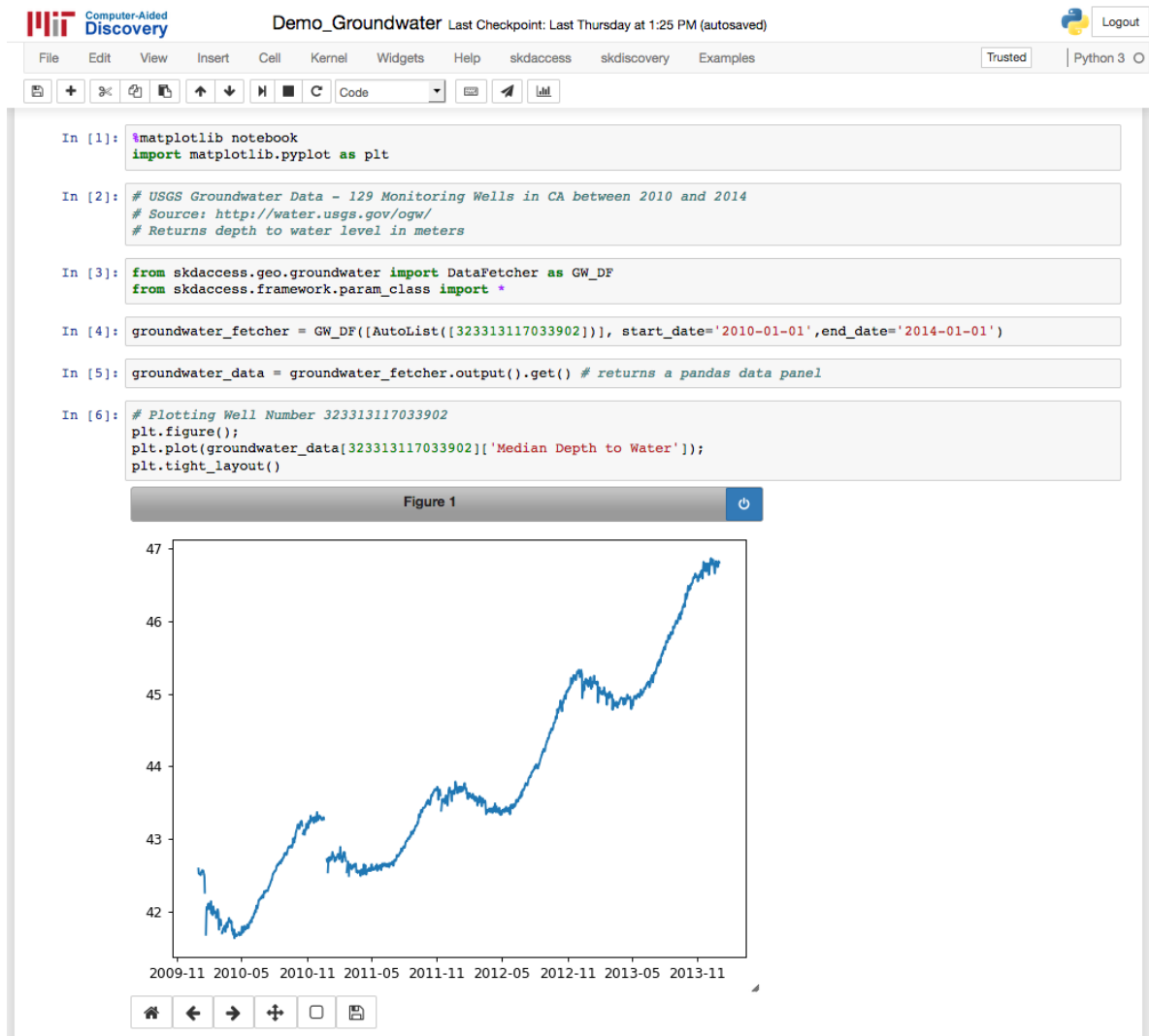
5.5 skdaccess.geo.grace



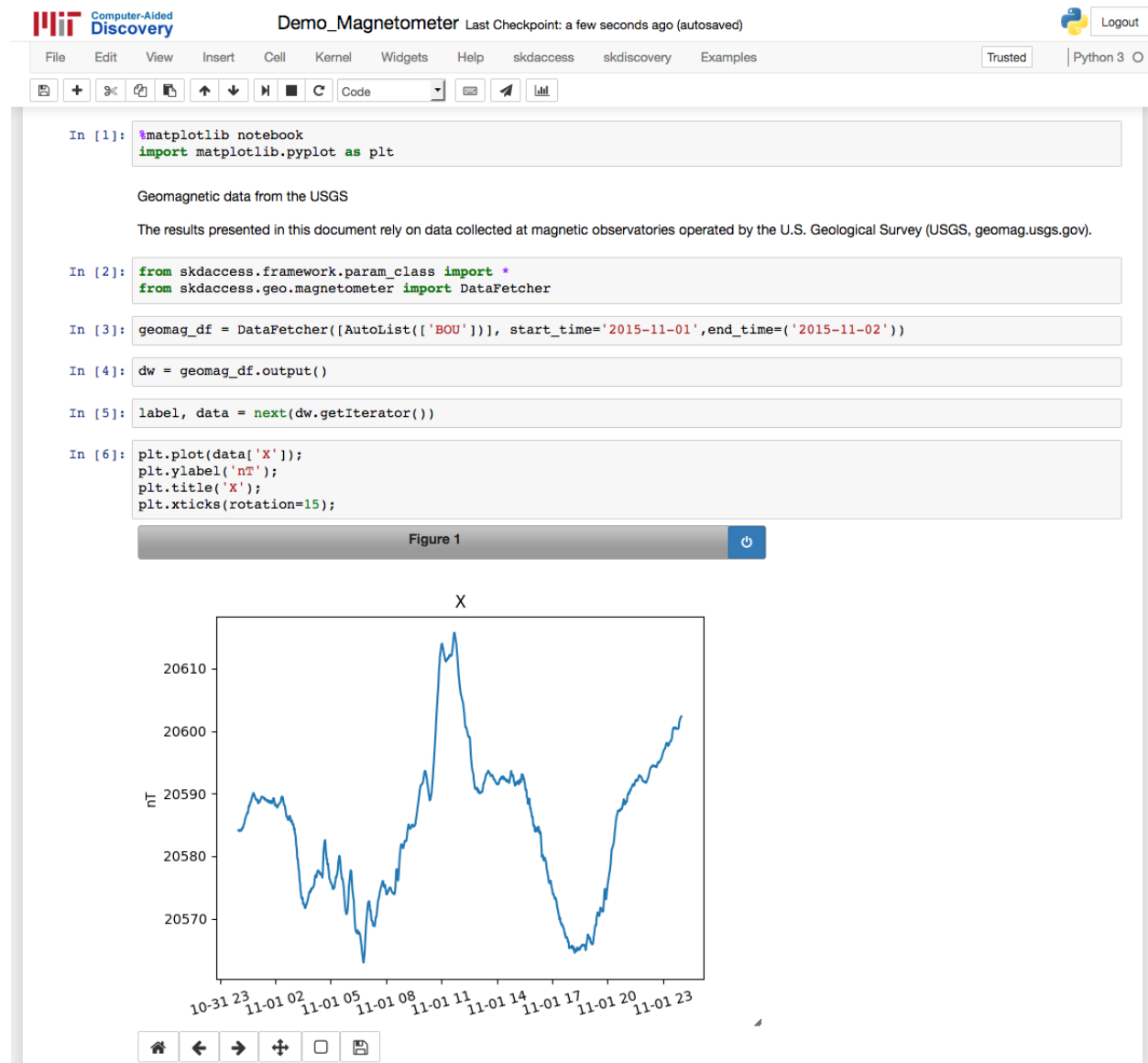
5.6 skdaccess.geo.grace.mascon



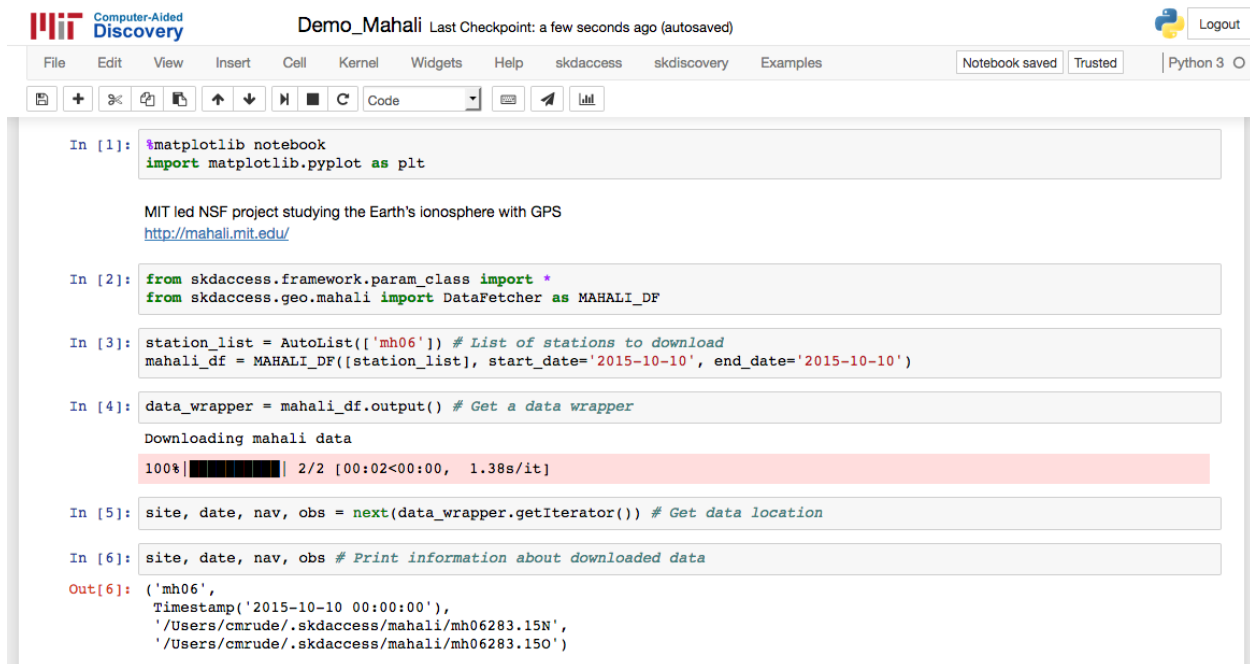
5.7 skdaccess.geo.groundwater



5.8 skdaccess.geo.magnetometer



5.9 skdaccess.geo.mahali.rinex



```
In [1]: %matplotlib notebook
import matplotlib.pyplot as plt

MIT led NSF project studying the Earth's ionosphere with GPS
http://mahali.mit.edu/

In [2]: from skdaccess.framework.param_class import *
from skdaccess.geo.mahali import DataFetcher as MAHALI_DF

In [3]: station_list = AutoList(['mh06']) # List of stations to download
mahali_df = MAHALI_DF([station_list], start_date='2015-10-10', end_date='2015-10-10')

In [4]: data_wrapper = mahali_df.output() # Get a data wrapper

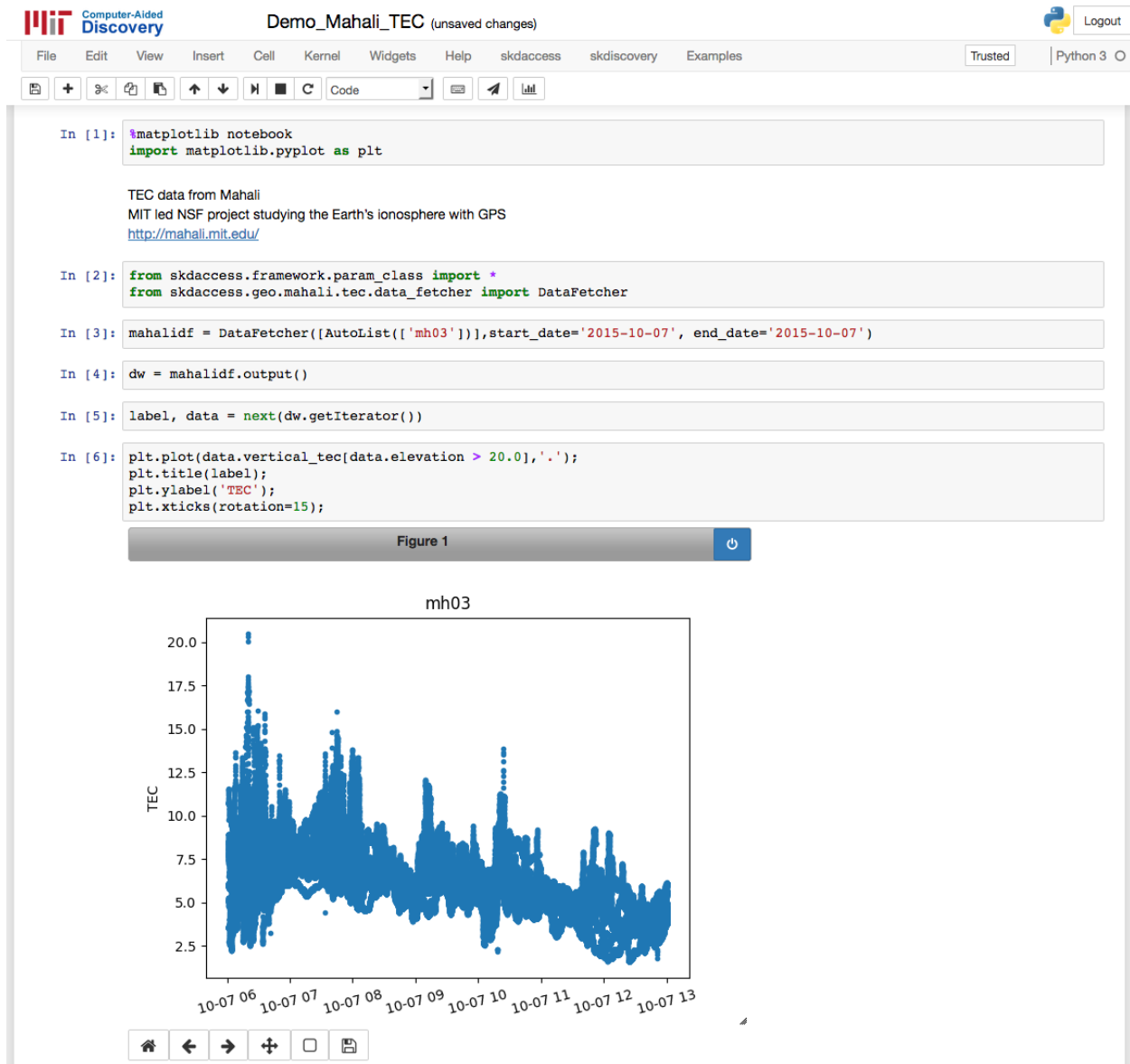
Downloading mahali data
100%|██████████| 2/2 [00:02<00:00, 1.38s/it]

In [5]: site, date, nav, obs = next(data_wrapper.getIterator()) # Get data location

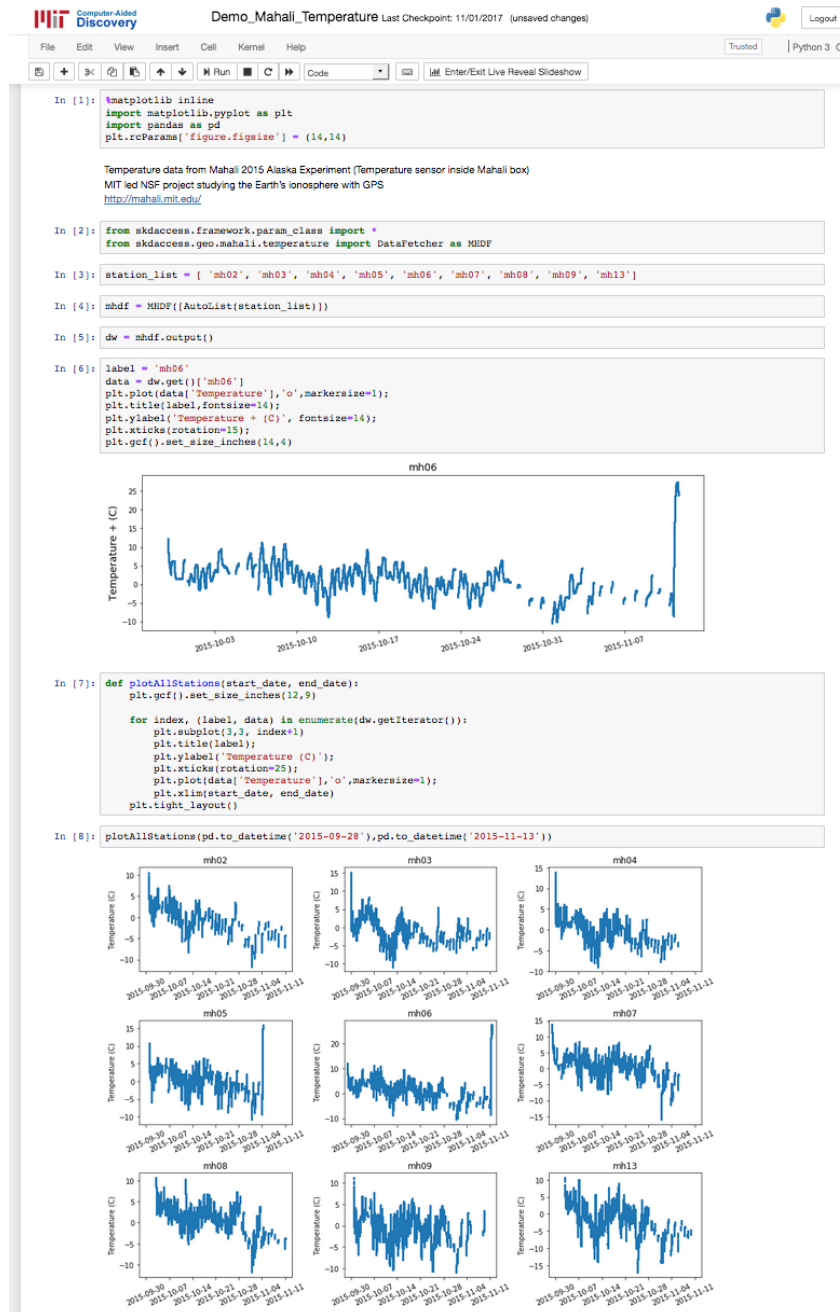
In [6]: site, date, nav, obs # Print information about downloaded data

Out[6]: ('mh06',
Timestamp('2015-10-10 00:00:00'),
'/Users/cmrude/.skdaccess/mahali/mh06283.15N',
'/Users/cmrude/.skdaccess/mahali/mh06283.15O')
```

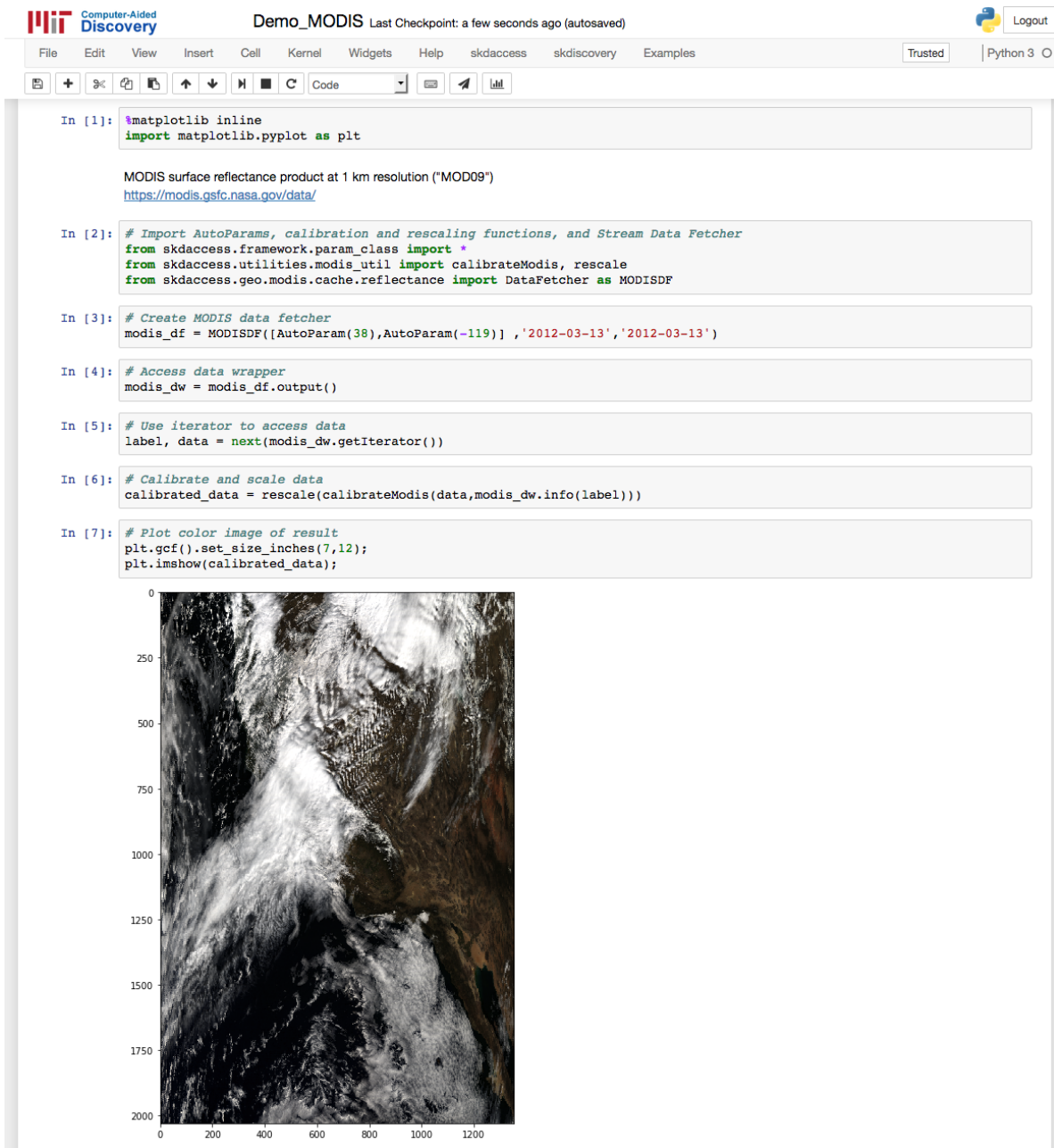
5.10 skdaccess.geo.mahali.tec



5.11 skdaccess.geo.mahali.temperature



5.12 skdaccess.geo.modis.cache.reflectance



5.13 skdaccess.geo.pbo

Computer-Aided Discovery

Demo_PBO Last Checkpoint: Sat Thursday at 1:25 PM (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help skdaccess skdiscovery Examples Trusted Python 3 O

```
In [1]: # Plate Boundary Observatory GPS Data
# Source: http://www.unavco.org/instrumentation/networks/status/pbo
# Time series data for GPS sensors (North, East, Up), displacement in meters versus time

In [2]: from skdaccess.geo.pbo import DataFetcher as PBO_DF
from skdaccess.framework.param_class import *

In [3]: %matplotlib notebook
import matplotlib.pyplot as plt

In [4]: # Latitude and Longitude range around Akutan Volcano
lat_range = AutoList((54,54.25))
lon_range = AutoList((-166, -165.6))
start_time = '2006-01-01'
end_time = '2015-06-01'

PBO_data_fetcher = PBO_DF(start_time, end_time, [lat_range, lon_range],mdyratio=.7)

In [5]: PBO_data = PBO_data_fetcher.output().get() # returns an ordered dictionary of data frames
100%|██████████| 6/6 [00:00<00:00, 17.98it/s]

In [6]: PBO_data['AV06'].head()
```

	HMMSSS	JJJJ.JJJJ	X	Y	Z	Sx	Sy	Sz	Rxy	Rxz	...	dN	dE	dU	Sn
2006-01-01	120000	53736.5	-3.629267e+06	-920656.48751	5.146731e+06	0.00339	0.00167	0.00460	0.508	-0.801	...	0.00945	0.00935	-0.01095	0.00172
2006-01-02	120000	53737.5	-3.629267e+06	-920656.48670	5.146731e+06	0.00323	0.00160	0.00441	0.506	-0.801	...	0.01064	0.00896	-0.01233	0.00165
2006-01-03	120000	53738.5	-3.629267e+06	-920656.48672	5.146731e+06	0.00332	0.00166	0.00450	0.495	-0.806	...	0.01108	0.00937	-0.01432	0.00166
2006-01-04	120000	53739.5	-3.629267e+06	-920656.48650	5.146731e+06	0.00338	0.00169	0.00457	0.492	-0.802	...	0.00803	0.00947	-0.02076	0.00170
2006-01-05	120000	53740.5	-3.629267e+06	-920656.48658	5.146731e+06	0.00331	0.00165	0.00446	0.490	-0.802	...	0.01132	0.00890	-0.01179	0.00167

5 rows x 24 columns

```
In [7]: plt.figure();
plt.plot(PBO_data['AV06']['dN']);
plt.tight_layout()
```

Figure 1

5.14 skdaccess.geo.sentinel_1

MIT

Computer-Aided
Discovery

Demo_Sentinel_1 Last Checkpoint: 05/04/2018 (unsaved changes)

Python

Logout

File Edit View Insert Cell Kernel Help Trusted Python 3

In [1]:

%matplotlib inline

In [2]:

```
import matplotlib.pyplot as plt
plt.rcParams['figure.dpi'] = 150
import numpy as np
from getpass import getpass
```

In [3]:

```
from skdaccess.geo.sentinel_1.cache import DataFetcher as S1DF
```

Supply Earth Data credentials

In [4]:

username='Enter username'

In [5]:

password = getpass()

Define urls for Sentinel 1 data and precise orbits

In [6]:

```
slc_url_list = ['https://datapool.asf.alaska.edu/SLC/SA/S1A_IW_SLC__1SSV_20141103T195043_20141103T195057_003122_00395A_']
satellite_url_list = ['https://slqc.asf.alaska.edu/aux_poeorb/S1A_OPER_AUX_POEORB_OPOD_20141124T123237_v20141102T225944_']
```

Create data fetcher

In [7]:

sldf = S1DF(slc_url_list, satellite_url_list, username, password, swath=3, polarization='VV')

Access data

In [8]:

sldw = sldf.output()

Retrieving SLC data
Retrieving orbit files
All files retrieved

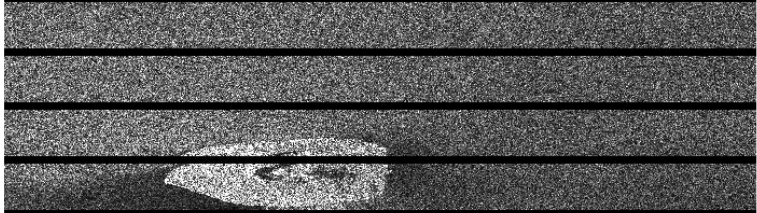
In [9]:

label, data = next(sldw.getIterator())

In [10]:


```
plt.title(label, fontsize=8)
plt.imshow(np.abs(data[:,10,:10]), vmin=0, vmax=100, cmap='gray', origin='lower')
plt.axis('off');
```

S1A_IW_SLC__1SSV_20141103T195043_20141103T195057_003122_00395A_F396.zip




20

5.15 skdaccess.geo.srtm











 Computer-Aided
Discovery

Demo_SRTM Last Checkpoint: 02/07/2018 (unsaved changes)

 Logout

File Edit View Insert Cell Kernel Help

Not Trusted Python 3

         Code 

```
In [1]: %matplotlib inline
import matplotlib.pyplot as plt
plt.rcParams['figure.dpi'] = 150
import numpy as np
from getpass import getpass

In [2]: from skdaccess.geo.srtm.cache import DataFetcher as SDF

Supply Earth Data credentials

In [3]: username='Enter username'

In [4]: password = getpass()
.....

Create data fetcher for elevation data from Shuttle Radar Topography

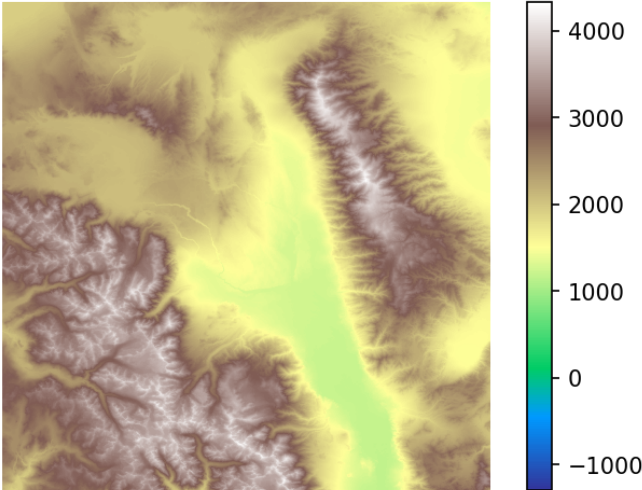
In [5]: sdf = SDF(lat_tile_start=37,lat_tile_end=37,lon_tile_start=-119,lon_tile_end=-119,
username=username,password=password)

In [6]: sdw = sdf.output()

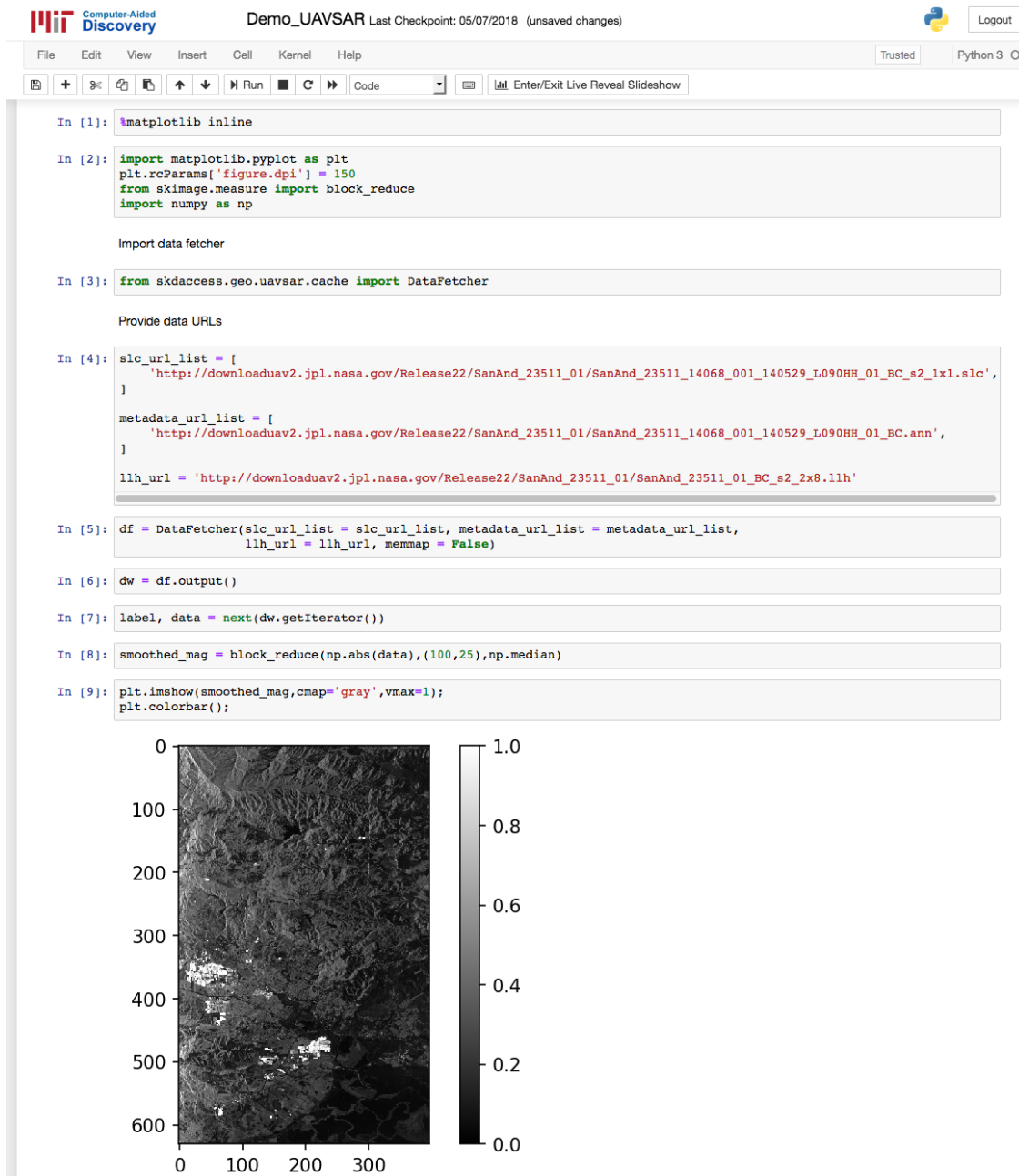
Access data

In [7]: label, data = next(sdw.getIterator())

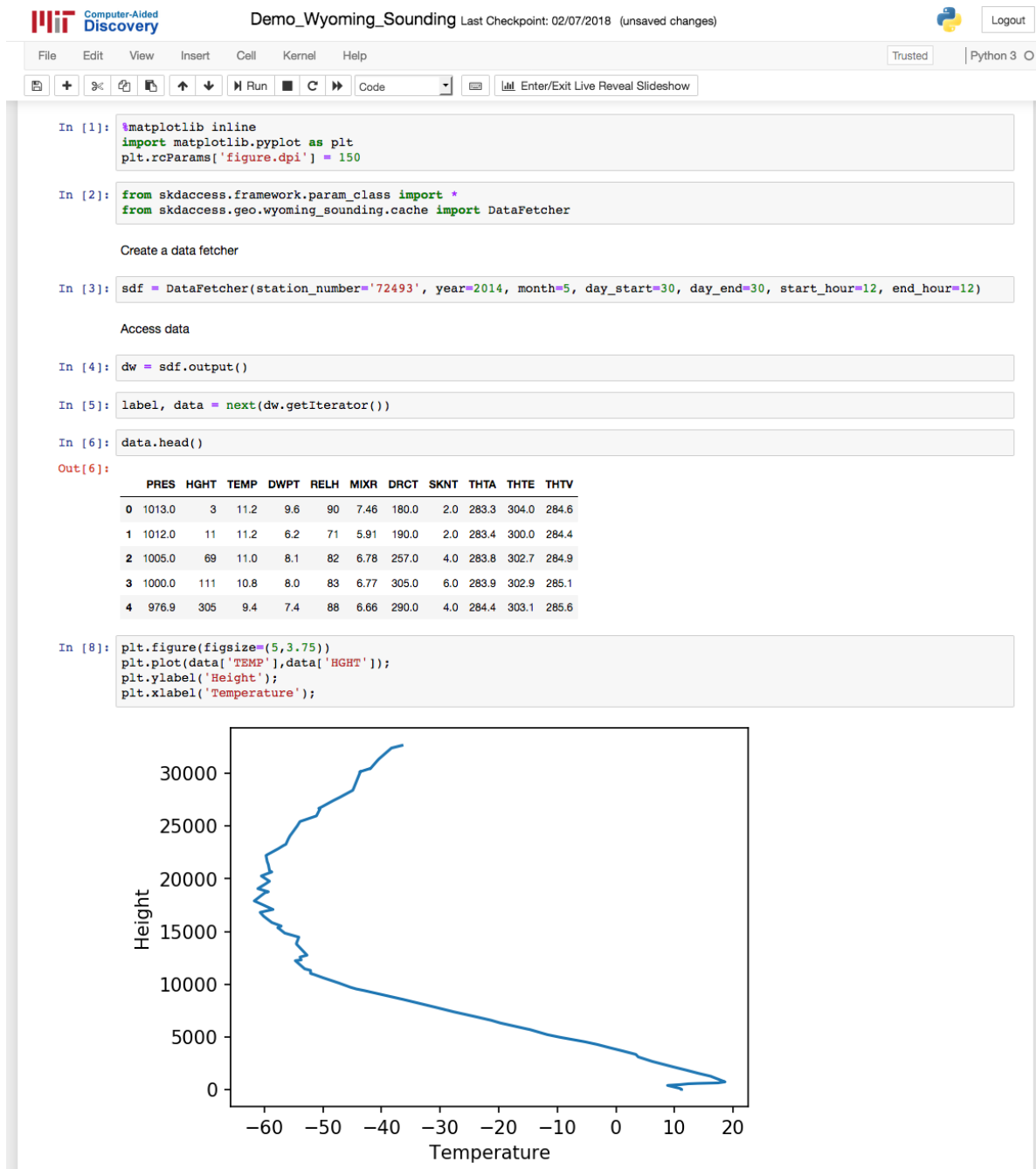
In [8]: plt.imshow(data,cmap='terrain',vmin=-1300);
plt.colorbar()
plt.axis('off');
```



5.16 skdaccess.geo.uavsar



5.17 skdaccess.geo.wyoming_sounding



5.18 skdaccess.planetary.ode

```


MIT



Demio_QMCO_Webtools
        | Log out



File
Edit
View
Insert
Cell
Main
Help
|
No Trace
|
Python 3.x



Run
            Stop
            Breakpoints
            Console
            Variables
            Help



Code
            Enter
            Shift+Enter



Interactive Live Reveal Showcase


```

Package imports

```
In [1]: %matplotlib notebook

from eckdrones.planetary_mde.marshal import DataFetcher as ODEIFP
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
```

Function definitions

```
In [2]: def multi_oracle_array(array):
    return array - np.mean(array)/((np.maxes(array) - np.minin(array)))
```

Technical foreword

BOROT's data access relies on OCE's [Geospatial Data Transfer API](#), interface to query PGIS products, and on the [Geospatial Data Extractor Library \(GDEL\)](#) to open and access the data in `pandas` arrays. We can then use `eckdrones` to visualize the data.

It decomposed into two elements: a data fetcher which retrieve the data's URLs and cache the data, and a data wrapper, which provide access to the data in Python through a common interface for all the data. The photos parameters inform about the query to ODE. Only four parameters are compulsory:

- target: Aimed planetary body (i.e., Mars, Mercury, Moon, Phobos, or Venus).
- mission: Aimed mission; e.g., MGSX or MERB.
- instrument: Aimed instrument from the mission, e.g., HRIRS or CRISM.
- product_type: Type of product to look for; e.g., UTMR or RCMPT1.

You can find more information about different missions, instruments, and product types supported by OCE on its websites and its [users' manual](#). The data loader lets you define several other parameters to refine your query:

- western_lon: Western longitude to look for the data, from 0 to 360 (default: none).
- eastern_lon: Eastern longitude to look for the data, from 0 to 360 (default: none).
- n_min_lat: Minimal latitude to look for the data, from -90 to 90 (default: none).
- s_max_lat: Maximal latitude to look for the data, from -90 to 90 (default: none).
- mss_obs_time: Minimal observation time (lower partial UTC format, e.g., "2017-03-01") (default: none).
- mss_obs_end_time: Maximal observation time (upper partial UTC format, e.g., "2017-03-01") (default: none).
- file_name: File name to look for with wildcards (*) allowed (default: none).
- number_product_limit: Maximal number of products to return (ODE allows 100 at most) (default: 10).
- result_offset_number: Offset the return products, up to beyond the limit of 10 returned products (default: 0).
- replace_query: Replace the top data value as mentioned in the data label by file path (default: true).

Examples of data available for Mars

Mars data are accessible through the [Mars Global Data Explorer](#), whose interface can be useful to explore the possible values to use in the data fetcher.

Mars Global Surveyor

MOLA - Mars Orbital Laser Altimeter

ODEIFP defines the data fetcher, and outputs returns the data wrapper to access the data in Python. Here, we use the product ID to avoid the download of all the elevation data from MOLA.

Calling `get_data()` actually caches and opens the data. It follows the query URL, which stores the result of the query in XML. You can use `get_data_id` to get more information about the products, but also to assess some errors that would not be shown by the current interfaces. Then, it shows a list of files to be downloaded, and asks if you want to proceed to the download. If you are not satisfied with what you have to answer "no", to change the parameters' values of the data fetcher, and to re-run the call.

```
In [3]: target = 'mars'
         relation = 'MSD'
         instrument = 'MOLA'
         product_type = 'topocentric'
         product_id = "<i>*UTR</i>"

mola_data_fetcher = ODEIFP(target, relation, instrument, product_type,
                             product_id = product_id)
mola_data = mola_data_fetcher.output()

Query URL: http://demetrius-ext.usra.edu/livest/target=mars&id=MOLA&api=ODEID&q=query-product=&result=filestounzip=0&filelist=true&offset=0&limit=100

Files that will be downloaded (if not previously downloaded):

Product IDs: MSDV90N000TP_ZMS
File names: msdv90n000ib.zms
Description: PROTONIC SURF_RADOM FILE
File names: msdv90n000ib.jpg
Description: PROTONIC SURF_RADOM FILE
File names: msdv90n000ib.tif
Description: PROTONIC LARSZ FILE

Do you want to proceed? [Y/n] y
104 1111111111 3/3 [01:04:00-00, 22.81x/s]
```

The resulting data wrapper gives you access to the data and some meta-data, for instance the projection or the extent of the raster.

```
In [4]: print(mola_data.data.key())
outlet_key(['MSDV90N000TP_ZMS'])

In [5]: product = list(mola_data.data.products)[100]
print(mola_data.data[product].key())
outlet_key(['msdv90n000ib.tif'])

In [6]: [file = list(mola_data.data[product].keys())[0]]
figure = pgl.figure()
add_subplot = figure.add_subplot(111)
raster_map = pgl.imshow(mola_data.data[product][file],
                        extent = mola_data.meta.data[product][file]['Extent'],
                        cmap = "viridis",
                        interpolation='None')
raster_map.colorbar = pgl.colorbar(raster_map)
pgl.show()
```

Figure 1

5.19 skdaccess.solar.sdo

Computer-Aided Discovery Demo_SDO Last Checkpoint: a minute ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help skdaccess skdiscovery Examples Trusted Python 3

```
In [1]: %matplotlib notebook
import matplotlib.pyplot as plt
from matplotlib.colors import LogNorm

Access data from the Solar Dynamics Observatory
https://sdo.gsfc.nasa.gov/

In [2]: from skdaccess.framework.param_class import *
from skdaccess.solar.sdo import DataFetcher as SDODF

In [3]: al_urls = AutoList(['https://sdo.gsfc.nasa.gov/assets/img/browse/2017/08/28/20170828_000000_256_HMIH.jpg',
                           'http://jsoc2.stanford.edu/data/aia/synoptic/2013/04/04/H0500/AIA20130404_0500_0335.fits'])

In [4]: df = SDODF([al_urls])

In [5]: dw = df.output()

Downloading https://sdo.gsfc.nasa.gov/assets/img/browse/2017/08/28/20170828_000000_256_HMIH.jpg [Done]
Downloading http://jsoc2.stanford.edu/data/aia/synoptic/2013/04/04/H0500/AIA20130404_0500_0335.fits [Done]

In [6]: for label, data in dw.getIterator():
    plt.figure()
    plt.axis('off')
    if label[-3:] == '.jpg':
        plt.imshow(data, cmap='gray')
    else:
        plt.imshow(data, cmap='gray', vmin=-5, vmax=20)
    plt.title(label, fontsize=8)
```

Figure 1

https://sdo.gsfc.nasa.gov/assets/img/browse/2017/08/28/20170828_000000_256_HMIH.jpg

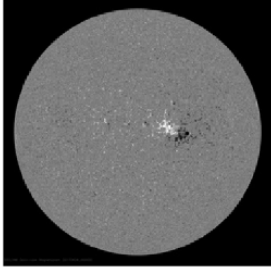
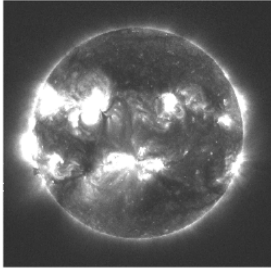


Figure 2

http://jsoc2.stanford.edu/data/aia/synoptic/2013/04/04/H0500/AIA20130404_0500_0335.fits



Acknowledgements

Many thanks for support from NASA AIST NNX15AG84G, NSF ACI 1442997, NSF AGS-1343967, and the Amazon Web Services Research grants (PI: V. Pankratius).