



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

Quick Start Guide

v1.9.3 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, 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:

3 Installation and Modes of Operation

The package can easily be 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 user's 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
```

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. geo. groundwater	Dictionary of Data Frames	USGS National Water Information System (https://waterservices.usgs.gov/rest/DV-Service.html)	≈ 1GB	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. 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)	≈ 1GB	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. 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. 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. modis	Dictionary of Numpy arrays	NASA MODIS (https://ladsweb.modaps.eosdis.nasa.gov/tools-and-services/)	≈ 100MB /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. mahali	Dictionary of rinex files	MIT-Haystack Observatory (http://apollo.haystack.mit.edu/mahali-data/)	≈ 10GB	MIT led NSF project studying the Earth's ionosphere with GPS (mahali.mit.edu). This data set uses a cache data fetcher.

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

MIT

Computer-Aided
Discovery

Demo_Kepler

Last Checkpoint: a minute ago (autosaved)

Logout

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	

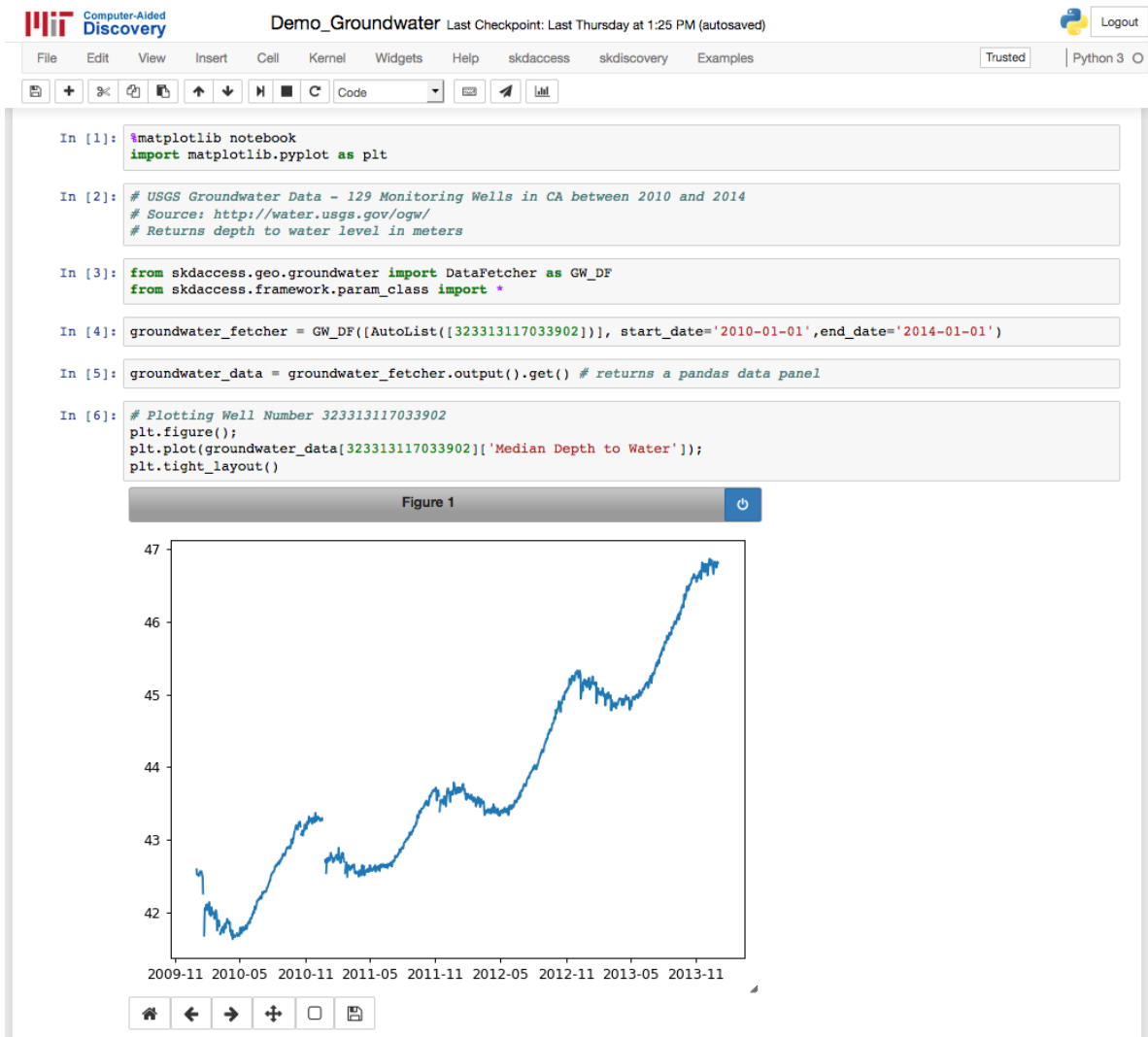
In [8]:

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.geo.groundwater



5.3 skdaccess.geo.pbo

MIT Computer-Aided Discovery Demo_PBO Last Checkpoint: Last Thursday at 1:25 PM (unsaved changes) Logout

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

```
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()

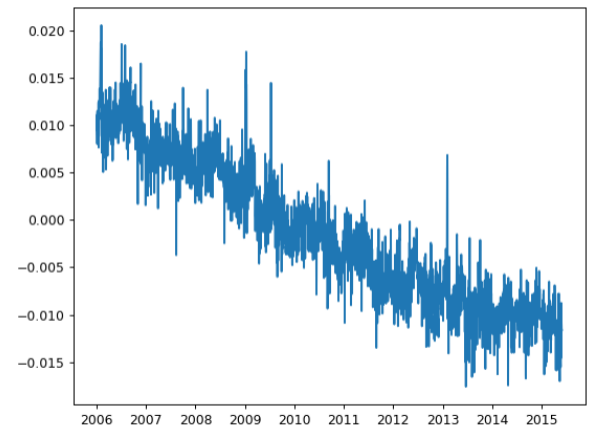
Out[6]:
```

	HMMSS	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

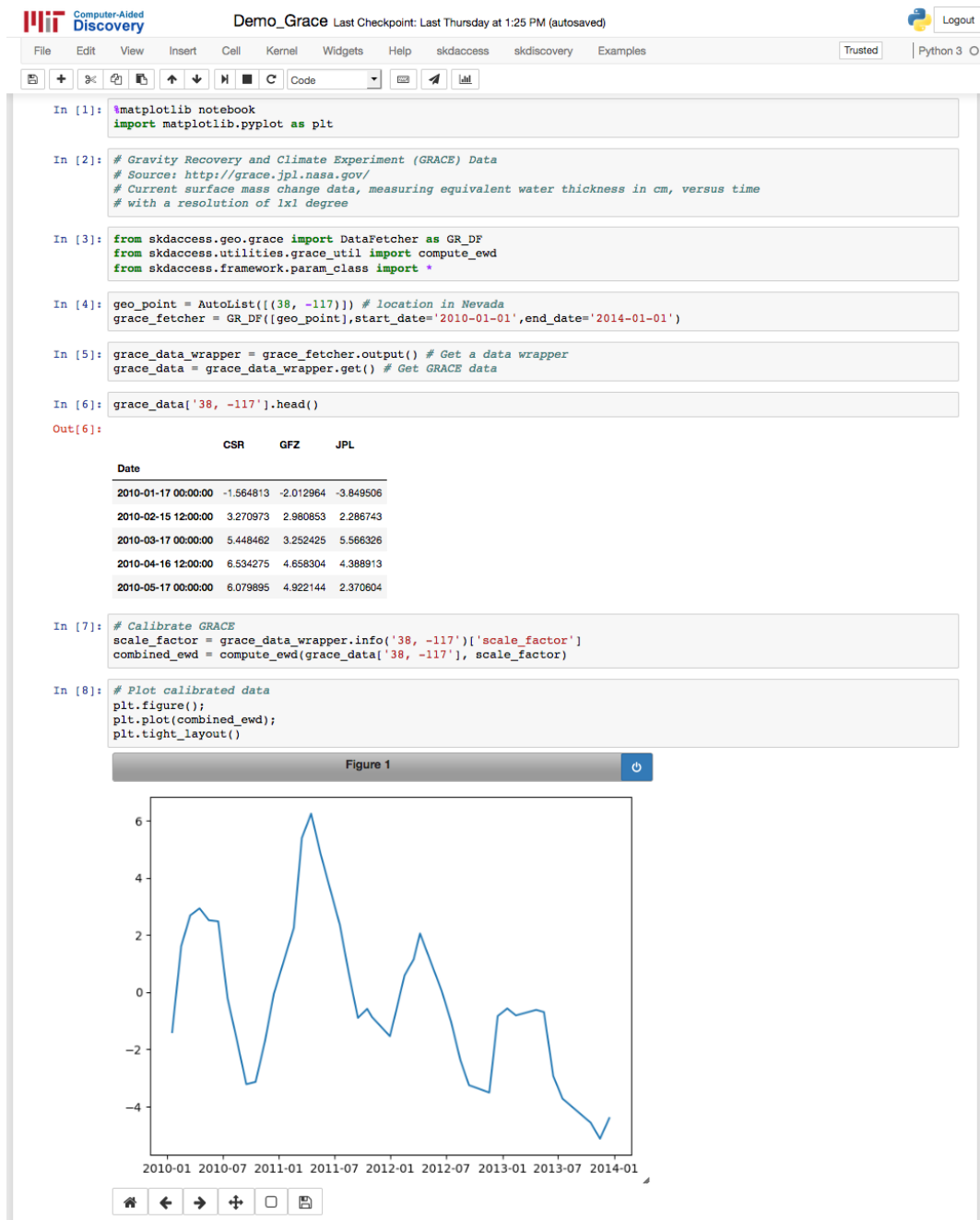
5 rows x 24 columns

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

Figure 1



5.4 skdaccess.geo.grace



Computer-Aided Discovery
DEMO_GLDAS Last Checkpoint: 21 minutes ago (unsaved changes)
 Logout

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

Code

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

Land hydrology model produced by NASA
https://grace.jpl.nasa.gov/data/get-data/land-water-content/
```

```
In [2]: from skdaccess.geo.gldas import DataFetcher as GLDAS_DF
from skdaccess.framework.param_class import *
```

```
In [3]: geo_point = AutoList([(38, -117)]) # location in Nevada
gldas_fetcher = GLDAS_DF([geo_point],start_date='2010-01-01',end_date='2014-01-01')
```

```
In [4]: data_wrapper = gldas_fetcher.output() # Get a data wrapper
label, data = next(data_wrapper.getIterator()) # Get GLDAS data
```

```
In [5]: data.head()
```

Out[5]:

	Equivalent Water Thickness (cm)	Uncertainty
2010-01-16	2.271377	NaN
2010-02-16	6.710746	NaN
2010-03-18	7.592598	NaN
2010-04-18	5.524107	NaN
2010-05-18	3.634455	NaN

```
In [6]: plt.plot(data['Equivalent Water Thickness (cm)']);
plt.xticks(rotation=15);
```

Figure 1

5.6 skdaccess.geo.modis.cache.reflectance

MIT Computer-Aided Discovery

Demo_MODIS Last Checkpoint: a few seconds ago (autosaved)

Logout

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

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

MODIS surface reflectance product at 1 km resolution ("MOD09")
<https://modis.gsfc.nasa.gov/data/>

In [2]: # Import AutoParams, calibration and rescaling functions, and Stream Data Fetcher
from skdaccess.framework.param_class import *
from skdaccess.utilities.modis_util import calibrateModis, rescale
from skdaccess.geo.modis.cache.reflectance import DataFetcher as MODISDF

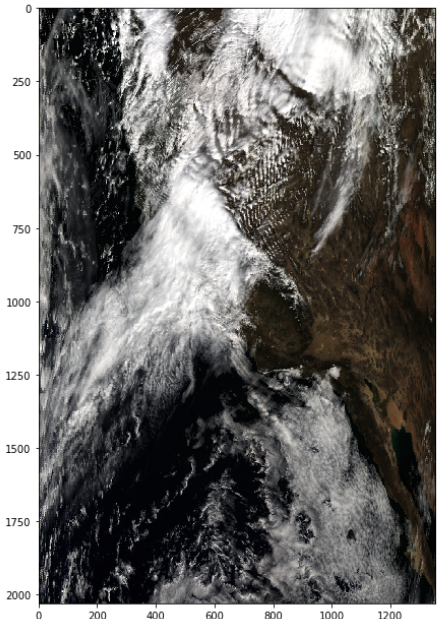
In [3]: # Create MODIS data fetcher
modis_df = MODISDF([AutoParam(38),AutoParam(-119)] , '2012-03-13', '2012-03-13')

In [4]: # Access data wrapper
modis_dw = modis_df.output()

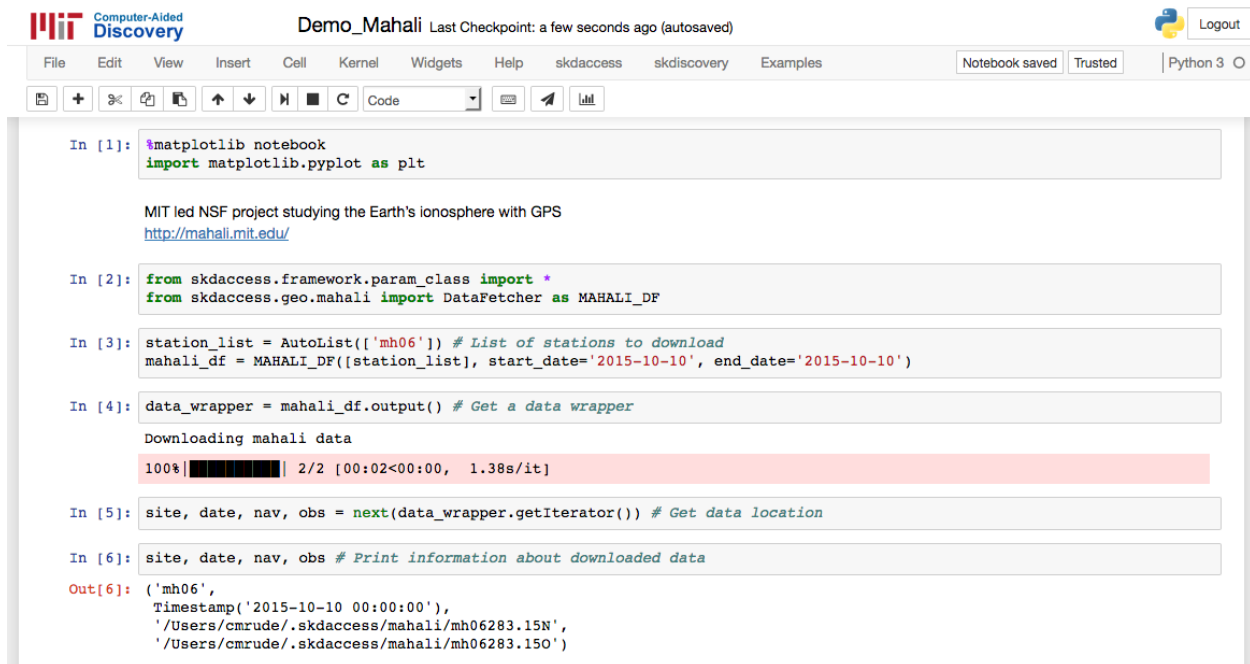
In [5]: # Use iterator to access data
label, data = next(modis_dw.getIterator())

In [6]: # Calibrate and scale data
calibrated_data = rescale(calibrateModis(data,modis_dw.info(label)))

In [7]: # Plot color image of result
plt.gcf().set_size_inches(7,12);
plt.imshow(calibrated_data);



5.7 skdaccess.geo.mahali



```
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')
```

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).