SCIKIT- data access
DATA INTERFACES FOR PYTHON

# Skdaccess: The **Scikit Data Access** Python Package Quick Start Guide

v1.9.17 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 plattform: 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 struc-ture | 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 *Kepler* Space Telescope (https://keplerscience.arc.nasa.gov/). This data set uses a cache data fetcher. |
| skdaccess. astro. spectra | Dictionary of Data Frames | Sloan Digital Sky Survey Science Archive Server (https://data.sdss.org/sas/) | 100KB / image | Spectra from the Sloan Digital Sky Server (https://www.sdss.org/dr14/spectro/). This data set uses a stream 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. engineering. la. traffic_counts | Dictionary of Data Frames | Los Angeles Open Data (https://data.lacity.org) | ≈ 0.1MB | Traffic count data in Los Angeles (https://data.lacity.org/A-Livable-and-Sustainable-City/LADOT-Traffic-Counts-Summary/94wu-3ps3). This data set uses a stream data fetcher. |
| skdaccess. finance. timeseries | Dictionary of Data Frames | Alpha Vantage (https://www.alphavantage.co/) | Data product depen-dent | Stock data obtained from Alpha Vantage (https://www.alphavantage.co/). This data set uses a stream data fetcher. |
| skdaccess. geo. era_interim | XArray Dataset | The University Corporation for Atmospheric Research (https://rda.ucar.edu/datasets/ds627.0/) | ≈ 0.1GB / day | 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) | ≈ 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.magnetometer | Dictionary of Data Frames | USGS National Geomagnetism Program (https://geomag.usgs.gov/products/downloads.php) | ≈ 1GB | 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/) | ≈ 10GB | 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/) | ≈ 1GB | 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/) | ≈ 0.1GB | 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/) | ≈ 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.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.sentinel_1 | Dictionary of Numpy arrays | Alaska Satellite Facility (https://www.asf.alaska.edu/sentinel/) | ≈ 1 − 10GB / 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/) | ≈ 100GB | 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 dependent | Synthetic Aperture Radar Single Look Complex data from the Uninhabited 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 Frames | 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 dependent | Planetary data from PDS Geosciences 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 Observatory (https://sdo.gsfc.nasa.gov/assets/img/browse/) and the Joint Science Operations Center (http://jsoc2.stanford.edu/data/aia/synoptic/ | Data product dependent | Images from the Solar Dynamics 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 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
```

The data will be downloaded into the current directory, and the .skdaccess config file located in the user's 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) — 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
```

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)
```

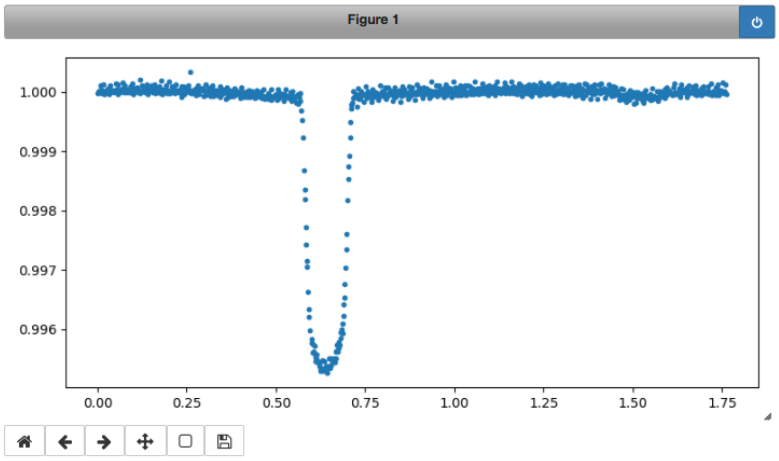
In [6]:
```
normalize(kepler_data['009941662'])
```

In [7]:
```
kepler_data['009941662'].head()
```

Out[7]:

| CADENCENO | TIME | TIMECORR | SAP_FLUX | SAP_FLUX_ERR | SAP_BKG | SAP_BKG_ERR | PDCSAP_FLUX | PDCSAP_FLUX_ERR | SAP_QUALITY | PSF_CEN |
|---|---|---|---|---|---|---|---|---|---|---|
| 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();
```



## 5.2 skdaccess.astro.spectra

Computer-Aided Discovery — Demo_SDSS_Spectra Last Checkpoint: 17 minutes ago (autosaved) — Logout

File Edit View Insert Cell Kernel Widgets Help — Trusted — Python 3

Run | Code | Enter/Exit Live Reveal Slideshow | nbdiff

Setup plotting libraries

In [1]:
```
%matplotlib inline
```

In [2]:
```
import matplotlib.pyplot as plt
plt.rcParams['figure.dpi'] = 150
```

Import data fetcher

In [3]:
```
from skdaccess.framework.param_class import *
from skdaccess.astro.spectra.stream import DataFetcher
```

Specify list of SDSS spectra URLs to retrieve

In [4]:
```
ap_spectra_url = AutoList([
    'https://dr14.sdss.org/sas/dr14/eboss/spectro/redux/v5_10_0/spectra/lite/4055/spec-4055-55359-0596.fits',
])
```

Create data fetcher

In [5]:
```
df = DataFetcher([ap_spectra_url])
```

Access data and metadata

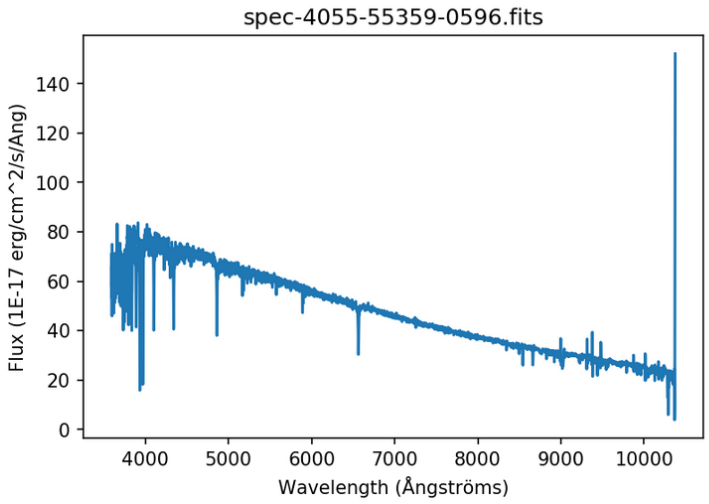
In [6]:
```
dw = df.output()
```

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

In [8]:
```
header = dw.info(label)
```

Plot spectra

In [9]:
```
plt.plot(10**data['loglam'], data['flux']);
plt.title(label.split('/')[-1]);
plt.ylabel('Flux ({})'.format(header['BUNIT']));
plt.xlabel('Wavelength (Ångströms)');
```



## 5.3 skdaccess.astro.voyager

Computer-Aided Discovery — Demo_Voyager Last Checkpoint: a few seconds ago (autosaved) — Logout

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

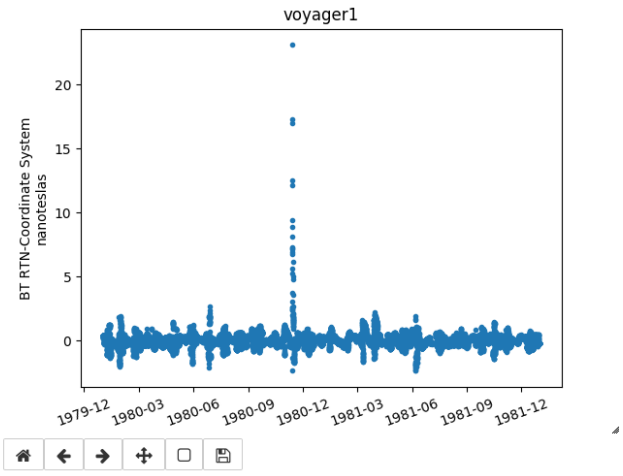
```
In [1]: %matplotlib notebook
        import matplotlib.pyplot as plt
```

Voyager mission data

Data is retrieved from the Space Physics Data Facility https://spdf.gsfc.nasa.gov/

```
In [2]: from skdaccess.astro.voyager import DataFetcher

In [3]: df = DataFetcher(1980,1981,spacecraft='voyager1')

In [4]: df.getMetadataFiles()
100%|██████████| 1/1 [00:00<00:00,  9.02it/s]
Out[4]: ['/Users/cmrude/.skdaccess/voyager/https/spdf.gsfc.nasa.gov/pub/data/voyager/voyager1/merged/vy1mgd.txt']

In [5]: dw = df.output()
100%|██████████| 2/2 [00:00<00:00,  3.53it/s]

In [6]: it = dw.getIterator()
        label, data = next(it)

In [7]: plt.plot(data.loc[data['BT']<900,'BT'],'.');
        plt.title(label)
        plt.ylabel(dw.info(label)['BT']['MEANING'] + '\n'
                   + dw.info(label)['BT']['UNITS/COMMENTS'])
        plt.xticks(rotation=20);
```

Figure 1



## 5.4 skdaccess.engineering.la.traffic_counts

Computer-Aided Discovery — Demo_Traffic_Counts (unsaved changes) — Logout

File Edit View Insert Cell Kernel Widgets Help — Not Trusted — Python 3

Run — Code — Enter/Exit Live Reveal Slideshow

In [1]:
```
%matplotlib inline
```

In [2]:
```
import matplotlib.pyplot as plt
plt.rcParams['figure.dpi'] = 150
```

In [3]:
```
import pandas as pd
```

In [4]:
```
# LA Traffic count data between 2014-01-14 and 2014-01-16
# Source: https://data.lacity.org/A-Livable-and-Sustainable-City/LADOT-Traffic-Counts-Summary/94wu-3ps3
```

In [5]:
```
from skdaccess.engineering.la.traffic_counts.stream import DataFetcher as TrafficDF
from skdaccess.framework.param_class import *
```

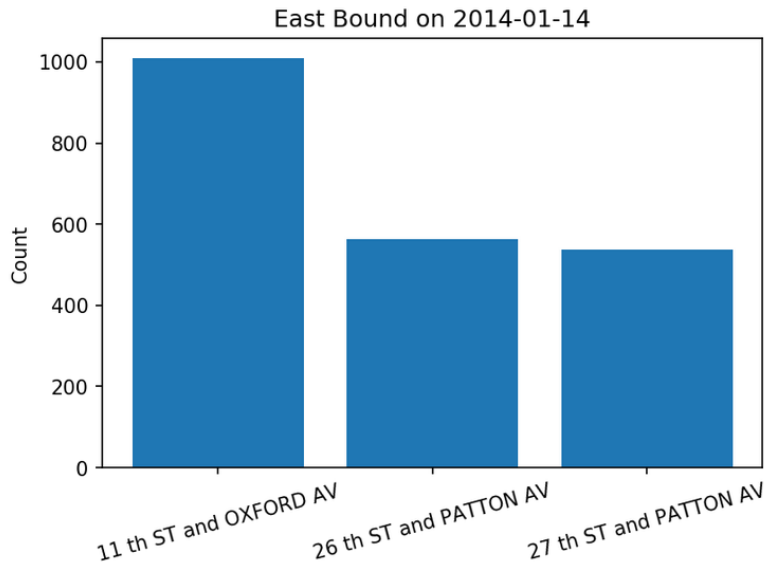
In [6]:
```
# Create traffic count data fetcher
# Note, use paramter app_token to supply an application token to prevent throttling
# See: https://dev.socrata.com/docs/app-tokens.html
traffic_df = TrafficDF(start_time='2014-01-14', end_time='2014-01-16')
```

In [7]:
```
# Create a data wapper
traffic_dw = traffic_df.output()
```

In [8]:
```
# Retrieve results
label, data = next(traffic_dw.getIterator())
```

In [9]:
```
# Select rows with east bound traffic on 2014-01-14
date = pd.to_datetime('2014-01-14')
cut_data = data[data.count_date == date]
cut_data = cut_data[cut_data.e_b != 0]
```

In [10]:
```
# Create plot
plt.title('East Bound on {}'.format(date.strftime('%Y-%m-%d')));
plt.ylabel('Count');
tick_labels = cut_data.primary_street + ' and ' + cut_data.cross_street
plt.bar(x=range(1,4), height=cut_data.e_b, tick_label=tick_labels);
plt.xticks(rotation=15);
```



## 5.5 skdaccess.finance.timeseries

Computer-Aided Discovery — Demo_Finance_Time_Series Last Checkpoint: a few seconds ago (unsaved changes) — Logout

File Edit View Insert Cell Kernel Widgets Help — Trusted — Python 3

Run — Code — Enter/Exit Live Reveal Slideshow — nbdiff

Initial imports

In [1]:
```
%matplotlib inline
```

In [2]:
```
import matplotlib.pyplot as plt
plt.rcParams['figure.dpi'] = 150
```

In [3]:
```
from skdaccess.framework.param_class import *
from skdaccess.finance.timeseries.stream import DataFetcher
```

Select which symbol to retrieve

In [4]:
```
stock_ap_list = AutoList(['SPY'])
```

Create a data fetcher

In [5]:
```
stockdf = DataFetcher([stock_ap_list], 'daily', '2017-06-01')
```

Access the data

In [6]:
```
dw = stockdf.output()
```

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

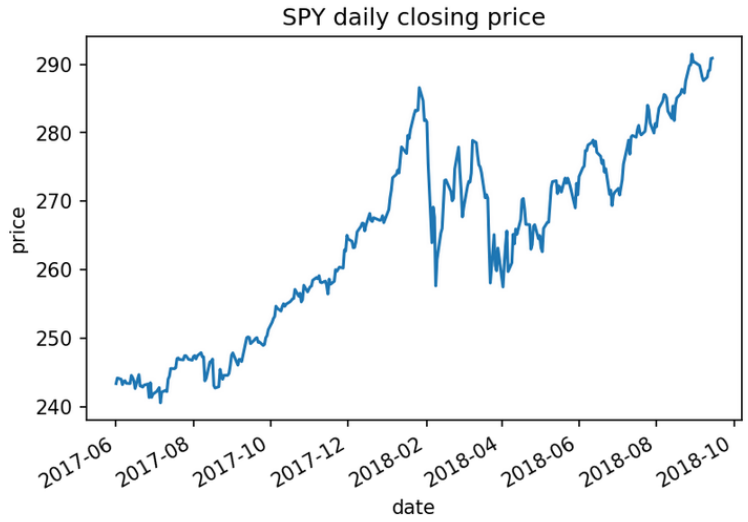
List the columns of the data

In [8]:
```
data.columns
```

Out[8]:
```
Index(['1. open', '2. high', '3. low', '4. close', '5. volume'], dtype='object')
```

Plot the closing price

In [9]:
```
ax = data['4. close'].plot(title='SPY daily closing price');
ax.set_ylabel('price');
```



## 5.6 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 — Markdown — 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)
```

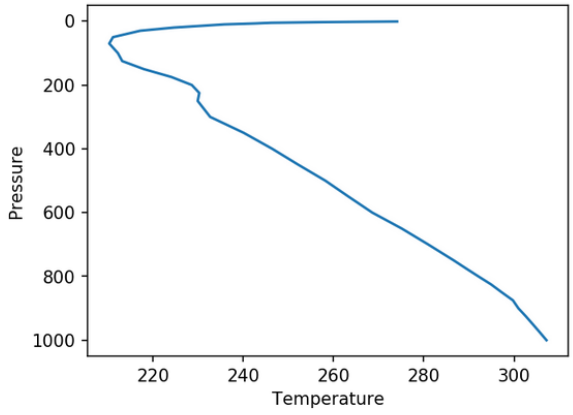
Access data

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');
```



## 5.7 skdaccess.geo.gldas

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
```

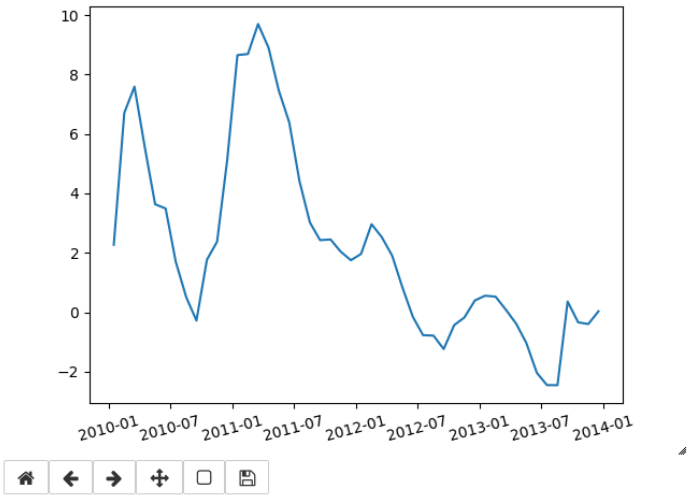
In [5]:
```
data.head()
```

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



## 5.8 skdaccess.geo.grace

Computer-Aided Discovery — Demo_GRACE Last Checkpoint: Last Thursday at 1:56 PM (unsaved changes) — Logout

File Edit View Insert Cell Kernel Widgets Help — Trusted — Python 3

Code

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

In [2]:
```
# Gravity Recovery and Climate Experiment (GRACE) Data
# Source: http://grace.jpl.nasa.gov/
# Current surface mass change data, measuring equivalent water thickness in cm, versus time
# with a resolution of 1x1 degree
```

In [3]:
```
from skdaccess.geo.grace import DataFetcher as GR_DF
from skdaccess.utilities.grace_util import computeEWD
from skdaccess.framework.param_class import *
```

In [4]:
```
geo_point = AutoList([(38, -117)]) # location in Nevada
grace_fetcher = GR_DF([geo_point],start_date='2010-01-01',end_date='2014-01-01')
```

In [5]:
```
grace_data_wrapper = grace_fetcher.output() # Get a data wrapper
grace_data = grace_data_wrapper.get() # Get GRACE data
```

In [6]:
```
grace_data['38, -117'].head()
```

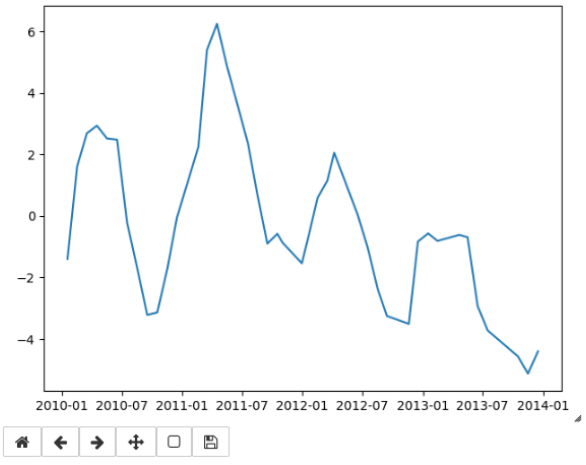
Out[6]:

| Date | CSR | GFZ | JPL |
|---|---|---|---|
| 2010-01-17 00:00:00 | -1.564813 | -2.012964 | -3.849506 |
| 2010-02-15 12:00:00 | 3.270973 | 2.980853 | 2.286743 |
| 2010-03-17 00:00:00 | 5.448462 | 3.252425 | 5.566326 |
| 2010-04-16 12:00:00 | 6.534275 | 4.658304 | 4.388913 |
| 2010-05-17 00:00:00 | 6.079895 | 4.922144 | 2.370604 |

In [7]:
```
# Calibrate GRACE
scale_factor = grace_data_wrapper.info('38, -117')['scale_factor']
combined_ewd = computeEWD(grace_data['38, -117'], scale_factor)
```

In [8]:
```
# Plot calibrated data
plt.figure();
plt.plot(combined_ewd);
plt.tight_layout()
```

Figure 1



## 5.9 skdaccess.geo.grace.mascon

Computer-Aided Discovery

Demo_GRACE_Mascon Last Checkpoint: 05/07/2018 (unsaved changes)

Logout

File Edit View Insert Cell Kernel Help

Trusted | Python 3

Run Code Enter/Exit Live Reveal Slideshow

In [1]:
```
%matplotlib inline
```

In [2]:
```
import matplotlib.pyplot as plt
plt.rcParams['figure.dpi']=150
```

In [3]:
```
# Gravity Recovery and Climate Experiment (GRACE) Data
# Source: http://grace.jpl.nasa.gov/
# Current surface mass change data, measuring equivalent water thickness in cm, versus time
# This data fetcher uses results from the Mascon solutions
```

In [4]:
```
from skdaccess.geo.grace.mascon.cache import DataFetcher as GR_DF
from skdaccess.framework.param_class import *
```

In [5]:
```
geo_point = AutoList([(38, -117)]) # location in Nevada
grace_fetcher = GR_DF([geo_point],start_date='2010-01-01',end_date='2014-01-01')
```

In [6]:
```
grace_data_wrapper = grace_fetcher.output() # Get a data wrapper
grace_label, grace_data = next(grace_data_wrapper.getIterator())# Get GRACE data
```

In [7]:
```
grace_data.head()
```

Out[7]:

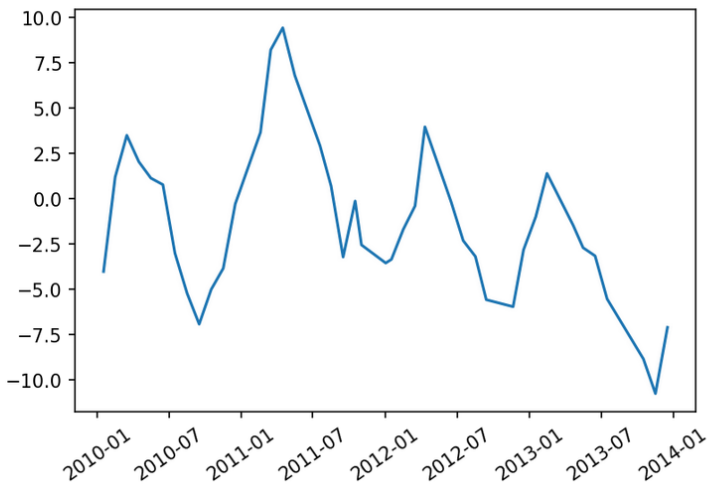
| | EWD | EWD_Error |
|---|---|---|
| 2010-01-16 12:00:00 | -6.871147 | 1.856682 |
| 2010-02-15 00:00:00 | 2.071728 | 1.847683 |
| 2010-03-16 12:00:00 | 6.026260 | 2.088941 |
| 2010-04-16 00:00:00 | 3.524163 | 1.866020 |
| 2010-05-16 12:00:00 | 1.974874 | 1.937367 |

Get scale factor

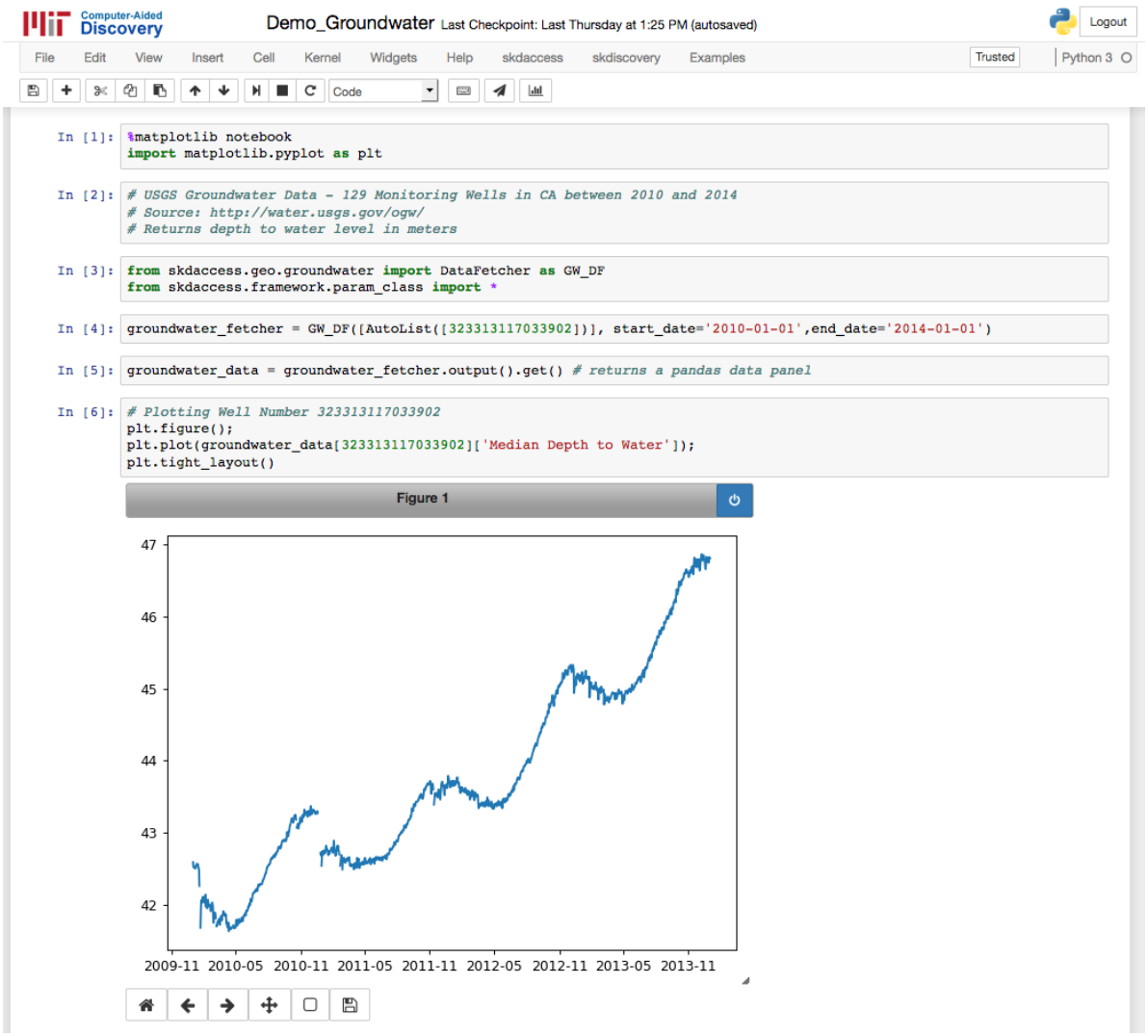
In [8]:
```
scale_factor = grace_data_wrapper.info(grace_label)['scale_factor']
```

Plot EWD × scale factor

In [9]:
```
plt.plot(grace_data['EWD']*scale_factor);
plt.xticks(rotation=35);
```



## 5.10 skdaccess.geo.groundwater



## 5.11 skdaccess.geo.magnetometer

Computer-Aided Discovery

Demo_Magnetometer Last Checkpoint: a few seconds ago (autosaved)

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
```

Geomagnetic data from the USGS

The results presented in this document rely on data collected at magnetic observatories operated by the U.S. Geological Survey (USGS, geomag.usgs.gov).

In [2]:
```
from skdaccess.framework.param_class import *
from skdaccess.geo.magnetometer import DataFetcher
```

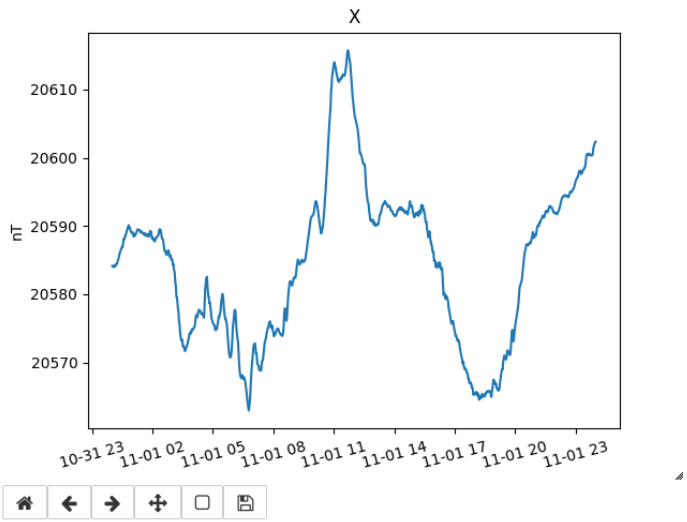
In [3]:
```
geomag_df = DataFetcher([AutoList(['BOU'])], start_time='2015-11-01',end_time=('2015-11-02'))
```

In [4]:
```
dw = geomag_df.output()
```

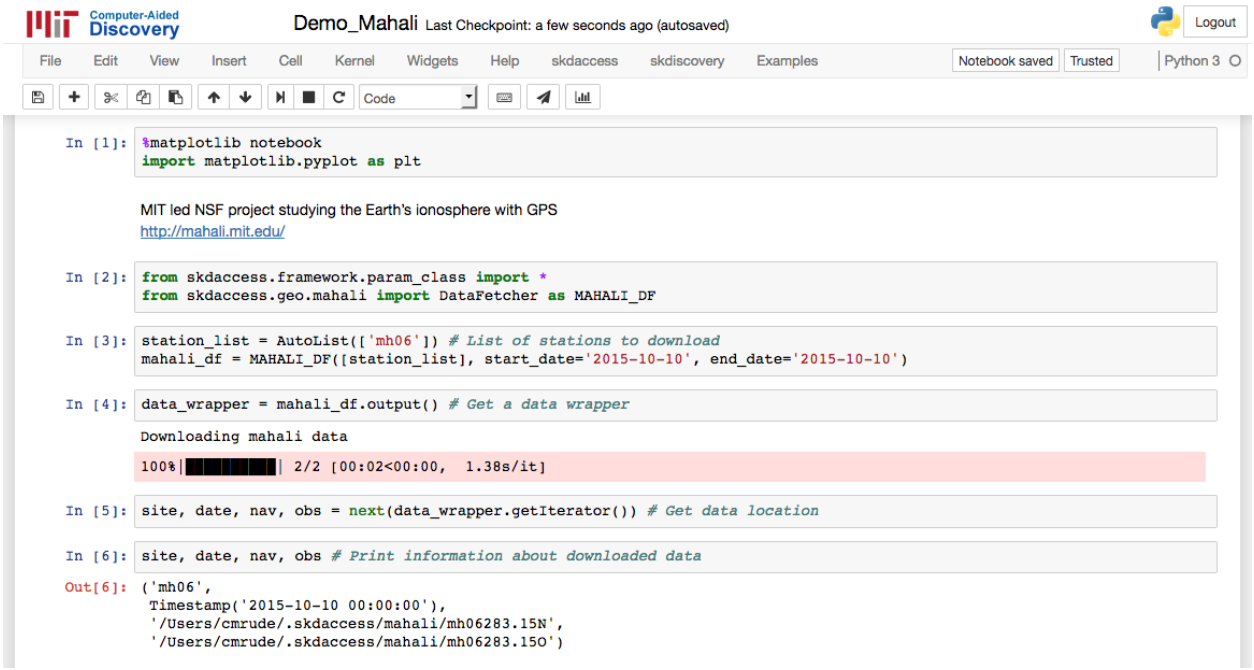
In [5]:
```
label, data = next(dw.getIterator())
```

In [6]:
```
plt.plot(data['X']);
plt.ylabel('nT');
plt.title('X');
plt.xticks(rotation=15);
```

Figure 1



## 5.12 skdaccess.geo.mahali.rinex



## 5.13 skdaccess.geo.mahali.tec

Computer-Aided Discovery — Demo_Mahali_TEC (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
```

TEC data from Mahali
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.tec.data_fetcher import DataFetcher
```

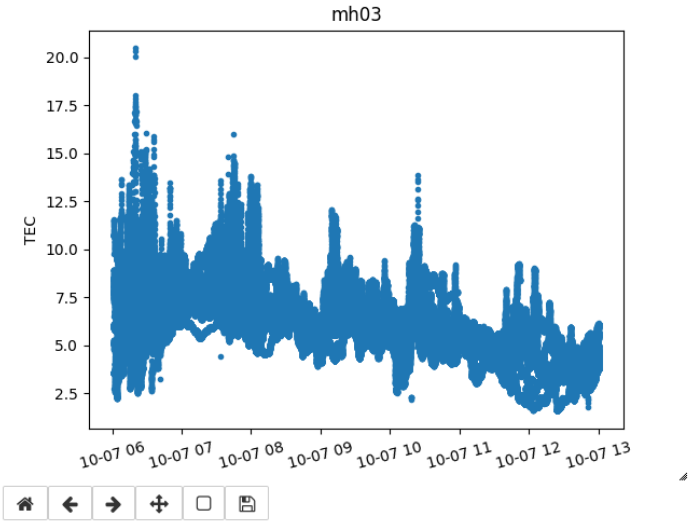
In [3]:
```
mahalidf = DataFetcher([AutoList(['mh03'])],start_date='2015-10-07', end_date='2015-10-07')
```

In [4]:
```
dw = mahalidf.output()
```

In [5]:
```
label, data = next(dw.getIterator())
```

In [6]:
```
plt.plot(data.vertical_tec[data.elevation > 20.0],'.');
plt.title(label);
plt.ylabel('TEC');
plt.xticks(rotation=15);
```

Figure 1



## 5.14 skdaccess.geo.mahali.temperature

Computer-Aided Discovery — Demo_Mahali_Temperature Last Checkpoint: 11/01/2017 (unsaved changes) — Logout

File Edit View Insert Cell Kernel Help — Trusted — Python 3

Run — Code — Enter/Exit Live Reveal Slideshow

In [1]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import pandas as pd
plt.rcParams['figure.figsize'] = (14,14)
```

Temperature data from Mahali 2015 Alaska Experiment (Temperature sensor inside Mahali box)
MIT led NSF project studying the Earth's ionosphere with GPS
http://mahali.mit.edu/

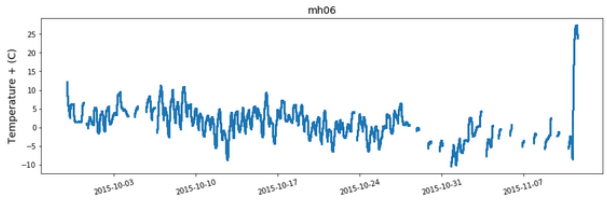
In [2]:
```
from skdaccess.framework.param_class import *
from skdaccess.geo.mahali.temperature import DataFetcher as MHDF
```

In [3]:
```
station_list = [ 'mh02', 'mh03', 'mh04', 'mh05', 'mh06', 'mh07', 'mh08', 'mh09', 'mh13']
```

In [4]:
```
mhdf = MHDF([AutoList(station_list)])
```

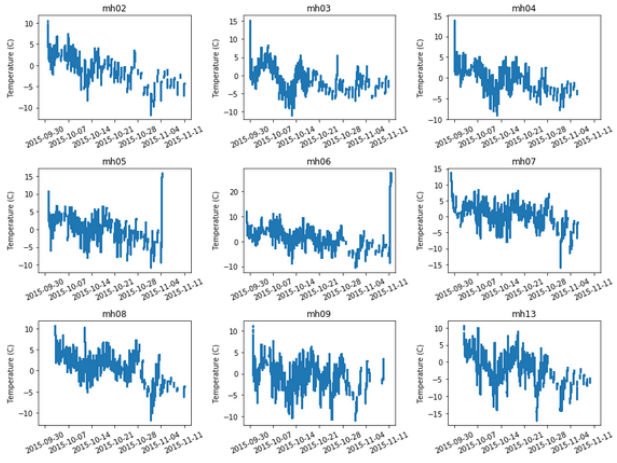
In [5]:
```
dw = mhdf.output()
```

In [6]:
```
label = 'mh06'
data = dw.get()['mh06']
plt.plot(data['Temperature'],'o',markersize=1);
plt.title(label,fontsize=14);
plt.ylabel('Temperature + (C)', fontsize=14);
plt.xticks(rotation=15);
plt.gcf().set_size_inches(14,4)
```




In [7]:
```
def plotAllStations(start_date, end_date):
    plt.gcf().set_size_inches(12,9)

    for index, (label, data) in enumerate(dw.getIterator()):
        plt.subplot(3,3, index+1)
        plt.title(label);
        plt.ylabel('Temperature (C)');
        plt.xticks(rotation=25);
        plt.plot(data['Temperature'],'o',markersize=1);
        plt.xlim(start_date, end_date)
    plt.tight_layout()
```

In [8]:
```
plotAllStations(pd.to_datetime('2015-09-28'),pd.to_datetime('2015-11-13'))
```



## 5.15 skdaccess.geo.modis.cache.reflectance

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

Code

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
```

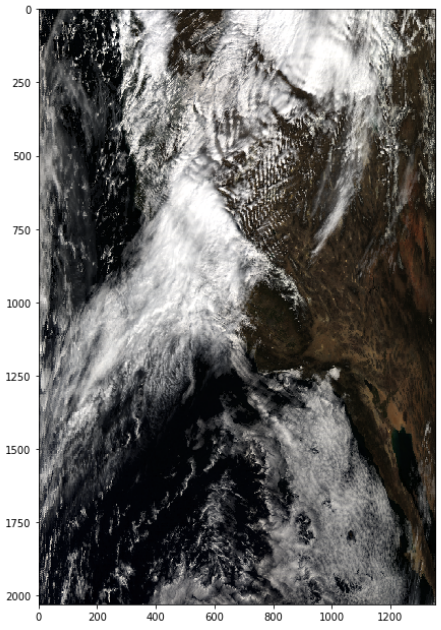
In [3]:
```
# Create MODIS data fetcher
modis_df = MODISDF([AutoParam(38),AutoParam(-119)] ,'2012-03-13','2012-03-13')
```

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);
```



## 5.16 skdaccess.geo.pbo

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

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

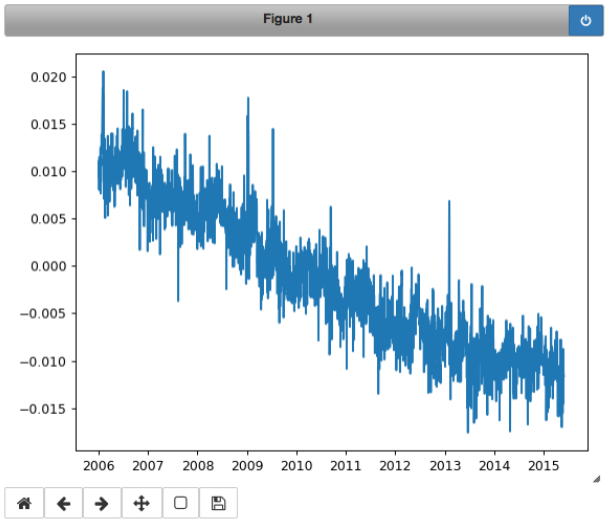
```
In [1]: # Plate Boundary Observatory GPS Data
        # Source: http://www.unavco.org/instrumentation/networks/status/pbo
        # Time series data for GPS sensors (North, East, Up), displacement in meters versus time

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]:

| | HHMMSS | JJJJJ.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 × 24 columns

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



## 5.17 skdaccess.geo.sentinel_1

Computer-Aided Discovery

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

Logout

File Edit View Insert Cell Kernel Help Trusted Python 3

Run Code Enter/Exit Live Reveal Slideshow

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://s1qc.asf.alaska.edu/aux_poeorb/S1A_OPER_AUX_POEORB_OPOD_20141124T123237_V20141102T225944_
```

Create data fetcher

In [7]:
```
s1df = S1DF(slc_url_list, satellite_url_list, username, password, swath=3, polarization='VV')
```

Access data

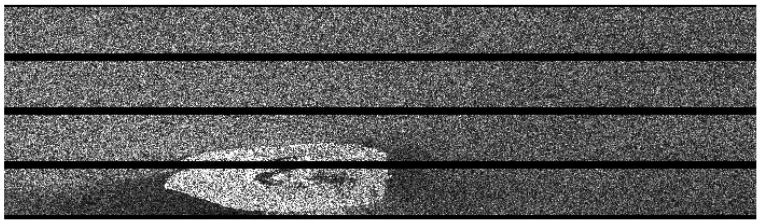
In [8]:
```
s1dw = s1df.output()
```

```
Retrieving SLC data
Retrieving orbit files
All files retrieved
```

In [9]:
```
label, data = next(s1dw.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



## 5.18 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

Run Code Enter/Exit Live Reveal Slideshow

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

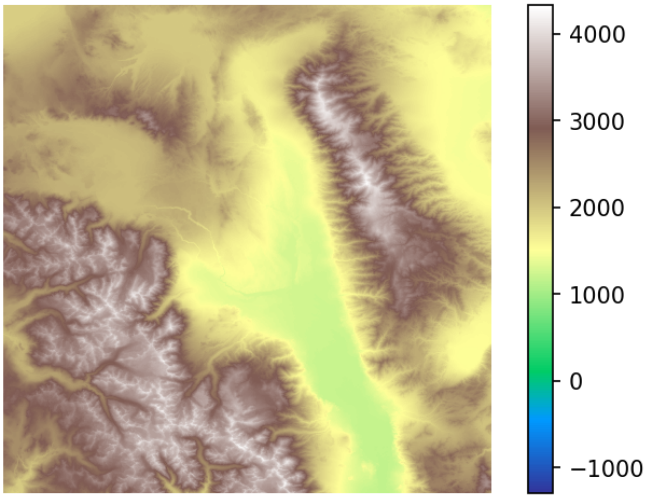
In [5]:
```
sdf = SDF(lat_tile_start=37,lat_tile_end=37,lon_tile_start=-119,lon_tile_end=-119,
          username=username,password=password)
```

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');
```



## 5.19 skdaccess.geo.uavsar

Computer-Aided Discovery

Demo_UAVSAR Last Checkpoint: 05/07/2018 (unsaved changes)

Logout

File Edit View Insert Cell Kernel Help Trusted Python 3

Run Code Enter/Exit Live Reveal Slideshow

In [1]:
```
%matplotlib inline
```

In [2]:
```
import matplotlib.pyplot as plt
plt.rcParams['figure.dpi'] = 150
from skimage.measure import block_reduce
import numpy as np
```

Import data fetcher

In [3]:
```
from skdaccess.geo.uavsar.cache import DataFetcher
```

Provide data URLs

In [4]:
```
slc_url_list = [
    'http://downloaduav2.jpl.nasa.gov/Release22/SanAnd_23511_01/SanAnd_23511_14068_001_140529_L090HH_01_BC_s2_1x1.slc',
]

metadata_url_list = [
    'http://downloaduav2.jpl.nasa.gov/Release22/SanAnd_23511_01/SanAnd_23511_14068_001_140529_L090HH_01_BC.ann',
]

llh_url = 'http://downloaduav2.jpl.nasa.gov/Release22/SanAnd_23511_01/SanAnd_23511_01_BC_s2_2x8.llh'
```

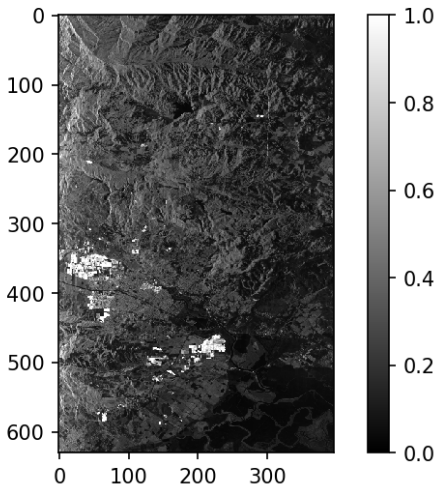
In [5]:
```
df = DataFetcher(slc_url_list = slc_url_list, metadata_url_list = metadata_url_list,
                 llh_url = llh_url, memmap = False)
```

In [6]:
```
dw = df.output()
```

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

In [8]:
```
smoothed_mag = block_reduce(np.abs(data),(100,25),np.median)
```

In [9]:
```
plt.imshow(smoothed_mag,cmap='gray',vmax=1);
plt.colorbar();
```



## 5.20 skdaccess.geo.wyoming_sounding

Computer-Aided Discovery

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

Logout

File Edit View Insert Cell Kernel Help

Trusted Python 3

Run Code Enter/Exit Live Reveal Slideshow

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

In [2]:
```
from skdaccess.framework.param_class import *
from skdaccess.geo.wyoming_sounding.cache import DataFetcher
```

Create a data fetcher

In [3]:
```
sdf = DataFetcher(station_number='72493', year=2014, month=5, day_start=30, day_end=30, start_hour=12, end_hour=12)
```

Access data

In [4]:
```
dw = sdf.output()
```

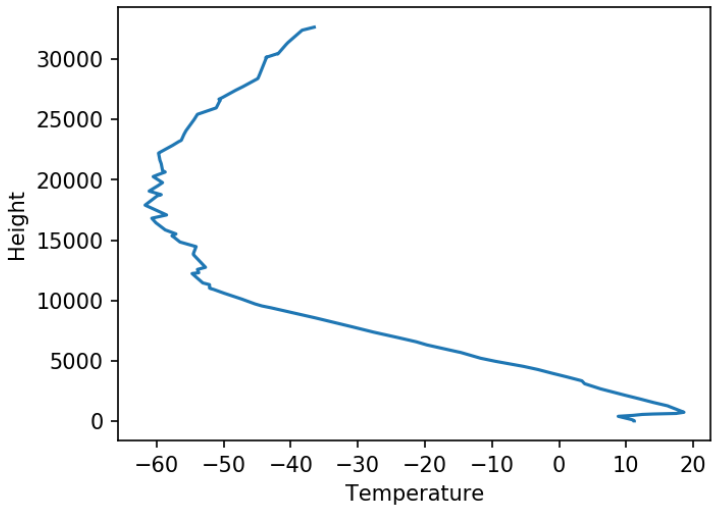
In [5]:
```
label, data = next(dw.getIterator())
```

In [6]:
```
data.head()
```

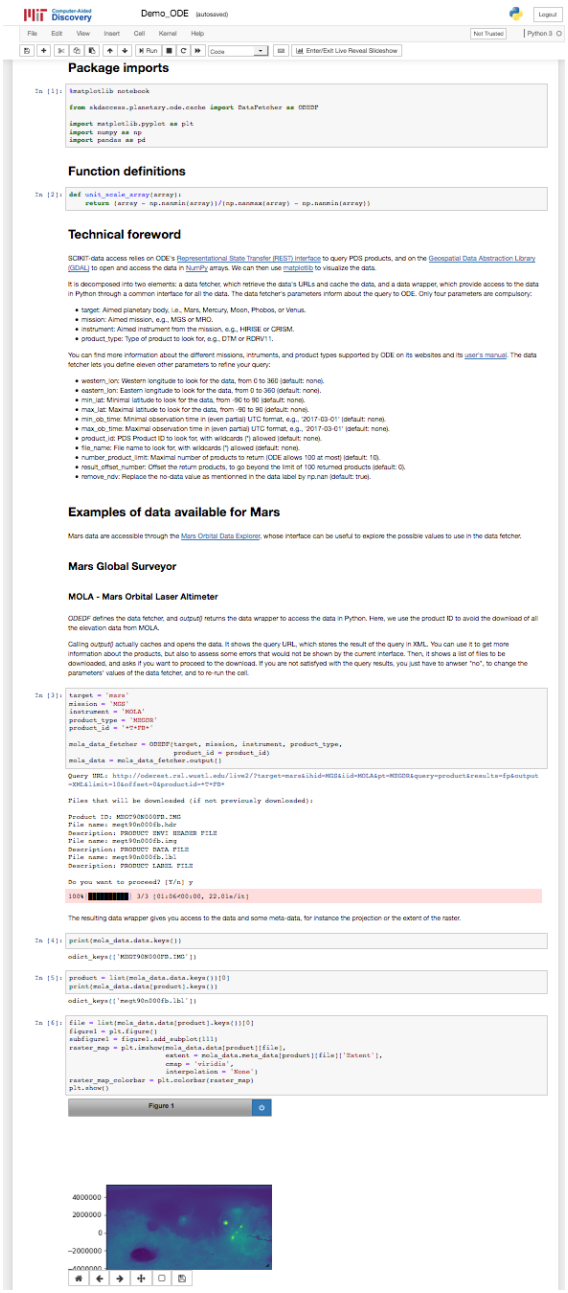
Out[6]:

| | PRES | HGHT | TEMP | DWPT | RELH | MIXR | DRCT | SKNT | THTA | THTE | THTV |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1013.0 | 3 | 11.2 | 9.6 | 90 | 7.46 | 180.0 | 2.0 | 283.3 | 304.0 | 284.6 |
| 1 | 1012.0 | 11 | 11.2 | 6.2 | 71 | 5.91 | 190.0 | 2.0 | 283.4 | 300.0 | 284.4 |
| 2 | 1005.0 | 69 | 11.0 | 8.1 | 82 | 6.78 | 257.0 | 4.0 | 283.8 | 302.7 | 284.9 |
| 3 | 1000.0 | 111 | 10.8 | 8.0 | 83 | 6.77 | 305.0 | 6.0 | 283.9 | 302.9 | 285.1 |
| 4 | 976.9 | 305 | 9.4 | 7.4 | 88 | 6.66 | 290.0 | 4.0 | 284.4 | 303.1 | 285.6 |

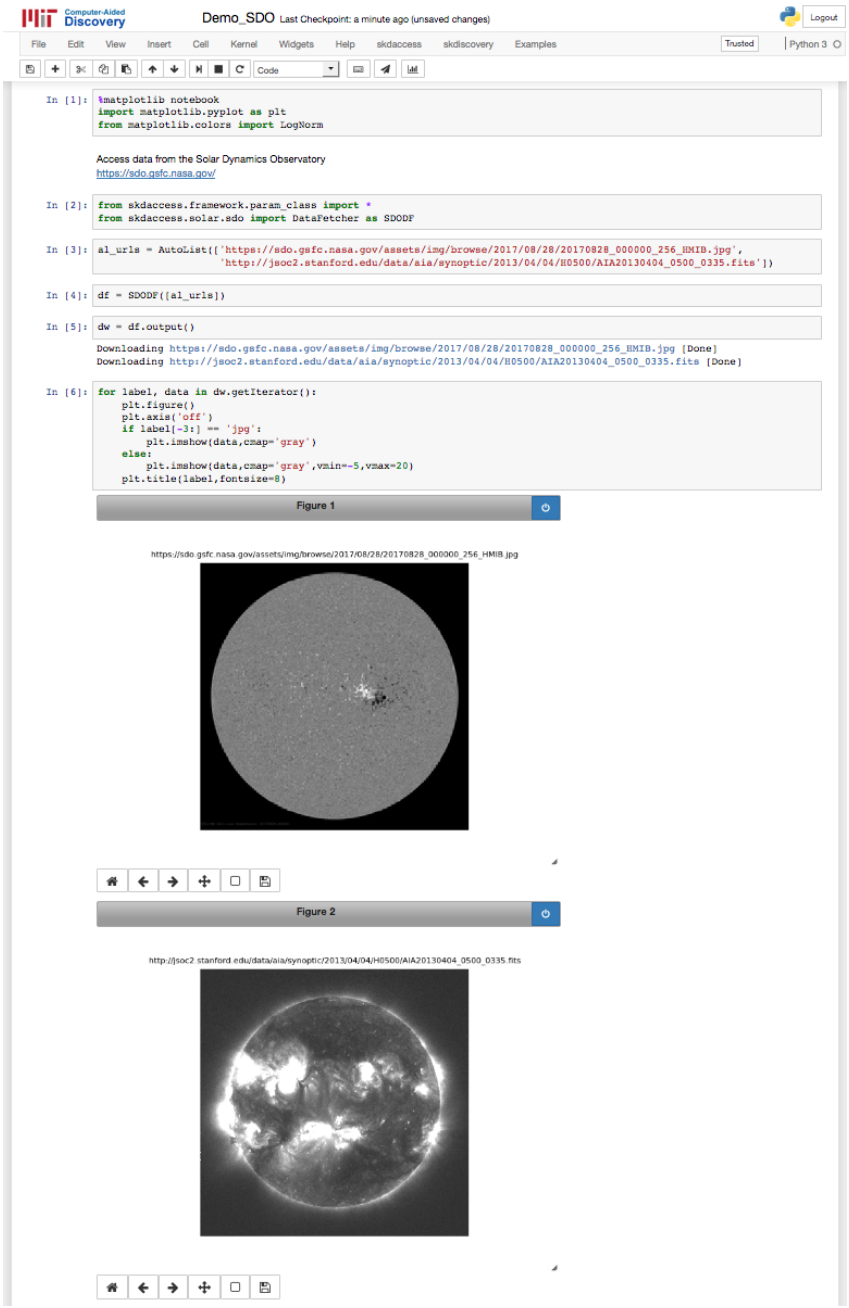
In [8]:
```
plt.figure(figsize=(5,3.75))
plt.plot(data['TEMP'],data['HGHT']);
plt.ylabel('Height');
plt.xlabel('Temperature');
```



## 5.21 skdaccess.planetary.ode



### 5.22 skdaccess.solar.sdo




## 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).