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

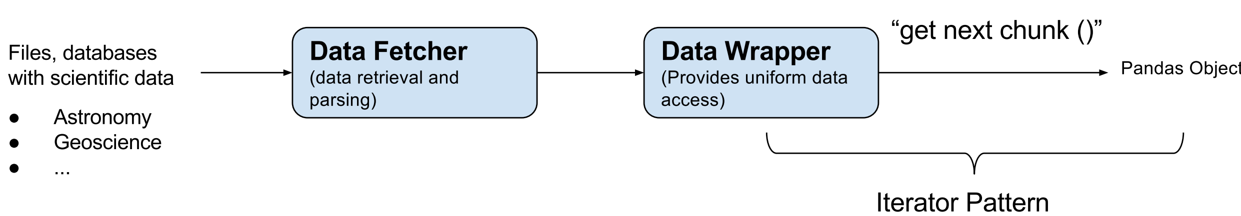
v1.1.1 for Python 3.4
https://pypi.python.org/pypi/skdaccess

Created and maintained by
MIT Haystack Observatory, Astro-&Geo-Informatics Group
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 skdacces

- API to import a data generator + function to get next data chunk (configurable)
- Eliminates the need to create parsers for each data set and simplifies the construction of scientific data processing pipelines.
- Enables studies involving data fusion and cross-comparisons from several sources.
- Skip parser development, dealing with physical file formats, etc.
- Can be used to download data locally or to a cloud node (e.g., Amazon Cloud). This feature simplifies distributing entire data sets or partitions of data to the cloud, and enables parallel processing in cloud computing environments.
- Easy expansion for more data sets in the future
- Skdaccess code is open source (MIT License)

2 Supported Data Sets

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

Skdaccess Namespace	**Data structure returned by get()**	**Original Source**	**Total Size**	**Description**
skdaccess.astro. kepler	Dictionary of Data Frames	The Mikulski Archive for Space Telescopes	≈ 1TB	Light curves for stars imaged by the *Kepler* Space Telescope
skdaccess. geo.groundwater	Pandas Panel	USGS National Water Information System	≈ 1GB	United States groundwater monitoring wells measuring the depth to water level.
skdaccess. geo.pbo	Pandas Panel	UNAVCO Plate Boundary Observatory	≈ 1GB	Daily GPS displacement time series measurements throughout the United States.
skdaccess. geo.grace	Pandas DataFrame	NASA Jet Propulsion Laboratory	≈ 1GB	Grace Tellus Monthly Mass Grids. 30-day measurements of changes in Earths gravity field to quantify equivalent water thickness.

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 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

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
        Groundwater - Ground water daily values from wells in California
```

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

```
> skdaccess
usage: skdaccess [-h] [-l] [-i LOCAL_DATA] [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
```

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 = fullDG.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

Demo_Kepler Last Checkpoint: 5 hours ago (autosaved)

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.framework.param_class import *
import numpy as np
```

In [4]:
```
kepler_fetcher = Kepler_DF([AutoList(['009941662'])], filter_window=50/24, normalize=True)
```

In [5]:
```
kepler_data = kepler_fetcher.output().get()
```

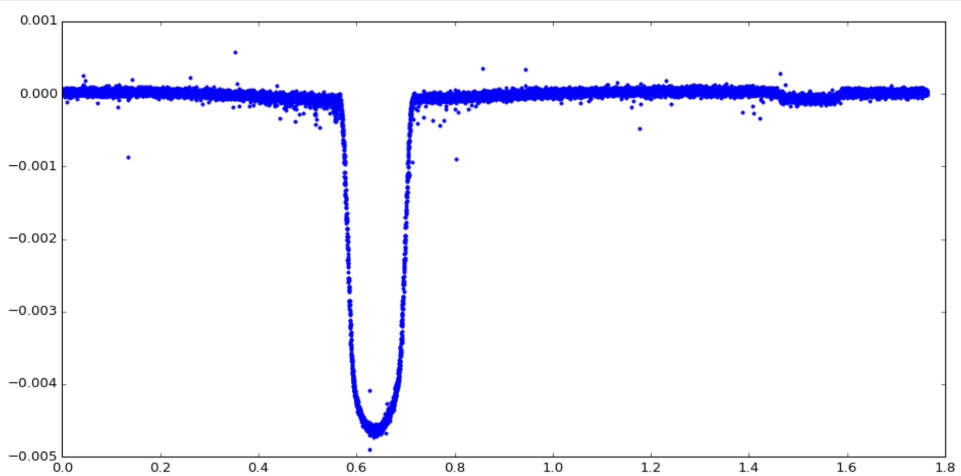
In [6]:
```
kepler_data['009941662'].head()
```

Out[6]:

	TIMECORR	CADENCENO	SAP_FLUX	SAP_FLUX_ERR	SAP_BKG	SAP_BKG_ERR	PDCSAP_FLUX	PDCSAP_FLUX_ERR	SAP_QU/
TIME									
120.539195	0.001042	568	1778533.750	33.049557	4841.642090	1.547178	-0.004503	32.609024	0
120.559629	0.001043	569	1778263.875	33.047188	4846.805664	1.546246	-0.004617	32.732418	0
120.580063	0.001044	570	1778347.750	33.048054	4848.539062	1.549641	-0.004565	32.837833	0
120.600498	0.001044	571	1778901.000	33.052914	4847.870117	1.543734	-0.004289	32.684124	0
120.620932	0.001045	572	1781658.250	33.081059	4852.192871	1.546612	-0.002727	32.769455	0

In [7]:
```
plt.figure(figsize=(12,6));
data = kepler_data['009941662'].iloc[0:15000]
plt.plot(np.array(data.index) % 1.7636, data['PDCSAP_FLUX'],'.');
plt.tight_layout();
```

Figure 1



5.2 skdaccess.geo.groundwater

MIT Computer-Aided Discovery — Demo_Groundwater Last Checkpoint: 5 hours ago (autosaved)

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

In [2]:
```
# USGS Groundwater Data - 129 Monitoring Wells in CA between 2010 and 2014
# Source: http://water.usgs.gov/ogw/
# Returns depth to water level in meters
```

In [3]:
```
from skdaccess.geo.groundwater import DataFetcher as GW_DF
```

In [4]:
```
groundwater_fetcher = GW_DF(start_date='2010-01-01',end_date='2014-01-01')
```

In [5]:
```
groundwater_data = groundwater_fetcher.output().get() # returns a pandas data panel
```

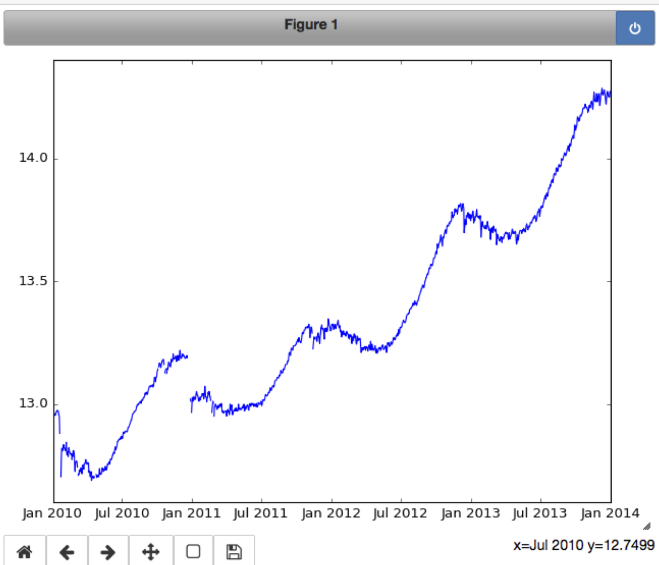
In [6]:
```
groundwater_data.loc['Water Depth'].head()
```

Out[6]:

	323313117033901	323313117033902	323313117033903	323313117033904	323313117033905	323527117050001	323527117050002	323
2010-01-01	12.896088	12.981432	13.024104	5.830824	NaN	13.203936	10.250424	6.14
2010-01-02	12.877800	12.960096	13.005816	5.797296	NaN	13.200888	10.146792	6.08
2010-01-03	12.877800	12.960096	13.002768	5.775960	NaN	13.213080	10.064496	6.05
2010-01-04	12.877800	12.960096	13.005816	5.760720	NaN	13.219176	9.982200	6.01
2010-01-05	12.877800	12.960096	13.005816	5.739384	NaN	13.222224	9.909048	5.97

5 rows × 129 columns

In [7]:
```
# Plotting Well Number 323313117033902
plt.figure();
plt.plot(groundwater_data.loc['Water Depth',:,'323313117033902']);
plt.tight_layout()
```



5.3 skdaccess.geo.pbo

Computer-Aided Discovery Demo_PBO Last Checkpoint: 6 hours ago (autosaved) Control Panel Logout

File Edit View Insert Cell Kernel Help Python 3

Code Cell Toolbar: None

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]:
```
# Alaska (for preprocessing common mode error removal)
lat_range = (50,75); lon_range = (-175, -130); start_time = '2006-01-01'; end_time = '2015-06-01'
# Akutan Volcano, AK, 15 km around 54.13308 N, -165.98555 E
ap_radius = AutoParam(15); ap_geopoint = AutoParam((54.13308, -165.98555))
PBO_data_fetcher = PBO_DF(start_time, end_time, lat_range, lon_range,
                          [ap_radius, ap_geopoint],mdyratio=.7,stabFlag=0)
```

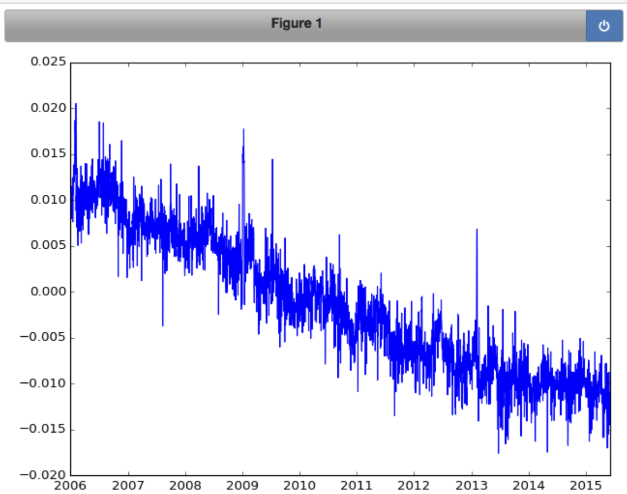
In [5]:
```
PBO_data = PBO_data_fetcher.output().get() # returns a pandas data panel
```

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

Out[6]:

	AV06	AV07	AV08	AV10	AV12	AV13	AV14
2006-01-01	0.00945	0.01176	0.00783	0.01702	-0.00152	0.01230	0.01235
2006-01-02	0.01064	0.01294	0.00769	0.01670	-0.00040	0.01362	0.01397
2006-01-03	0.01108	0.01318	0.00882	0.01849	0.00086	0.01296	0.01485
2006-01-04	0.00803	0.01211	0.00618	0.02182	-0.00054	0.01136	0.01393
2006-01-05	0.01132	0.01406	0.00678	0.01728	-0.00010	0.01642	0.01433

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



5.4 skdaccess.geo.grace

Demo_Grace Last Checkpoint: 6 hours ago (unsaved changes)

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.framework.param_class import *
```

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

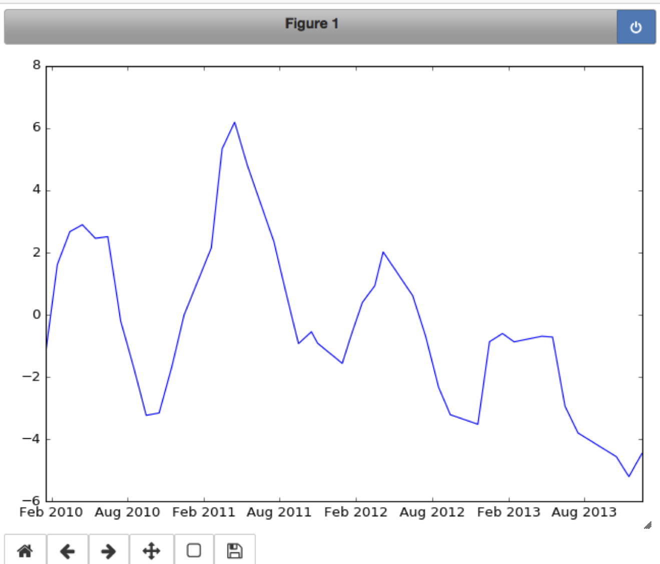
In [5]:
```
grace_data = grace_fetcher.output().get() # returns a pandas data panel
```

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

Out[6]:

	Grace	Uncertainty
2010-01-17	-1.340180	2.412851
2010-02-15	1.612133	2.412851
2010-03-17	2.670002	2.412851
2010-04-16	2.894872	2.412851
2010-05-17	2.458127	2.412851

In [7]:
```
plt.figure();
plt.plot(grace_data['Grace']);
plt.tight_layout()
```



Acknowledgements

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