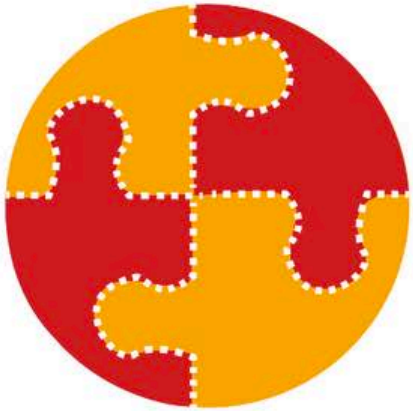
Data Science with Scikit Data Access and Scikit Discovery

Cody Rude, Guillaume Rongier, and Victor Pankratius

NASA AIST14-NNX15AG84G, NASA AIST16-80NSSC17K0125, NSF ACI-1442997, and NSF AGS-1343967

Outline

- Scikit Data Access
  - Overview
  - Example with Magnetometer Data
- Scikit Discovery
  - Overview
  - Akutan Volcano Example



SCIKIT-data access

DATA INTERFACES FOR PYTHON

- Import scientific data from various sources through one easy Python API

- Write your own "DataFetcher.py" and have it added to the package

- Use iterator patterns for each data source (configurable data fetchers + wrappers to get next data chunk)
- Skip parser programming and file format handling
- Enjoy a common namespace for all data and make data fusion simpler

- 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
- Open source (MIT License)

Example of Geomagnetic data from magnetic observatories operated by the U.S. Geological Survey (geomag.usgs.gov)

Initial Imports

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

Initialize Data Fetcher

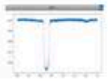
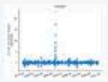
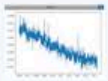
In [3]:
```
# Defining search parameters
station_list = AutoList(['BOU'])

# Fetch data
geomag_df = Geomag_DF([station_list],
                      start_time='2015-11-01',
                      end_time='2015-11-02')

geomag_dw = geomag_df.output()
```

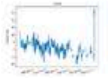
Access the data using an iterator interface

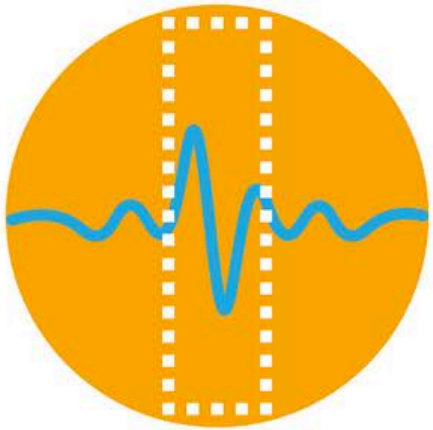
In [4]:
```
# Data can be accessed one data frame at a time
label, data = next(geomag_dw.getIterator())
data.head()
```

Out[4]:

	X	Y	Z	F
2015-11-01 00:00:00	20584.156	3245.273	47363.740	52279.830
2015-11-01 00:01:00	20584.149	3245.212	47363.729	52279.810
2015-11-01 00:02:00	20584.098	3245.271	47363.666	52279.739
2015-11-01 00:03:00	20584.003	3245.548	47363.608	52279.673
2015-11-01 00:04:00	20583.988	3246.008	47363.570	52279.649

Namespace	Description	Preview	Data Source
astro.kepler	Light curves for stars imaged by the NASA Kepler Space Telescope		https://keplerscience.arc.nasa.gov
astro.voyager	Data from the Voyager mission		https://spdf.gsfc.nasa.gov/
geo.groundwater	USGS United States groundwater monitoring wells measuring the depth to water level		https://waterservices.usgs.gov
geo.pbo	UNAVCO EarthScope - Plate Boundary Observatory (PBO): Daily GPS displacement time series measurements throughout the United States		http://www.unavco.org/projects/major-projects/pbo/pbo.html

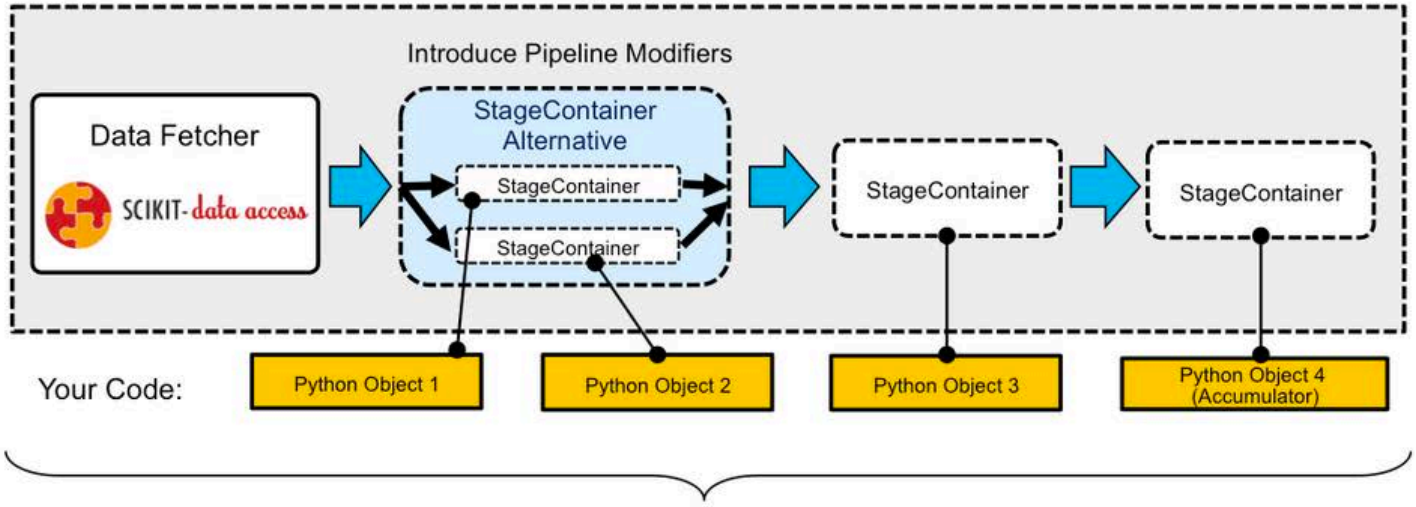
geo.mahali.rinex	Rinex files from the MIT led NSF project studying the Earth's ionosphere with GPS		http://mahali.mit.edu
geo.mahali.tec	Total Electron Content from the MIT led NSF project studying the Earth's ionosphere with GPS		http://mahali.mit.edu
geo.mahali.temperature	Temperature data from the MIT led NSF project studying the Earth's ionosphere with GPS		http://mahali.mit.edu
solar.sdo	Images from the Solar Dynamics Observatory		https://sdo.gsfc.nasa.gov/



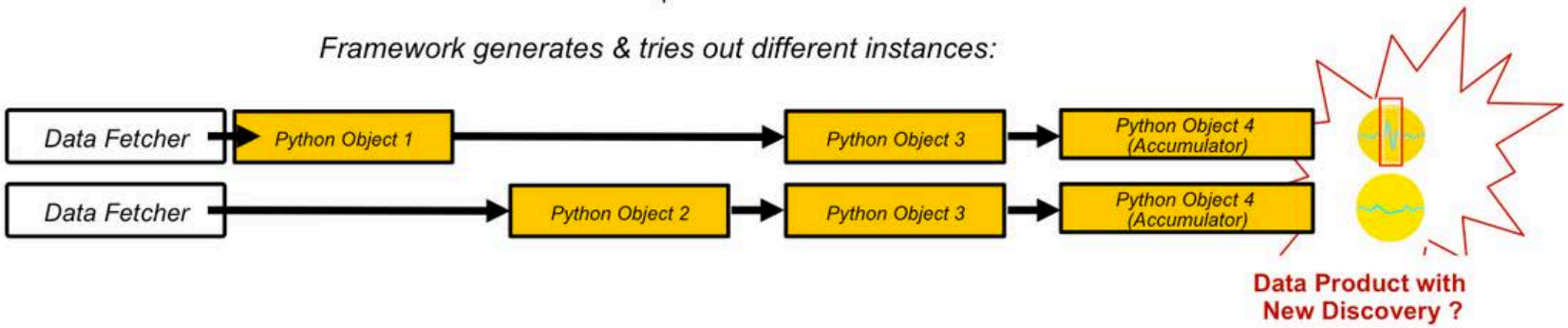
SCIKIT-discovery

PYTHON TOOLKIT FOR COMPUTER-AIDED DISCOVERY

Define Data Discovery Pipeline Possibilities with a Synoptic Model



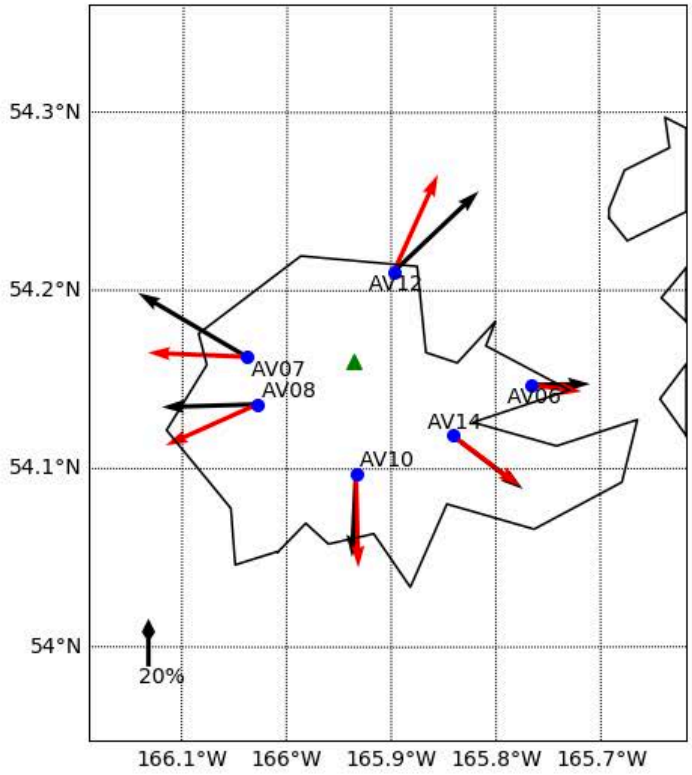
Framework generates & tries out different instances:



Create a pipeline that models deformation motion using an expanding magma chamber

In [16]:
```
pipeline_model_plot = DiscoveryPipeline(pbodf, [sc_stab, sc_geo, sc_station, sc_tf,
                                                sc_mf, sc_gca, sc_model,
                                                sc_model_plot])

pipeline_model_plot.plotPipelineInstance()
```

In [17]:
```
# Run mogi pipeline
pipeline_model_plot.run()
```



The different stages in the pipeline can automatically be changed

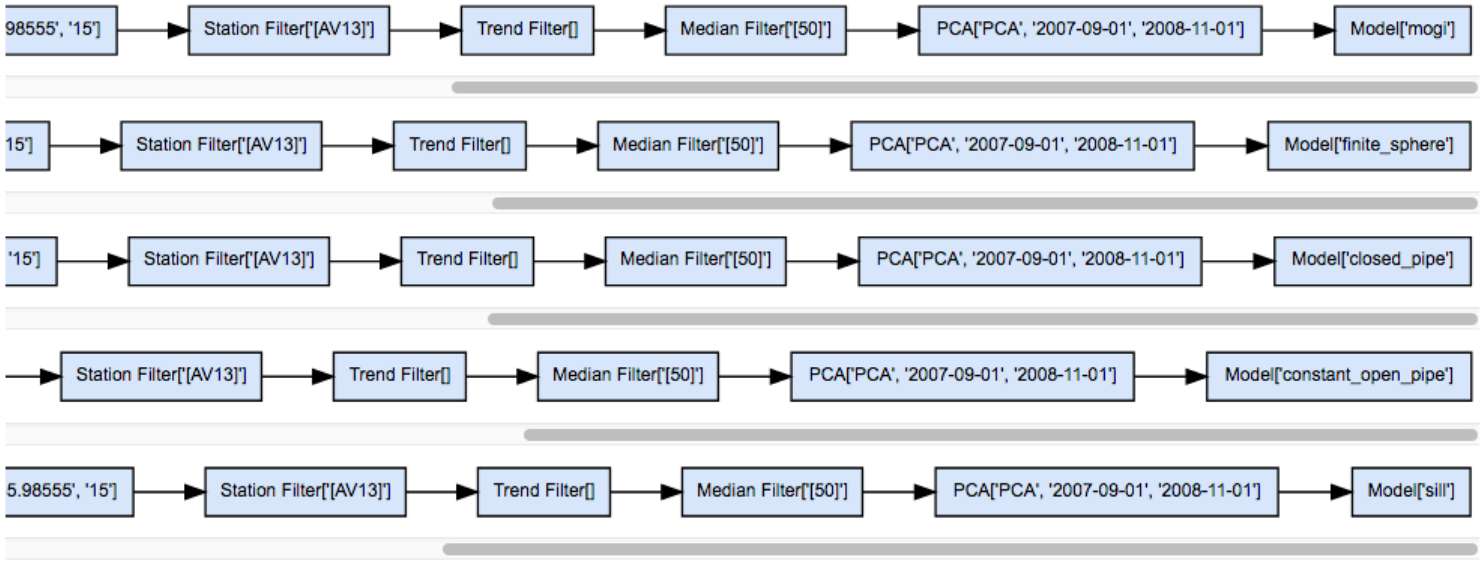
Create a item that can cycle through different models

In [18]:
```
# Multiple source models estimation
# Set name (location) of PCA results
pca_results = 'PCA'
# Set model types
ap_mogi_model = AutoParamListCycle(('mogi','finite_sphere','closed_pipe',
                                    'constant_open_pipe','sill'))
# Create mogi analysis item
ana_model_cycle = Mogi_Inversion('Model', [ap_mogi_model],
                            pca_name = pca_results)
# Create stagecontainer for mogi model
sc_model_cycle = StageContainer(ana_model_cycle)
```

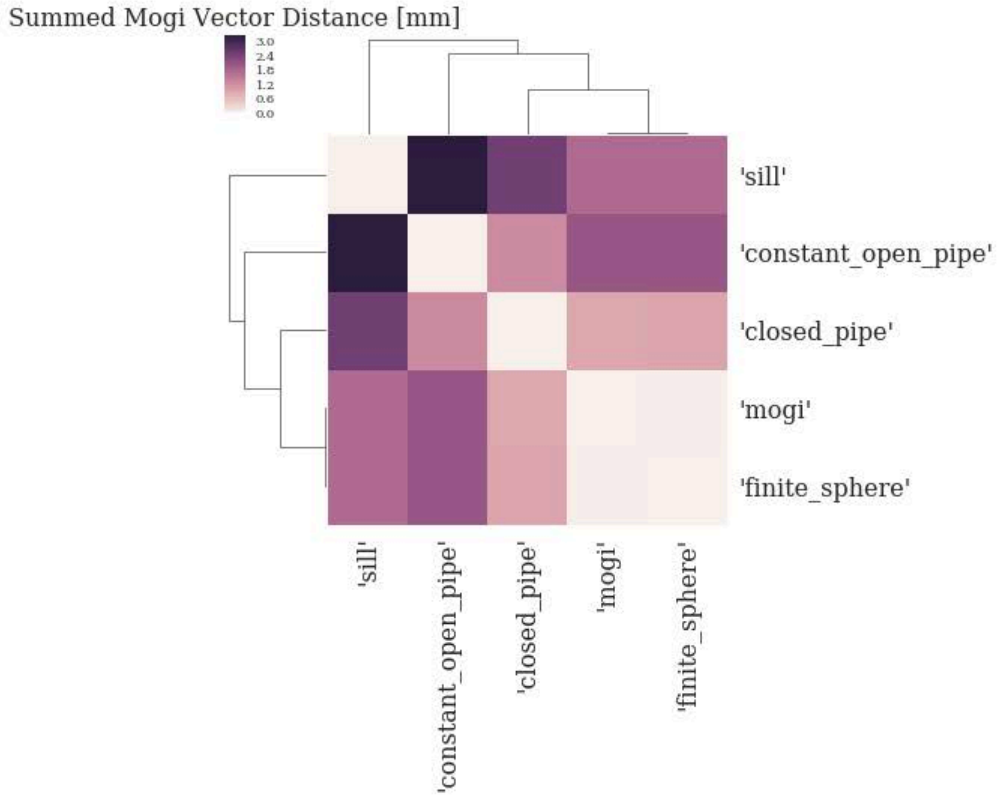
In [19]:
```
pipeline_model = DiscoveryPipeline(pbodf, [sc_stab, sc_geo, sc_station, sc_tf,
                                           sc_mf, sc_gca, sc_model_cycle])
pipeline_model.plotPipelineInstance()
```

Run the pipeline 5 times

In [20]:
```
pipeline_model.reset()
pipeline_model.run(num_runs=5, verbose=True)
```



In [21]:
```
calc_distance_map(pipeline_model,'Model','Model','MogiVector',plotFlag=True,
                  fontsize=16);
plt.tick_params('y', labelsize='small')
plt.gcf().set_size_inches(6,6)
```



Software available at:

- https://github.com/MITHaystack/scikit-dataaccess
- https://github.com/MITHaystack/scikit-discovery

Scientific case studies located at:

- https://github.com/MITHaystack/science-casestudies

Thank you!

Questions?

NASA AIST14-NNX15AG84G, NASA AIST16-80NSSC17K0125, NSF ACI-1442997, and NSF AGS-1343967