Computer-Aided Discovery:
Insight Generation with Machine Support

NASA GSFC, 4/19/2017

Victor Pankratius

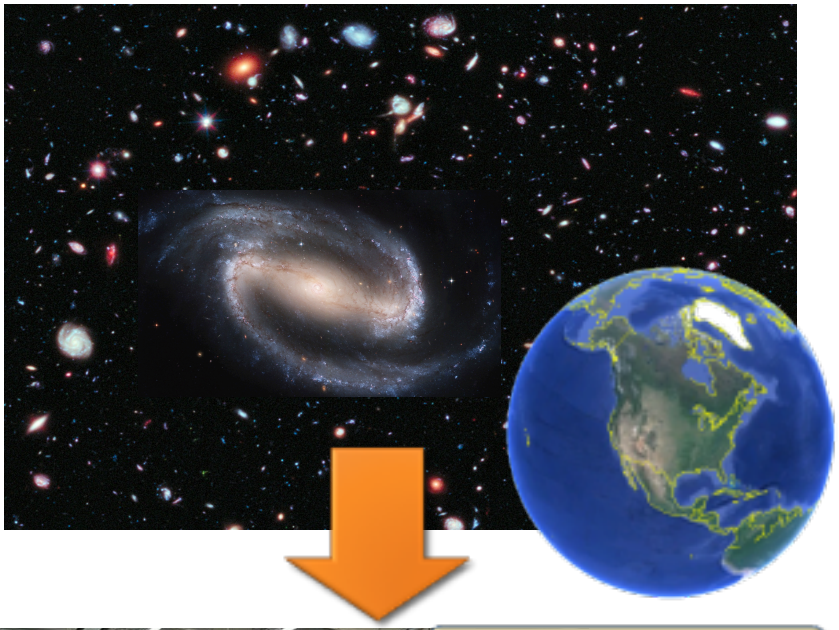
Head of Astro-&Geo-Informatics Group
Massachusetts Institute of Technology
Haystack Observatory

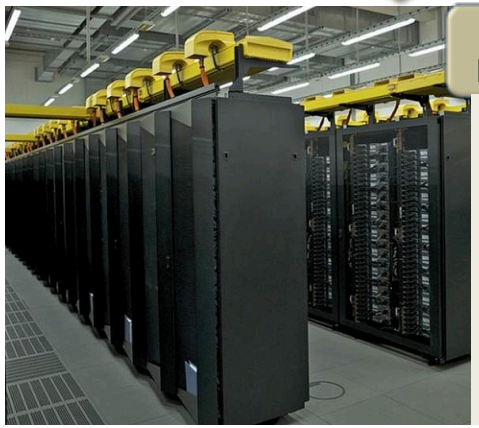
Email: pankrat@mit.edu
Web: victorpankratius.com

NASA AIST NNX15AG84G
V. Pankratius (PI), T. Herring (co-I)

Data – all the time – everywhere on earth and in space

- Scalable machine assistance is needed to help humans in the discovery process
- Overcome human cognitive limits through algorithmic support
  - The scientific discovery process becomes a search process across multidimensional data sets. Scientific question answering by matching theory variants to empirical data sets.





Software-based Instruments / Backends

- Algorithms
- Parallel Computing
- Search, Classification
- Signal Processing
- Imaging
- Simulations
- Software Engineering
- Data Mining

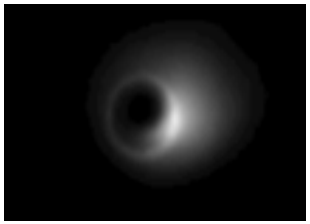
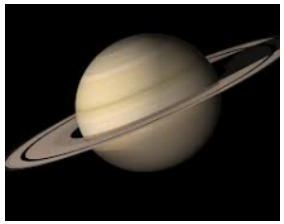
Computer-Aided Discovery

Challenge

"Pragmatics"	Discovery and Insight Generation	• What does feature X imply? • How does it fit into the theoretical context? • Does it contradict or confirm established models?
"Semantics"	Data Exploration, Analysis, Mining	"This is feature X"
"Syntax"	Data	Numbers correctly encode actual measurements
	Signals	

Semiotic Layers

Real-World Phenomena

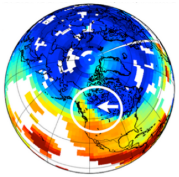
ACI, PI Pankratius
1442997

Computer-Aided Discovery: Overview

Empirical Data

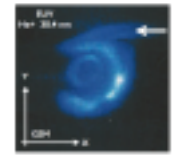
Theory

Instrument 1



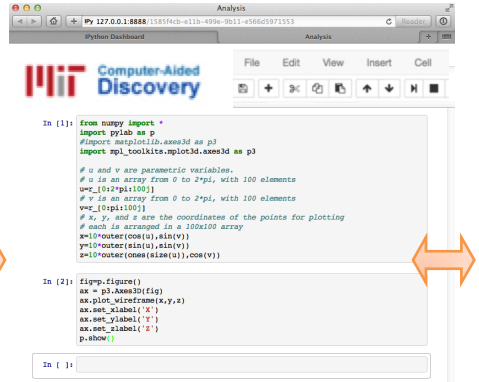
"physics"

Instrument 2



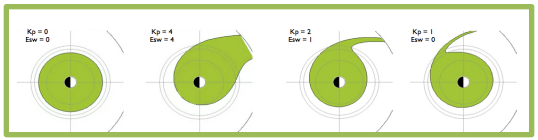
Data /Compute Center

Cloud-Based Analysis Interface



Artificial Intelligence

Scientist

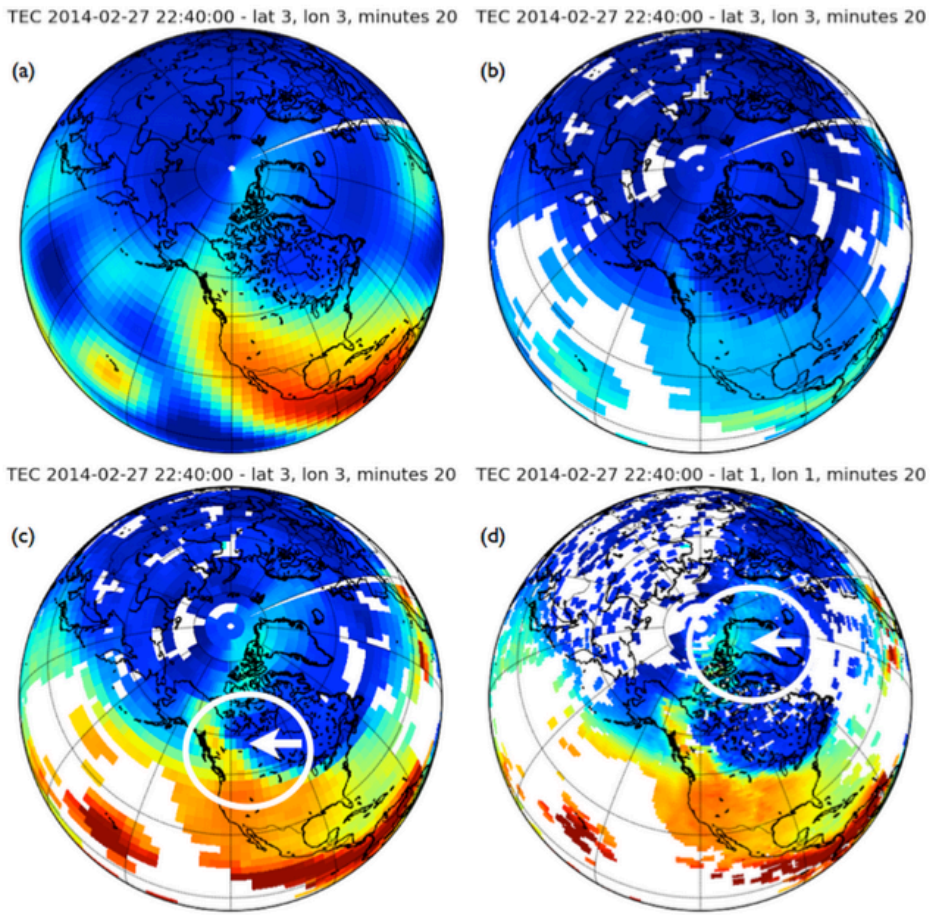


Model 1 Model 2 Model 3 Model 4

Search results – would you like to find more like:

Gather scientist's feedback

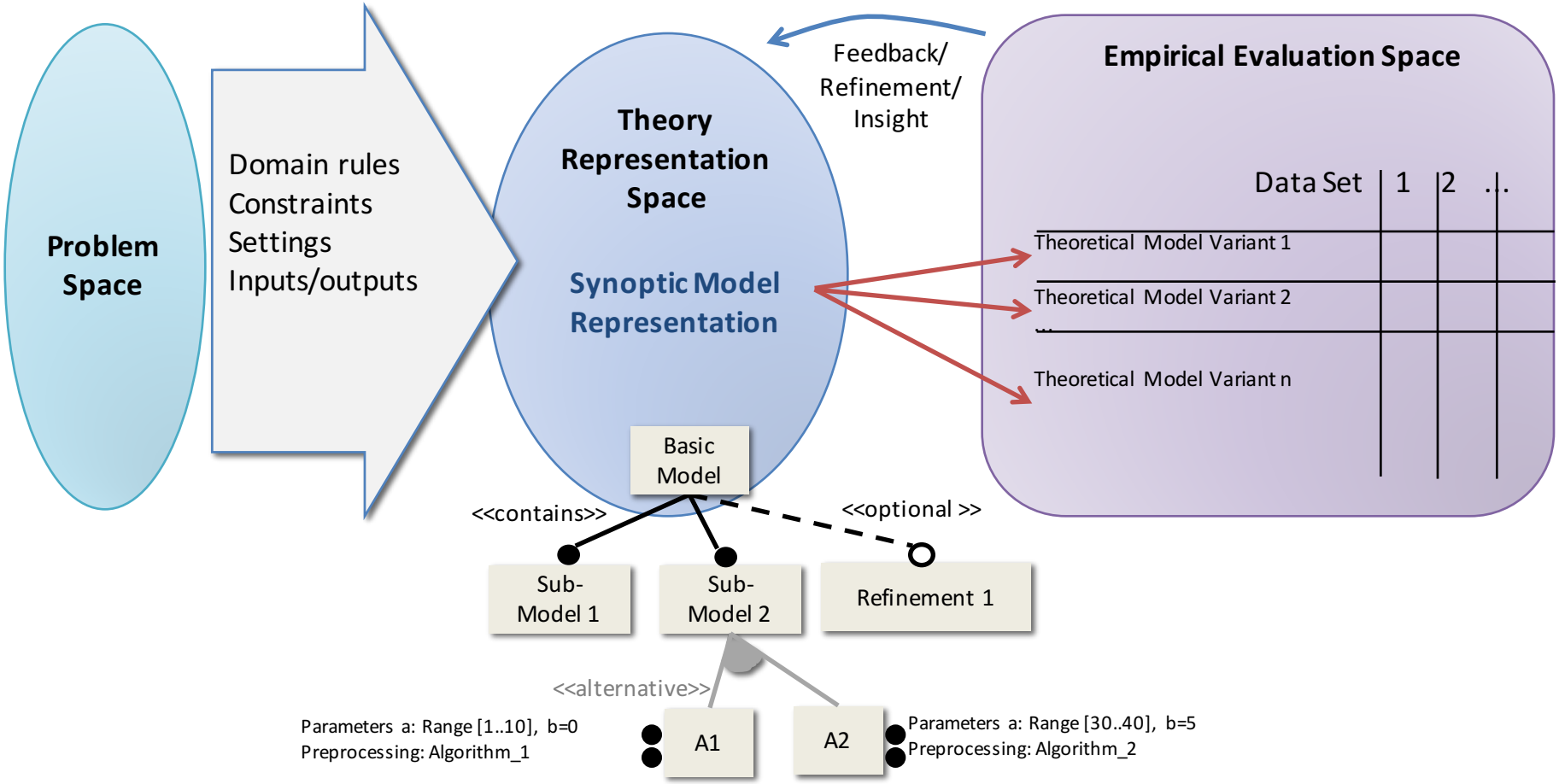
Example: Impact of Choice of Processing Workflows



Figures showing geophysical phenomena (global TEC views) over the northern hemisphere on 2014-02-27

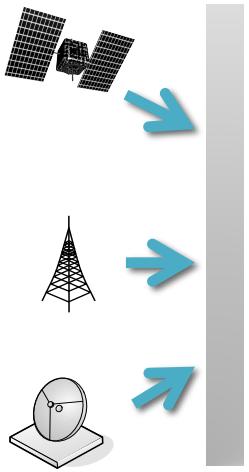
- Choice of workflow processing parameters can affect the visibility and discovery of interesting phenomena

Computer-Aided Discovery: Conceptual Approach



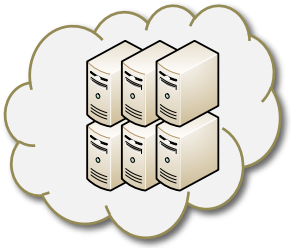
Computer-Aided Discovery: Infrastructure

"Sensors"	Data & Compute Cloud	Service Layer	Cloud-Based Analysis Interface	Scientist



Data Integrators, indexing, etc.

- Generate compute jobs
- Move computation to data locations

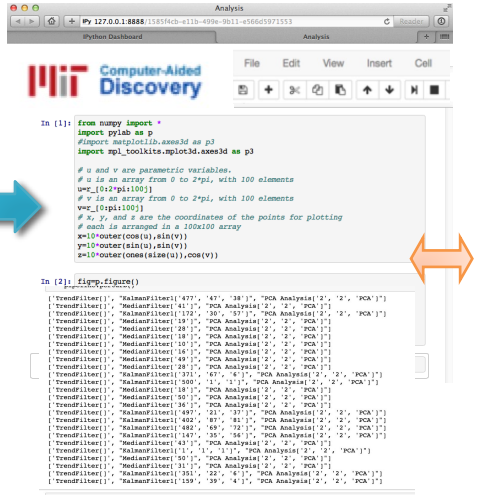


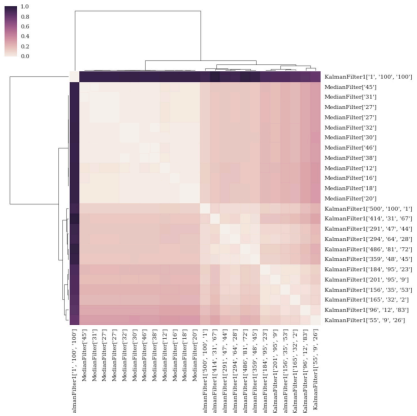
Execution Crawlers

- Computer-Aided Discovery Engine

Web Service APIs

Direct service integration into other tools

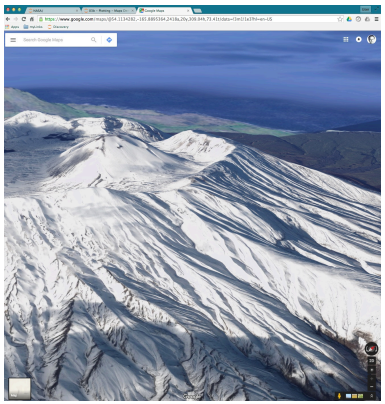


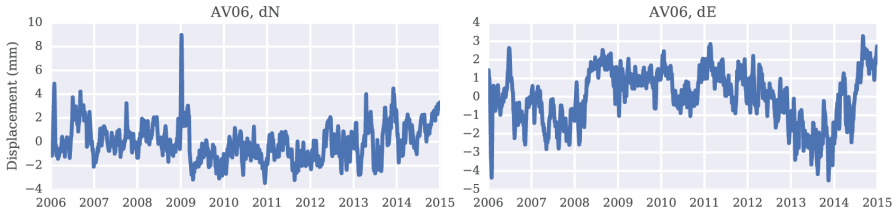


Massachusetts Institute of Technology

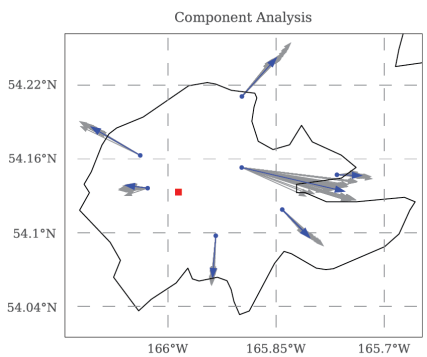
Application Examples

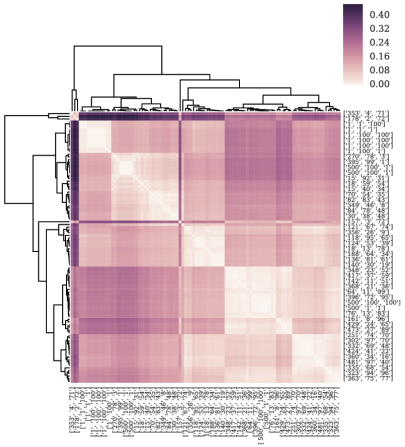
Geoscience / Volcanics



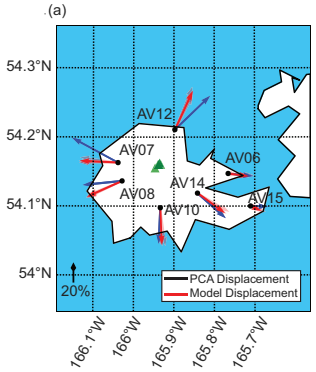


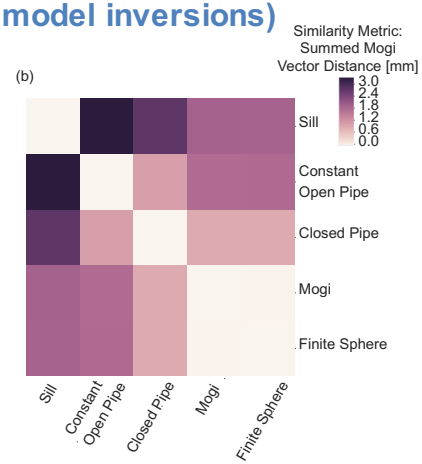
Variants of earth deformation models





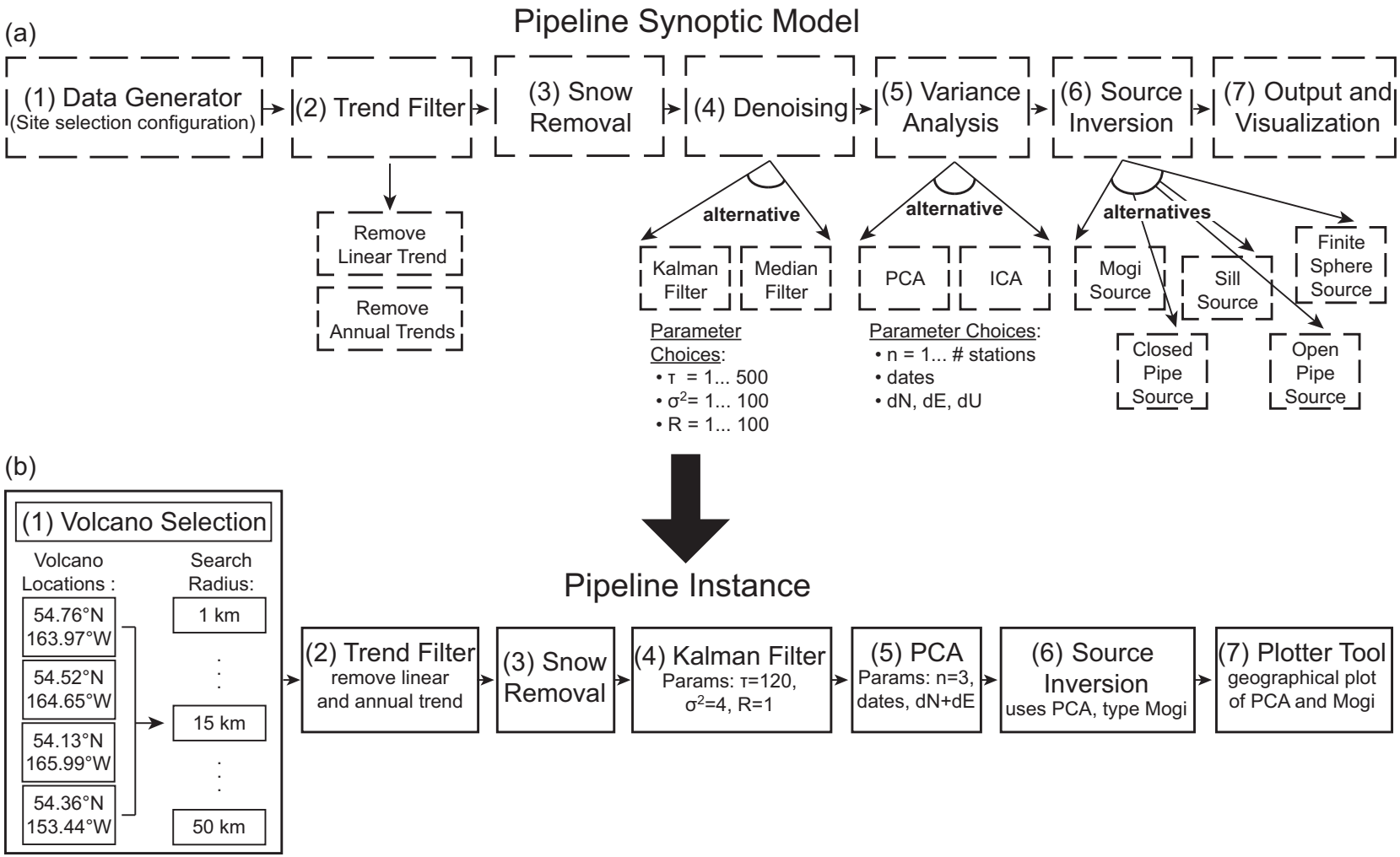
Variants of magma source models (5 different source model inversions)





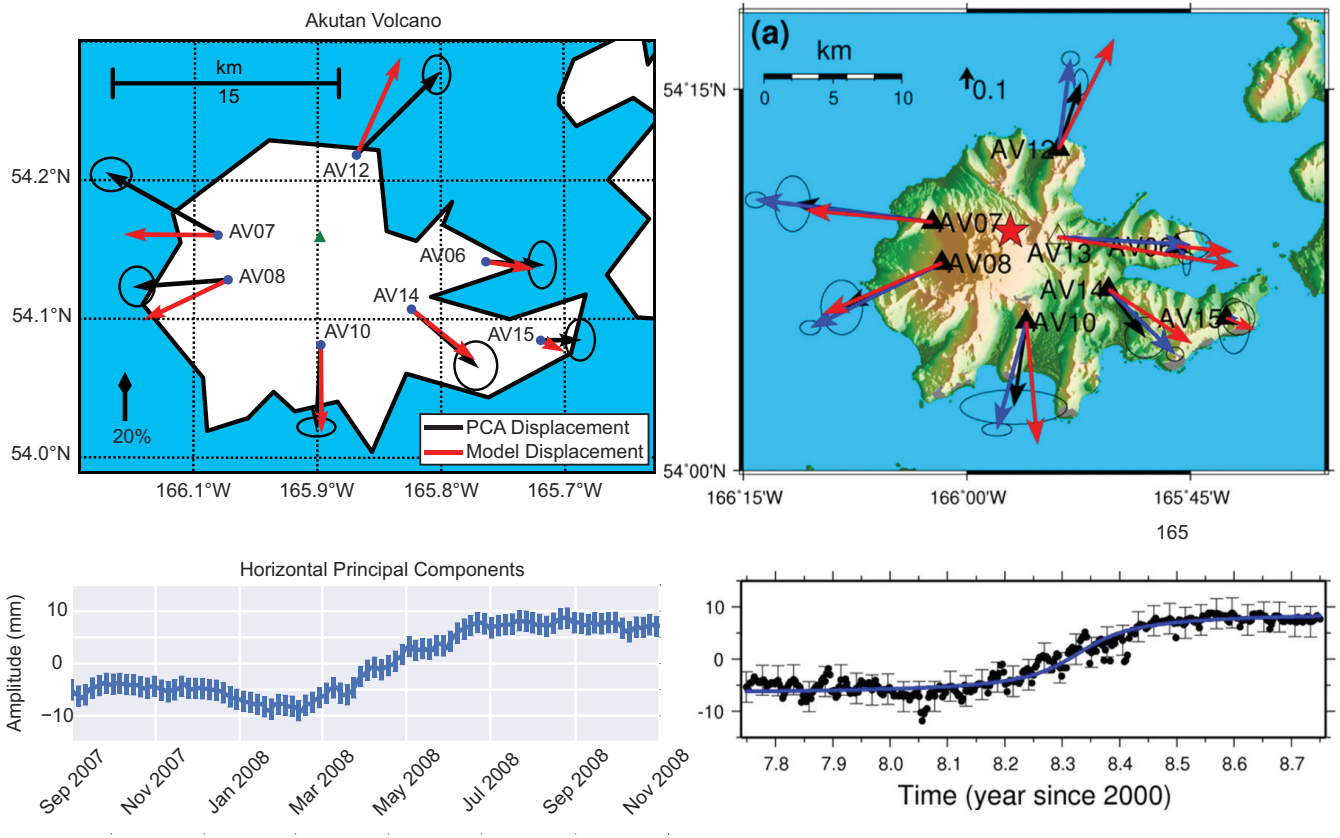
[J.Li, C.Rude, D.Blair, M.Gowanlock, T.Herring, V.Pankratius. Journal of Volcanology and Geothermal Research, 2016]

Volcanics: Event Discovery Pipeline



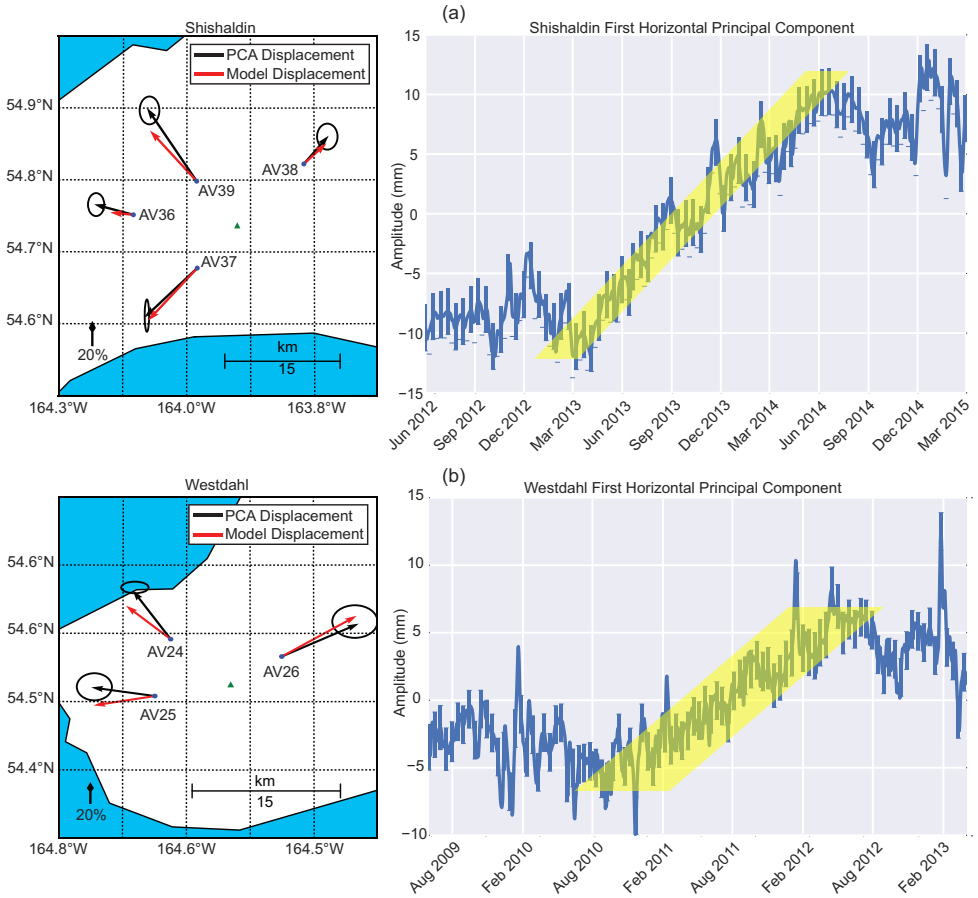
Volcanics: Validation of Methodology – Example

Comparison of our results (left) with Ji&Herring 2011 (right)



Volcanics: New Discoveries

- Using our framework, we detected two previously unreported inflation events at Shishaldin and Westdahl (highlighted in yellow)



Volcanics: Exploration of Model Configuration Space

- Framework generates multiple configurations to explore different model assumptions (e.g., noise characteristics)
- Capability for parallel execution of various configurations via Amazon Cloud
- Framework provides heatmap as exploratory visualization tool to compare models

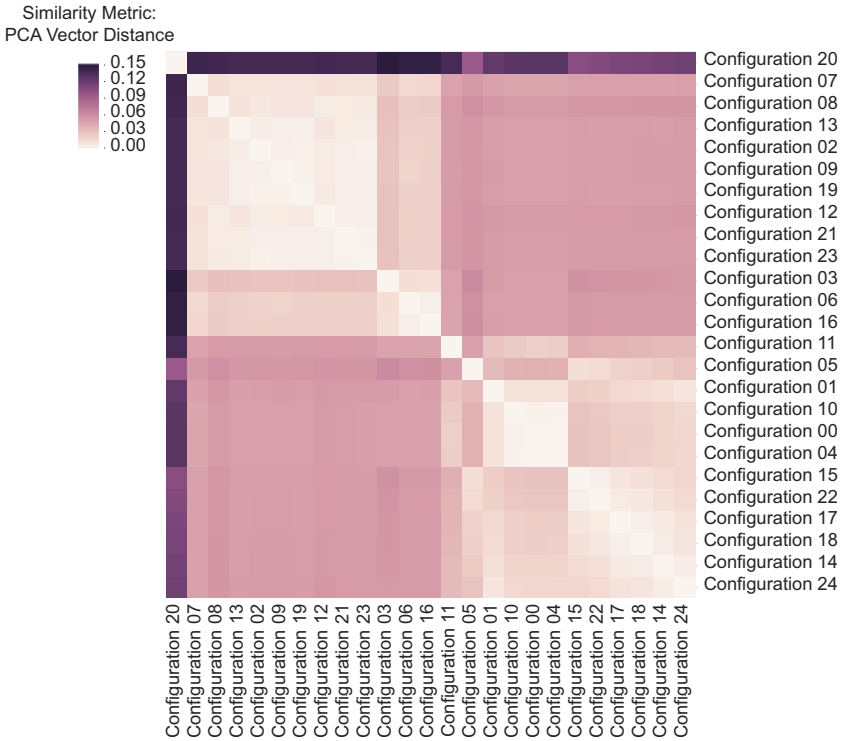


Figure 6: A heatmap showing the sum of the euclidean distance between the eigenvectors at each PBO station for each sensitivity analysis run at Akutan. Colors reflect the distance between results from different configurations as measured by the difference between PCA eigenvectors at each PBO station. Visualization of configurations provides natural groupings of potential model assumptions for the given Akutan empirical data set. The actual parameters for each configuration are provided in the Appendix

Volcanics: Framework Helps Explore Different Magma Source Models

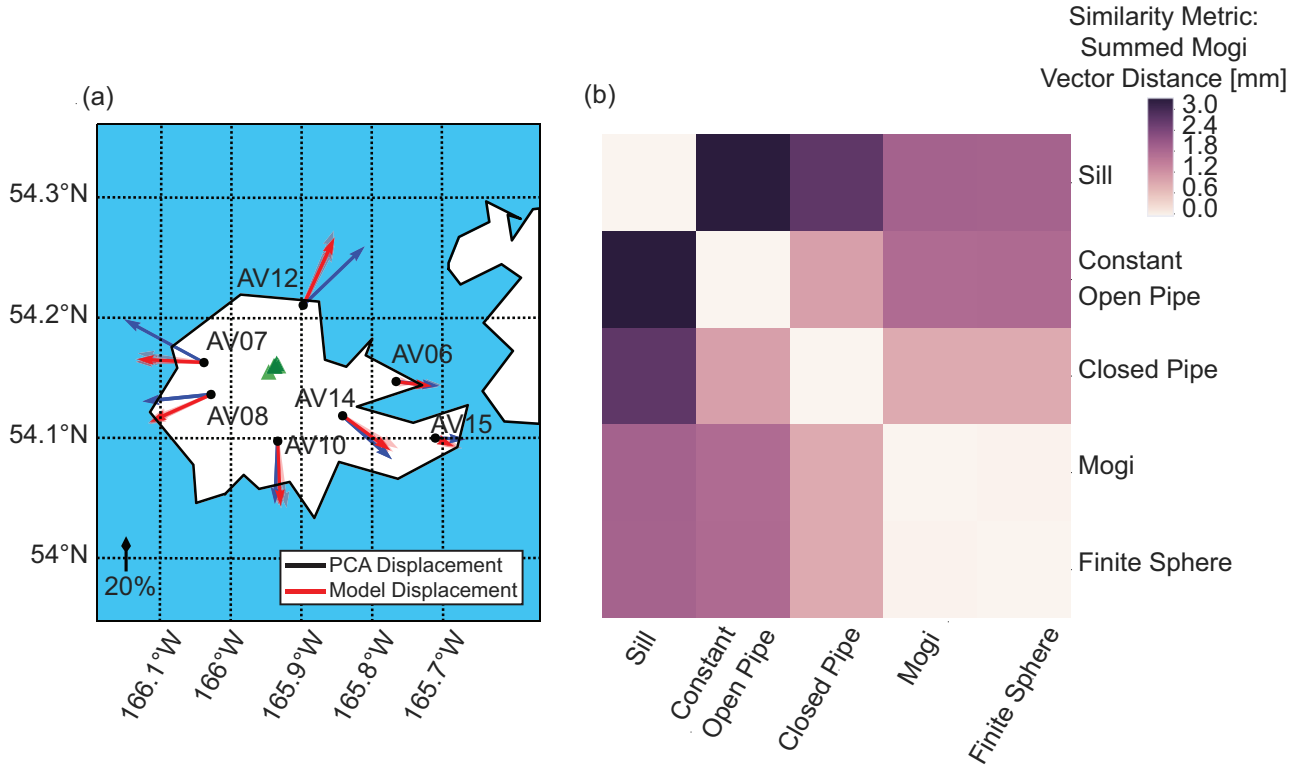
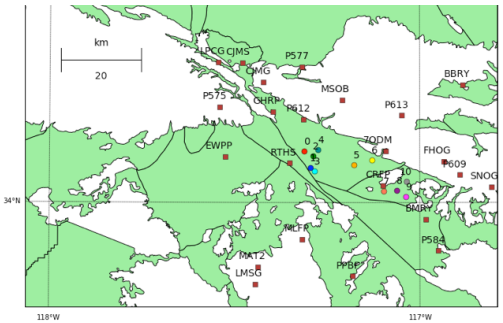


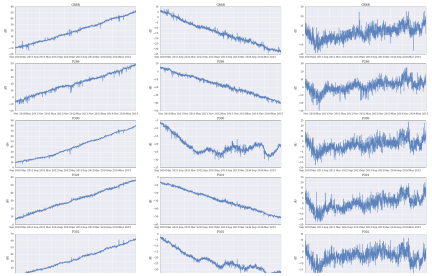
Figure 7: Comparison of 5 different source model inversions, Mogi, finite sphere, closed pipe, constant (width) open pipe, and sill, for the motion described by the eigenvectors for Akutan displayed on (a) a geographical plot with the eigenvectors in blue, the different inversion displacements in red (the solid color is the mean), and the different magma source locations in green (the solid color shows the average location). The 20% arrow indicates the scaling of the vectors relative to the respective first horizontal PCA component's amplitude.(b) A heatmap that compares the summed Euclidean distance [mm] between model displacement vectors at each GPS station for different model sources

Application Examples

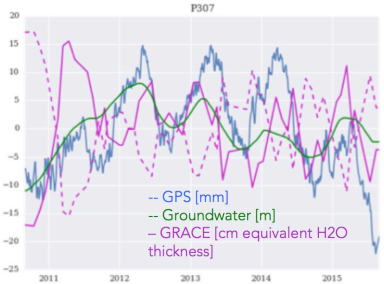
Groundwater Studies

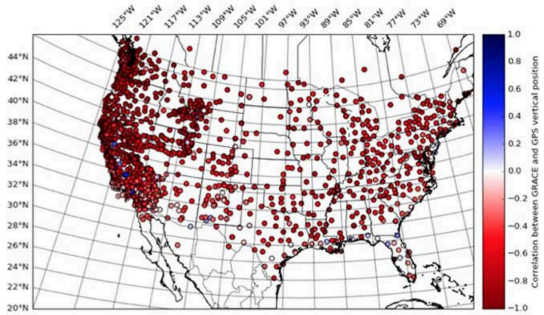


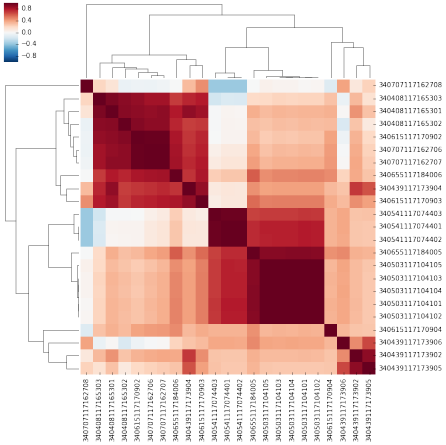
GPS (North, East, vertical)



+ Data Fusion: GRACE,...







- Group wells by behavior
- Red groups represent wells whose water levels are highly correlated in time

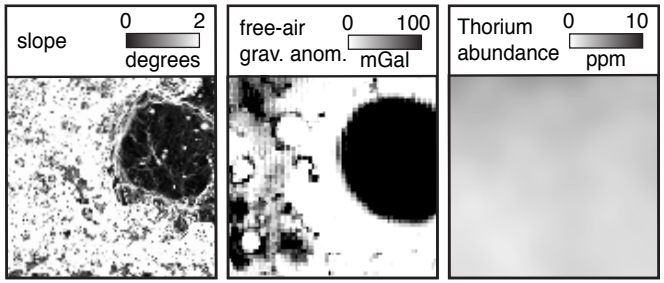
[C.Rude, J.Li, M.Gowanlock, T.Herring, V.Pankratius, AGU 2016]

MIT Haystack Observatory
Massachusetts Institute of Technology

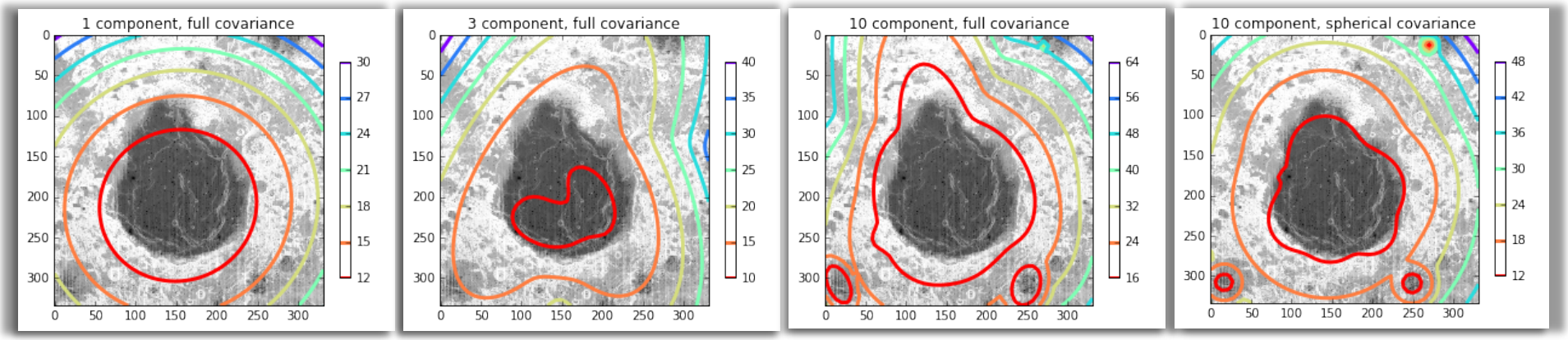
Application Examples

Planetary Science / Site Selection: Moon

Moon Example:



Variants of combined site selection models



Plotted Over Overall Rank (darker = better)

[D.Blair, M.Gowanlock, J.Li, C.Rude, T.Herring, V.Pankratius, LPSC 2016]

Computer-Aided Discovery: Spacecraft Site Selection

- Suitable landing & exploration sites needed for both human and robotic interplanetary missions, asteroids, etc.
- On a given body, there are many possible site candidates - how to choose?
- Want an optimal choice in terms of both science and engineering goals & constraints

Mission Architecture:

- Flat enough for low-risk landing
- Efficiently reachable from Earth
- Within rover range of other sites

Scientific Value:

- Outcrops or cliffs
- Signs of hydrologic activity
- Volcanic features
- Impact craters

Operating Conditions:

- Temperature
- Topography
- Sunlight availability
- Communications availability

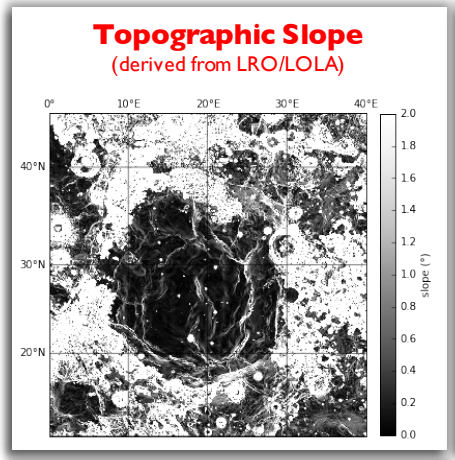
Planetary Protection:

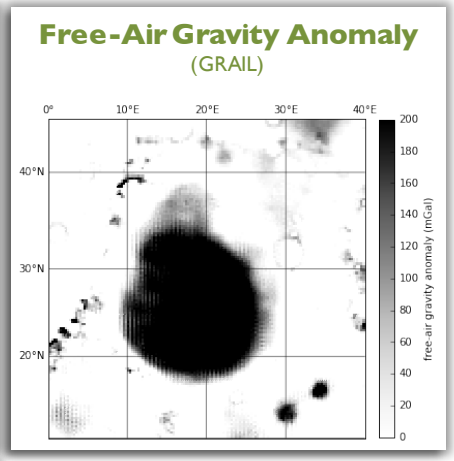
- Avoid possible present-day biomes
- Ensure non-harmful end-of-life equipment locations

Computer-Aided Discovery: Site Selection

Step 1: Data Fusion

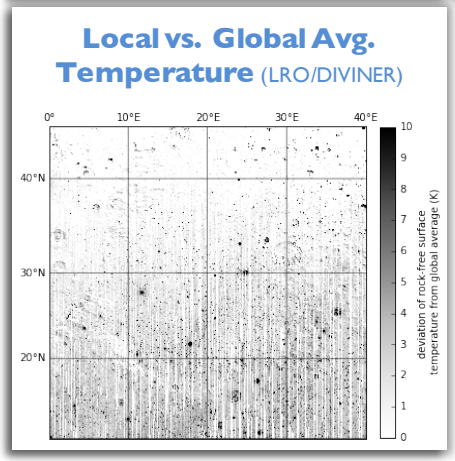
flatter = better (safer landing)



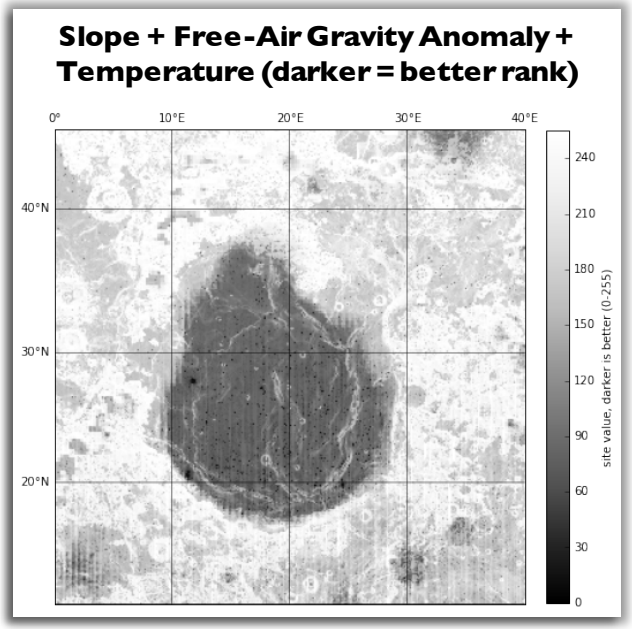


higher = better (thinner crust)

higher = better (more cohesive rock)

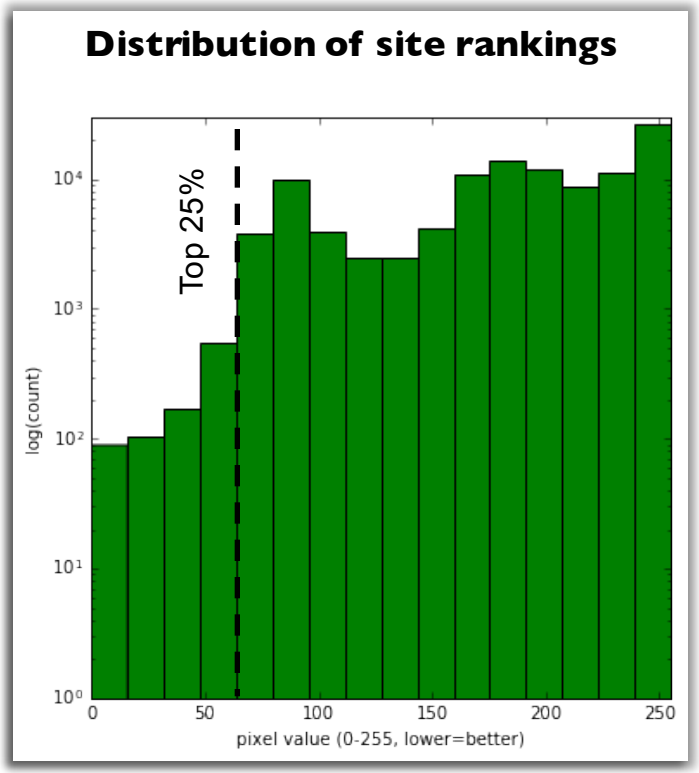


combination with all three data sets weighted equally

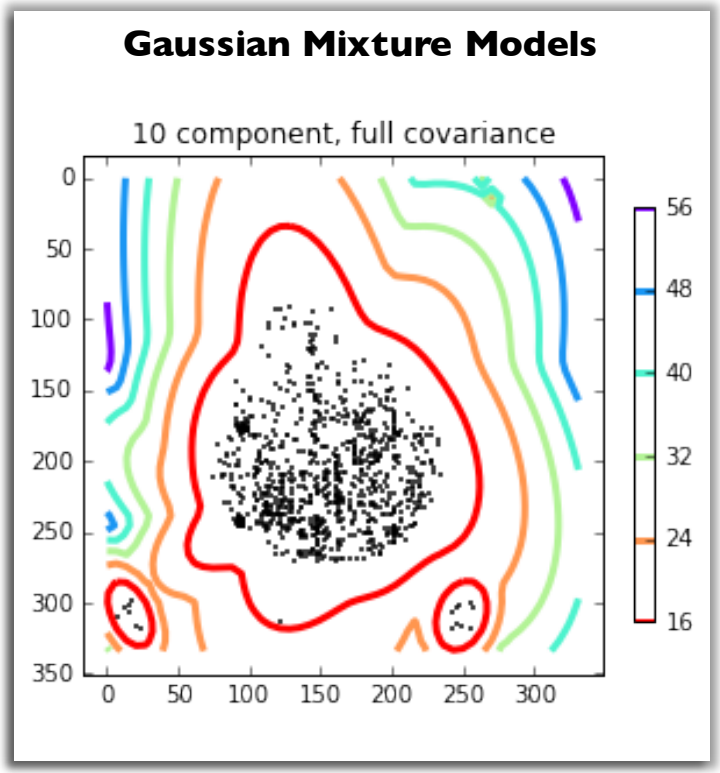


Computer-Aided Discovery: Site Selection

Step 2

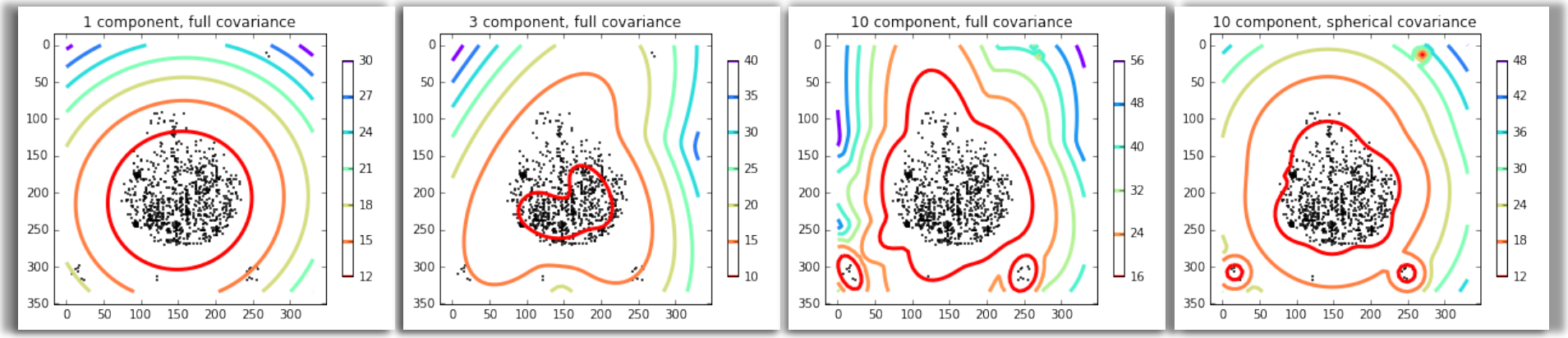


Step 3

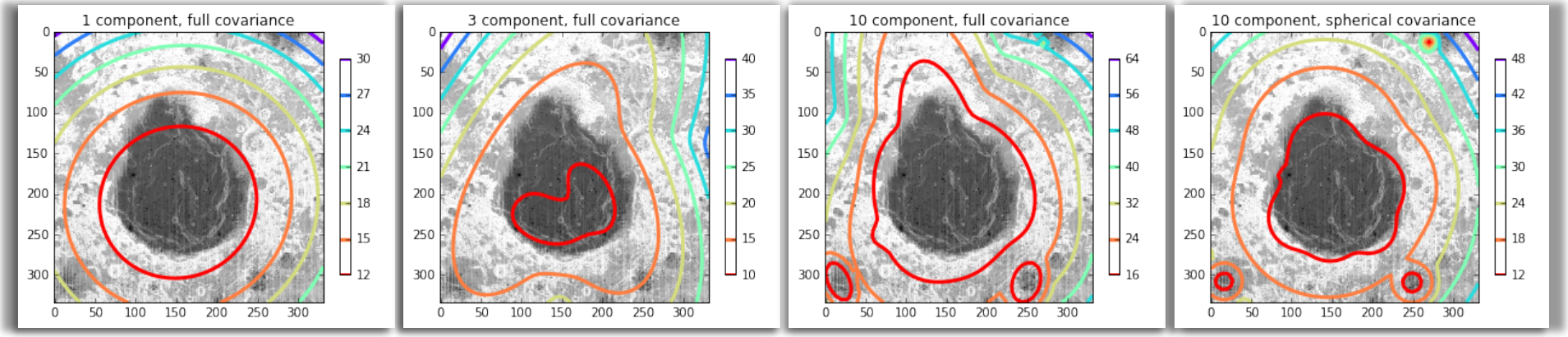


Computer-Aided Discovery: Site Selection

Plotted Over Sites in Top 25%

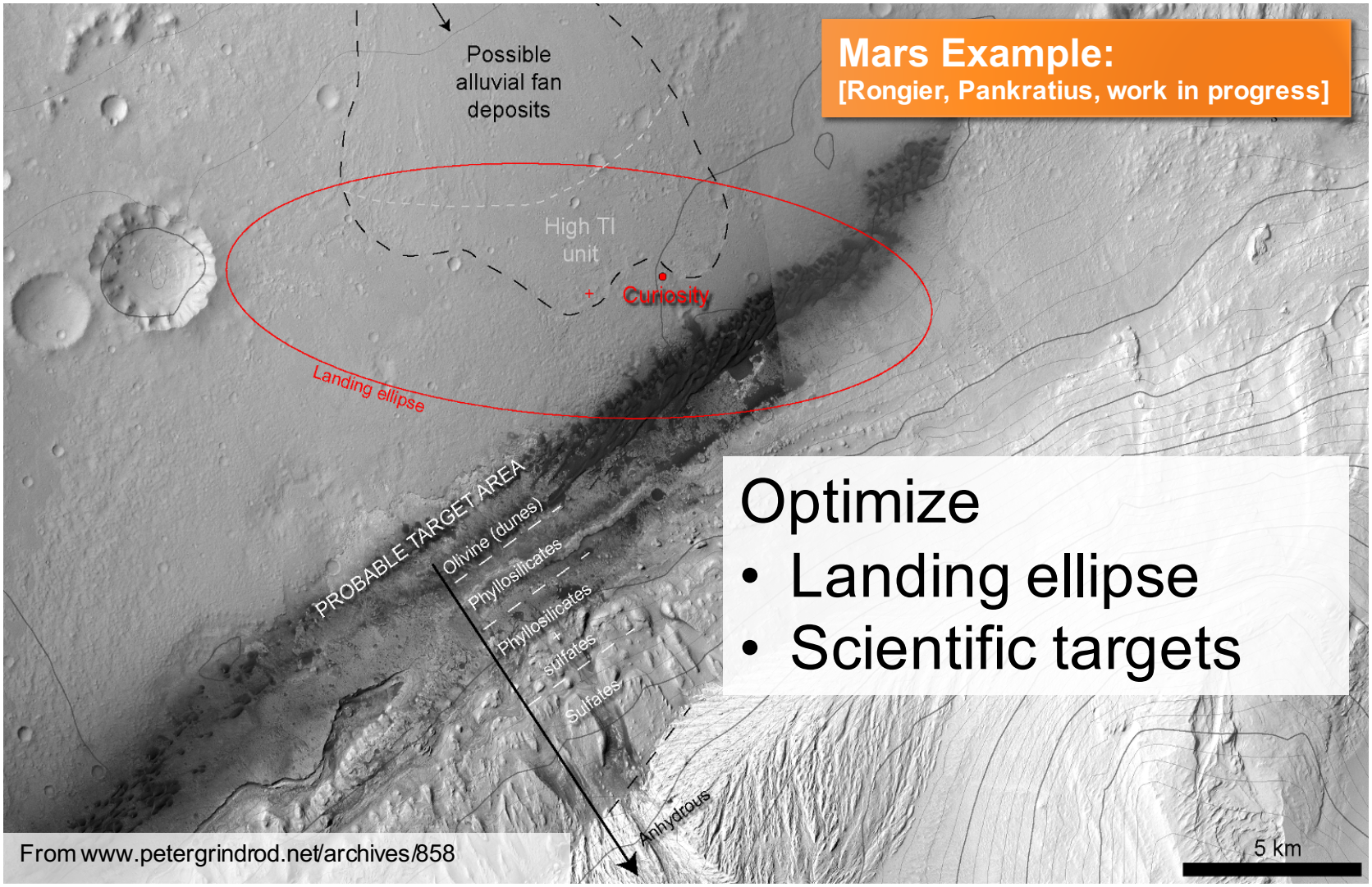


Plotted Over Overall Rank (darker = better)



Application Examples

Planetary Science / Site Selection: Mars



Mars Example:
[Rongier, Pankratius, work in progress]

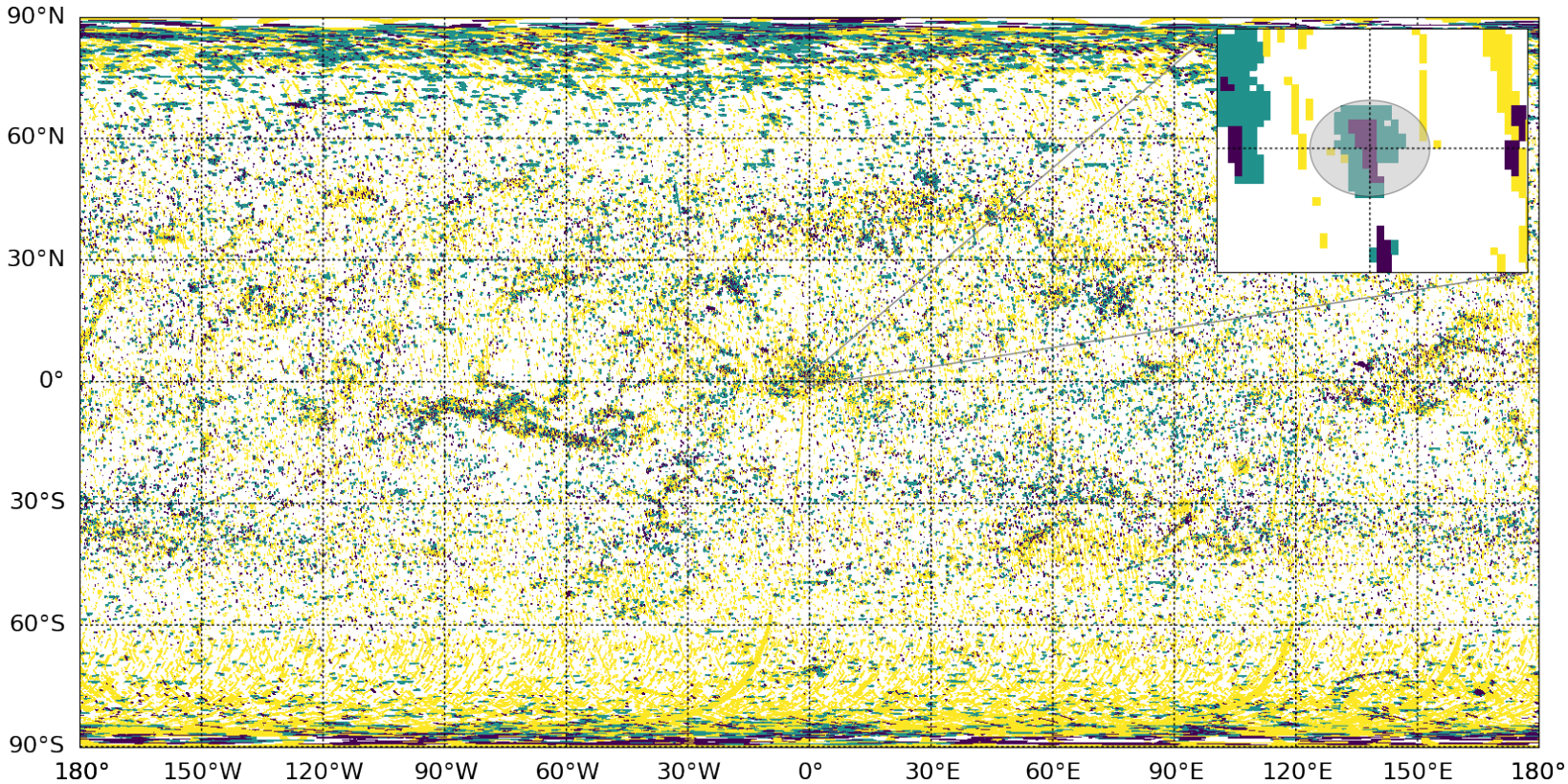
Optimize
- Landing ellipse
- Scientific targets

From www.petergrindrod.net/archives/858

Computer-Aided Discovery: Mars Site Selection

High resolution data volumes of Mars growing → how to leverage?

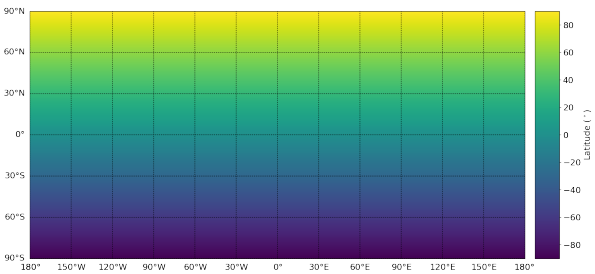
- HiRISE (camera, 0.25-0.13 m/px)
- CRISM (spectrometer, 18-36 m/px)
- MOC (camera, 1.5-12 m/px)



Computer-Aided Discovery: Mars Site Selection

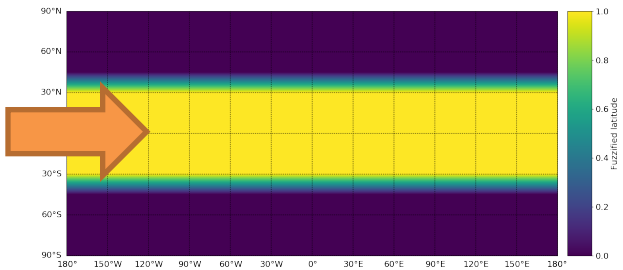
→ *use Fuzzy Logic to determine how favorable a landing site is*

Initial area



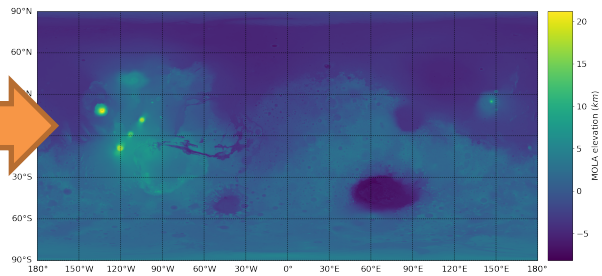
1) Area fuzzification:

1: high utility, 0: no utility

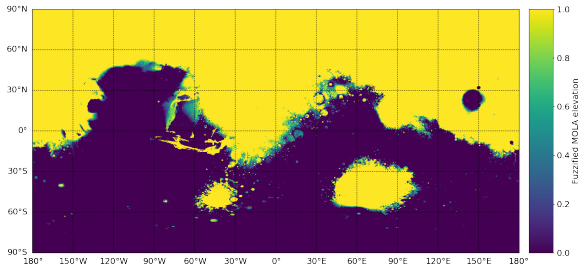


2) Criteria fuzzification

e.g., elevation

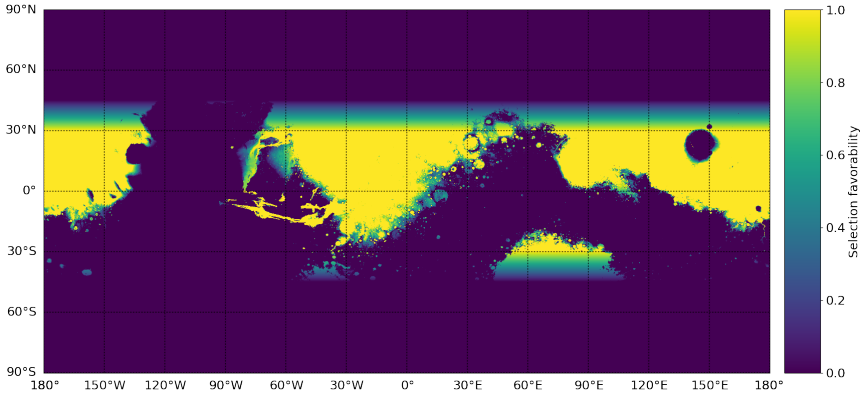


3)



Compute fuzzy OR, AND, etc.

4) Intermediate Product

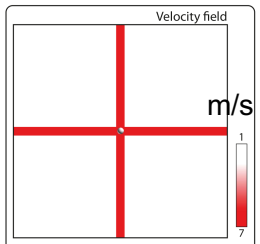


MIT Haystack Observatory
Massachusetts Institute of Technology

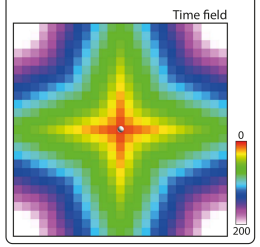
Computer-Aided Discovery: Mars Site Selection

→ now: reachability - how to get around with rover

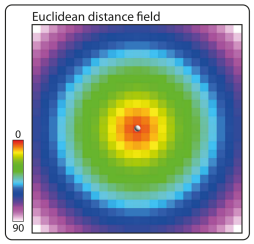
5) Fast Marching Algorithm



Define a velocity profile over space, landing site in the middle

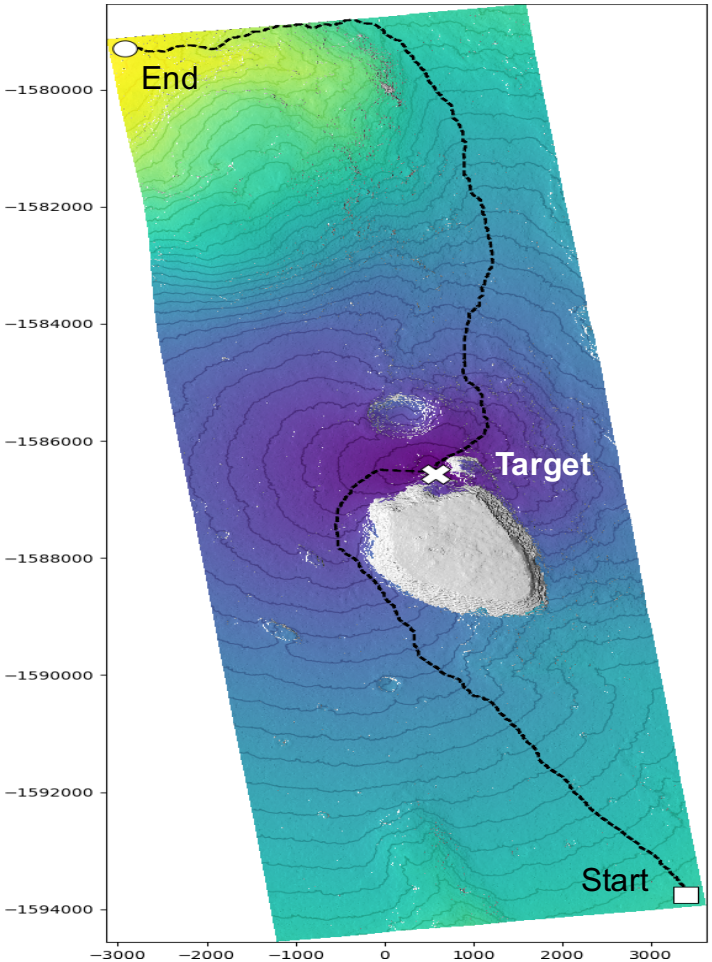


Pick a point → color = how much time to get to landing site



Pick a point → color = how "far" from landing site. Can be non-Euclidean (hills, obstacles, etc.)

Sethian, 1996: www.pnas.org/content/93/4/1591.short
Rongier et al. 2014



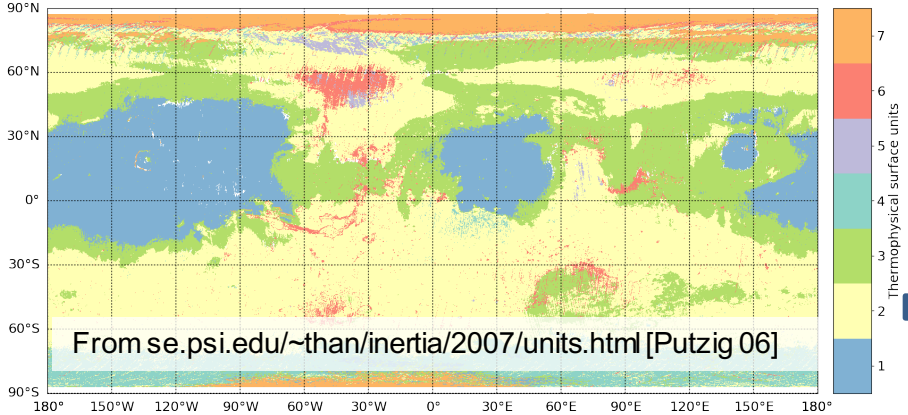
Example: HiRISE DTM
http://www.uahirise.org/dtm/dtm.php?ID=ESP_029815_1530

Computer-Aided Discovery: Mars Site Selection

→ *now: reachability - how to get around with rover*

6a) Terrain Classification

(based on thermal inertia and albedo)



1 - Bright unconsolidated fines

2 - Sand, rocks, and bedrock; some duricrust

3 - Duricrust; some sand, rocks and bedrock

4 - Low density mantle or dark dust?

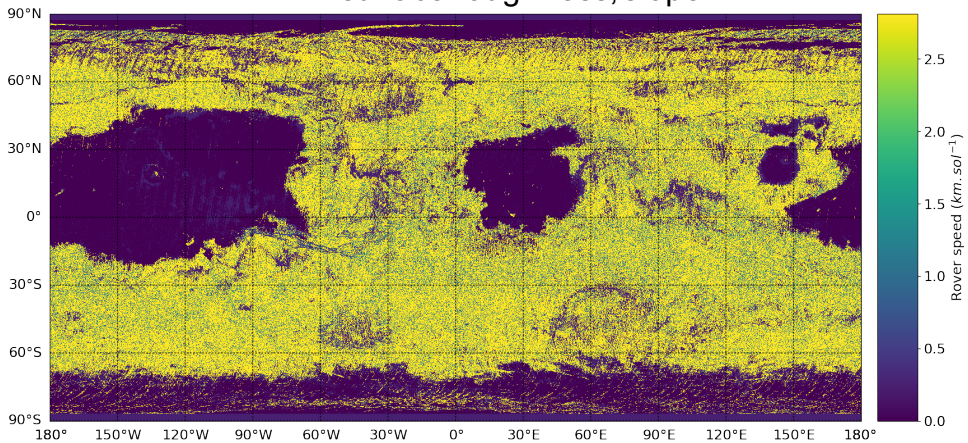
5 - As 2, but little or no fines

6 - Rocks, bedrock, duricrust, and polar ice

7 - As 1, thermally thin at higher inertia

6b) Rover speed

depending on terrain, surface roughness, slope



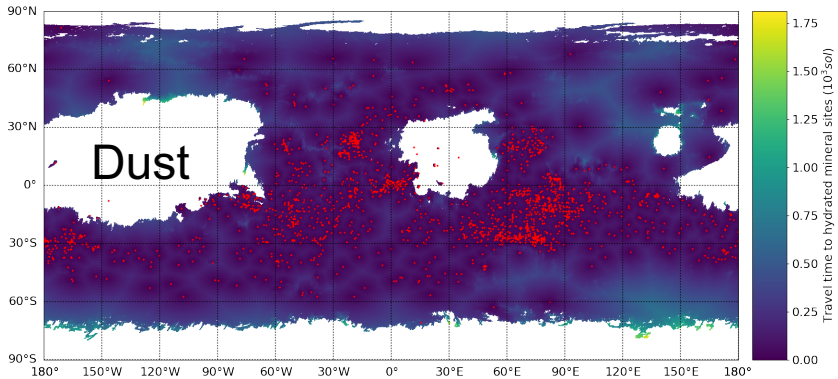
MIT Haystack Observatory
Massachusetts Institute of Technology

Computer-Aided Discovery: Mars Site Selection

→ *now: reachability - how to get around with rover*

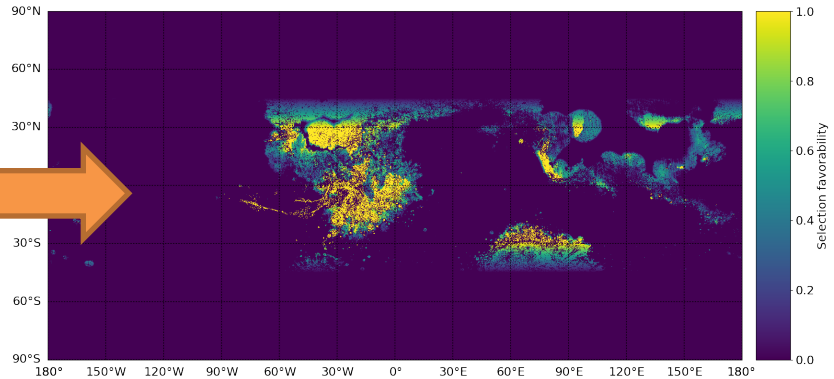
7) Fast Marching + Terrain

→ travel times to a selected location

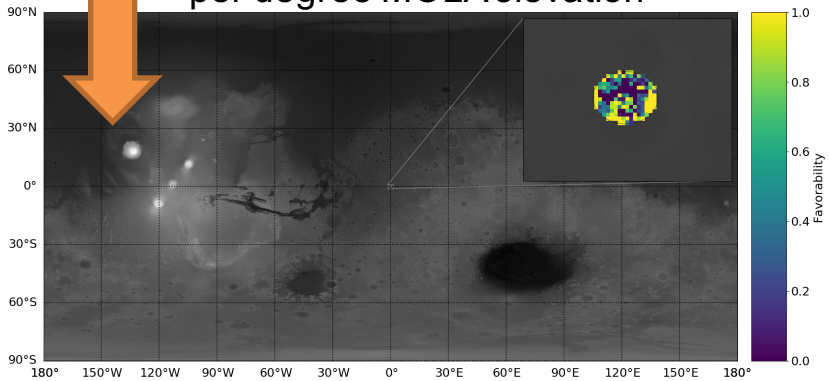


8) Combine with other criteria

e.g., latitude, elevation, thermal inertia, albedo, slope, vertical roughness, hydrated mineral sites, valley networks, HiRISE, CRISM and MOC footprints



Search for 25 × 20 km ellipse on 20 pixel per degree MOLA elevation

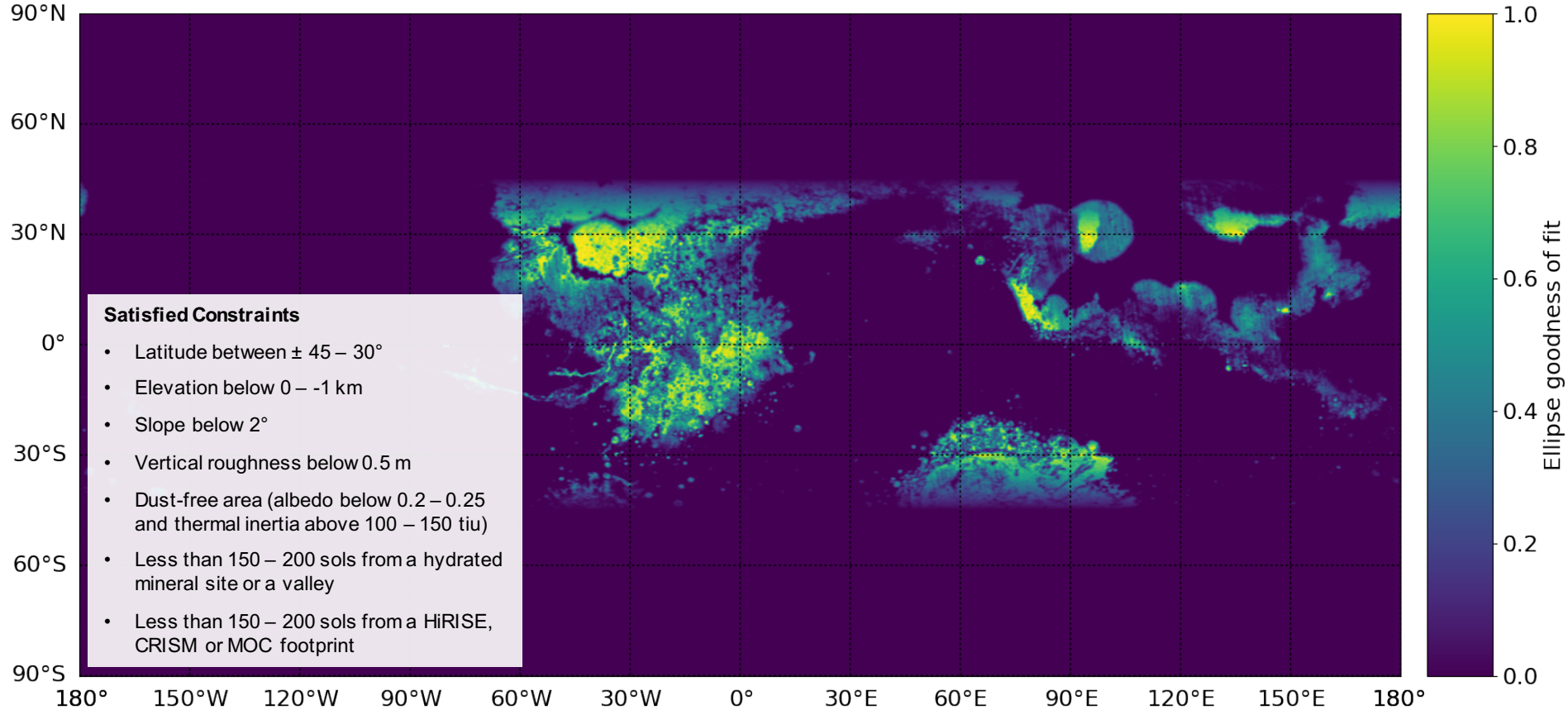


Result

MIT Haystack Observatory
Massachusetts Institute of Technology

Computer-Aided Discovery: Mars Site Selection

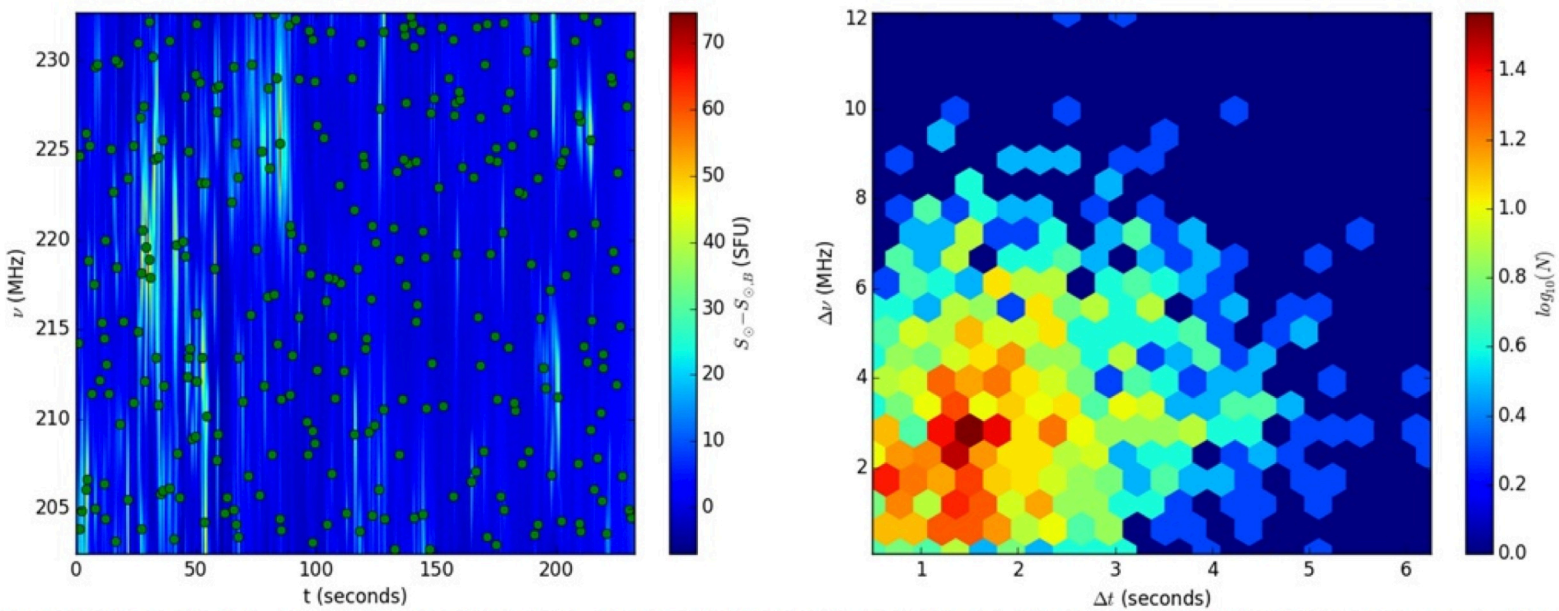
- The selection model is based on the average favorability inside the landing ellipse
- Goodness of fit of a 25 × 20 km ellipse taking into account both engineering and scientific constraints:
- Computation time ≈ 7h30min



Application Examples

Solar Science

Wavelet Based Characterization of Low Radio Frequency Solar Emissions

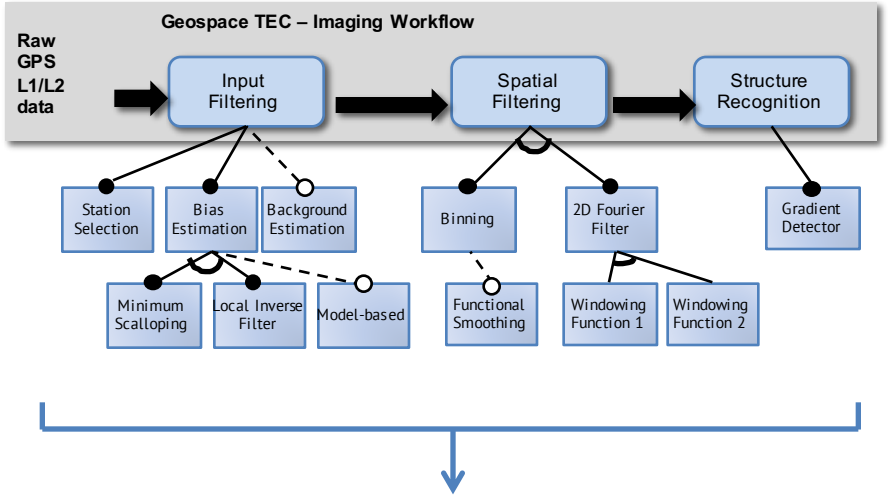


The left panel shows an example pre-processed dynamic spectrum from the MWA which has a resolution of 0.5 s and 40 kHz. The circles mark the features identified by our wavelet decomposition technique. The right panel shows the distribution of the detected features in the $\Delta t - \Delta\nu$ plane on a log scale.

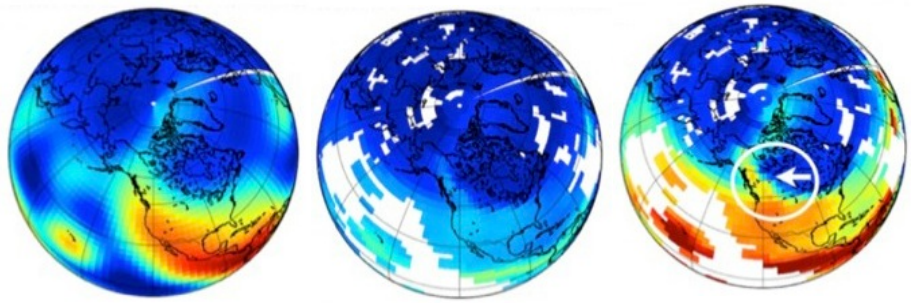
[A.Suresh, R.Sharma, D.Bharati Das, D.Oberoi, V.Pankratius, C.Lonsdale, and the MWA Collaboration. AGU 2016]

Application Examples

Ionosphere & Space Weather



Variants of ionospheric feature models



[V.Pankratius et al., IEEE Intelligent Systems, 2016]

Application Examples

Astronomy

...consider

- Light curves
- Pulsars
- Variable stars
- Stellar flares
- ...

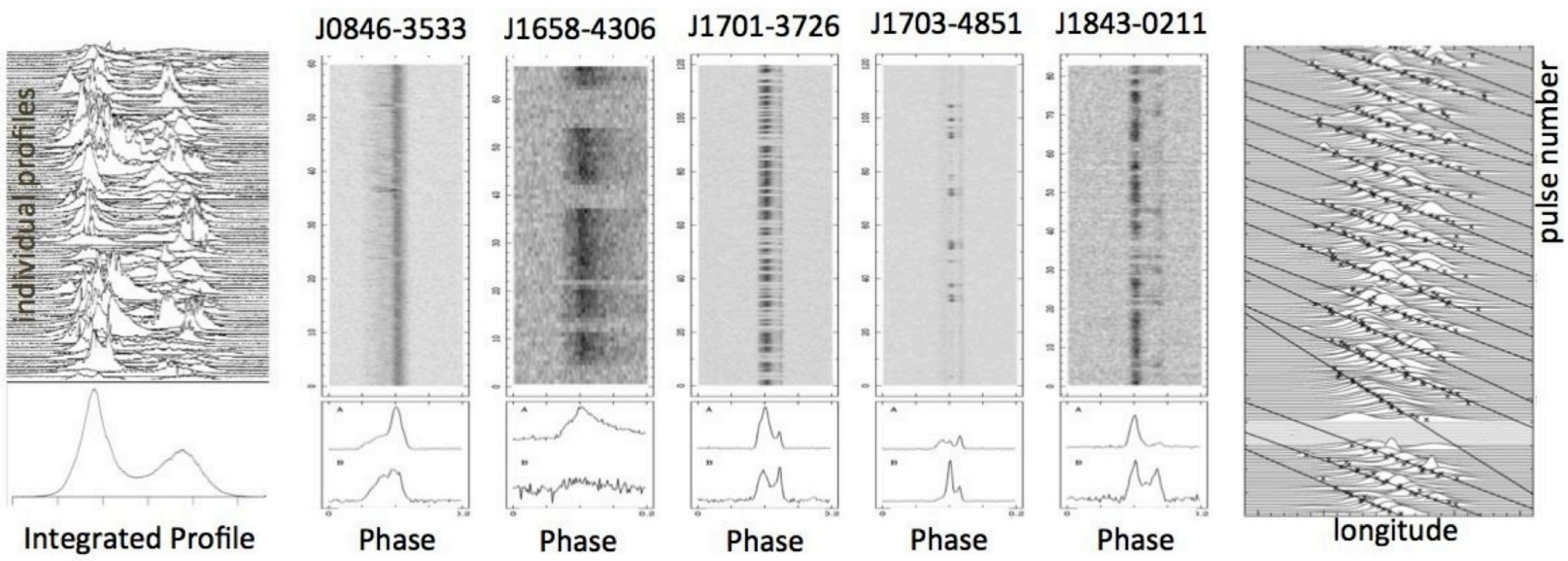
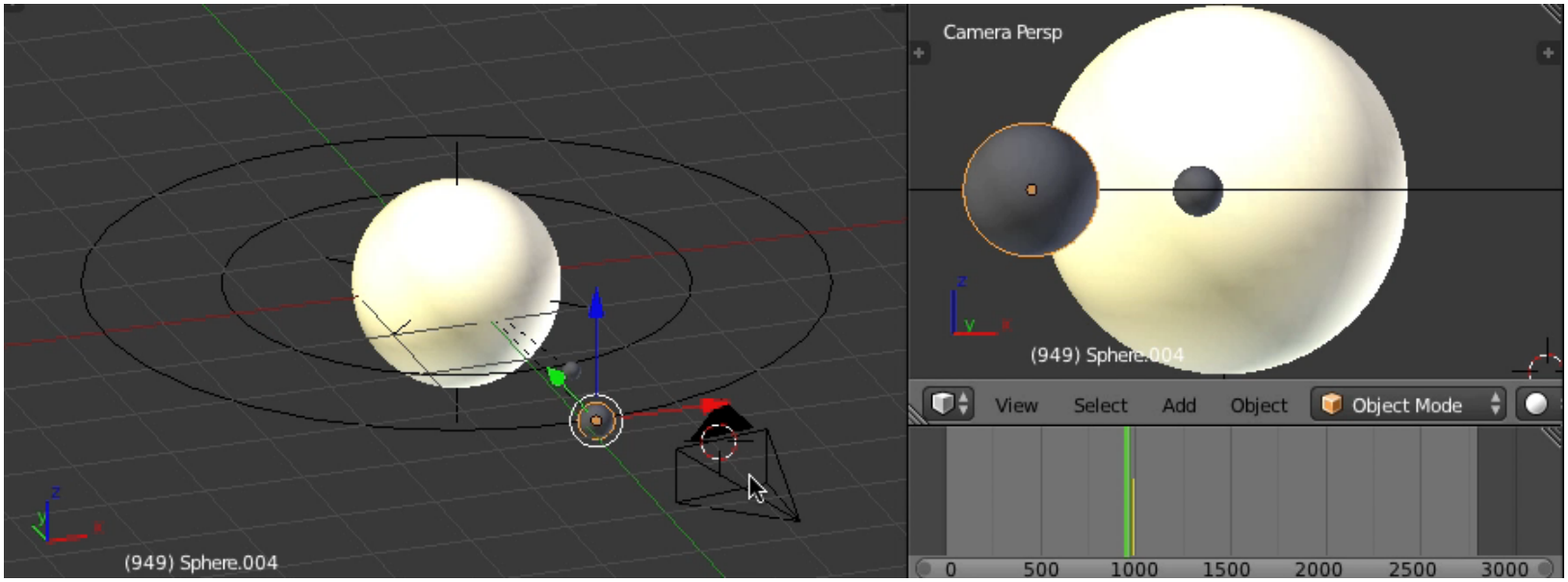


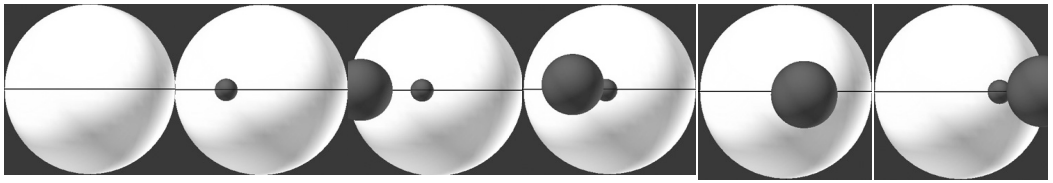
Image sources: [Seiradakis&Wielebinski A&A 2004, VanLeeuwen et al. A&A 2002, Wang et al. MNRAS 2007]

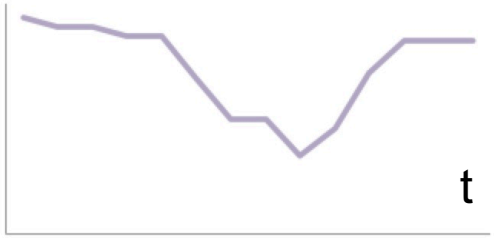
Application Examples

Astronomy / Exoplanets

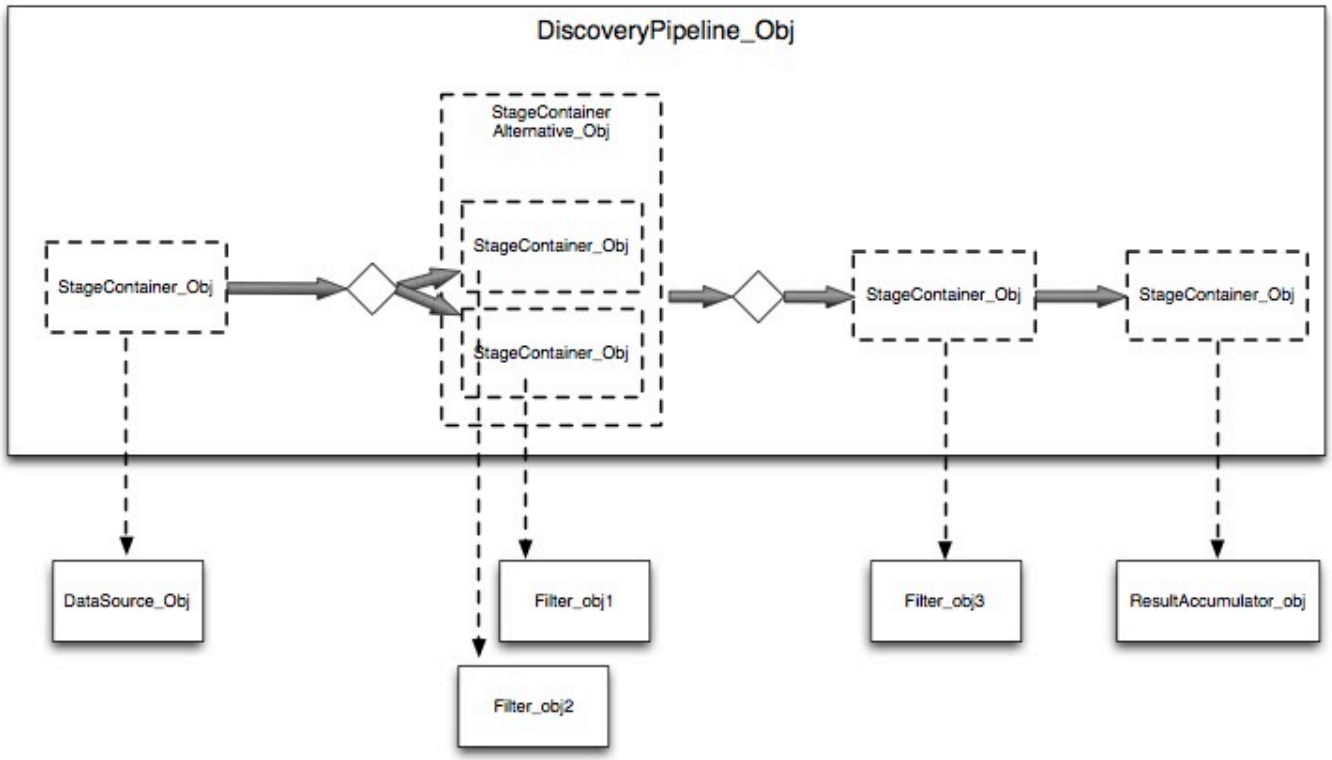
Light Curve Model Generation Example







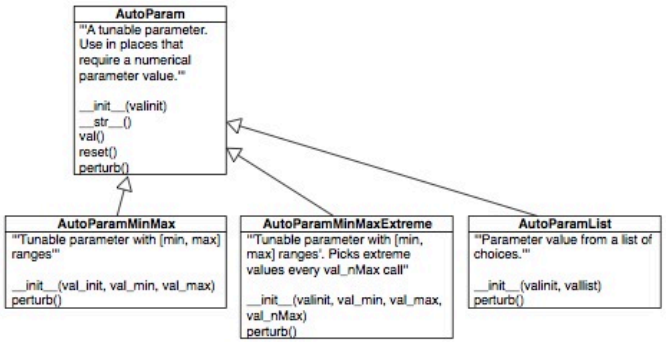
Discovery Framework Architecture Overview



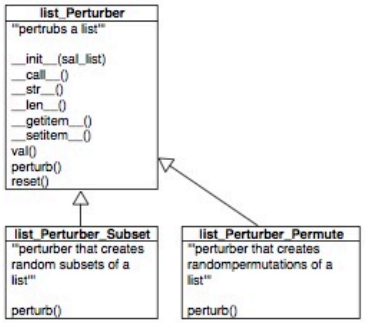
Python Framework for Computer-Aided Discovery - Victor Pankratius

Discovery Framework Architecture Overview

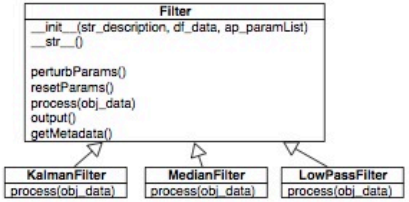
Autoparams



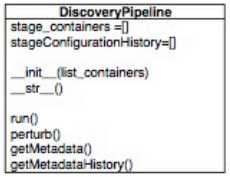
Perturbers



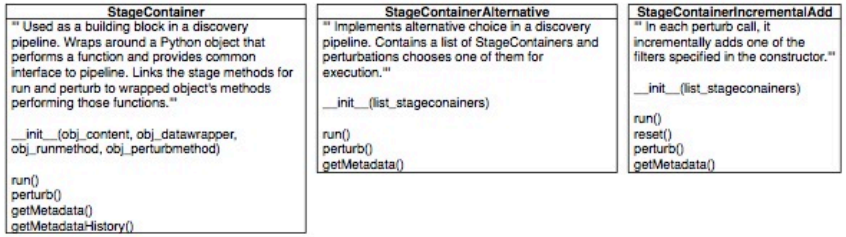
Filters



Discovery Pipeline



Stage Containers



Data Import: skdaccess

Python Data Access Package Release

https://github.com/skdaccess/skdaccess

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

Quick Start Guide

v1.0.0 for Python 3.4

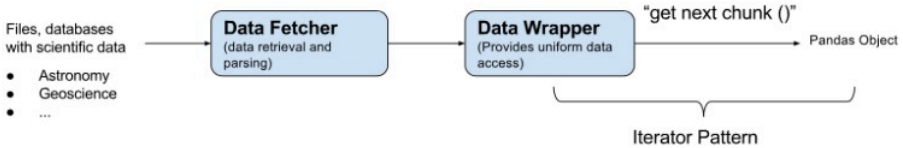
https://pypi.python.org/pypi/skdaccess

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 Sci-kit 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)

Install

pip install skdaccess

Documentation

See https://github.com/skdaccess/skdaccess/tree/master/skdaccess/docs

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 the equivalent water thickness.

Scikit Data Access

- Example usage for USGS groundwater data

Computer-Aided Discovery

06 - SKDACCESS Data Demo Last Checkpoint: an hour ago (unsaved changes)

Control Panel Logout

File Edit View Insert Cell Kernel Help Python 3

Code Cell Toolbar: None

In [2]:
```
from skdaccess.geo.groundwater import DataFetcher as GW_DF
```

Select groundwater stations in california with data within specified time range

In [3]:
```
groundwater_fetcher = GW_DF(start_date='2010-01-01',end_date='2014-01-01')
```

Get an iterator to the data

In [4]:
```
data_wrapper = groundwater_fetcher.output()
my_iterator = data_wrapper.getIterator()
```

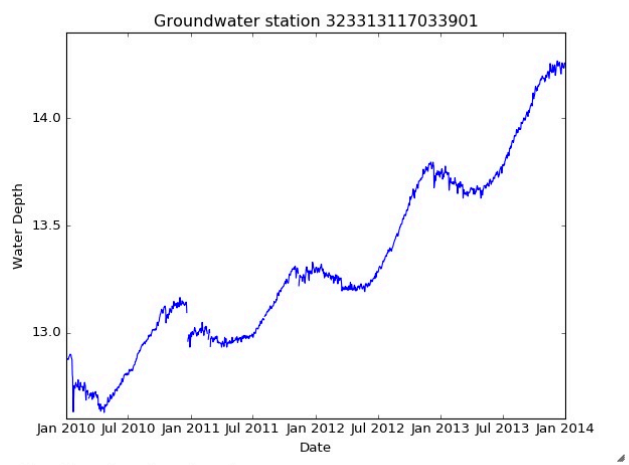
Each groundwater station can be accessed one at time

In [5]:
```
label, data, err = next(my_iterator)
```

Plot the first station

In [6]:
```
plt.plot(data);
plt.ylabel(label);
plt.xlabel("Date")
plt.title('Groundwater station ' + data.name);
```

Figure 1



MIT HAYSTACK OBSERVATORY
Massachusetts Institute of Technology

Code Example: Discovery Pipeline

- Works in your local Web browser; pipeline actually executed on server / cloud environment that hosts all data and executes computations. Only results go back to browser.
- Geographic region and time interval of interest selection
- Filter stage container setup for flexible analysis pipeline
- Perturb functions for exploring variable parameter values
- On top of this code, we can add GUI visualizations that hide the code from the average user, but experts can still edit code if needed.

```
# import all packages and modules
```

```
data_type = 'pbo'
geo_point = [54.13308, -165.98555] #Akutan
# geo_point = [59.3626, -153.435] #Augustine
start_time = '2004-01-01'
end_time = '2014-01-01'
site_radius = acf.AutoParamMinMax(20,5,50)
stab_degree = acf.AutoParamMinMax(25,10,40)
testing1 = acf.DataGenerator(data_type,geo_point,start_time,
                             end_time,[site_radius,stab_degree])
t1data = testing1.output()
storeName = pickle.load(open(testing1.storeName_fn,'rb'))
```

```
ftr_tf = acf.TrendFilter('TrendFilter', [])
sc_tf = acf.StageContainer(ftr_tf, ftr_tf.process,
                           ftr_tf.perturbParams,ftr_tf.resetParams)

ap_tau = acf.AutoParamMinMaxExtreme(120,1,500,5)
ap_sigmaSq = acf.AutoParamMinMaxExtreme(5.5,1,100,5)
ap_R = acf.AutoParamMinMaxExtreme(1,1,100,5)
ftr_kf1 = acf.KalmanFilter('KalmanFilter1',
                           [ap_tau, ap_sigmaSq, ap_R])
sc_kf1 = acf.StageContainer(ftr_kf1, ftr_kf1.process,
                            ftr_kf1.perturbParams,
                            ftr_kf1.resetParams)

# numH = acf.AutoParamMinMax(4,2,8)
# numV = acf.AutoParamMinMax(4,2,8)
# atyp = acf.AutoParamList('PCA',['PCA','ICA'])
numH = acf.AutoParamMinMax(4,2,6)
numV = acf.AutoParamMinMax(4,2,6)
atyp = acf.AutoParamList('PCA',['PCA'])
pca_A = acf.Component_Analysis('PCA Analysis',[numH,numV,atyp])
sc_pca = acf.StageContainer(pca_A, pca_A.process,
                            pca_A.perturbParams, pca_A.resetParams)
```

```
data_ls = acf.list_Perturber(['AV06','AV07','AV08','AV10','AV12','AV14','AV15'])
t1data.setStationList(data_ls)
```

```
pipeline_kf1 = acf.DiscoveryPipeline(t1data,[sc_tf, sc_kf1, sc_pca])
pipeline_kf1.run()
t1ddata = t1data.get()
pipeline_kf1.dataReset()
eigvecs = [pca_A.getEigenvectors(storeName)]
stations = pca_A.results['stations']
viz.multi_CAplot(eigvecs,geo_point,stations,storeName)
```

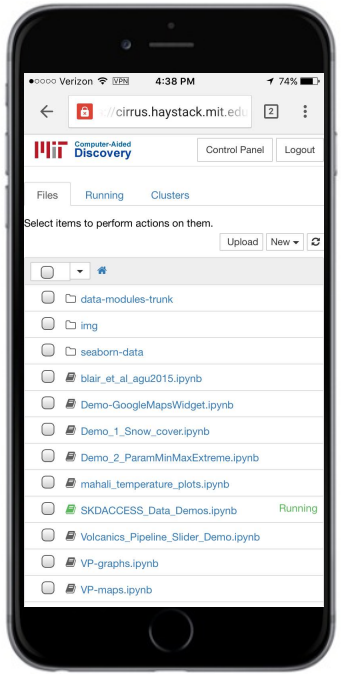
```
perturbRuns = acf.ResultAccumulator()
for ii in tqdm.tqdm(range(50)):
    pipeline_kf1.perturb()
    pipeline_kf1.run()
    pipeline_kf1.dataReset()
    perturbRuns.add(pca_A.getEigenvectors(storeName))
stations = pca_A.results['stations']

viz.multi_CAplot(perturbRuns.getall(),geo_point,stations,storeName)
mapres = viz.calc_distance_map(pipeline_kf1,storeName,1,'PCA Analysis')
sns.clustermap(mapres)
```

Proof of concept: Transparently offloading a processing pipeline to Amazon

```
ubuntu@ip-172-31-43-31:~$ /home/ubuntu/.local/bin/dispynode.py -s haystackstuff --ext_ip_addr 52.10.130.251
2016-02-25 16:48:08,100 - dispynode - dispynode version 4.6.4
2016-02-25 16:48:08,108 - dispynode - serving 2 cpus at 52.10.130.251:51348
Enter "quit" or "exit" to terminate dispynode,
  "stop" to stop service, "start" to restart service,
  "cpus" to change CPUs used, anything else to get status:

  Serving 2 CPUs
  Completed:
    0 Computations, 0 jobs, 0.000 sec CPU time
  Running:
    Client 1: amazon_run @ 127.0.0.1 running 0 jobs

Enter "quit" or "exit" to terminate dispynode,
  "stop" to stop service, "start" to restart service,
  "cpus" to change CPUs used, anything else to get status:

  Serving 2 CPUs
  Completed:
    0 Computations, 0 jobs, 0.000 sec CPU time
  Running:
    Client 1: amazon_run @ 127.0.0.1 running 0 jobs

Enter "quit" or "exit" to terminate dispynode,
  "stop" to stop service, "start" to restart service,
  "cpus" to change CPUs used, anything else to get status:

  Serving 2 CPUs
  Completed:
    0 Computations, 0 jobs, 0.000 sec CPU time
  Running:
    Client 1: amazon_run @ 127.0.0.1 running 0 jobs

Enter "quit" or "exit" to terminate dispynode,
  "stop" to stop service, "start" to restart service,
  "cpus" to change CPUs used, anything else to get status:

  Serving 2 CPUs
  Completed:
    0 Computations, 0 jobs, 0.000 sec CPU time
  Running:
    Client 1: amazon_run @ 127.0.0.1 running 2 jobs

Enter "quit" or "exit" to terminate dispynode,
  "stop" to stop service, "start" to restart service,
  "cpus" to change CPUs used, anything else to get status:

  Serving 2 CPUs
  Completed:
    1 Computations, 2 jobs, 7.509 sec CPU time
  Running:

Enter "quit" or "exit" to terminate dispynode,
  "stop" to stop service, "start" to restart service,
  "cpus" to change CPUs used, anything else to get status:
```

```
In [4]: ap_tau = AF.AutoParamMinMaxExtreme(120,1,500,5)
        ap_sigmaSq = AF.AutoParamMinMaxExtreme(5.5,1,100,5)
        ap_R = AF.AutoParamMinMaxExtreme(1,1,100,5)

        ftr_kf = AF.KalmanFilter('KalmanFilter', [ap_tau, ap_sigmaSq, ap_R])

        ftr_tf = AF.TrendFilter('TrendFilter', [])

        ac_data = AF.DataAccumulator('Data',[])

        sc_tf = AF.StageContainer(ftr_tf)
        sc_kf = AF.StageContainer(ftr_kf)
        sc_data = AF.StageContainer(ac_data)


        pipeline = AF.DiscoveryPipelineV2(data_generator, [sc_tf, sc_kf, sc_data])

In [5]: pipeline.run(2, amazon=True)

        2016-02-25 11:48:14,903 - dispy - Storing fault recovery information in "_dispy_20160225114814"
        INFO:dispy:Storing fault recovery information in "_dispy_20160225114814"

                                  Node |  CPUs |    Jobs |    Sec/Job | Node Time Sec
        ---------------------------------------------------------------------------
         52.10.130.251 (ip-172-31-43-31 |     2 |       2 |       3.755 |         7.509

        Total job time: 7.509 sec
```

Cloud environment can enable Big Data Computer-Aided Discovery on phones!



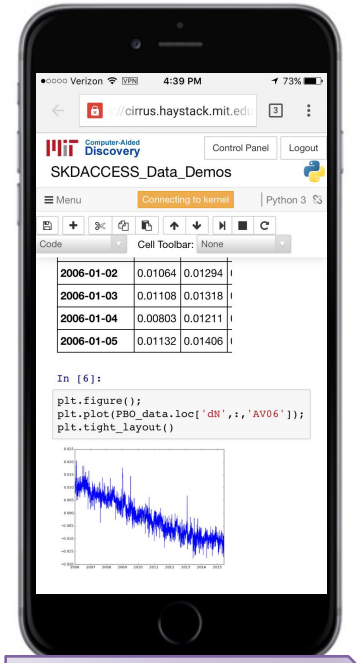
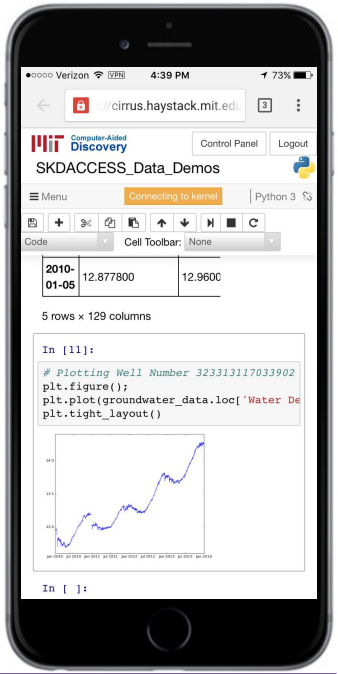
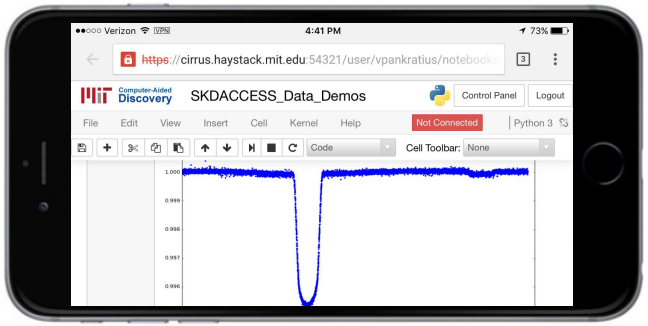


Plate Boundary Observatory Data



USGS Groundwater Data

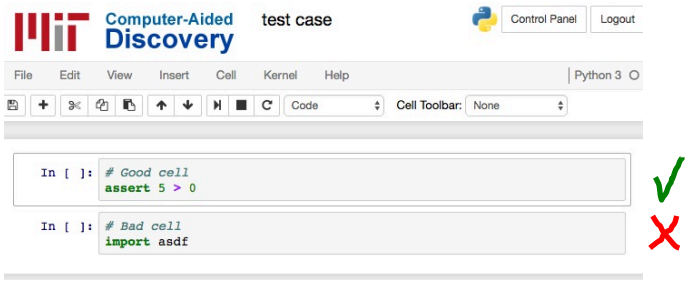


NASA Kepler Data

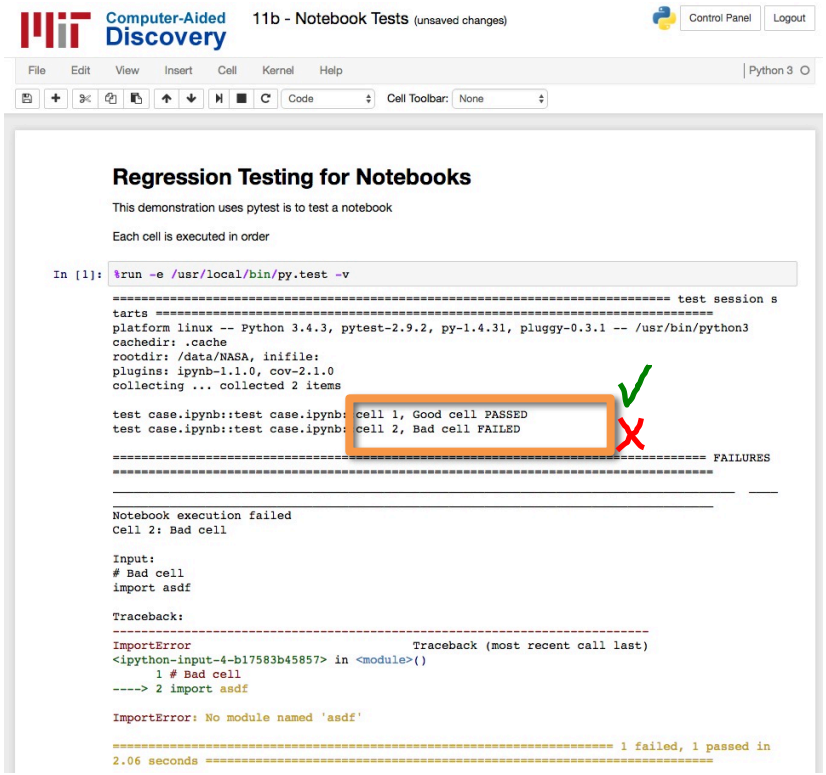
Testing

- We created a regression testing infrastructure for the Python notebooks based on pytest
- If notebook code, imports, dependencies change, our script re-runs all available notebooks, tests for errors/warnings, and produces “pass/fail” outputs

Test Case Example Notebook



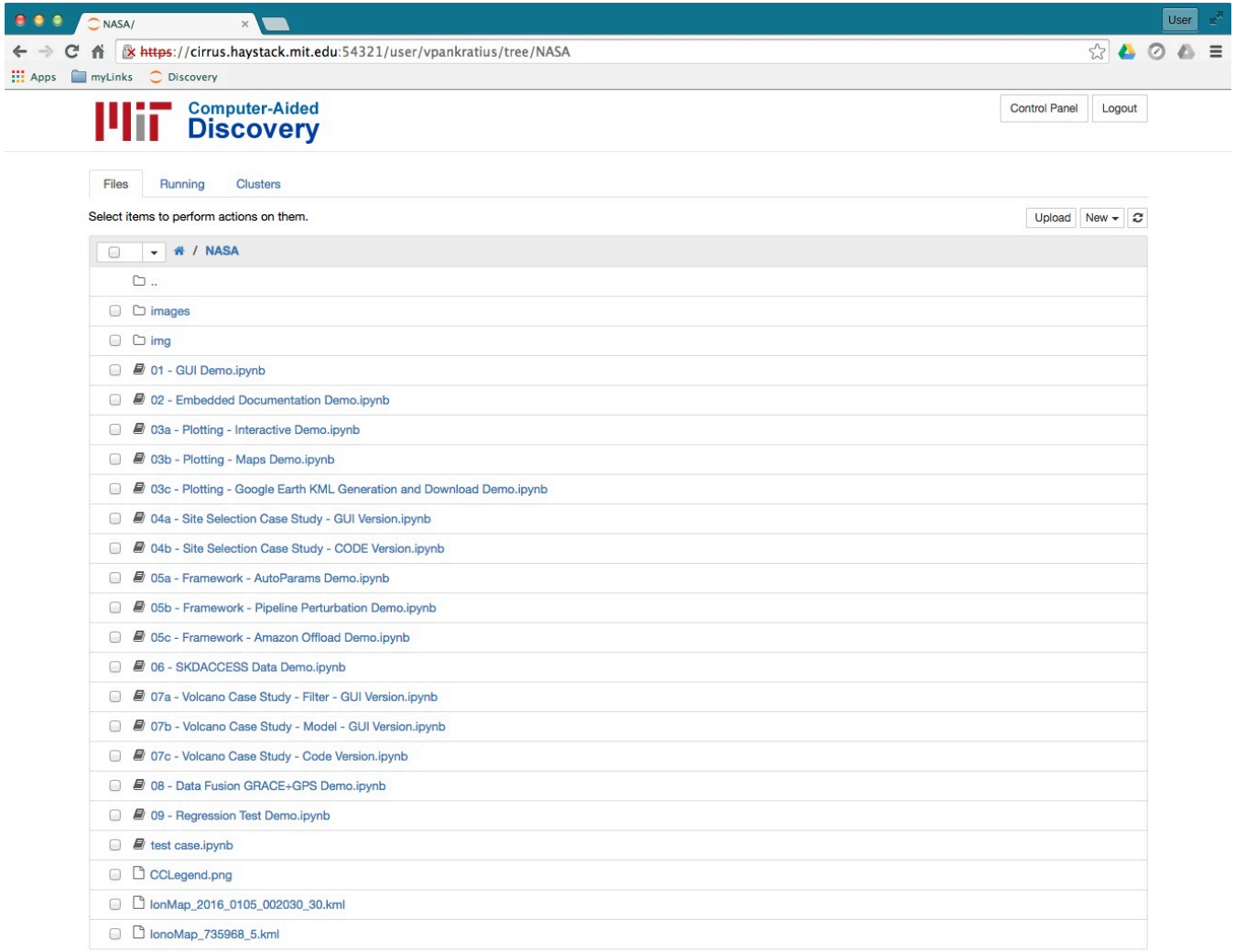
Regression Test



Deployment / Installation Plans

- System deployable as cloud instance on Amazon (right now with manual setup)
- Aiming for final “product” that will consist of an image that users can easily install (without requiring compilation, Python package installs, etc.)
- Also create VirtualBox virtual machine file for demo, can run locally

Demos


NASA/
User
https://cirrus.haystack.mit.edu:54321/user/vpankratius/tree/NASA
Apps
myLinks
Discovery
Computer-Aided Discovery
Control Panel
Logout
Files
Running
Clusters
Select items to perform actions on them.
Upload
New
/ NASA
..
images
img
01 - GUI Demo.ipynb
02 - Embedded Documentation Demo.ipynb
03a - Plotting - Interactive Demo.ipynb
03b - Plotting - Maps Demo.ipynb
03c - Plotting - Google Earth KML Generation and Download Demo.ipynb
04a - Site Selection Case Study - GUI Version.ipynb
04b - Site Selection Case Study - CODE Version.ipynb
05a - Framework - AutoParams Demo.ipynb
05b - Framework - Pipeline Perturbation Demo.ipynb
05c - Framework - Amazon Offload Demo.ipynb
06 - SKDACCESS Data Demo.ipynb
07a - Volcano Case Study - Filter - GUI Version.ipynb
07b - Volcano Case Study - Model - GUI Version.ipynb
07c - Volcano Case Study - Code Version.ipynb
08 - Data Fusion GRACE+GPS Demo.ipynb
09 - Regression Test Demo.ipynb
test case.ipynb
CCLegend.png
IonMap_2016_0105_002030_30.kml
IonoMap_735968_5.kml

1

Computer-Aided Discovery

01 - GUI Demo Last Checkpoint: an hour ago (unsaved changes)

Control Panel Logout

File Edit View Insert Cell Kernel Help Python 3

Code Cell Toolbar: None

GUI Demo

Demonstrating GUI capabilities

Import widgets

In [1]:
```
from ipywidgets import widgets
```

Create a simple function and call it using a widget

In [2]:
```
def f(x):
    print(x)
widgets.interact(f,x=10)
```

x 10

```
10
```

Out[2]:
```
<function __main__.f>
```

Functions can be wrapped using a decorator

In [3]:
```
@widgets.interact(x=True, y=1.0)
def g(x, y):
    print(x, y)
```

x ☑

y 1

```
True 1.0
```

Dropdown lists can also be made

In [4]:
```
widgets.interact(f, x=('apples','oranges'));
```

x apples

```
apples
```

A dictionary can be used to assign values to dropdown items

In [5]:
```
widgets.interact(f, x={'one': 10, 'two': 20});
```

x one

```
10
```

Multiple parameters of a function can be adjusted

In [6]:
```
from IPython.display import display
def func(a, b):
    return a+b
myParam = widgets.interactive(func, a=10, b=20)
display(myParam)
```

a 10

b 20

```
30
```

The resulting value can be retrieved

In [7]:
```
myParam.result  #use this to get value
```

Out[7]:
```
30
```


MIT Haystack Observatory
Massachusetts Institute of Technology

2

Computer-Aided Discovery

02 - Embedded Documentation Demo Last Checkpoint: 30 minutes ago (autosaved)

Control Panel | Logout

File Edit View Insert Cell Kernel Help | Python 3

Markdown | Cell Toolbar: None

Computer-Aided Discovery

Embedded Documentation Example

Markdown can be used to include scientific / usage documentation.

Images can be embedded in the documentation cell.

it is also possible to embed images in to regular notebook cells

In [1]:
```
from IPython.display import Image
```

In [2]:
```
Image(filename='images/logo.png')
```

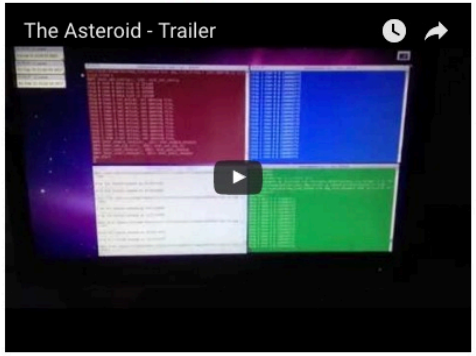
Out[2]:

Computer-Aided Discovery

Video can also be embedded into the notebook

In [3]:
```
from IPython.display import YouTubeVideo
YouTubeVideo('fO2j-gV36WA')
```

Out[3]:



3a

Computer-Aided Discovery

03a - Plotting - Interactive Demo Last Checkpoint: 30 minutes ago (unsaved changes)

Control Panel Logout

File Edit View Insert Cell Kernel Help | Python 3

Code Cell Toolbar: None

Interactive Plots

Plots can be embedded directly into the notebook. Plots can interact with user mouse clicks.

Initial imports

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

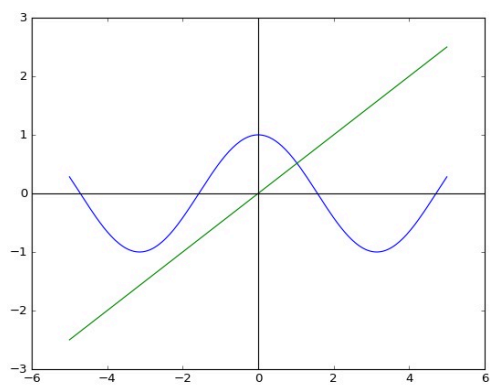
Generate two functions to plot

In [2]:
```
x = np.linspace(-5,5,200)
y1 = np.cos(x)
y2 = 0.5*x
```

Plot the results and add two axes

In [3]:
```
plt.plot(x,y1);
plt.plot(x,y2);
plt.axhline(0, color='black');
plt.axvline(0, color='black');
```

Figure 1



3b

Computer-Aided Discovery

03b - Plotting - Maps Demo Last Checkpoint: 31 minutes ago (unsaved changes)

Control Panel Logout

File Edit View Insert Cell Kernel Help Python 3

Code Cell Toolbar: None

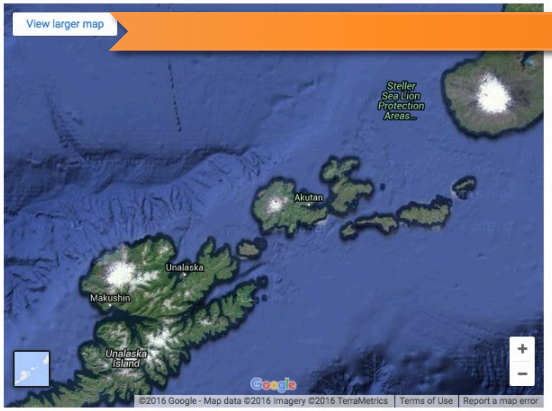
Map Plotting Demonstration

A demonstration of mapping visualization capabilities. In particular, we showcase embedding external Google Maps into the notebook and the utility of the Basemap package to generate geographic plots, with markers and annotations directly made onto the plot

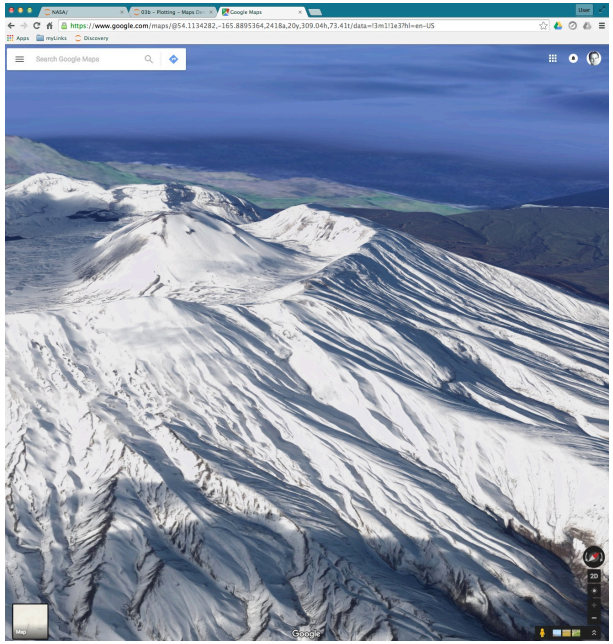
- Example of embedded Google Map display
- Example of Basemap plotting the MA region, with markers for Haystack and MIT
- Example of Basemap plotting, not restricted to the Earth, of a region on the Moon used in the lunar site selection example

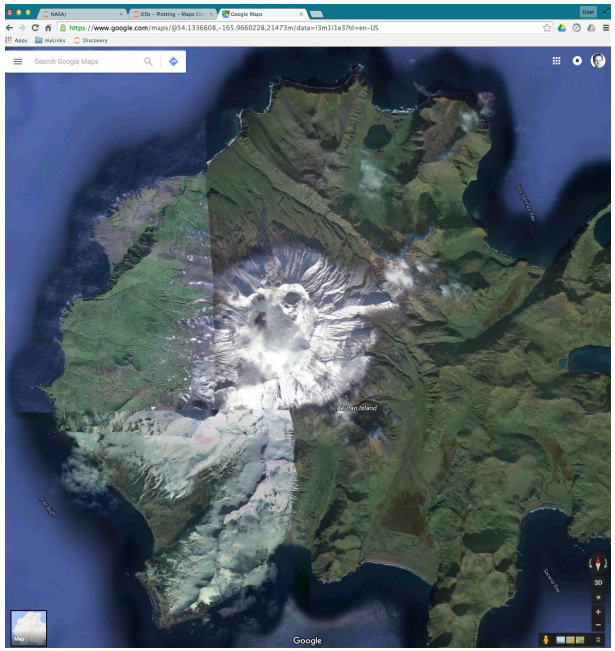
In [1]:
```
# Imports
%matplotlib inline
import matplotlib.pyplot as plt
from IPython.display import HTML, display
from mpl_toolkits.basemap import Basemap
import numpy as np
plt.rcParams['figure.figsize'] = (10.0, 8.0)
plt.rcParams['font.size'] = 12
plt.rcParams['svg.fonttype'] = 'none'
```

In [2]:
```
# Embedded Google Map Example
def eGoogleMap(Lat,Lon,zlevel=8):
    gmap_str = 'https://www.google.com/maps/embed/v1/view?key=AIzaSyCsYEeegZyzlqjy5DPQcGHQtJ-g0xRSWgc&center='
    gmap_str = gmap_str + str(Lat) + ',' + str(Lon) + '&zoom=' + str(zlevel) + '&maptype=satellite'
    display(HTML('<iframe width="600" height="450" frameborder="0" style="border:0" src="'+gmap_str+'"></iframe>'))

eGoogleMap(Lat=54.133056, Lon=-165.985556)
```



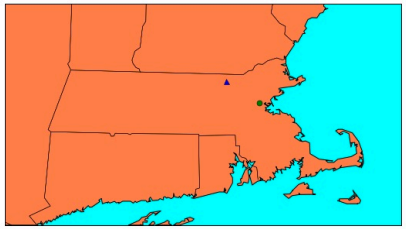
Click on "view larger map"



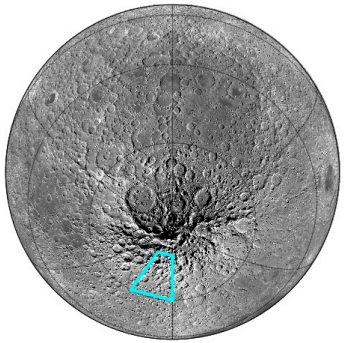


3b

```
In [3]: # Basemap Mapping Example
bmap = Basemap(llcrnrlat=41, urcrnrlat=43.5, llcrnrlon=-74, urcrnrlon=-69.5, projection='cyl', resolution='i')

plt.figure(figsize=(7.5,6));
plt.subplot(111);
# Draw just coastlines, no lakes
for i,cp in enumerate(bmap.coastpolygons):
     if bmap.coastpolygontypes[i]<2:
        bmap.plot(cp[0],cp[1],'k-')

bmap.fillcontinents(color='coral',lake_color='coral')
bmap.drawmapboundary(fill_color='aqua')
bmap.drawstates()
bmap.drawcountries()
bmap.drawmeridians(np.arange(0,360,30))
bmap.drawparallels(np.arange(-90,90,30))

# markers plotting Haystack and (approximately) MIT
bmap.plot(-71.4882, 42.6233, markersize=7, marker='^', color='blue', latlon=True)
bmap.plot(-71.1091, 42.3856, markersize=7, marker='o', color='green', latlon=True)

plt.tight_layout();
```



```
In [4]: # Lunar Mapping Example
N = -50; S = -90
W = 0; E = 360
m2 = Basemap(projection='ortho',lon_0=np.mean((W,E)),lat_0=np.mean((S,N)),resolution='l')

m2.warpimage(image="images/LRO-LOLA_WAC_16ppd.png", alpha=1)

m2.drawparallels(np.arange(-90.,120.,30.))
m2.drawmeridians(np.arange(0.,420.,60.))
m2.drawmapboundary(fill_color='aqua')

# Boxed region corresonds with the region used in the lunar site selection demo
xs = [0,30,30,0,0]
ys = [-60,-60,-80,-80,-60]
m2.plot(xs, ys,latlon = True, linewidth=4.0, color='cyan');
```



3c

Computer-Aided Discovery

03c - Plotting - Google Earth KML Generation and Download Demo Last Checkpoint: 40 minutes ago (autosaved)

Control Panel | Logout

File Edit View Insert Cell Kernel Help | Python 3

Code | Cell Toolbar: None

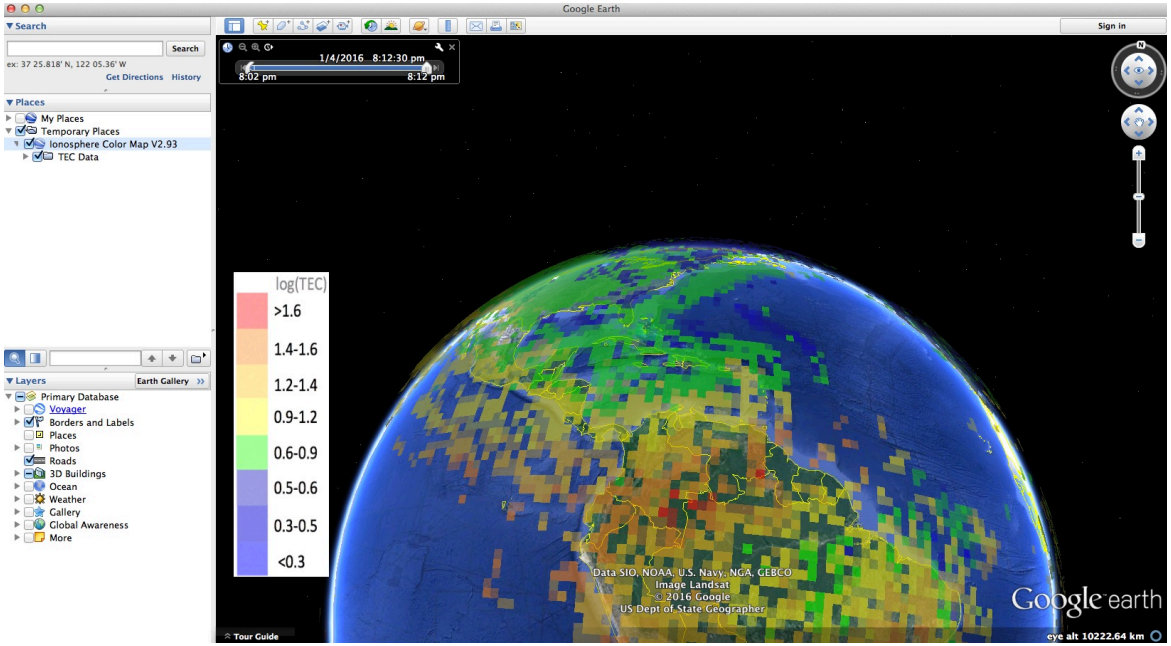
Google Earth KML Generation & Downloading Demo

A demonstration of an html download interface to obtain generated kml files for Google Earth visualization. The script interfaces directly with Haystack's Madrigal Database to grab global total electron content (TEC) values for visualizing the Earth's ionosphere. It generates a kml file that can be downloaded.

In [1]:
```
# Import
%matplotlib inline
import google_earth_kml
```

In [2]:
```
# Run the script and download the kml file here, or from the jupyter notebook homepage
google_earth_kml.generate_ionosphere_map(start_date='2016-01-05 0:2:30', duration=5)
```

DOWNLOAD KML FILE



4a

Site Selection Case Study - GUI Version

A demonstration of interactive (GUI) for choosing parameters for selecting sites of interest for lunar missions and fitting Gaussian Mixture Models (GMM) that characterize the distribution of regions of interest

- Slider(s) to interactively set search criterion parameter values
- Additional Slider stage displayed after fusing datasets to select the number of GMM components

For details, see publication:

Improving Spacecraft Site Selection Through Computer-Aided Discovery And Data Fusion.
David Blair, Michael Gowanlock, Justin Li, Cody Rude, Tom Herring, Victor Pankratius, 47th Lunar and Planetary Science Conference (LPSC), 2016
http://www.hou.usra.edu/meetings/lpsc2016/pdf/1987.pdf

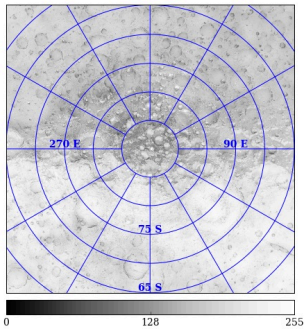
In [1]:
```
%matplotlib inline
# GUI Lunar Exploration Site Selection
import lunar_site_selection
lunar_site_selection.run()
```

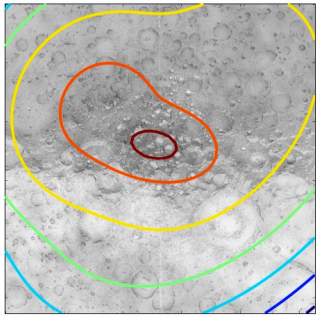
Slope 50
H 50
Th 50
Avg. Ill. 50
Avg. Earth Vis. 50

Fuse Datasets (using above relative weights)

Num. Comp. 3

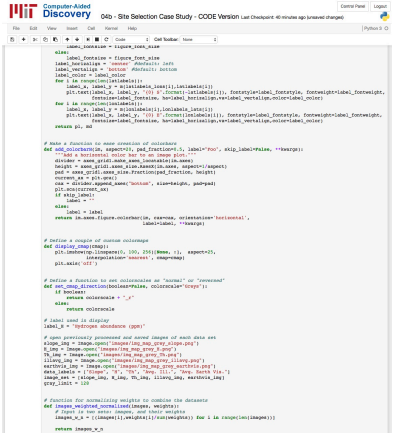
Fit Gaussian Mixture Model

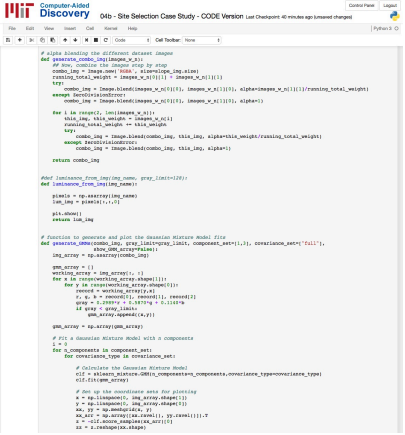


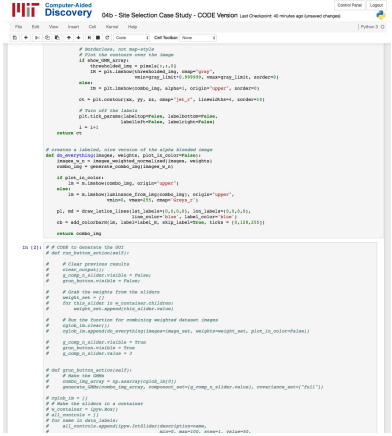


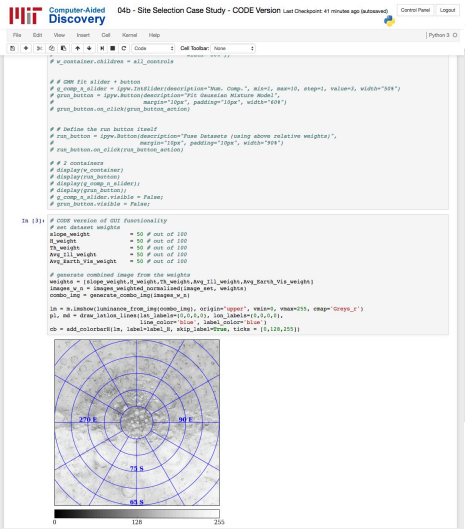
4b

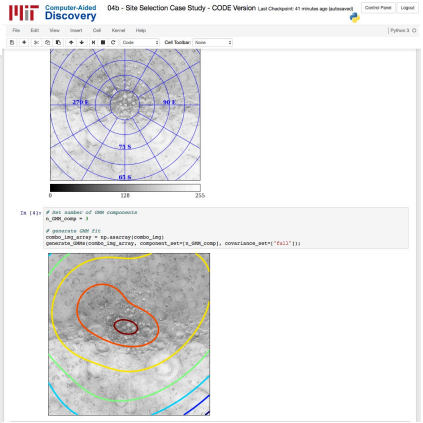
Site Selection Case Study - CODE Version
A demonstration of a code implementation for choosing parameters for selecting sites of interest for lunar missions and fitting Gaussian Mixture Models (GMM) that characterize the distribution of regions of interest
Setup, function definition, and data loading in the first cell
Commented version of how to create the GUI in the second cell
Site selection through code in the third and fourth cells
For details, see publication:
Improving Spacecraft Site Selection Through Computer-Aided Discovery And Data Fusion.
David Blair, Michael Gowanlock, Justin Li, Cody Paule, Tom Herring, Victor Pankratius, 47th Lunar and Planetary Science Conference (LPSC), 2016
http://www.hou.usra.edu/meetings/lpsc2016/pdf/1867.pdf


Computer-Aided Discovery
04b - Site Selection Case Study - CODE Version


Computer-Aided Discovery
04b - Site Selection Case Study - CODE Version


Computer-Aided Discovery
04b - Site Selection Case Study - CODE Version


Computer-Aided Discovery
04b - Site Selection Case Study - CODE Version


Computer-Aided Discovery
04b - Site Selection Case Study - CODE Version

5a

Computer-Aided Discovery — 05a - Framework - AutoParams Demo — Last Checkpoint: a few seconds ago (unsaved changes)

File Edit View Insert Cell Kernel Help | Python 3

Code | Cell Toolbar: None

Framework - AutoParam Demo

An overview of the different AutoParam classes within the skdaccess/skmdapt package. AutoParams and AutoList classes allow for the definition of parameters which can be perturbed, according to this class type. This formulation allows for multiple parameters to be perturbed within a pipeline using a single command.

- AutoParams - single value parameters
- AutoList - parameter of a list of values

In [1]:
```
# skdaccess param classes
from skdaccess.framework.param_class import *
```

In [2]:
```
# AutoParamMinMax
apMM = AutoParamMinMax(5,1,100)
print(apMM())
for ii in range(5):
    apMM.perturb()
    print(apMM())
```

```
5
52
31
94
4
18
```

In [3]:
```
# AutoParamMinMax (generates the min or max extreme value ever 'extreme' number of steps)
apMME = AutoParamMinMax(5,1,100,extreme=2)
print(apMME())
for ii in range(5):
    apMME.perturb()
    print(apMME())
```

```
5
89
100
9
100
31
```

In [4]:
```
# AutoParamList (returns randomly selected value from list)
apList = AutoParamList(3,[1,2,3,1,3,5,1,5,9])
print(apList())
for ii in range(5):
    apList.perturb()
    print(apList())
```

```
3
9
2
1
1
3
```

Computer-Aided Discovery — 05a - Framework - AutoParams Demo — Last Checkpoint: a few seconds ago (unsaved changes)

File Edit View Insert Cell Kernel Help | Python 3

Code | Cell Toolbar: None

```
9
100
31
```

In [4]:
```
# AutoParamList (returns randomly selected value from list)
apList = AutoParamList(3,[1,2,3,1,3,5,1,5,9])
print(apList())
for ii in range(5):
    apList.perturb()
    print(apList())
```

```
3
9
2
1
1
3
```

In [5]:
```
# AutoParamListCycle (get next value in list after each perturbation)
apLCycle = AutoParamListCycle([1,2,3,1,3,5,1,5,9])
print(apLCycle())
for ii in range(5):
    apLCycle.perturb()
    print(apLCycle())
```

```
1
2
3
1
3
5
```

In [6]:
```
# AutoListCycle (cycles through a list of given values)
aListC = AutoListCycle([[1,2],[3,5],[5,9],[7,3],[4,1]])
print(aListC())
for ii in range(5):
    aListC.perturb()
    print(aListC())
```

```
[1, 2]
[3, 5]
[5, 9]
[7, 3]
[4, 1]
[1, 2]
```

In [7]:
```
# AutoListPermute (permutes a list)
aListP = AutoListPermute([1,2,3,4,5,6,7])
print(aListP())
for ii in range(5):
    aListP.perturb()
    print(aListP())
```

```
[1, 2, 3, 4, 5, 6, 7]
[4, 5, 3, 6, 7, 2, 1]
[5, 3, 1, 6, 4, 2, 7]
[1, 3, 6, 2, 4, 5, 7]
[3, 2, 7, 6, 5, 1, 4]
[7, 1, 2, 6, 4, 5, 3]
```

5b

Computer-Aided Discovery — 05b - Framework - Pipeline Perturbation Demo Last Checkpoint: 2 minutes ago (autosaved) — Control Panel — Logout

File Edit View Insert Cell Kernel Help — Python 3

Code — Cell Toolbar: None

Framework - Pipeline Perturbation Demonstration

A demonstration of pipeline functionality in perturbing and visualizing AutoParams

- Visualization of configurations of pipeline parameters
- Example of easy functionality for running multiple configurations
- Demonstration of visualization tools for comparing the results of different configurations

In [1]:
```
# Imports

# Standard libraries and notebook plotting setup
%matplotlib notebook
import matplotlib.pyplot as plt
plt.rcParams['figure.figsize'] = (12.0, 6.0)
from IPython.display import display
import numpy as np
import pandas as pd

# skdaccess PBO Data Fetcher and param class
from skdaccess.geo.pbo import DataFetcher as PBODF
from skdaccess.framework.param_class import *

# skmdapt analysis items
from skmdapt import DiscoveryPipeline
import skmdapt.filters as mfilters
import skmdapt.analysis as analysis
import skmdapt.accumulators as sdacc

# skmdapt visualization tools
from skmdapt.visualization.multi_dist import calc_distance_map
from skmdapt.visualization import multiCaPlot

# skmdapt framework items
import skmdapt.framework.stagecontainers as ACF


# ----------------------------------------------------------------------------------------------
# Region of interest for stabilization, all of Alaska
lat_range = (50,70)
lon_range = (-175, -155)

# Time range of data of interest
start_time = '2006-01-01'
end_time = '2015-06-01'

# Akutan Volcano, Akutan Island, Alaska
ap_geopoint = AutoParam((54.13308, -165.98555))

# Interest in GPS Stations within 15 km
ap_radius = AutoParam(15)

# Create the Data Fetcher to load the data and apply stabilization to remove commone mode error
data_generator = PBODF(start_time, end_time, lat_range, lon_range,
                       [ap_radius, ap_geopoint],mdyratio=.7,stab_flag=1)
```

Computer-Aided Discovery — 05b - Framework - Pipeline Perturbation Demo Last Checkpoint: 3 minutes ago (autosaved) — Control Panel — Logout

File Edit View Insert Cell Kernel Help — Python 3

Code — Cell Toolbar: None

```
# ----------------------------------------------------------------------------------------------
# Assemble Stage Containers

# From before, reduce Akutan station set and remove AV13
data_generator.setStationList(['AV06','AV07','AV08','AV10','AV12','AV14','AV15'])

# Linear trend filter
ftr_tf = mfilters.TrendFilter('TrendFilter', [])
sc_tf = ACF.StageContainer(ftr_tf)

# Kalman Smoother
ap_tau = AutoParamMinMax(120,1,500,extreme=5)
ap_sigmaSq = AutoParamMinMax(4,1,100,extreme=5)
ap_R = AutoParamMinMax(1,1,100,extreme=5)
ftr_kf = mfilters.KalmanFilter('KalmanFilter',[ap_tau, ap_sigmaSq, ap_R])
sc_kf = ACF.StageContainer(ftr_kf)

# Median filter
ap_window = AutoParamMinMax(10,5,50,extreme=10)
ftr_mf = mfilters.MedianFilter('MedianFilter', [ap_window])
sc_mf = ACF.StageContainer(ftr_mf)

# Alternative Stage Container bundling Kalman and Median filters
sc_akm = ACF.StageContainerAlternative([sc_kf,sc_mf])

# Principal Component Analysis with specified time window
pca_times = ('2007-09-01','2008-11-01')
ap_start_time = AutoParam(pca_times[0])
ap_end_time = AutoParam(pca_times[1])
ap_hcomp = AutoParam(3)
ap_ctype = AutoParam('PCA')
ap_column_names = AutoParam(('dN','dE'))
ana_GPCA = analysis.General_Component_Analysis('GPCA', [ap_hcomp, ap_ctype, ap_start_time, ap_end_time, ap_column_names])
sc_gpca = ACF.StageContainer(ana_GPCA)
```

```
Opening /data/skdaccess/pbo_data.h5 in read-only mode
```

Pipeline runs of multiple configurations (verbose, non-verbose)

In [2]:
```
# set up the pipeline again, with sc_akm being a Kalman or a Median filter
pipeline_multi = DiscoveryPipeline(data_generator, [sc_tf, sc_akm, sc_gpca])

# run this pipeline 5 times as example, displaying the configurations
pipeline_multi.run(5,verbose=True)
# run it 45 times more, without display
pipeline_multi.run(45,verbose=False)
```

```
TrendFilter[] → KalmanFilter['120', '4', '1'] → GPCA['3', 'PCA', '2007-09-01', '2008-11-01', "('dN', 'dE')"]
TrendFilter[] → MedianFilter['45'] → GPCA['3', 'PCA', '2007-09-01', '2008-11-01', "('dN', 'dE')"]
TrendFilter[] → KalmanFilter['85', '70', '52'] → GPCA['3', 'PCA', '2007-09-01', '2008-11-01', "('dN', 'dE')"]
TrendFilter[] → KalmanFilter['155', '76', '15'] → GPCA['3', 'PCA', '2007-09-01', '2008-11-01', "('dN', 'dE')"]
TrendFilter[] → MedianFilter['25'] → GPCA['3', 'PCA', '2007-09-01', '2008-11-01', "('dN', 'dE')"]
```

5b

Multiple visualization functions for examining multiple configurations

In [3]:
```
# two ways of visualizing the perturbed configurations

# geographical overlay plot of all PCA results
multiCaPlot(pipeline_multi);

# heatmap showing similarities based on differences in PCA vector results
calc_distance_map(pipeline_multi,2,'GPCA',histIdx=True,fontsize=6);
```

Figure 1

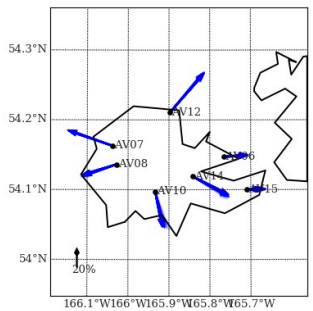
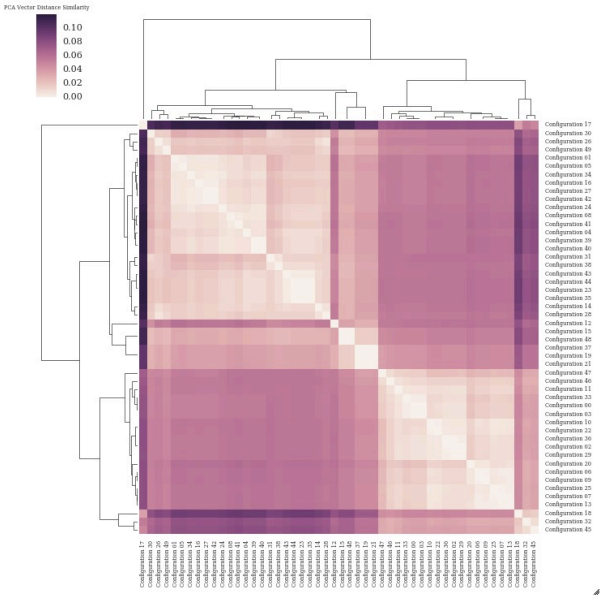


Figure 2



MIT Haystack Observatory
Massachusetts Institute of Technology

5c

Computer-Aided Discovery — 05c - Framework - Amazon Offload Demo Last Checkpoint: an hour ago (unsaved char

File Edit View Insert Cell Kernel Help

Code | Cell Toolbar: None

Framework - Offloading Pipeline Processing to Amazon Demo

This notebook demonstrates offloading work to an amazon server

Initial imports

In [1]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import matplotlib
plt.rcParams['figure.figsize'] = (14.0, 3.0)
import numpy as np
import pandas as pd
import re
```

skdaccess imports

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

skmdapt imports

In [3]:
```
from skmdapt import DiscoveryPipeline
from skmdapt.framework.stagecontainers import *
from skmdapt.table.filters import MedianFilter
from skmdapt.accumulators import DataAccumulator
```

Configure groundwater data fetcher

In [4]:
```
# Setup time range
start_date = '2000-01-01'
end_date = '2015-12-31'

# Select station
station_id = 340503117104104

# Create Data Fetcher
gwdf = GWDF([AutoList([station_id])],start_date,end_date,wrapper_type='table')
```

Create Pipeline

In [5]:
```
ap_window = AutoParamListCycle([pd.to_timedelta('0D'),
                                pd.to_timedelta('15D'),
                                pd.to_timedelta('40D'),
                                pd.to_timedelta('70D'),
                                pd.to_timedelta('150D'),
                                pd.to_timedelta('300D')])

fl_median = MedianFilter('Median Filter',[ap_window],interpolate=False)
sc_median = StageContainer(fl_median)


acc_data = DataAccumulator('Data Accumulator',[])
sc_data = StageContainer(acc_data)

pipeline = DiscoveryPipeline(gwdf,[sc_median, sc_data])
```

Computer-Aided Discovery — 05c - Framework - Amazon Offload Demo Last Checkpoint: an hour ago (unsaved changes) | Control Panel | Logout

File Edit View Insert Cell Kernel Help | Python 3

Code | Cell Toolbar: None

Display pipeline

In [6]:
```
pipeline.plotPipelineInstance()
```

Run the pipeline, offloading the processing to a node on Amazon.

While running, the amazon node can display the jobs:

1. ubuntu@ip-172-31-43-31: ~ (ssh)

```
ubuntu@ip-172-31-43-31:~$ dispynode.py -s 'the different tries during construction' --ext_ip_addr 52.39.135.37
2016-06-21 14:54:15 dispynode - dispynode version 4.6.14
2016-06-21 14:54:15 asyncoro - version 4.1 with epoll I/O notifier
2016-06-21 14:54:15 dispynode - serving 2 cpus at 52.39.135.37:51348

Enter "quit" or "exit" to terminate dispynode,
  "stop" to stop service, "start" to restart service,
  "cpus" to change CPUs used, anything else to get status:

  Serving 2 CPUs
  Completed:
    0 Computations, 0 jobs, 0.000 sec CPU time
  Running:
    Client 1: amazon_run @ 127.0.0.1 running 2 jobs


Enter "quit" or "exit" to terminate dispynode,
  "stop" to stop service, "start" to restart service,
  "cpus" to change CPUs used, anything else to get status:
```

In [7]:
```
pipeline.run(num_runs=6,amazon=True)

2016-06-22 14:39:58 asyncoro - version 4.1 with epoll I/O notifier
2016-06-22 14:39:58 dispy - Storing fault recovery information in "_dispy_20160622143958"

                                 Node |  CPUs |     Jobs |     Sec/Job | Node Time Sec
-----------------------------------------------------------------------------------------
 52.39.135.37 (ip-172-31-43-31) |     2 |        6 |       4.267 |        25.600

Total job time: 25.600 sec, wall time: 15.477 sec, speedup: 1.654
```

Plot the results

In [8]:
```
# Get the results
results = pipeline.getResults()
metadata = pipeline.getMetadataHistory()
# Loop over each pipeline run
for index,(run,info) in enumerate(zip(results,metadata)):
    # Plot the depth to water level
    plt.plot(run['Data Accumulator'].loc['Water Depth',:,:]);
    # Set xlabel
    plt.xlabel('Date');
    # Set ylabel
    plt.ylabel("Depth to Water Level");
    # Set title
    plt.title('Median Filter Window: ' + re.search("'(.*)days",info[1]).group(1) + 'Days');

    #Create new figure
    plt.figure();
```

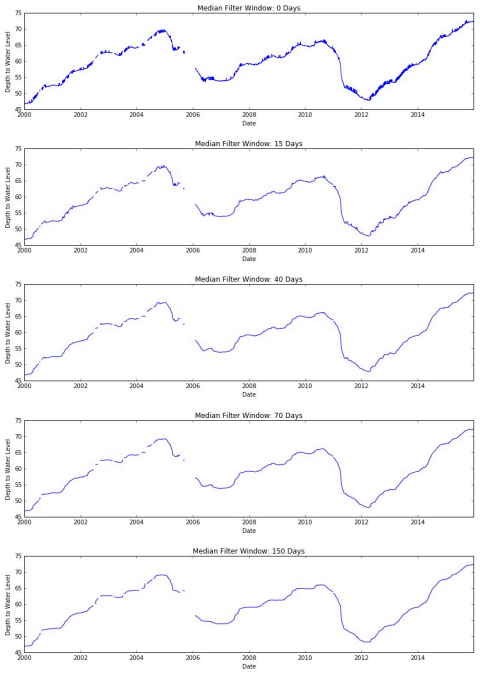
Offload this generated pipeline to Amazon Cloud

Results

Computer-Aided Discovery — 05c - Framework - Amazon Offload Demo Last Checkpoint: an hour ago (autosaved) | Control Panel | Logout

File Edit View Insert Cell Kernel Help | Python 3

Code | Cell Toolbar: None



6

Computer-Aided Discovery — 06 - SKDACCESS Data Demo Last Checkpoint: an hour ago (unsaved changes)

Control Panel | Logout

File Edit View Insert Cell Kernel Help | Python 3

Code | Cell Toolbar: None

Scientific Data Access using Scikit Data Access (skdaccess)

Open-source code: https://github.com/skdaccess/skdaccess
Code Doxygen Documentation: https://github.com/skdaccess/skdaccess/blob/master/skdaccess/docs/skdaccess_doxygen.pdf
User Guide: https://github.com/skdaccess/skdaccess/blob/master/skdaccess/docs/skdaccess_manual.pdf

DEMO: Accessing time series data using an iterator
Source: http://water.usgs.gov/ogw/
Returns depth to water level in meters

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

import skdaccess

In [2]:
```
from skdaccess.geo.groundwater import DataFetcher as GW_DF
```

Select groundwater stations in california with data within specified time range

In [3]:
```
groundwater_fetcher = GW_DF(start_date='2010-01-01',end_date='2014-01-01')
```

Get an iterator to the data

In [4]:
```
data_wrapper = groundwater_fetcher.output()
my_iterator = data_wrapper.getIterator()
```

Each groundwater station can be accessed one at time

In [5]:
```
label, data, err = next(my_iterator)
```

Computer-Aided Discovery — 06 - SKDACCESS Data Demo Last Checkpoint: an hour ago (unsaved changes)

Control Panel | Logout

File Edit View Insert Cell Kernel Help | Python 3

Code | Cell Toolbar: None

In [2]:
```
from skdaccess.geo.groundwater import DataFetcher as GW_DF
```

Select groundwater stations in california with data within specified time range

In [3]:
```
groundwater_fetcher = GW_DF(start_date='2010-01-01',end_date='2014-01-01')
```

Get an iterator to the data

In [4]:
```
data_wrapper = groundwater_fetcher.output()
my_iterator = data_wrapper.getIterator()
```

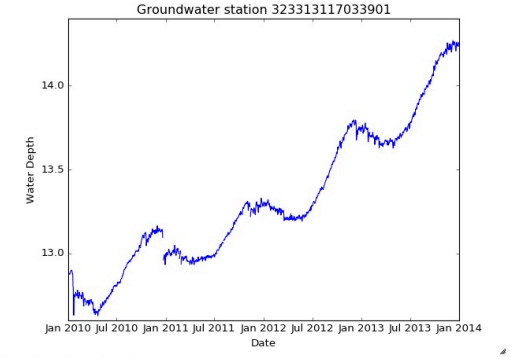
Each groundwater station can be accessed one at time

In [5]:
```
label, data, err = next(my_iterator)
```

Plot the first station

In [6]:
```
plt.plot(data);
plt.ylabel(label);
plt.xlabel("Date")
plt.title('Groundwater station ' + data.name);
```

Figure 1



7a

Computer-Aided Discovery

07a - Volcano Case Study - Filter - GUI Version Last Checkpoint: an hour ago (autosaved)

Control Panel Logout

File Edit View Insert Cell Kernel Help Python 3

Markdown Cell Toolbar: None

Volcano Case Study - Filter Parameter Perturbation GUI

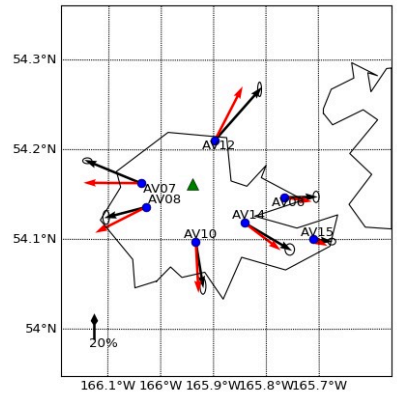
A demonstration of interactive (GUI) filter perturbation of an volcano analysis pipeline.

- Dropdown menu with two different types of smoothing filters
- Slider(s) to interactively set filter parameter values
- Example values: (tau = 120, sigma^2 = 4, R = 1)

For details, see
Computer Aided Detection of Transient Inflation Events at Alaska Volcanoes using GPS Measurements from 2005-2015
Justin Li, Cody Rude, David Blair, Michael Gowanlock, Thomas Herring, Victor Pankratius
submitted to Journal of Volcanology and Geothermal Research, May 2016

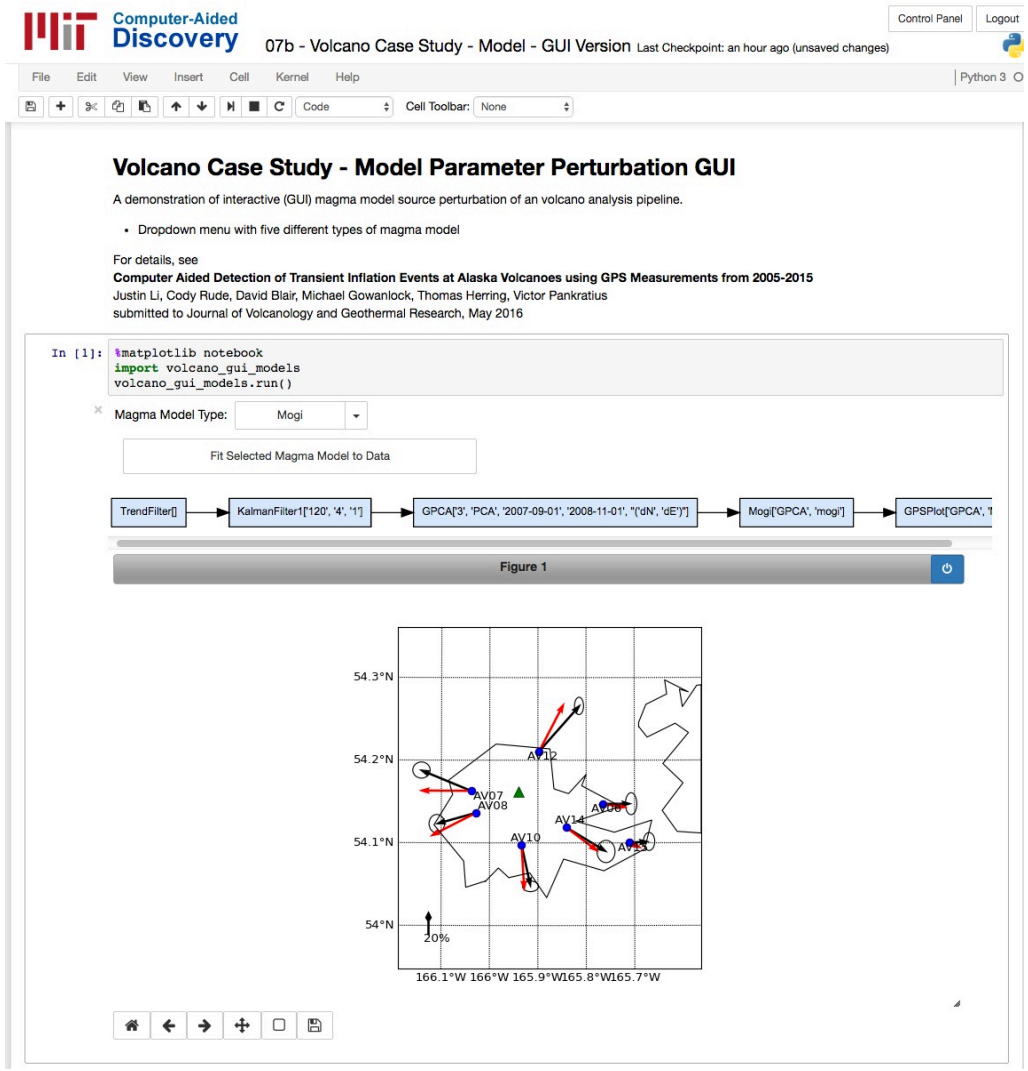
In [1]:
```
%matplotlib notebook
import volcano_gui_filters
volcano_gui_filters.run()
```

TrendFilter[] → KalmanFilter1['120', '4', '189'] → GPCA['3', 'PCA', '2007-09-01', '2008-11-01', "('dN', 'dE')"] → Mogi['GPCA', 'mogi'] → GPSPlot['GPC



Massachusetts Institute of Technology

7b


Computer-Aided Discovery
07b - Volcano Case Study - Model - GUI Version Last Checkpoint: an hour ago (unsaved changes)
Control Panel
Logout
File Edit View Insert Cell Kernel Help
Python 3
Code
Cell Toolbar: None
Volcano Case Study - Model Parameter Perturbation GUI
A demonstration of interactive (GUI) magma model source perturbation of an volcano analysis pipeline.
• Dropdown menu with five different types of magma model
For details, see
Computer Aided Detection of Transient Inflation Events at Alaska Volcanoes using GPS Measurements from 2005-2015
Justin Li, Cody Rude, David Blair, Michael Gowanlock, Thomas Herring, Victor Pankratius
submitted to Journal of Volcanology and Geothermal Research, May 2016
In [1]:
%matplotlib notebook
import volcano_gui_models
volcano_gui_models.run()
Magma Model Type:
Mogi
Fit Selected Magma Model to Data
TrendFilter[]
KalmanFilter1['120', '4', '1']
GPCA['3', 'PCA', '2007-09-01', '2008-11-01', "['dN', 'dE']"]
Mogi['GPCA', 'mogi']
GPSPlot['GPCA', '
Figure 1
54.3°N
54.2°N
54.1°N
54°N
AV12
AV07
AV08
AV14
AV10
20%
166.1°W 166°W 165.9°W165.8°W165.7°W

7c

Computer-Aided Discovery

07c - Volcano Case Study - Code Version Last Checkpoint: an hour ago (unsaved changes)

Control Panel Logout

File Edit View Insert Cell Kernel Help Python 3

Code Cell Toolbar: None

Volcano Case Study - Analysis Demonstration

A demonstration of the code used for the scientific analysis of transient event detection in Alaskan volcanoes, using as a specific example Akutan volcano (Akutan Island, Alaska).

- Highlights the usage of our skdaccess and skmdapt packages
- Includes comment blocks showing how the scientist provides expertise as the code runs
- Generates results that are in submission to publication

For details, see
Computer Aided Detection of Transient Inflation Events at Alaska Volcanoes using GPS Measurements from 2005-2015
Justin Li, Cody Rude, David Blair, Michael Gowanlock, Thomas Herring, Victor Pankratius
submitted to Journal of Volcanology and Geothermal Research, May 2016

In [1]:
```
# Imports
# Standard libraries and notebook plotting setup
%matplotlib notebook
import matplotlib.pyplot as plt
plt.rcParams['figure.figsize'] = (12.0, 6.0)
from IPython.display import display
import numpy as np
import pandas as pd

# skdaccess PBO Data Fetcher and param class
from skdaccess.geo.pbo import DataFetcher as PBODF
from skdaccess.framework.param_class import *

# skmdapt analysis items
from skmdapt import DiscoveryPipeline
import skmdapt.filters as mfilters
import skmdapt.analysis as analysis
import skmdapt.accumulators as sdacc
import skmdapt.framework.stagecontainers as ACF
```

Uses the skdaccess package to load the PBO data at Akutan

In [2]:
```
# Region of interest for stabilization, all of Alaska
lat_range = (50,75)
lon_range = (-175, -130)

# Time range of data of interest
start_time = '2006-01-01'
end_time = '2015-06-01'

# Akutan Volcano, Akutan Island, Alaska
ap_geopoint = AutoParam((54.13308, -165.98555))

# Interest in GPS Stations within 15 km
ap_radius = AutoParam(15)

# Create the Data Fetcher to load the data and apply stabilization to remove commone mode error
data_generator = PBODF(start_time, end_time, lat_range, lon_range,
                       [ap_radius, ap_geopoint],mdyratio=.7,stab_flag=1)
```

```
Opening /data/skdaccess/pbo_data.h5 in read-only mode
```

Uses the skmdapt package to build an analysis pipeline

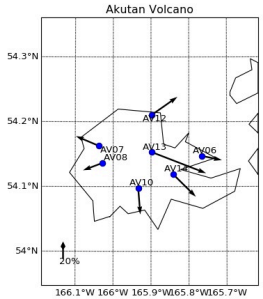
In [3]:
```
# Assemble Stage Containers

# Linear trend filter
ftr_tf = mfilters.TrendFilter('TrendFilter', [])
sc_tf = ACF.StageContainer(ftr_tf)

# Kalman Smoother
ap_tau = AutoParamMinMax(120,1,500,extreme=5)
ap_sigmaSq = AutoParamMinMax(4,1,100,extreme=5)
ap_R = AutoParamMinMax(1,1,100,extreme=5)
ftr_kf = mfilters.KalmanFilter('KalmanFilter',[ap_tau, ap_sigmaSq, ap_R])
sc_kf = ACF.StageContainer(ftr_kf)

# Time series data plotter tool
ac_plotter = sdacc.Plotter('Plotter',[])
sc_plotter = ACF.StageContainer(ac_plotter)

# Data accumulator for storing processed data
data_acc = sdacc.DataAccumulator('PBO_data',[])
sc_da = ACF.StageContainer(data_acc)

# Principal Component Analysis with full time window
pca_times = ('2006-01-01','2015-06-01')

ap_start_time = AutoParam(pca_times[0])
ap_end_time = AutoParam(pca_times[1])
ap_hcomp = AutoParam(3)
ap_ctype = AutoParam('PCA')
ap_column_names = AutoParam(('dN','dE'))
ana_GPCA = analysis.General_Component_Analysis('GPCA', [ap_hcomp, ap_ctype, ap_start_time, ap_end_time, ap_column_names])
sc_gpca = ACF.StageContainer(ana_GPCA)

# Geographic plot of the Volcano and GPS PCA motions
ac_hplot = sdacc.GPSHPlotter('GPSPlot', [AutoParam('GPCA')])
sc_hplot = ACF.StageContainer(ac_hplot)

# Compile the pipeline from the stage containers
pipeline = DiscoveryPipeline(data_generator, [sc_tf, sc_kf, sc_da, sc_gpca, sc_hplot])
```

Run the pipeline, seeing a visualization of the pipeline stages and parameters, direct results, and computed results

In [4]:
```
# Graphical flowchart of pipeline components
pipeline.plotPipelineInstance()

# Run first pass of the pipeline
pipeline.run()
plt.title('Akutan Volcano');


# PCA components
pcac = pipeline.RA_results[0]['GPCA']['CA'].components_
# Processing PBO data
pdata = pipeline.RA_results[0]['PBO_data']

# building error bars
herr_mat = [];
for site in pdata.minor_axis:
    herr_mat.append(pdata.loc['Sn',pca_times[0]:pca_times[1],site]**2)
    herr_mat.append(pdata.loc['Se',pca_times[0]:pca_times[1],site]**2)
herr_mat = np.array(herr_mat)
herr_pca = np.sqrt(np.abs(np.dot(pcac**2,herr_mat))).T

# Plot 1st Horizontal PCA component
plt.figure();
plt.errorbar(pd.date_range(start=pca_times[0],end=pca_times[1],freq='D'),
             pipeline.getResults()[0]['GPCA']['Projection'][:,0],
             yerr = herr_pca[:,0], errorevery=10);
plt.title('Horizontal Principal Components'); plt.xlabel('Time'); plt.ylabel('Amplitude (mm)');
```

TrendFilter[] → KalmanFilter['120', '4', '1'] → PBO_data[] → GPCA['3', 'PCA', '2006-01-01', '2015-06-01', *{'dN', 'dE'}'] → GPSPlot['GPCA']

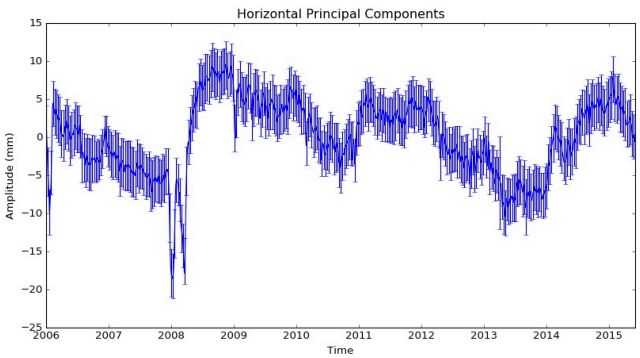
Figure 1



MIT Haystack Observatory
Massachusetts Institute of Technology

7c

Figure 2



After the above first pass, incorporate scientist's expertise

From the first-pass PCA of Akutan, the results aid the scientist in determining that:

- AV13's excessively large motion indicates some station error, and
- A transient event appears to occur in the the 09/2007 to 11/2008 range

So, for the next stage of analysis, remove AV13 from the analysis and narrow the time window for the PCA

In [5]:
```
# First-pass PCA of Akutan aids the scientist in determining that:
# (1) AV13's excessively large motion indicates some station error
# (2) and a transient event appears to occur in the the 09/2007 to 11/2008 range

# So, for a second pass, reduce Akutan station set and remove AV13
data_generator.setStationList(['AV06','AV07','AV08','AV10','AV12','AV14','AV15'])

# And refine the PCA time window
pca_times = ('2007-09-01','2008-11-01')


# Update Relevant Stage Containers (PCA)

# Principal Component Analysis with specified time window
ap_start_time = AutoParam(pca_times[0])
ap_end_time = AutoParam(pca_times[1])
ap_hcomp = AutoParam(3)
ap_ctype = AutoParam('PCA')
ap_column_names = AutoParam(('dN','dE'))
ana_GPCA = analysis.General_Component_Analysis('GPCA', [ap_hcomp, ap_ctype, ap_start_time, ap_end_time, ap_column_names])
sc_gpca = ACF.StageContainer(ana_GPCA)
```

Computer-Aided Discovery — 07c - Volcano Case Study - Code Version Last Checkpoint: an hour ago (autosaved)

Control Panel | Logout

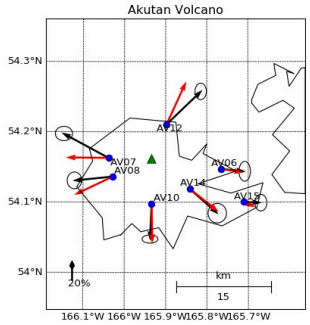
File Edit View Insert Cell Kernel Help | Python 3

Code | Cell Toolbar: None

```
# Geographic plot of the Volcano, GPS PCA motions, and Mogi model fitting results
ac_hplot = sdacc.GPSHPlotter('GPSPlot', [AutoParam('GPCA'), AutoParam('Mogi')], KF_tau=ap_tau.val, errorEllipses=True)
sc_hplot = ACF.StageContainer(ac_hplot)


# Add two additional stages

# Data accumulator for storing unprocessed data
rdata_acc = sdacc.DataAccumulator('raw_PBO_data',[])
sc_rda = ACF.StageContainer(rdata_acc)

# Mogi model estimation
modelType = AutoParam('mogi')
ana_mogi = analysis.Mogi_Inversion('Mogi', [AutoParam('GPCA'),modelType])
sc_mogi = ACF.StageContainer(ana_mogi)


# Recompile the pipeline from the stage containers
pipeline = DiscoveryPipeline(data_generator, [sc_rda, sc_tf, sc_kf, sc_da, sc_gpca, sc_mogi, sc_hplot])

# Graphical flowchart of pipeline components
pipeline.plotPipelineInstance()

# Re-Run the pipeline
pipeline.run()

# Some adjustments to figure scaling for viewing
bmap = pipeline.RA_results[0]['GPSPlot']
plt.title('Akutan Volcano')
bmap.drawmapscale(-165.76, 53.98, -165.94, 54.18, 15, fontsize = 12);
```

raw_PBO_data[] → TrendFilter[] → KalmanFilter['120', '4', '1'] → PBO_data[] → GPCA['3', 'PCA', '2007-09-01', '2008-11-01', '('dN', 'dE')']

Figure 3



7c

Computer-Aided Discovery

07c - Volcano Case Study - Code Version Last Checkpoint: an hour ago (unsaved changes)

File Edit View Insert Cell Kernel Help | Python 3

Code Cell Toolbar: None

Validate the targeted analysis

The geographical result plot has improved, with no station now showing excessively significant motion.

Examination of the improved time window PCA motion and a comparison of stations on opposite sides of the volcano showing an inflation motion validate the scientist's parameter selection and the detection and measurement of a transient inflation event at Akutan.

In [6]:
```
# Examine the improved time window PCA Results and validate the observed transient inflation event

# PCA components
pcac = pipeline.RA_results[0]['GPCA']['CA'].components_
# Raw PBO data
rdata = pipeline.RA_results[0]['raw_PBO_data']
# Processing PBO data
pdata = pipeline.RA_results[0]['PBO_data']

# building error bars
herr_mat = [];
for site in pdata.minor_axis:
    herr_mat.append(pdata.loc['Sn',pca_times[0]:pca_times[1],site]**2)
    herr_mat.append(pdata.loc['Se',pca_times[0]:pca_times[1],site]**2)
herr_mat = np.array(herr_mat)
herr_pca = np.sqrt(np.abs(np.dot(pcac**2,herr_mat))).T

# Plot 1st Horizontal PCA component
plt.figure();
plt.errorbar(pd.date_range(start=pca_times[0],end=pca_times[1],freq='D'),
             pipeline.getResults()[0]['GPCA']['Projection'][:,0],
             yerr = herr_pca[:,0], errorevery=5);
plt.title('Horizontal Principal Components'); plt.xlabel('Time'); plt.ylabel('Amplitude (mm)');

# Plot raw motion differenced between dE AV06 and AV07 as event validation
plt.figure();
diff_06_07 = rdata['dE',:,'AV06'] - rdata['dE',:,'AV07']
plt.plot(diff_06_07[:'2010']);
plt.title('AV06 dE - AV07 dE'); plt.xlabel('Time'); plt.ylabel('Displacement Difference [mm]');
```

Figure 4

Figure 4

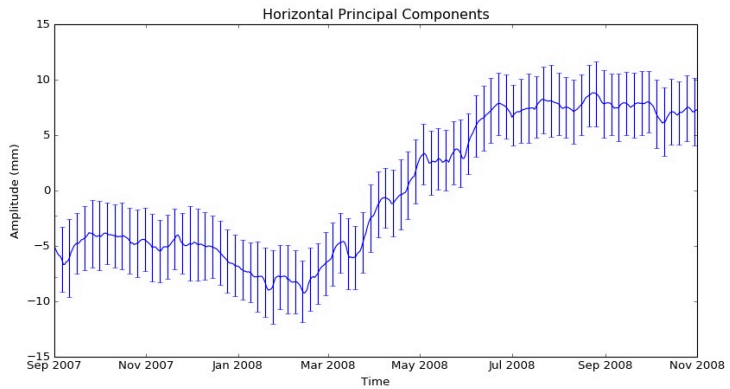
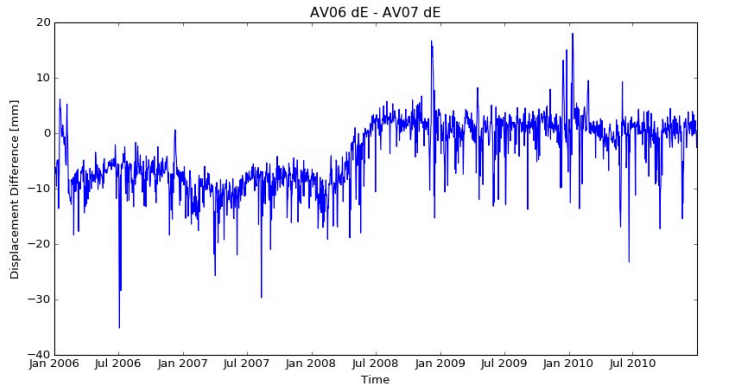


Figure 5



Massachusetts Institute of Technology

8

Computer-Aided Discovery — 08 - Data Fusion GRACE+GPS Demo — Last Checkpoint: an hour ago (unsaved changes)

Control Panel | Logout

File Edit View Insert Cell Kernel Help | Python 3

Markdown | Cell Toolbar: None

Data Fusion GRACE + GPS Demo

This notebook demonstrates data fusion by correlating the vertical gps position with equivelent water thickness from GRACE

Initial Imports:

In [1]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import matplotlib
import numpy as np
from mpl_toolkits.basemap import Basemap
```

skdaccess imports

In [2]:
```
from skdaccess.framework.param_class import *
from skdaccess.geo.pbo import DataFetcher as PBODF
from skdaccess.geo.grace import DataFetcher as GDF
```

skmdapt imports

In [3]:
```
from skmdapt import DiscoveryPipeline
from skmdapt.framework.stagecontainers import *
from skmdapt.table.filters import KalmanFilter
from skmdapt.table.accumulators import Plotter
from skmdapt.table.analysis import Correlate
from skmdapt.table.fusion import GraceFusion
```

Selecting region of interest and creating data fetcher

In [4]:
```
# Select Date range
start_date='2010-09-01'
end_date='2015-06-01'

# Select point of interest
lat = 36.5
lon = -119.5
ap_geopoint = AutoParam((lat,lon))

# Select radius around point of ineterest
ap_radius = AutoParam(100)

# Select range to perform reference frame stabilization
stab_lat_range = (lat-1.5,lat+1.5)
stab_lon_range = (lon-1.5,lon+1.5)


# Create Data Fetcher
pbo_dg = PBODF(start_date,end_date, stab_lat_range, stab_lon_range,
               [ap_radius, ap_geopoint],mdyratio=0.75,wrapper_type='table')
```

```
Opening /data/skdaccess/pbo_data.h5 in read-only mode
```

Computer-Aided Discovery — 08 - Data Fusion GRACE+GPS Demo — Last Checkpoint: an hour ago (unsaved changes)

Control Panel | Logout

File Edit View Insert Cell Kernel Help | Python 3

Markdown | Cell Toolbar: None

Creating a pipeline

In [5]:
```
# Setup Kalman Filter Parameters
ap_tau = AutoParamMinMax(120,1,500,5)
ap_sigmaSq = AutoParamMinMax(2,1,100,5)
ap_R = AutoParamMinMax(1,1,100,5)

# Create Kalman Filter
fl_kf = KalmanFilter('KalmanFilter', [ap_tau, ap_sigmaSq, ap_R], uncertainty_clip=5)
sc_kf = StageContainer(fl_kf)

# Create Plotter
ac_plotter = Plotter('Plotter',column_names=['dU','Grace'], num_columns=2)
sc_plotter = StageContainer(ac_plotter)

# Create GRACE Fusion pipeline item
metadata = PBODF.getStationMetadata()
fs_grace = GraceFusion('GRACE Fusion',metadata)
sc_grace = StageContainer(fs_grace)

# Create Correlation pipeline item
ana_corr = Correlate("Correlate",['dU','Grace'],local_match=True)
sc_corr = StageContainer(ana_corr)

# Create Pipeline
pbo_pipe = DiscoveryPipeline(pbo_dg, [sc_kf,sc_grace,sc_corr])
```

Display the pipeline

In [6]:
```
pbo_pipe.plotPipelineInstance()
```

Run the pipeline

In [7]:
```
pbo_pipe.run()
```

Retrieve Correlation results from the pipeline

In [8]:
```
corr = pbo_pipe.getResults(0)['Correlate']
```

Create map to plot results

In [9]:
```
# Bounding lat and lon coordinates
llat = 34
ulat = 38
llon =-122
rlon = -118

# Create Map
bmap = Basemap(llcrnrlat=llat, urcrnrlat=ulat, llcrnrlon=llon,
               urcrnrlon=rlon,projection='gnom', lon_0=np.mean([llon,rlon]),
               lat_0=np.mean([llat,ulat]), resolution='h')
```


Massachusetts Institute of Technology

8

Computer-Aided Discovery

08 - Data Fusion GRACE+GPS Demo Last Checkpoint: an hour ago (unsaved changes)

File Edit View Insert Cell Kernel Help

Markdown Cell Toolbar: None

Plot results on map

In [10]:
```
# Setup Figure
fig     = plt.figure()
ax      = fig.gca()
fig.set_size_inches(12,16);


# Draw state and country lines
bmap.drawstates();
bmap.drawcountries();

# Plot station locations and correlation values
pbo_lat_list = []
pbo_lon_list = []
for label, data in corr.iteritems():
    pbo_lat_list.append(metadata[label]['Lat'])
    pbo_lon_list.append(metadata[label]['Lon'])
    bmap.plot(metadata[label]['Lon'], metadata[label]['Lat'],'.',latlon=True)
    x,y = bmap(metadata[label]['Lon'], metadata[label]['Lat']);
    ax.annotate(round(data.loc['dU','Grace'],2), xy=(x,y), xytext=(11,11),
                ha='right', textcoords='offset points', fontsize=14);


# Draw a shaded relief map
bmap.shadedrelief();

# Add lines of constant latitude and longitude
bmap.drawmeridians(np.arange(llon,rlon,1.5),labels=[0,0,0,1],fontsize=12)
bmap.drawparallels(np.arange(llat,ulat,1.5),labels=[1,0,0,0],fontsize=12)

# Set plot title
plt.title("Correlations between Vertical GPS Positions and GRACE in Southern California",fontsize=16);
```

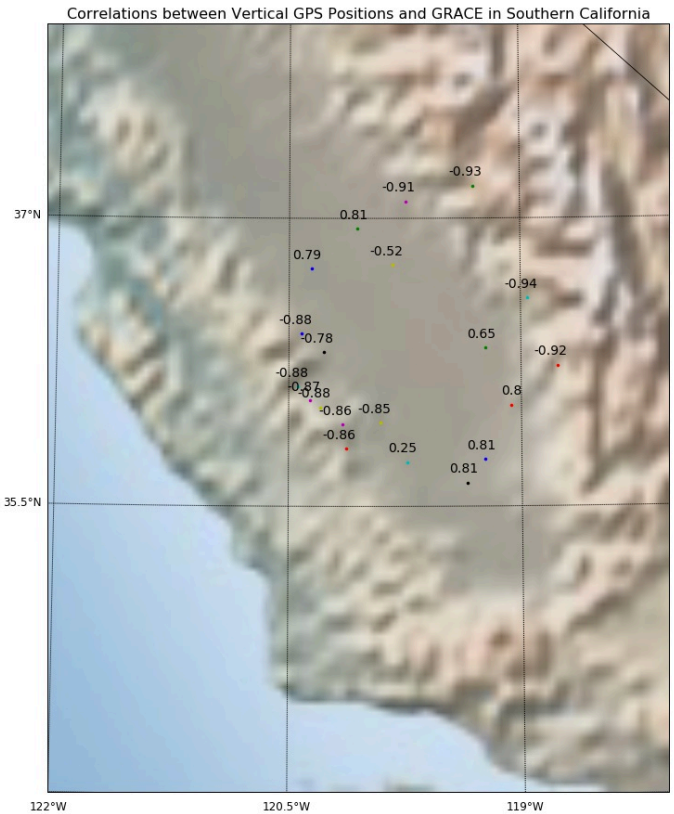
Computer-Aided Discovery

08 - Data Fusion GRACE+GPS Demo Last Checkpoint: an hour ago (autosaved)

Control Panel Logout

File Edit View Insert Cell Kernel Help Python 3

Markdown Cell Toolbar: None



Interpretation of Figure

- Points represent GPS station corrdinates
- Each station shows GPS vertical time series correlation with GRACE equivalent water thickness data product (of 1 deg x 1 deg monthly square)
- Stations with positive correlation are located on the aquifier
- Stations with negative correlation are located on bedrock

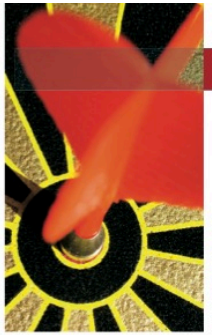
MIT Haystack Observatory
Massachusetts Institute of Technology

References

- **Computer-Aided Discovery: Towards Scientific Insight Generation with Machine Support.**
  Victor Pankratius, Justin Li, Michael Gowanlock, David M. Blair, Cody Rude, Tom Herring, Frank Lind, Philip J. Erickson, Colin Lonsdale, IEEE Intelligent Systems 31(4), July/August 2016
  http://doi.ieeecomputersociety.org/10.1109/MIS.2016.60

- **Computer Aided Detection of Transient Inflation Events at Alaska Volcanoes using GPS Measurements from 2005-2015**
  Justin Li, Cody Rude, David Blair, Michael Gowanlock, Thomas Herring, Victor Pankratius, Journal of Volcanology and Geothermal Research 327, Nov 2016
  http://dx.doi.org/10.1016/j.jvolgeores.2016.10.003

- **Improving Spacecraft Site Selection Through Computer-Aided Discovery And Data Fusion.**
  David Blair, Michael Gowanlock, Justin Li, Cody Rude, Tom Herring, Victor Pankratius, 47th Lunar and Planetary Science Conference (LPSC), 2016
  http://www.hou.usra.edu/meetings/lpsc2016/pdf/1987.pdf



EXPERT OPINION

Editor: **Daniel Zeng**, University of Arizona and Chinese Academy of Sciences, zengdaniel@gmail.com

Computer-Aided Discovery: Toward Scientific Insight Generation with Machine Support

Victor Pankratius, Justin Li, Michael Gowanlock, David M. Blair, Cody Rude, Tom Herring, Frank Lind, Philip J. Erickson, and Colin Lonsdale, *Massachusetts Institute of Technology*

Recent technical advances have enabled growing data volumes in astronomy and geoscience.[1] Scientists are now challenged to create insights from a deluge of data.[2] We are in the midst of a fundamental change, transitioning swiftly from an era in which data was scarce to an era in which data exceeds our ability to extract meaning from it, and the scientific community is facing an *analysis wall*.

Planetary and space-based sensor networks are generating continuous streams of data to monitor our environment, characterize diverse phenomena, and, in many cases, predict natural hazards. Computer science and intelligent systems are now called to action to develop a new breed of systems to extract insight from large datasets and different types of datasets (such as optical, radar, and GPS time series). Furthermore, data fusion from different instruments is gaining importance in making new discoveries of natural phenomena and ruling out false positives, especially because making the right connections can often be nonintuitive for humans.

Looking at information processing from the semiotics point of view, there are several layers. As Figure 1 shows, digitized sensor signals become data that represents the starting point for more complex analyses. The syntax layer essentially provides syntactically valid data—that is, numbers that codify actual valid measurements. The next layer on top typically aims to introduce semantics (for example, "this data represents feature X") by employing data exploration, analysis, and mining techniques. However, this alone is not enough to advance scientific progress, and scientists need support for pragmatics: What does it imply that a certain feature has been identified? What does a finding mean, and how does it fit into the big theoretical picture? Does it contradict or confirm previously established models and findings? How can the researcher test concepts and ideas effectively? Many of the implementation tasks that result from such questions are currently left to the individual researcher, who must artfully deduce the tools and workflows that lead to adequate answers.

Why Scientists Need Machine Support for Discovery Search

We can view the scientific discovery process essentially as a search process. This search space is defined not only by the large and diverse scientific datasets themselves, but also by the choices humans can make in the workflow processing that data (for example, choosing the parameters, order, or which algorithm, method, or filter to use in a certain stage of a processing pipeline). The workflow is assumed to be a sequence of processing steps that has the expressive power of a Turing machine.

As our case studies show, the choices made in the configuration of the processing workflow can drastically affect our ability to make discoveries. For example, if a set of processing steps highlights large-scale phenomena in a data product, discoveries of previously unknown small-scale phenomena will be suppressed. In addition, some natural phenomena might be rare or counter-intuitive in nature, so humans require machine assistance in configuring a potentially large parameter space to create data processing and discovery workflows.

The value of discovery automation also arises because of the sheer size of datasets on the order

1541-1672/16/$33.00 © 2016 IEEE
Published by the IEEE Computer Society

Thanks!

@vpankratius

pankrat@mit.edu

ACI, AGS INSPIRE
PI Pankratius
ACI1442997, AGS-1343967

AIST NNX15AG84G
PI Pankratius

BOSE Foundation

Acknowledgement: This material is based upon work supported by the NSF and NASA. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the view of the NSF.