Computer-Aided Discovery in Earth Science

NASA ESTF, 6/14/2017

Victor Pankratius

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

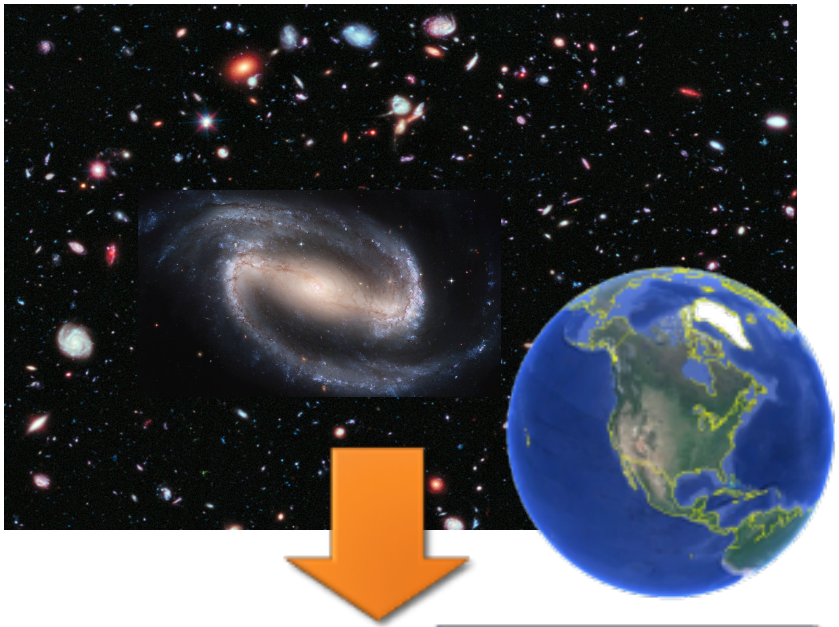
pankrat@mit.edu
victorpankratius.com

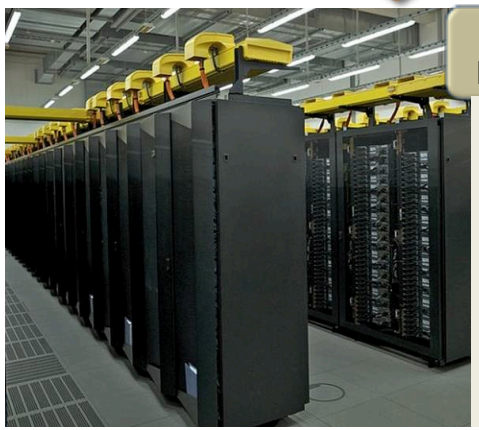
NASA AIST NNX15AG84G
V. Pankratius (PI), T. Herring (co-I)

Many thanks to contributors:
D. Blair, P. Erickson, J. Li, F. Lind, M. Gowanlock, G. Rongier, B. Rideout, C. Rude

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
- Scientific discovery process = search process across multidimensional data
- Scientific question answering
  → matching theory variants to empirical data sets





Software-based Instruments / Backends

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

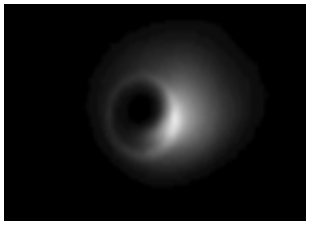
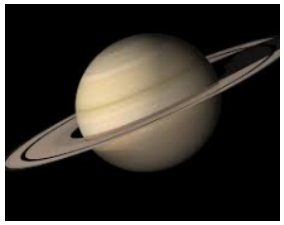
Computer-Aided Discovery

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

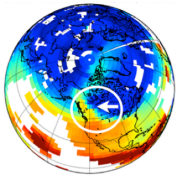

ACI, PI Pankratius
1442997

Computer-Aided Discovery: Overview

Empirical Data

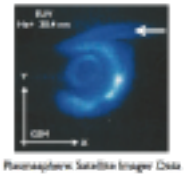
Theory

Instrument 1



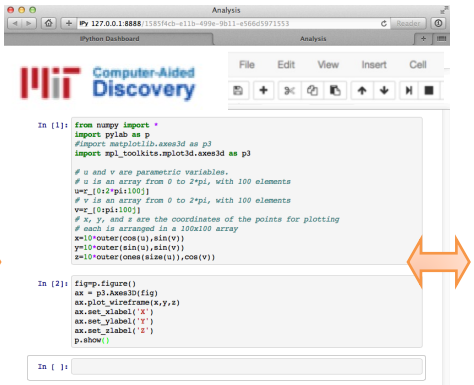
"physics"

Instrument 2



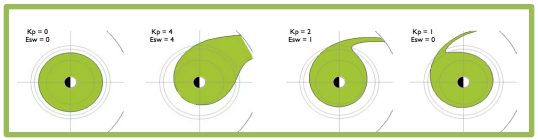
Data /Compute Center

Cloud-Based Analysis Interface



Scientist

Artificial Intelligence

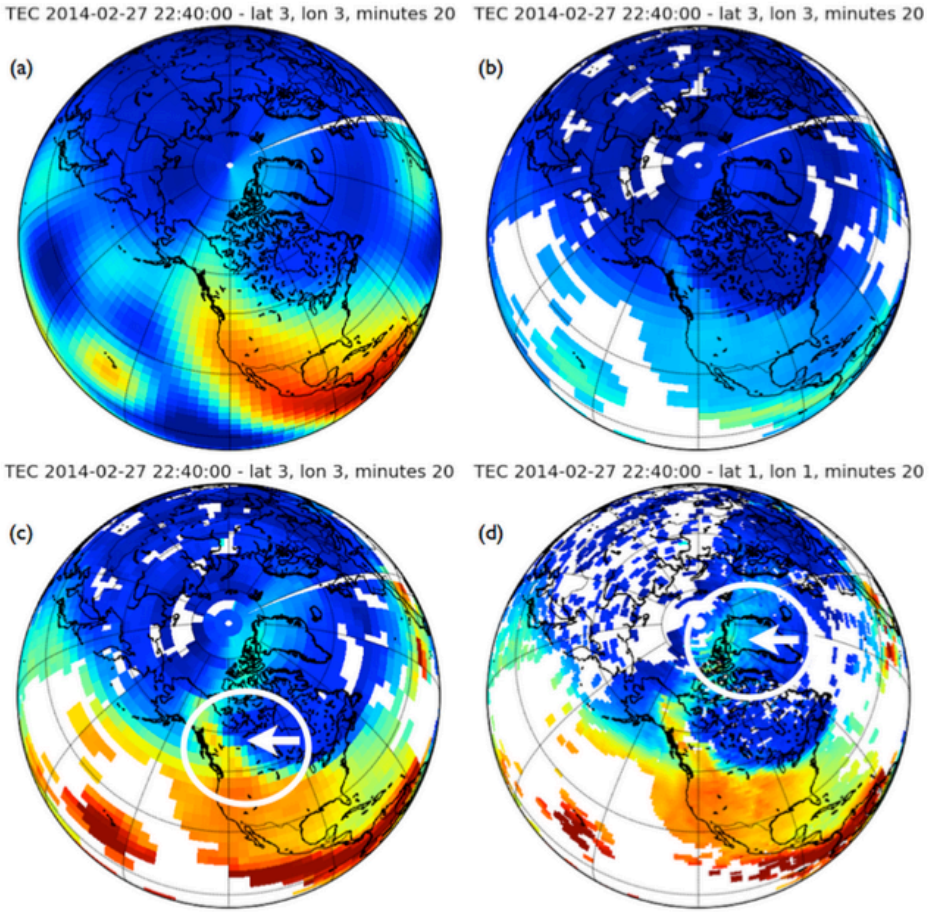


Model 1 Model 2 Model 3 Model 4

Search results – would you like to find more like:

Gather scientist's feedback

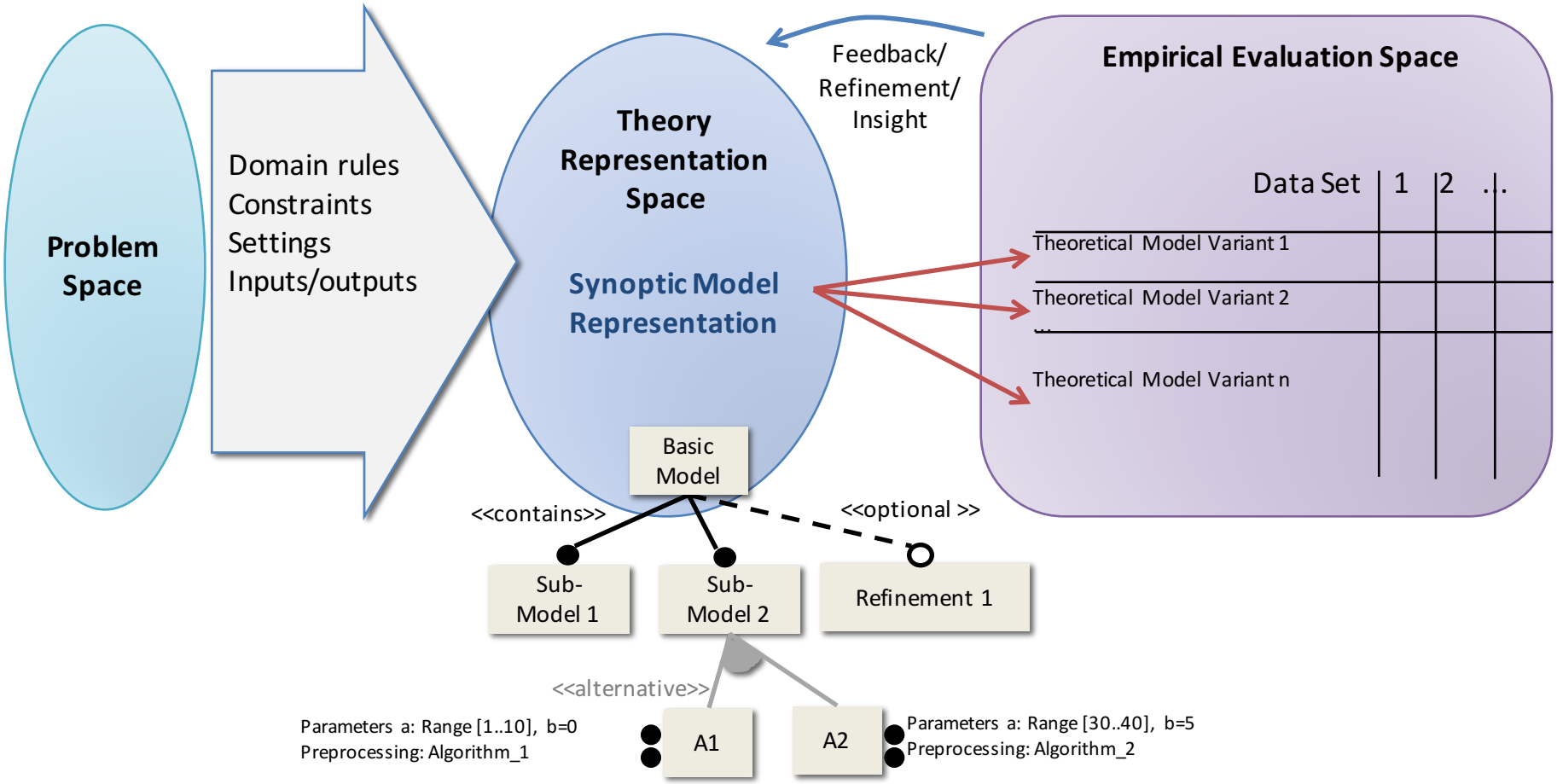
Example: Impact of Choice of Processing Workflows



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

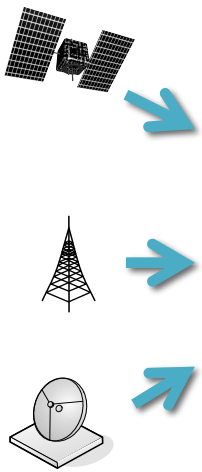
- Choice of workflow processing parameters can affect the visibility and discovery of interesting phenomena

Computer-Aided Discovery: Conceptual Approach



Computer-Aided Discovery: Infrastructure

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



Data Integrators, indexing, etc.

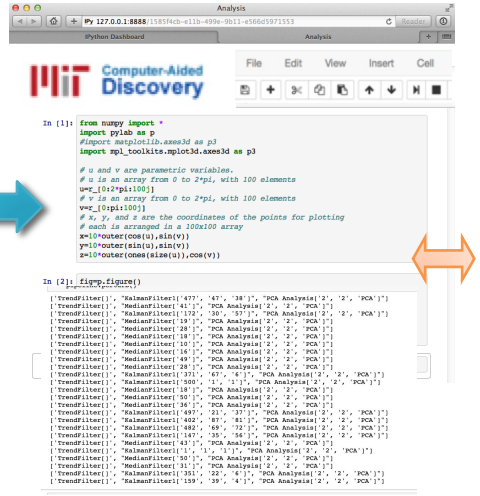
- Generate compute jobs
- Move computation to data locations

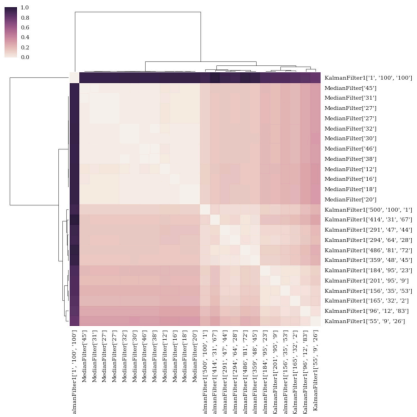
Execution Crawlers

- Computer-Aided Discovery Engine

Web Service APIs

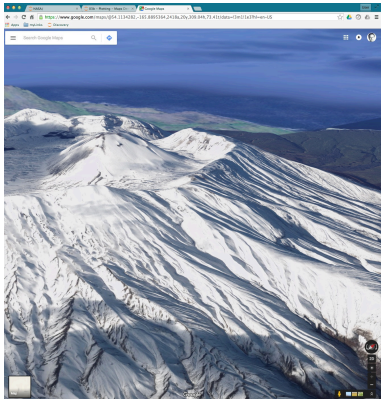
Direct service integration into other tools

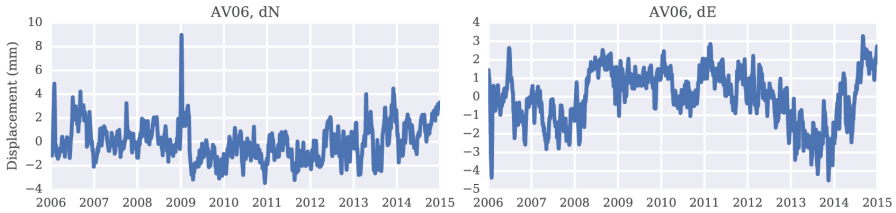




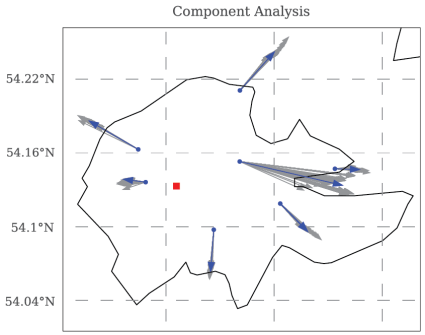
Application Examples

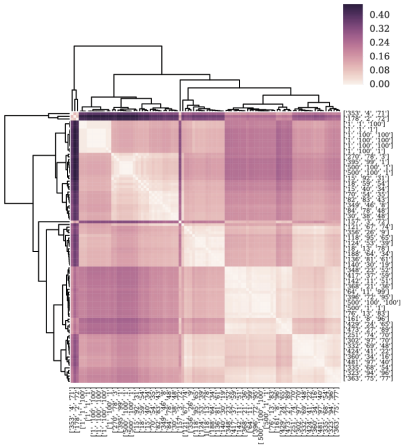
Geoscience / Volcanics



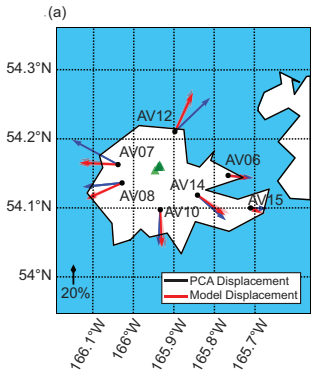


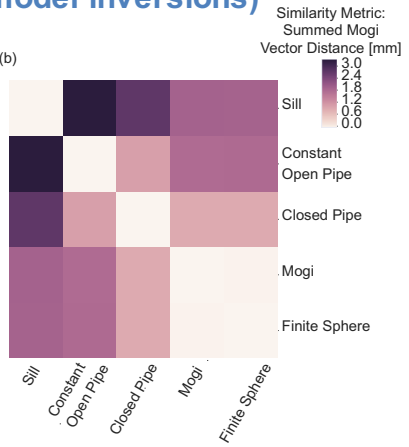
Variants of earth deformation models





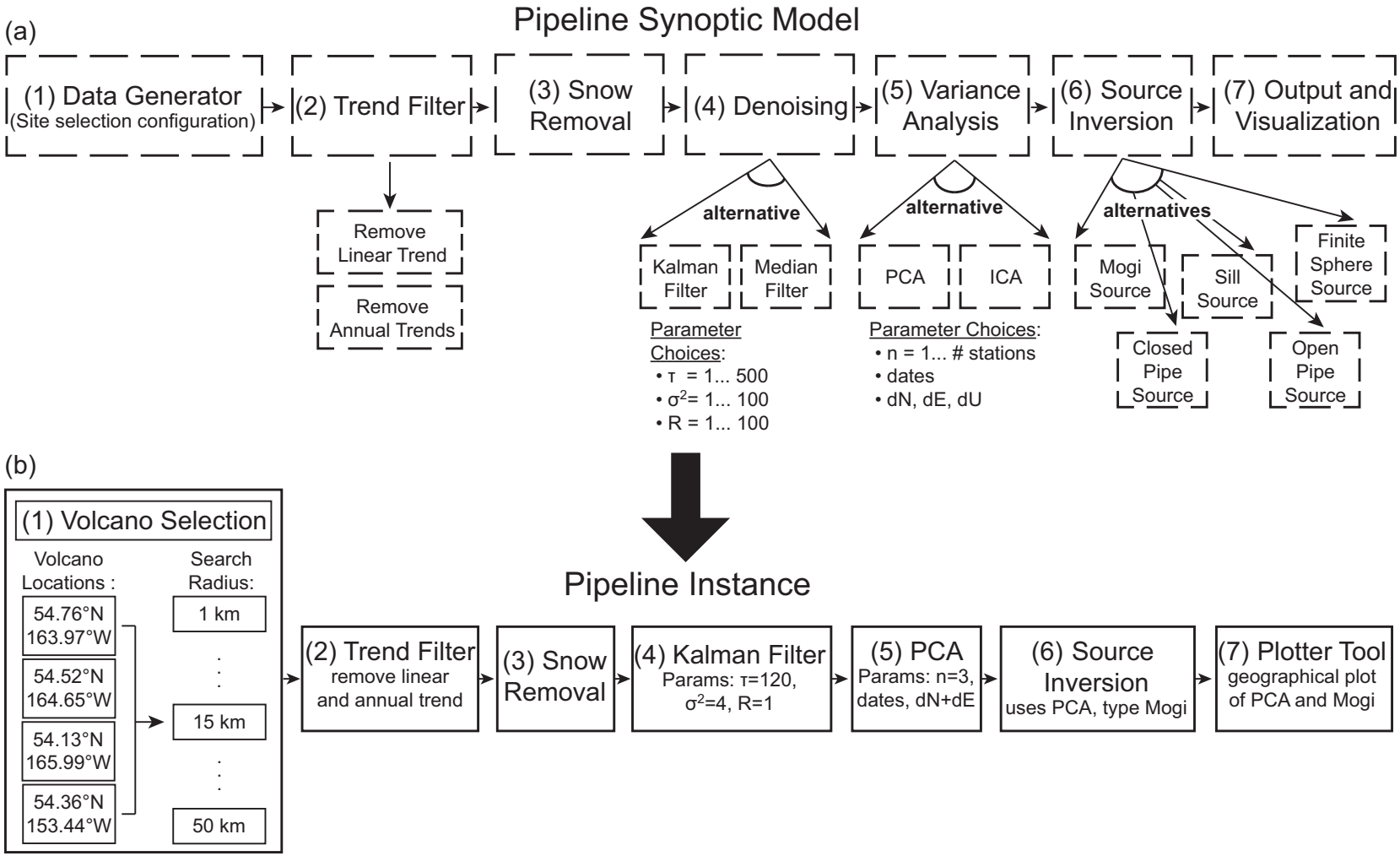
Variants of magma source models (5 different source model inversions)





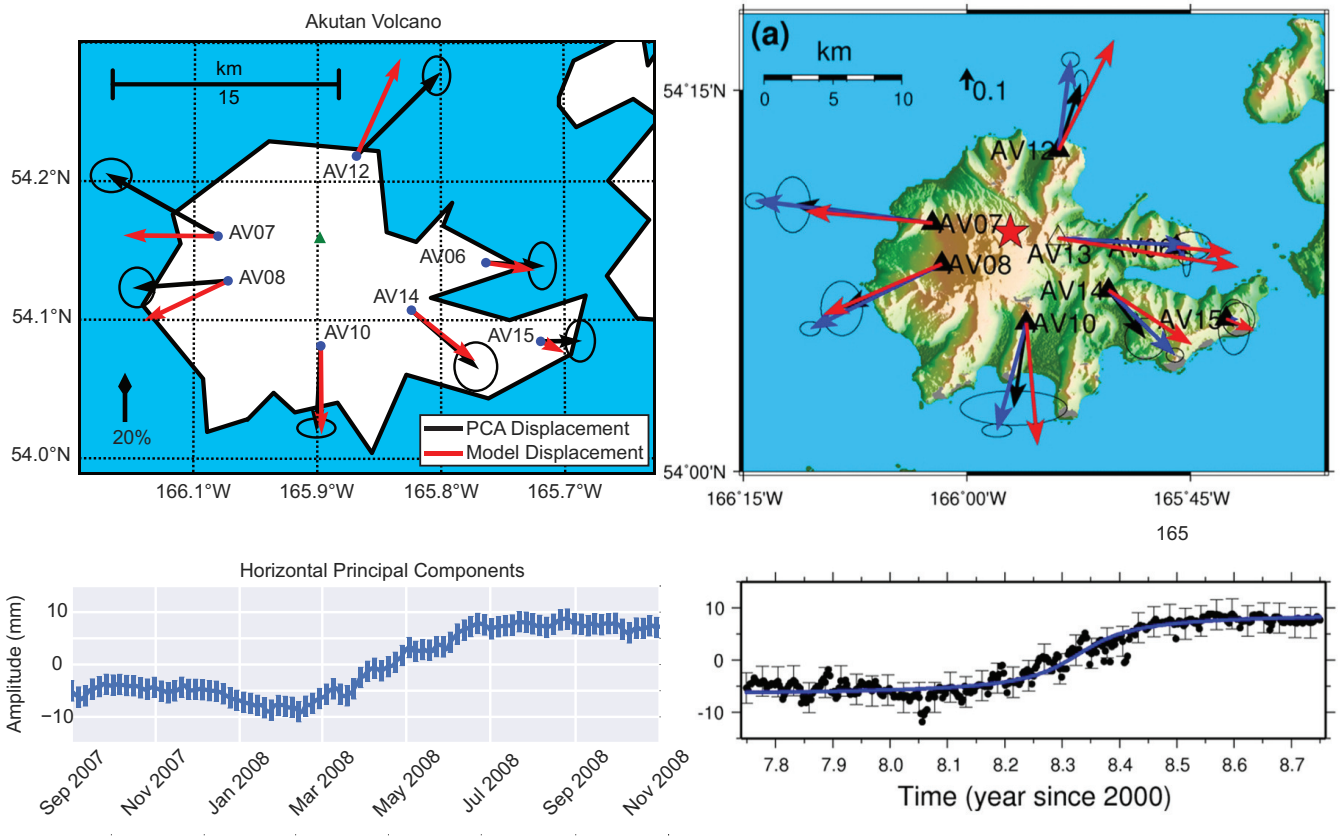
[J.Li, C.Rude, D.Blair, M.Gowanlock, T.Herring, V.Pankratius. Journal of Volcanology and Geothermal Research, 2016]

Volcanics: Event Discovery Pipeline



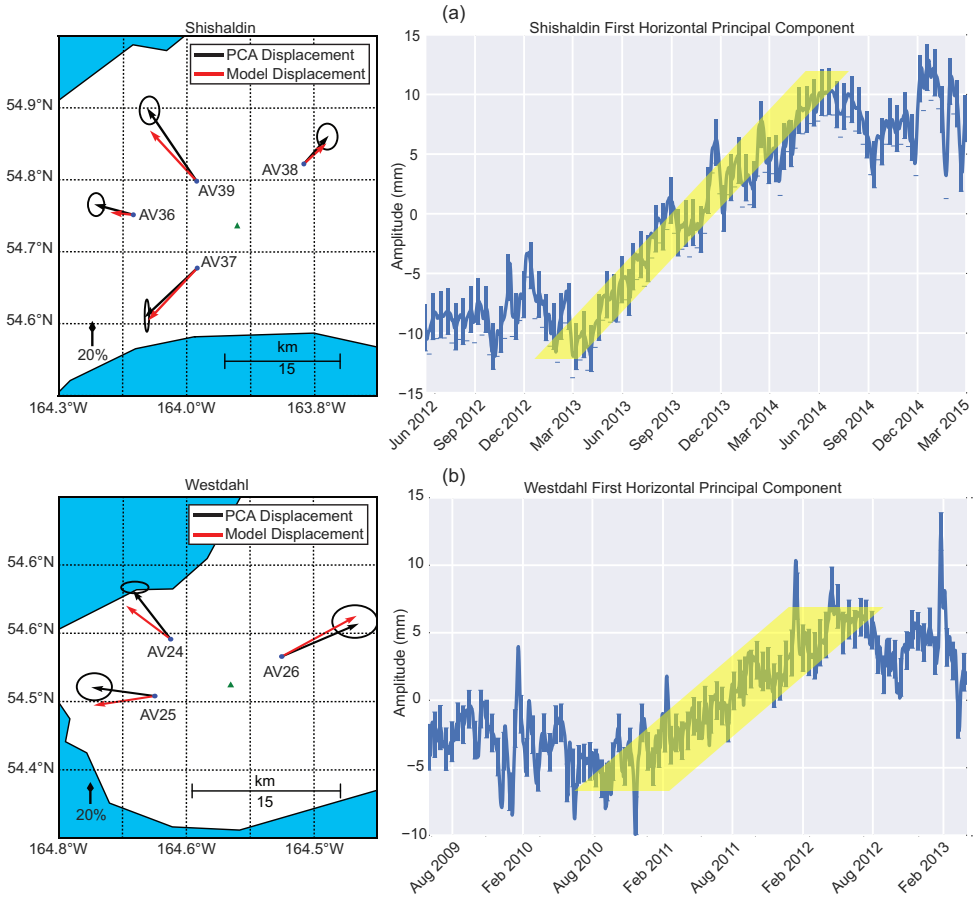
Volcanics: Validation of Methodology – Example

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

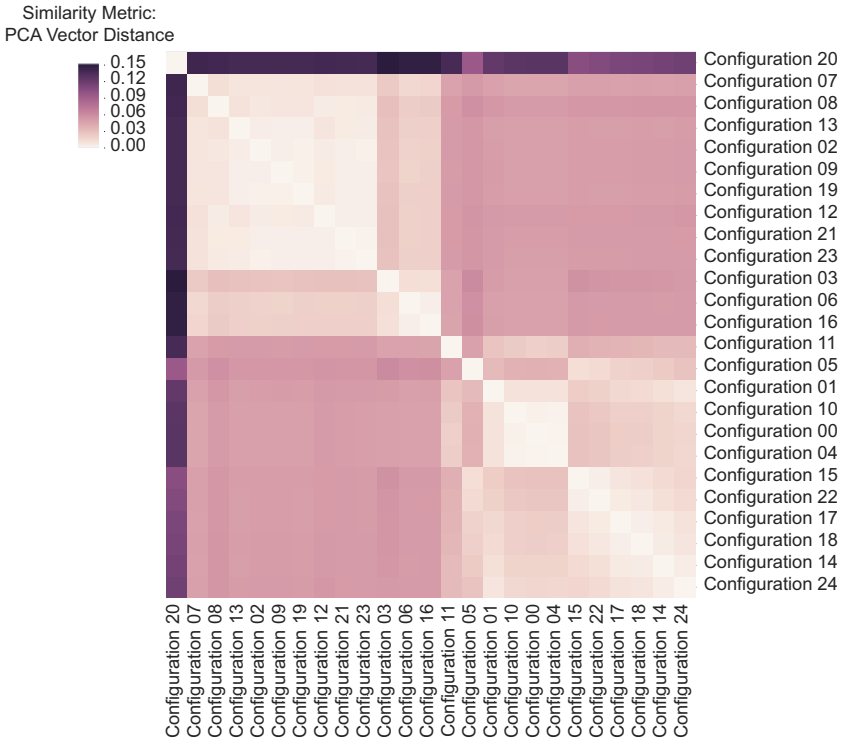


Figure 6: A heatmap showing the sum of the euclidean distance between the eigenvectors at each PBO station for each sensitivity analysis run at Akutan. Colors reflect the distance between results from different configurations as measured by the difference between PCA eigenvectors at each PBO station. 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

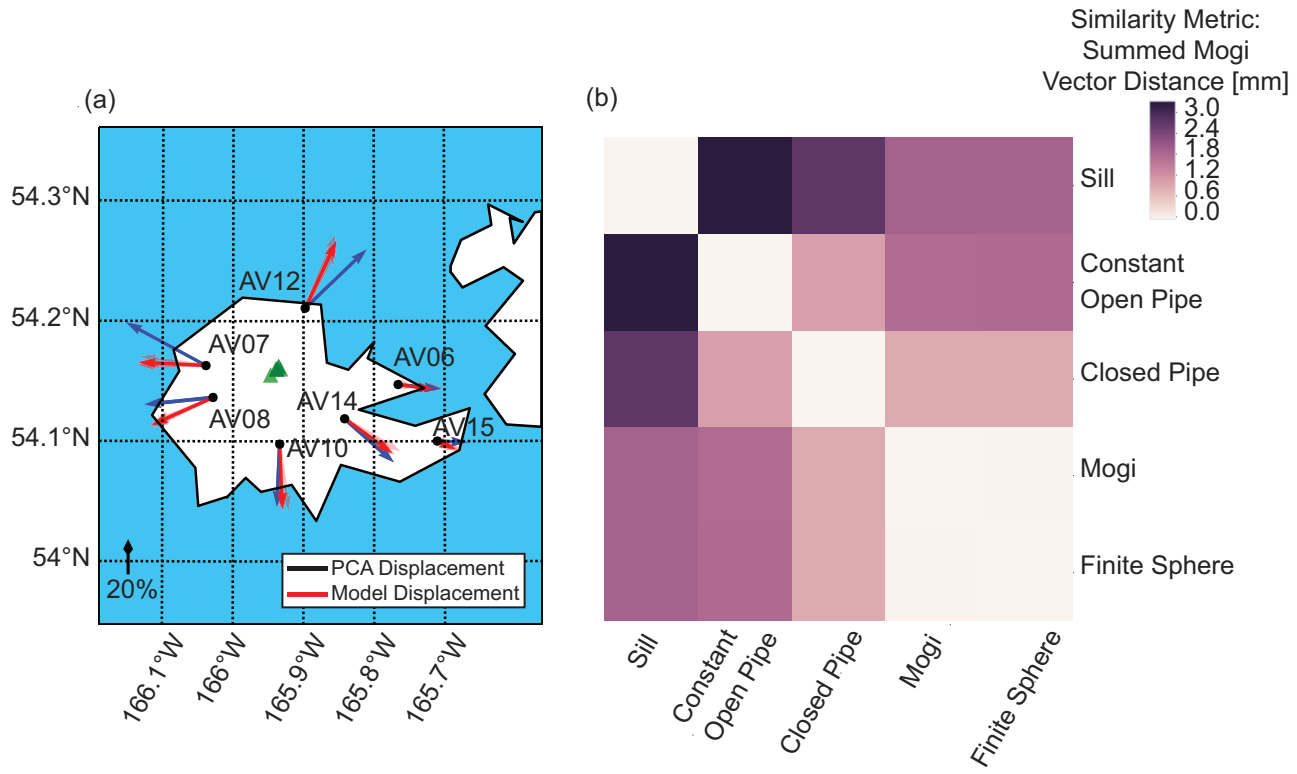
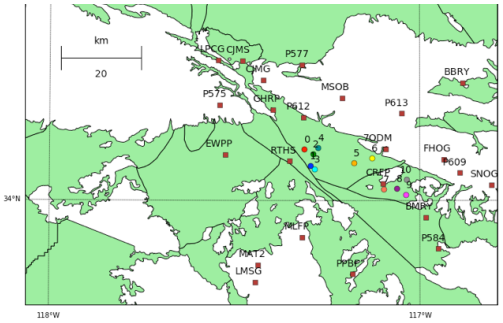


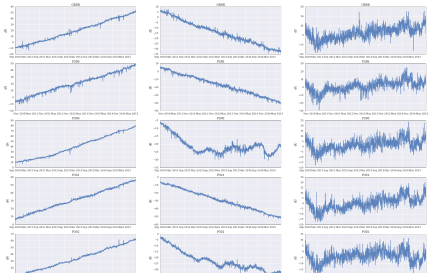
Figure 7: Comparison of 5 different source model inversions, Mogi, finite sphere, closed pipe, constant (width) open pipe, and sill, for the motion described by the eigenvectors for Akutan displayed on (a) a geographical plot with the eigenvectors in blue, the different inversion displacements in red (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

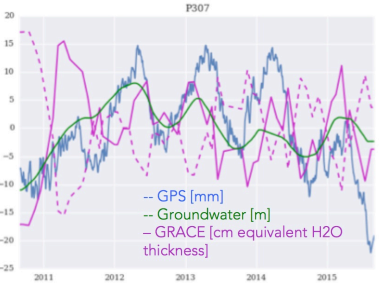
Groundwater Studies

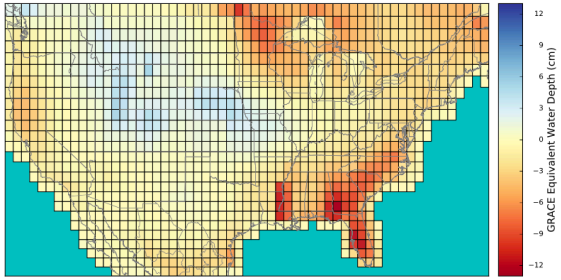


GPS (North, East, vertical)

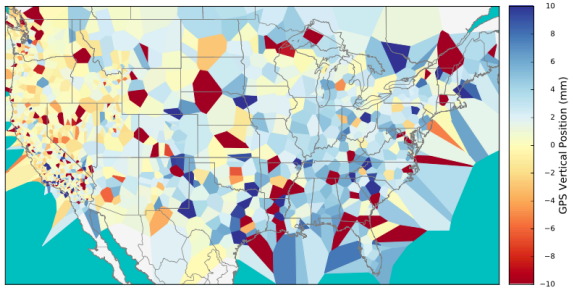


+ Data Fusion: GRACE,...



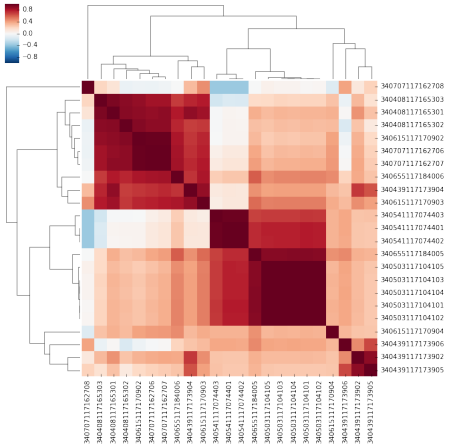


GRACE, Feb 2012, trends (annual, seasonal, linear) removed



vertical position GPS measurements from the PBO Feb 2012, Voronoi interpolation

[Surface Response to Changes in Terrestrial Water Storage Across the United States, C. Rude, J. Li, G. Rongier, M. Gowanlock, T. Herring, V. Pankratius, in preparation]

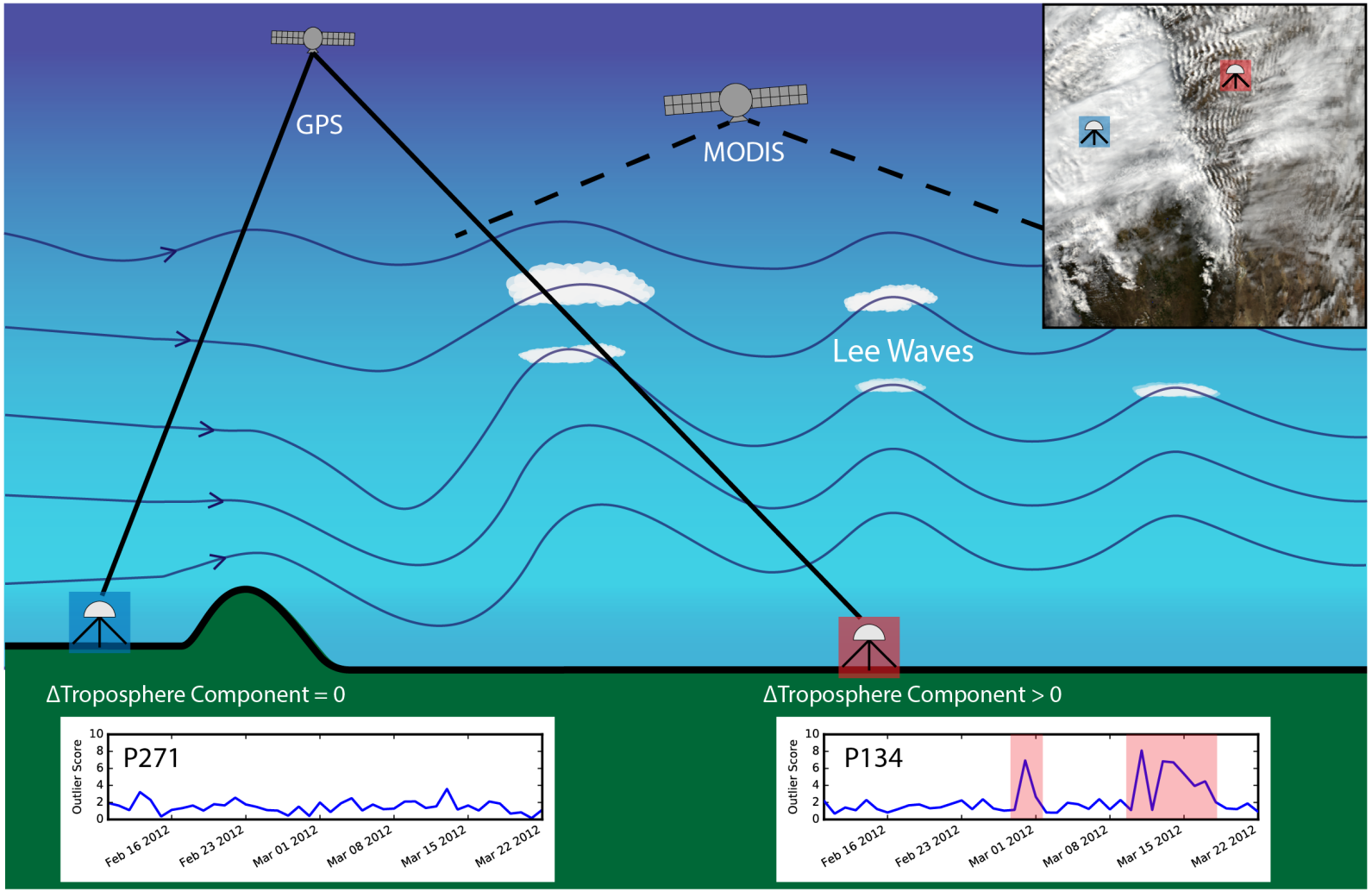


- Group wells by behavior
- Red groups represent wells whose water levels are highly correlated in time

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

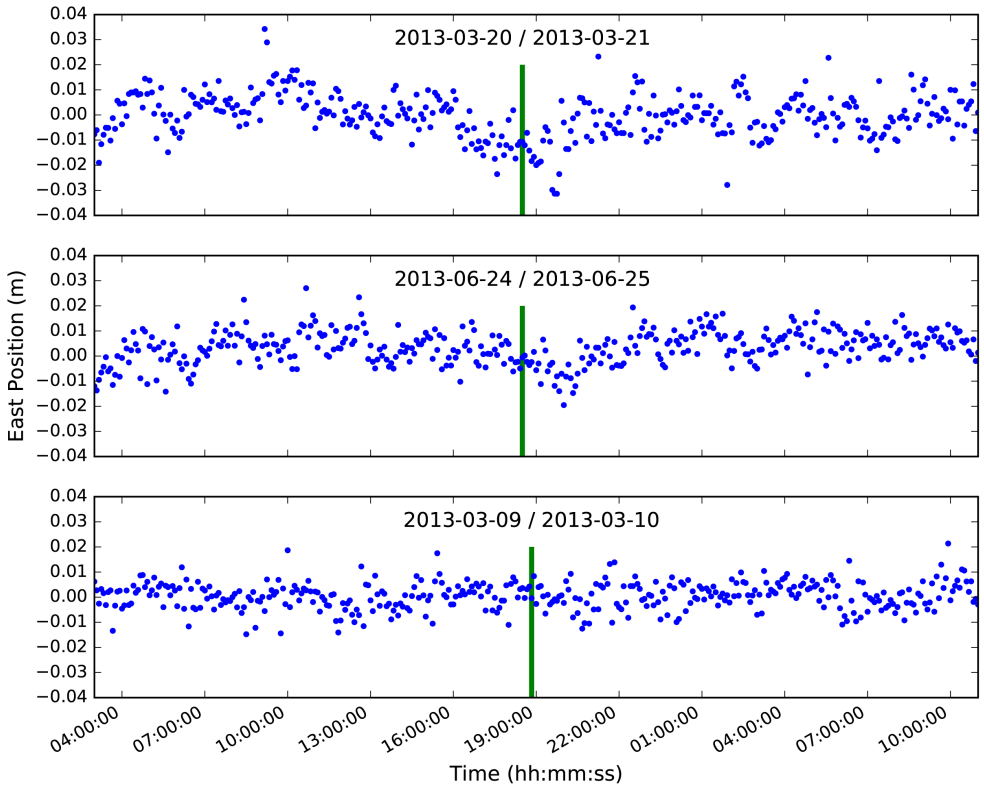
Application Examples

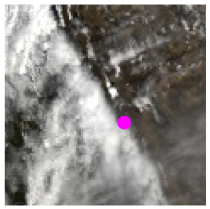
Lee Wave Detection: Data Fusion & Deep Learning

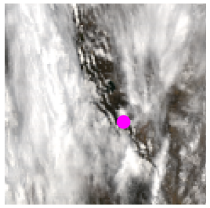


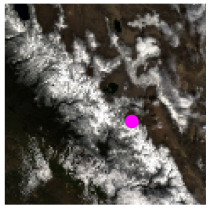
[Automatic Detection of Atmospheric Mountain Waves using Data Fusion and Deep Learning, J. Li, C. Rude, V. Pankratius, M. Gowanlock, T. Herring, in preparation]

Lee Waves: MODIS + GPS Data Fusion





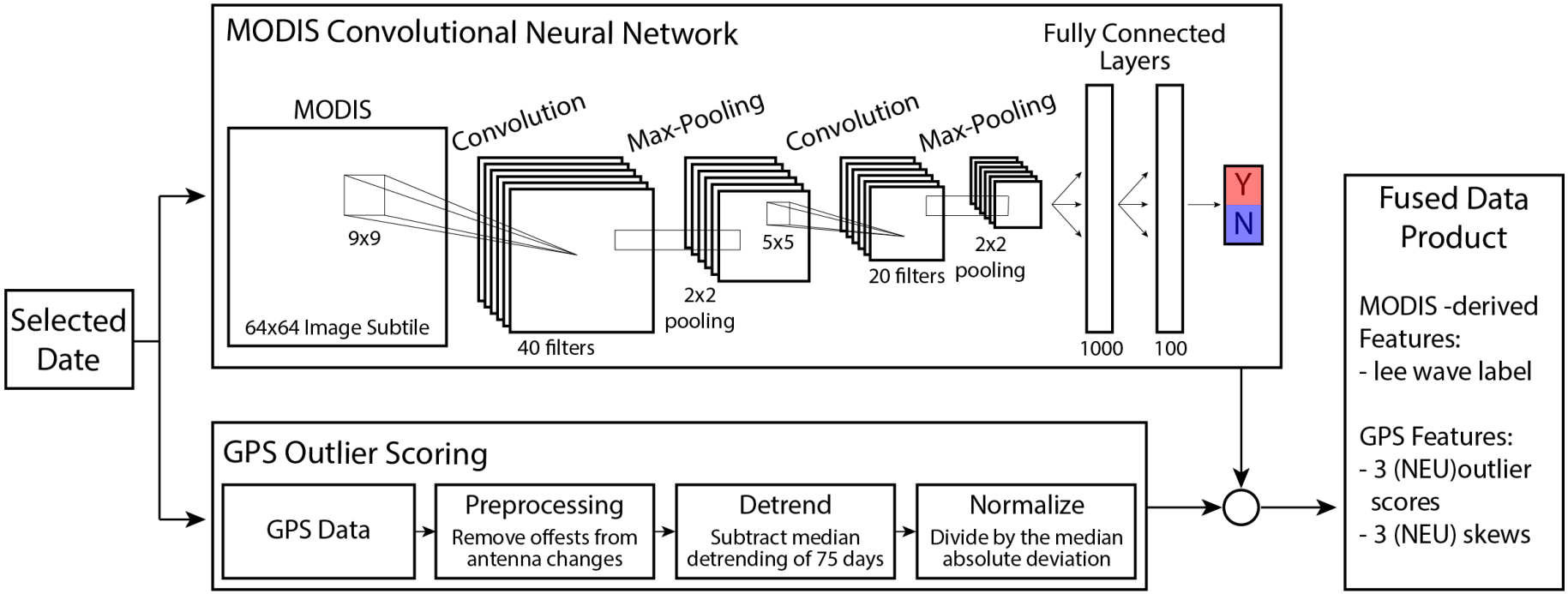




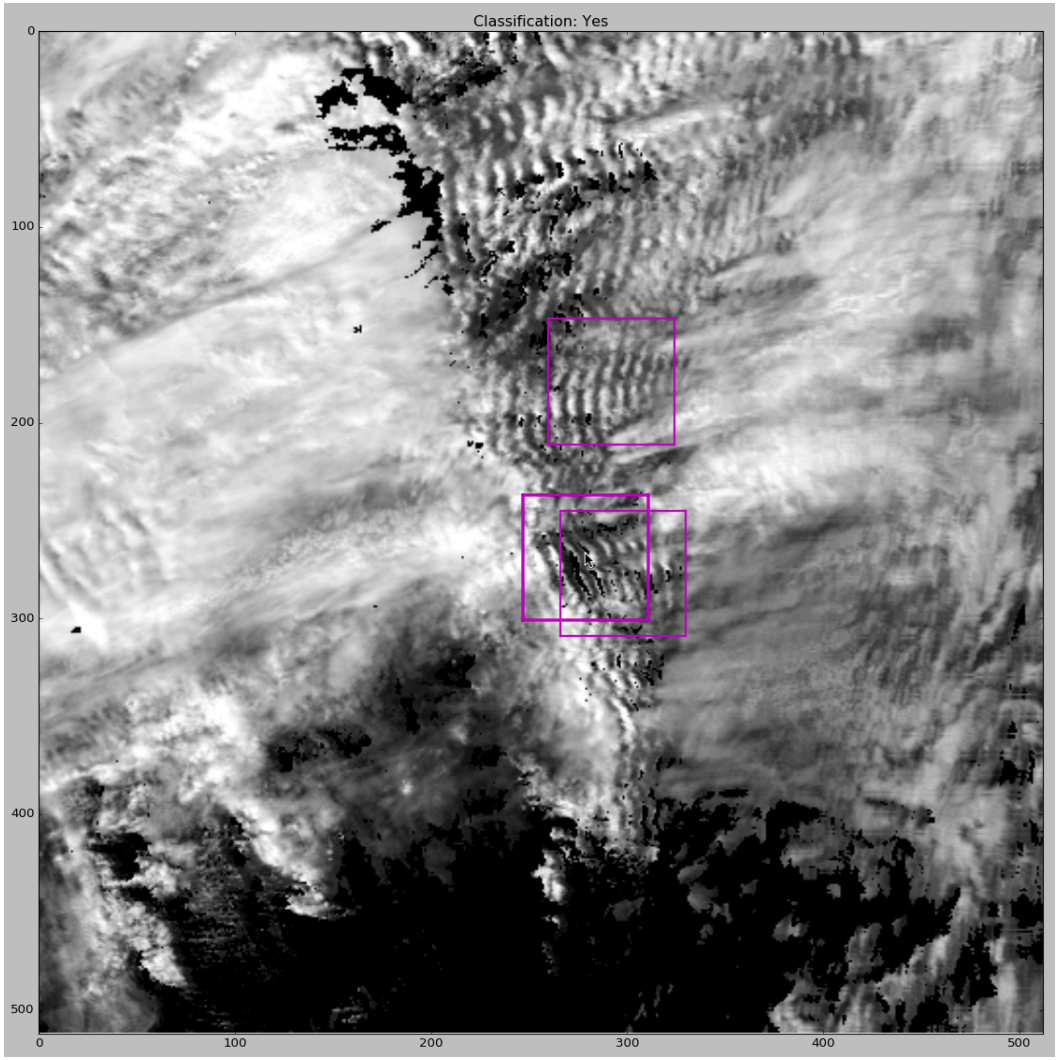
- Improving capability of lee wave detection
- Enables higher time resolutions
- GPS data: 5-min NGL vs 1-day PBO
- Outlier patterns around time of lee wave events from MODIS are better represented

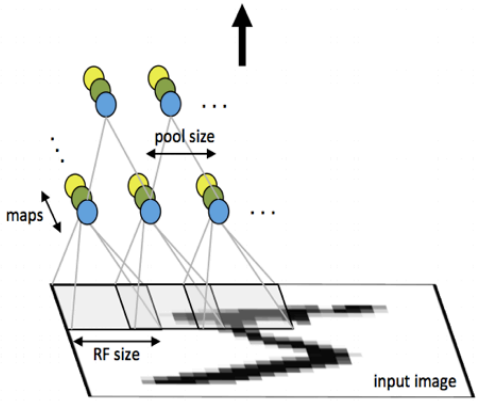
Lee Waves: Data Fusion & Deep Learning

- Combining data streams from GPS data and MODIS imagery
- Apply convolutional neural network (Theano) to detect lee waves in the MODIS image
- Calculated an outlier score for GPS stations
- Fuse into a single product for analysis



Lee Waves: GUI for Neural Network Training

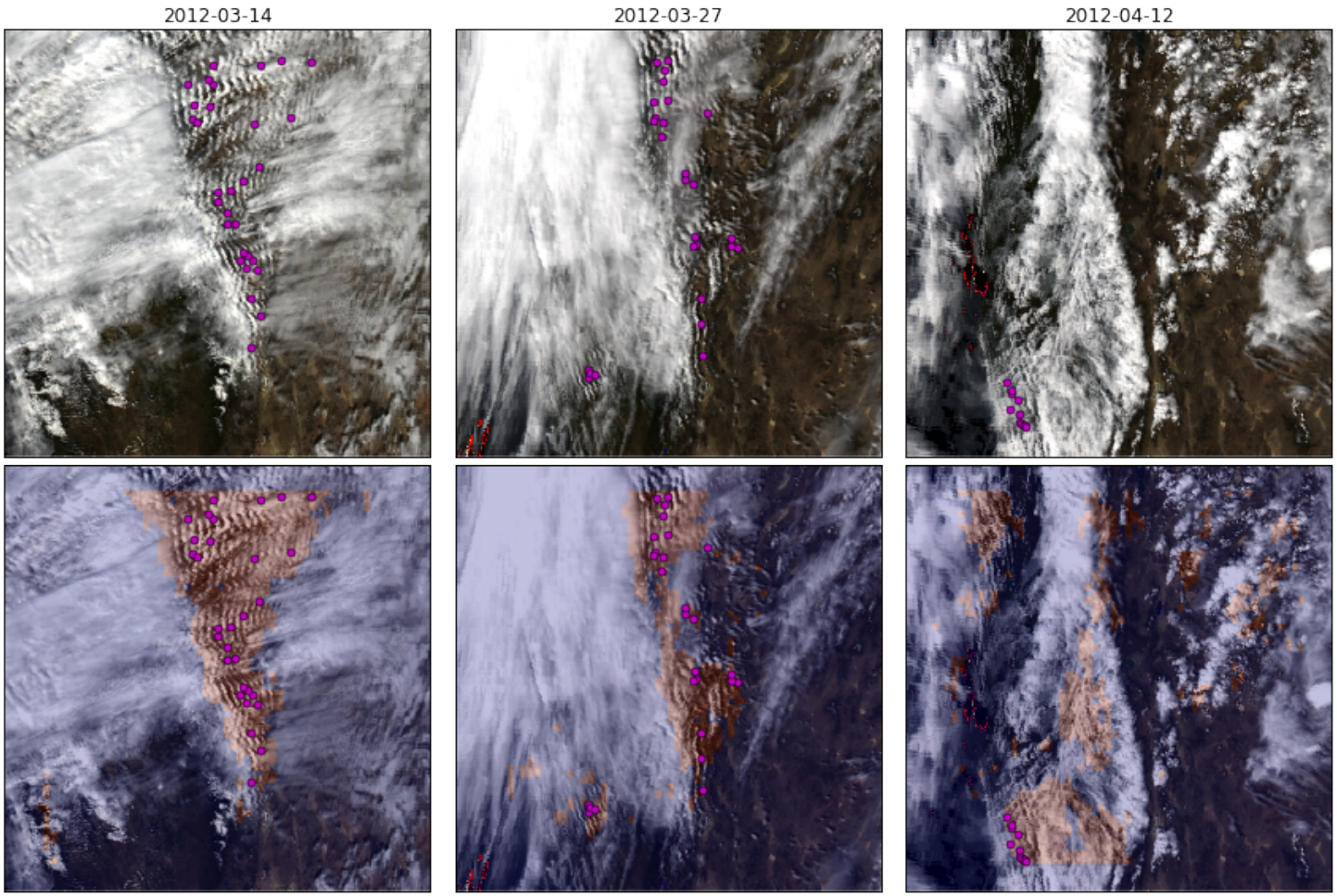




http://ufldl.stanford.edu/tutorial/supervised/ConvolutionalNeuralNetwork/

Supervised learning on 64x64 subtiles

Lee Waves: Neural Network Detection Results



3/14/2012 3/27/2012 4/12/2012

Victor Pankratius – AIST NNX15AG84G
MIT Haystack Observatory
Massachusetts Institute of Technology

Application Example: 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

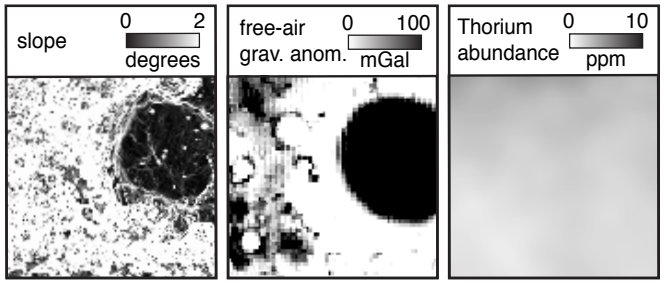
Planetary Protection:

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

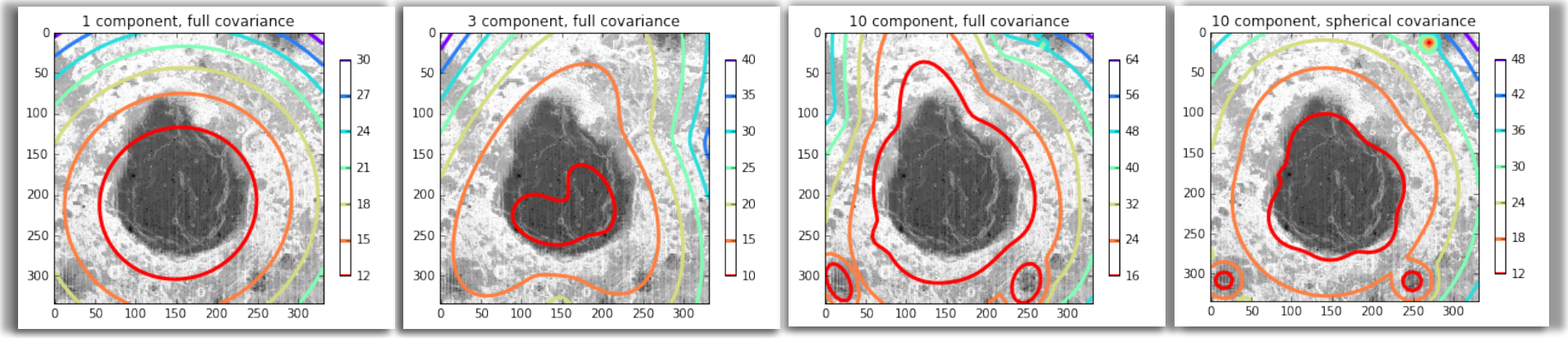
Application Examples

Planetary Science / Site Selection: Moon

Moon Example:



Variants of combined site selection models

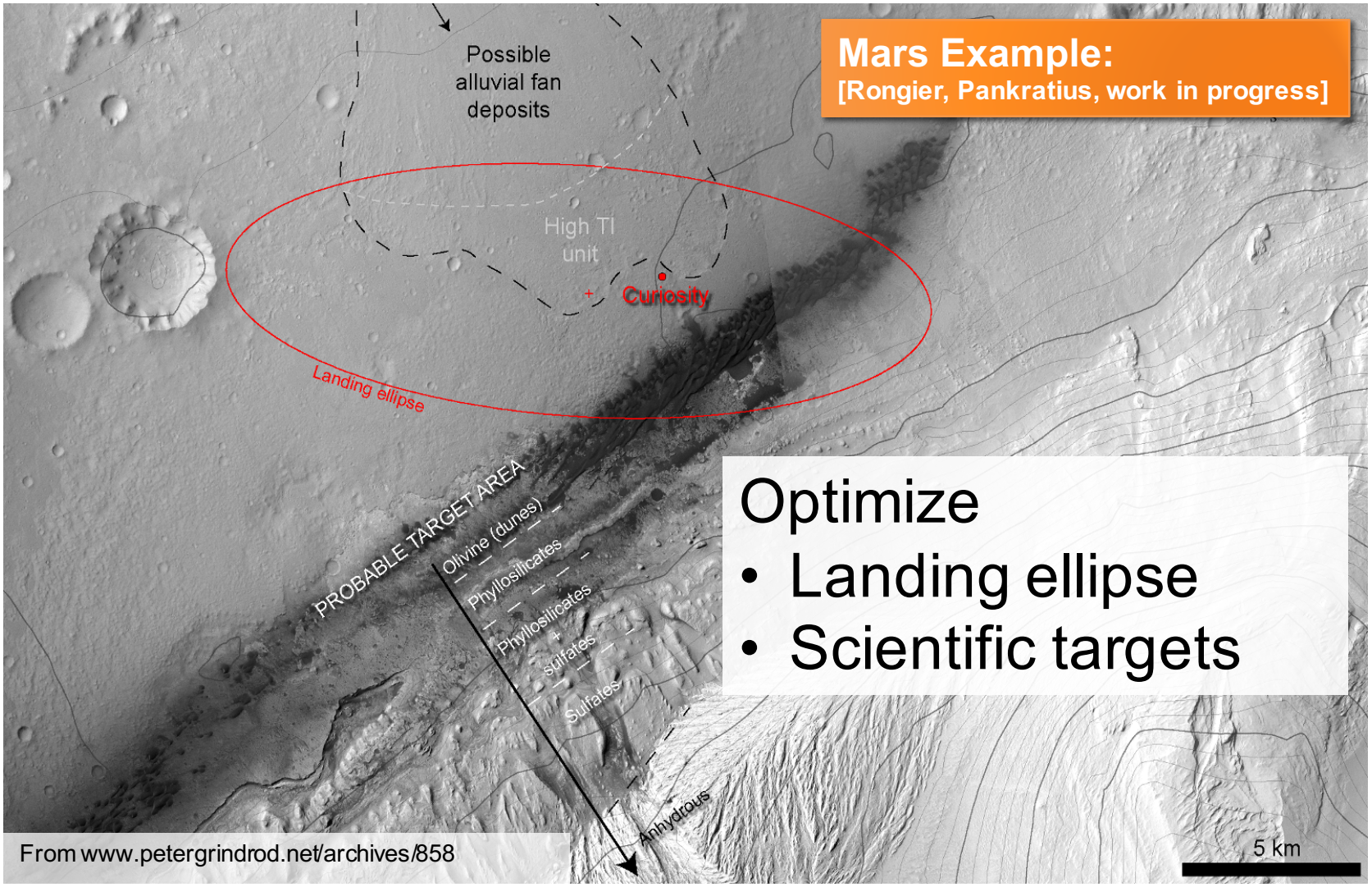


Plotted Over Overall Rank (darker = better)

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

Application Examples

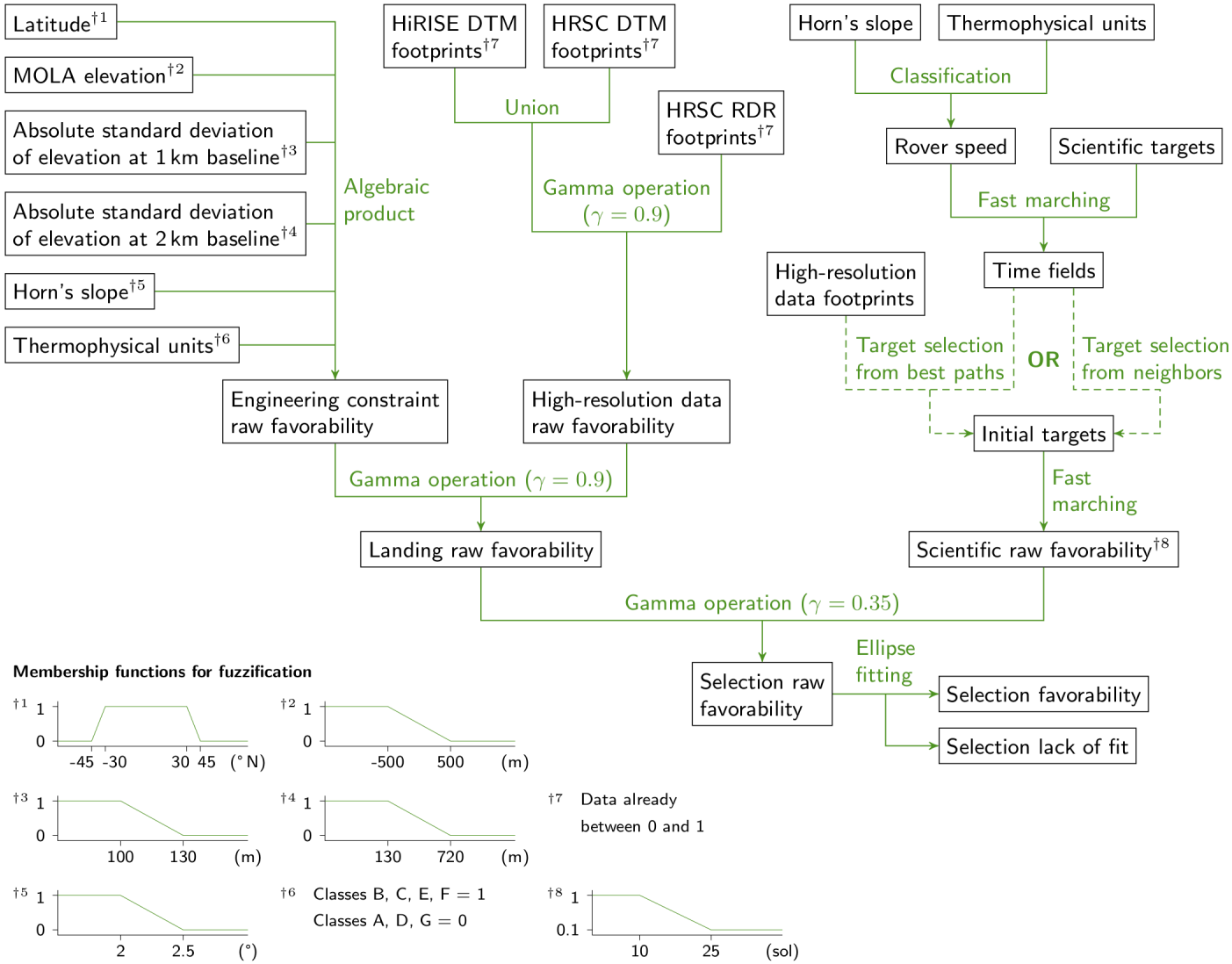
Planetary Science / Site Selection: Mars



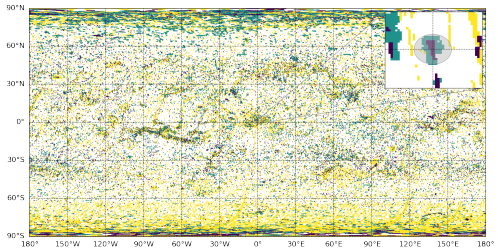
Mars Example:
[Rongier, Pankratius, work in progress]

Optimize
- Landing ellipse
- Scientific targets

Planetary Science / Site Selection: Mars Workflow



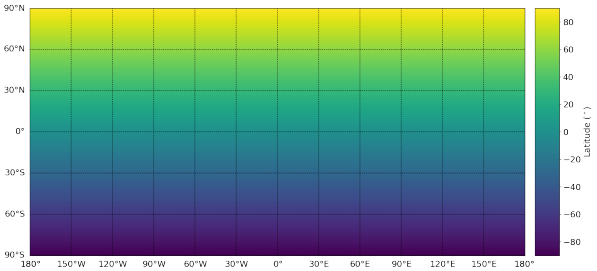
- HiRISE (camera, 0.25-0.13 m/px)
- CRISM (spectrometer, 18-36 m/px)
- MOC (camera, 1.5-12 m/px)



Membership functions for fuzzification

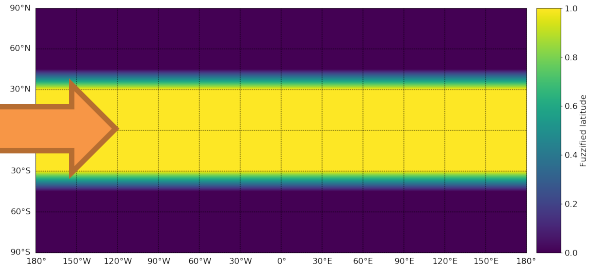
Application: Mars Site Selection

Initial area



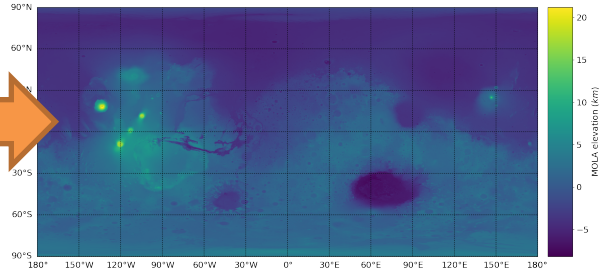
1) Area fuzzification:

1: high utility, 0: no utility

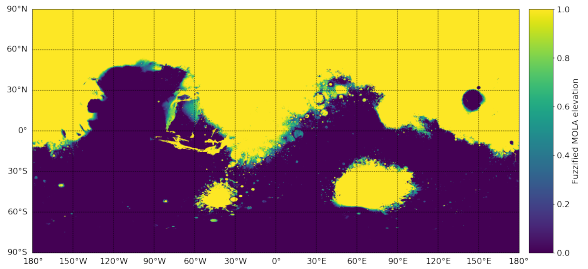


2) Criteria fuzzification

e.g., elevation

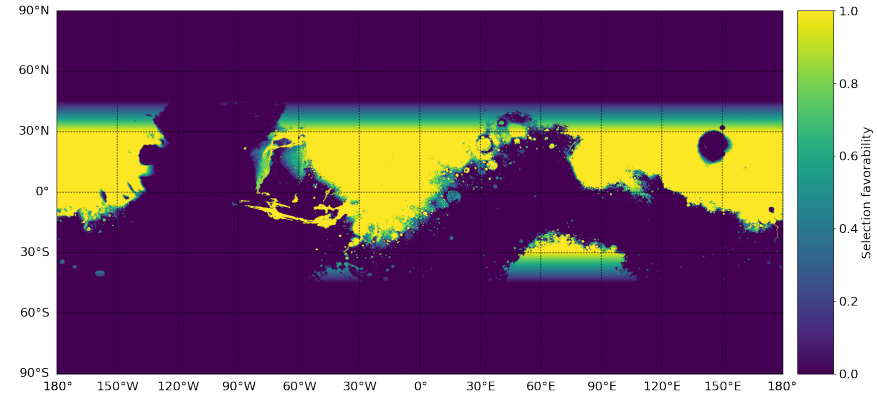


3)



Compute fuzzy OR, AND, etc.

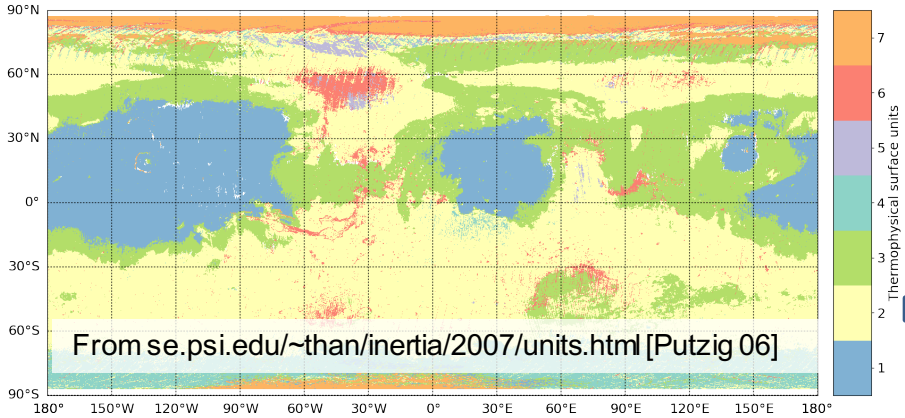
4) Intermediate Product



Application: Mars Site Selection

5a) 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?

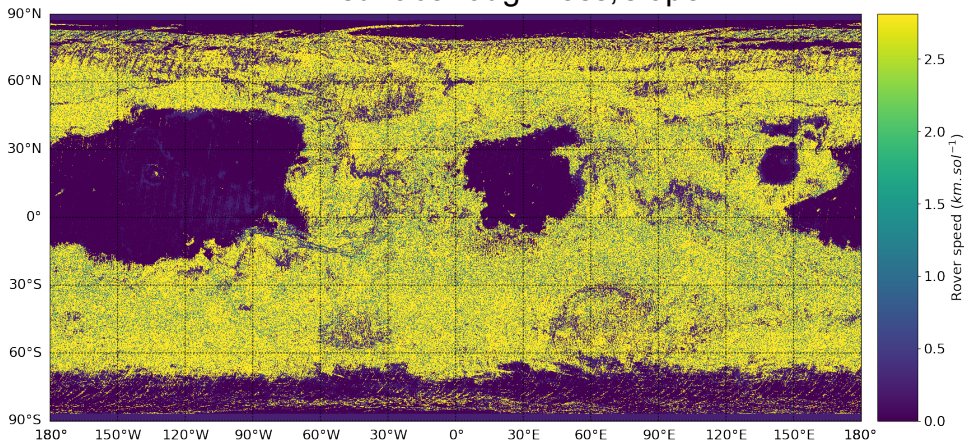
5 - As 2, but little or no fines

6 - Rocks, bedrock, duricrust, and polar ice

7 - As 1, thermally thin at higher inertia

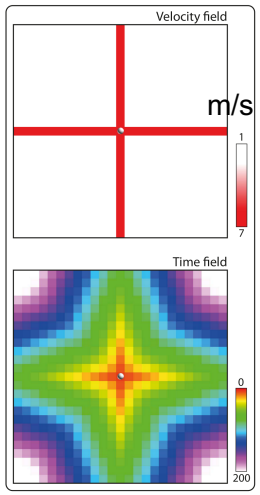
5b) Rover speed

depending on terrain, surface roughness, slope



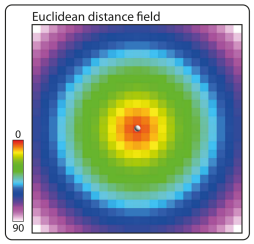
Application: Mars Site Selection

6) Fast Marching Algorithm



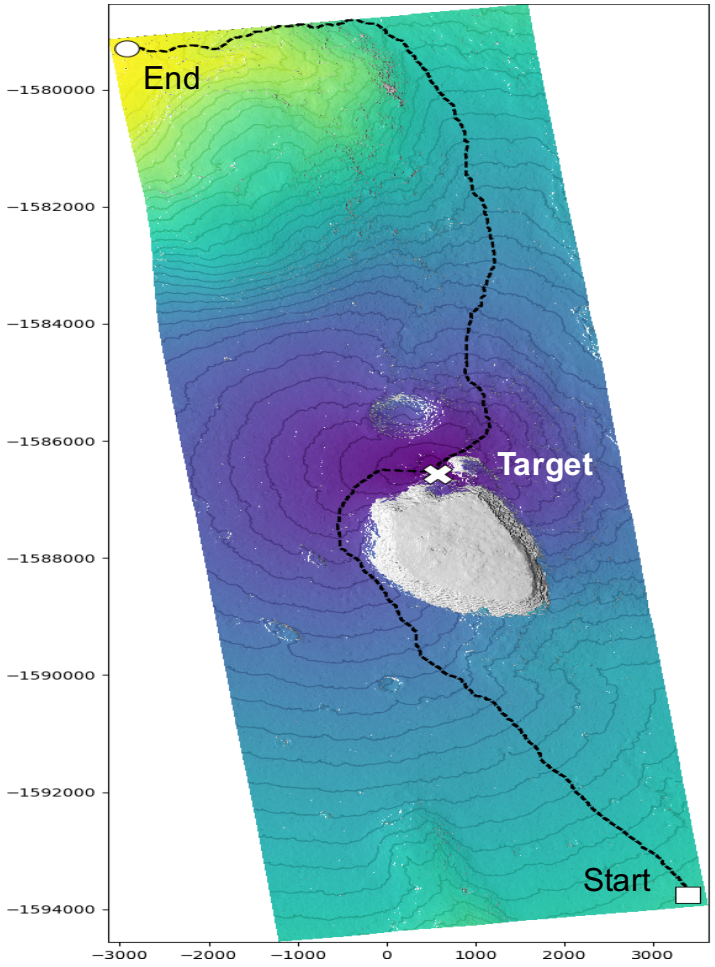
Define a velocity profile over space, landing site in the middle

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

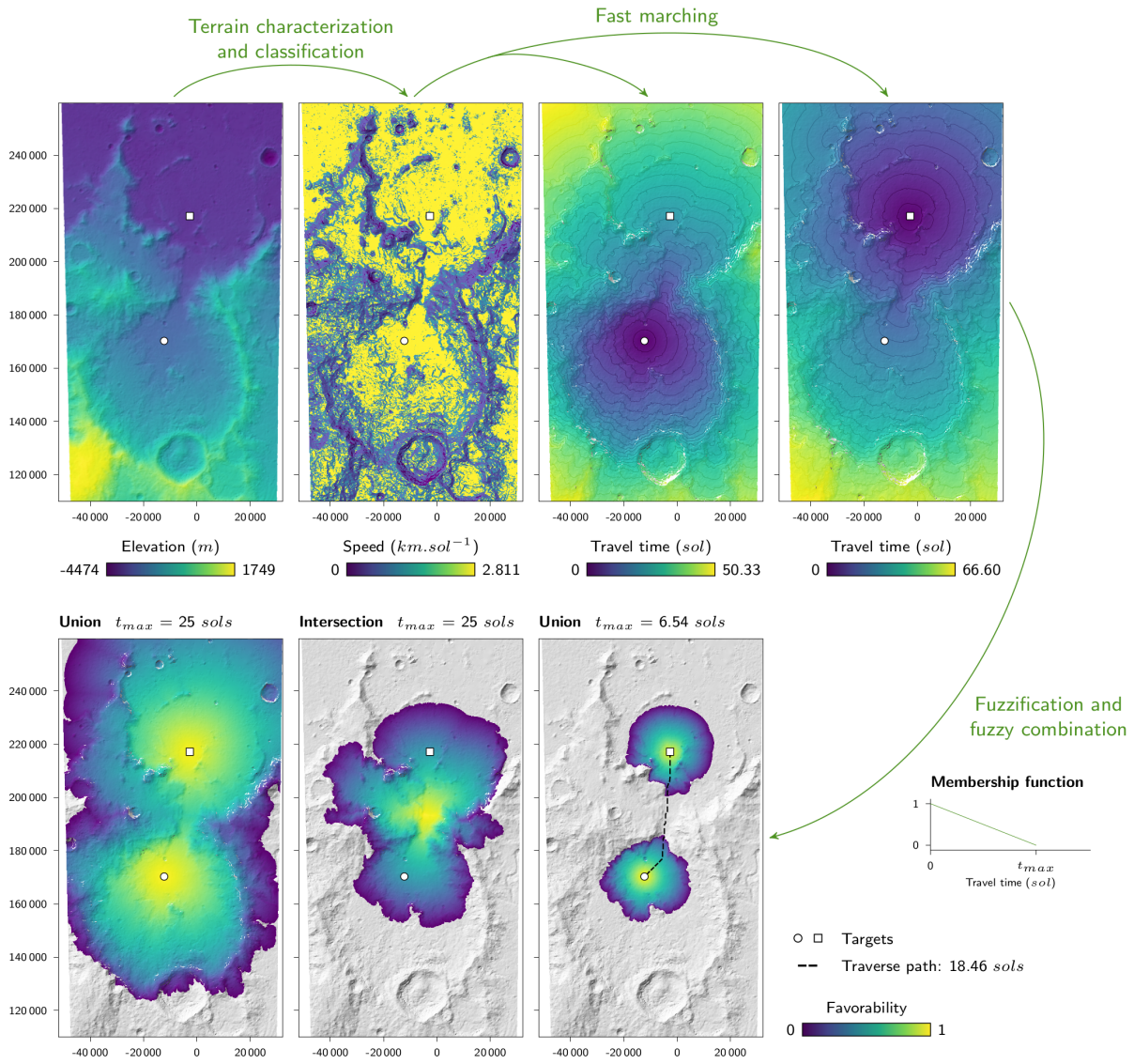


Example: HiRISE DTM
http://www.uahirise.org/dtm/dtm.php?ID=ESP_029815_1530

Application: Mars Site Selection

6) Fast Marching

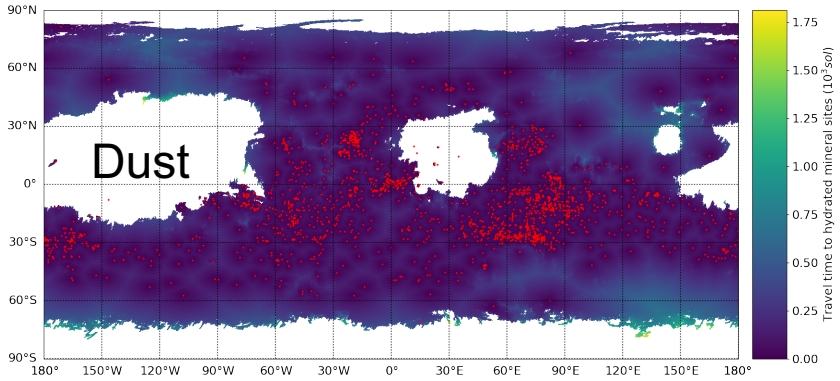
Traverse path analysis to better estimate the potential landing areas



Application: Mars Site Selection

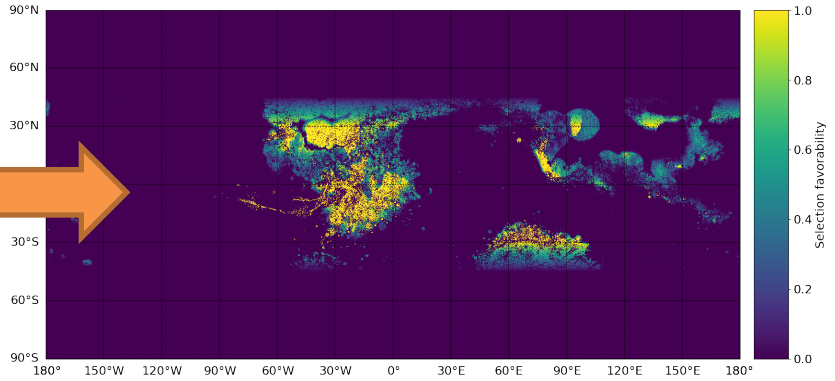
7) Fast Marching + Terrain

→ travel times to a selected location

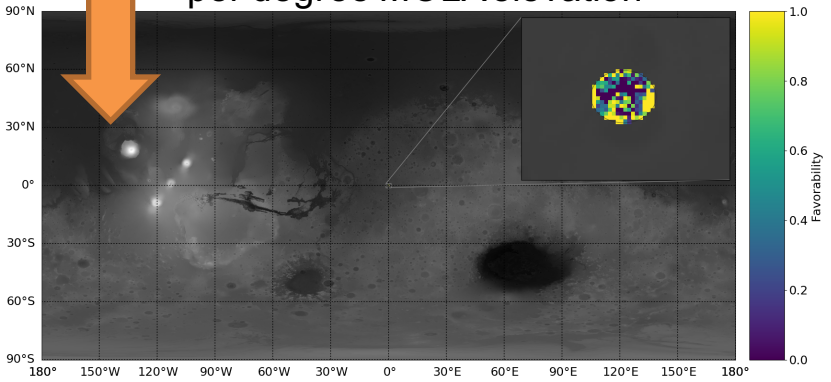


8) Combine with other criteria

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



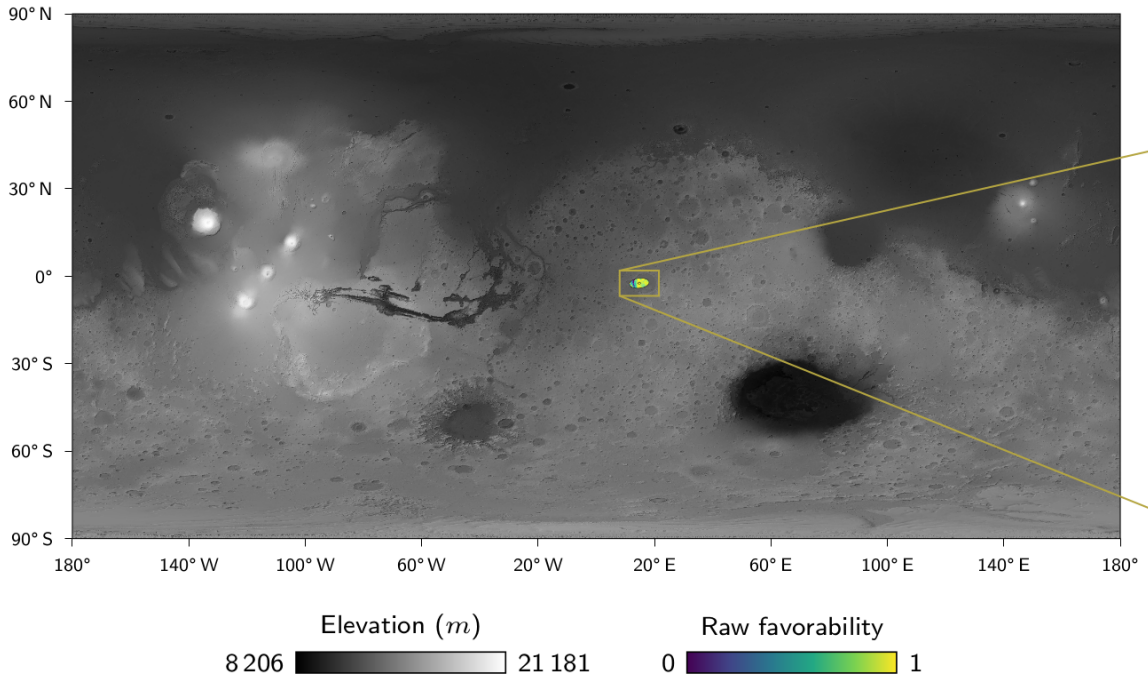
Search for 25 × 20 km ellipse on 20 pixel per degree MOLA elevation

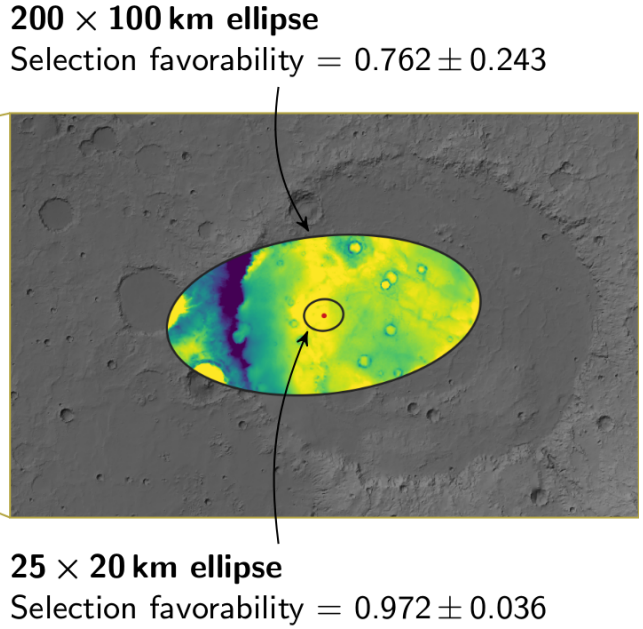


Result

Application: Mars Site Selection

- Raw favorability = final fuzzy logic map with landing constraints and scientific targets
- Selection favorability = average of the raw favorability inside the landing ellipse
- Selection lack of fit = standard deviation inside the landing ellipse





Victor Pankratius – AIST NNX15AG84G

Computer-Aided Discovery: Mars Site Selection

4 scenarios of high-priority scientific targets:

1. Standing-water environments at the Hesperian-Noachian boundary
2. Standing-water environments at the Early to Middle Noachian
3. Past groundwater environments
4. Past standing-water and groundwater environments, and volcanic structures and units

Computer-Aided Discovery: Mars Site Selection Alternative Scenario Analysis

(1)

Past standing-water near the Noachian–Hesperian boundary

5 654 paths, 3.47 (3–8) targets/path
Traverse time (sols): 19.83 (1.47–25)
Landing favorability: 0.50 (0–0.96)

Noachian–Hesperian contacts	5 654
Hydrous mineral exposures	5 559
Sedimentary rock exposures	5 382
Polygonal ridge networks	1 330
Valley networks	426
Deltaic deposits	234
Crater wall channels	164
Hes. to Am. volcanic units	156
Young deltas	138
Gullies	107
Chloride-bearing materials	105
Dune fields	101
Crater floor channels	62
Crater floor valley deposits	55
Open basin lakes	42
Crater floor dunes	34
Wrinkle ridges	20
Graben axes	16
Closed-basin lakes	11
Fresh craters	11
Crater floor fractured	10
Scarps	9
Outflow channels	6
Alluvial fans	3

(2)

Past standing-water on Early to Middle Noachian units

16 231 paths, 3.66 (3–8) targets/path
Traverse time (sols): 20.89 (2.32–25)
Landing favorability: 0.39 (0–0.96)

Early to Middle Noachian units	16 231
Hydrous mineral exposures	13 738
Sedimentary rock exposures	11 986
Valley networks	4 507
Chloride-bearing materials	2 738
Open basin lakes	1 952
Polygonal ridge networks	1 682
Closed-basin lakes	905
Young deltas	791
Deltaic deposits	754
Outflow channels	712
Crater wall channels	647
Crater floor valley deposits	526
Graben axes	382
Crater floor channels	257
Dune fields	255
Glacial features	195
Crater floor fractured	195
Gullies	182
Carbonate-bearing rocks	115
Crater floor landslide deposits	112
Crater floor dunes	102
Crater floor tectonics	96
Hes. to Am. volcanic units	87
Subaqueous fans	62
Wrinkle ridges	51
Channels no valleys	45
Scarps	39
Fresh craters	30
Slope streaks	6
Ridges	6
Deep craters	5
Dissected mantle	2

(3)

Past standing- and ground-water with volcanic structures and units

1 038 paths, 3.27 (3–5) targets/path
Traverse time (sols): 19.61 (0–24.99)
Landing favorability: 0.13 (0–0.70)

Hydrous mineral exposures	997
Hes. to Am. volcanic units	742
Sedimentary rock exposures	386
Crater floor fractured	377
Volcanoes	296
Deep craters	150
Scarps	141
Channels no valleys	125
Valley networks	61
Chloride-bearing materials	48
Fresh craters	28
Crater floor dunes	28
Wrinkle ridges	11
Graben axes	5

(4)

Past groundwater

91 paths, 4.22 (3–6) targets/path
Traverse time (sols): 18.88 (5.06–24.91)
Landing favorability: 0.36 (0–0.73)

Hydrous mineral exposures	83
Deep craters	51
Closed-basin lakes	42
Crater floor fractured	41
Sedimentary rock exposures	31
Crater floor dunes	29
Outflow channels	21
Crater wall channels	21
Dune fields	15
Crater floor tectonics	11
Scarps	10
Polygonal ridge networks	8
Chloride-bearing materials	8
Crater floor channels	5
Crater floor landslide deposits	4
Young deltas	2
Graben axes	2

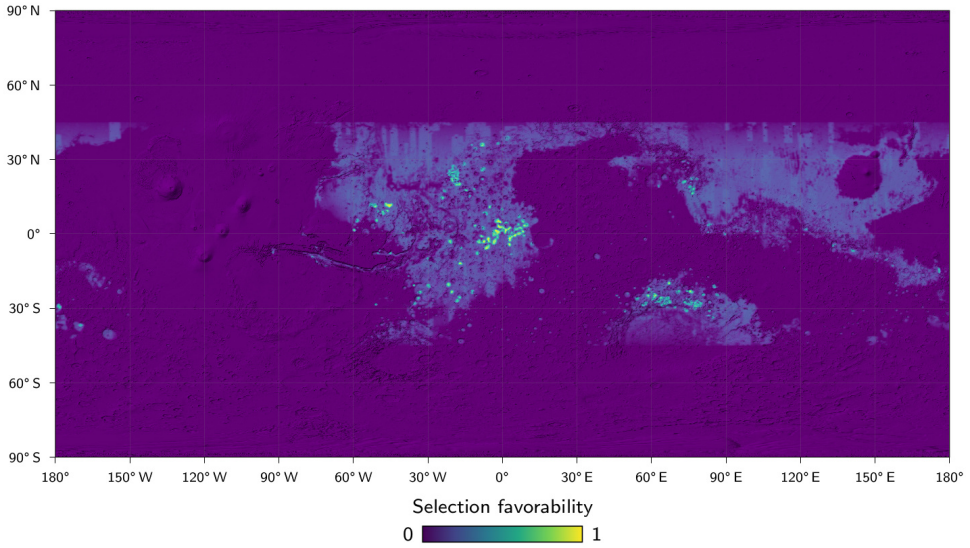
Computer-Aided Discovery: Mars Site Selection

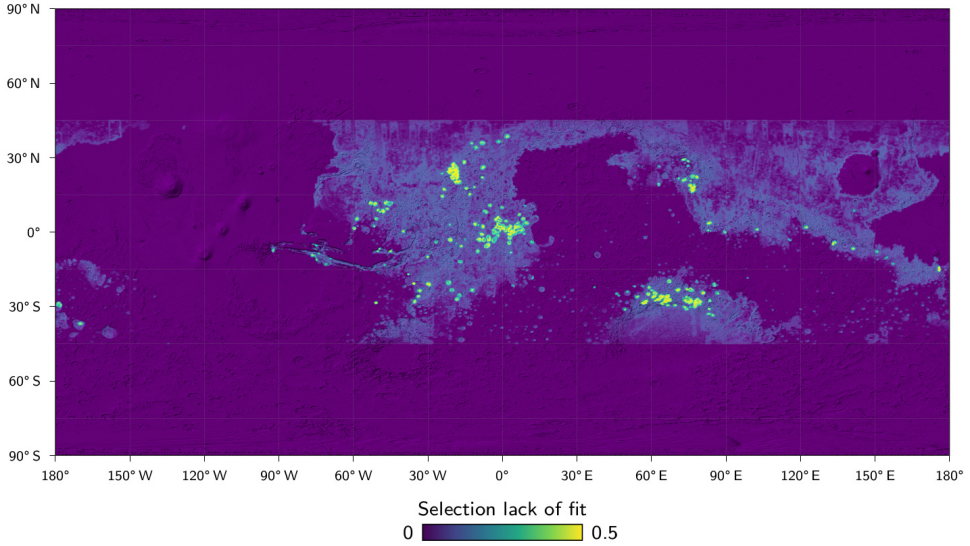
- Final favorability map combining the four scenarios and the landing constraints for the entire planet, at 20 ppd

- Large parts of the Noachian terrains are inaccessible due to elevation constraints during landing, but they include many interesting targets

Satisfied Constraints

- Latitude between ± 45 – 30°
- Elevation below 0 – -1 km
- Slope below 2°
- Vertical roughness below 0.5 m
- Dust-free area (albedo below 0.2 – 0.25 and thermal inertia above 100 – 150 tiu)
- Less than 150 – 200 sols from a hydrated mineral site or a valley
- Less than 150 – 200 sols from a HiRISE, CRISM or MOC footprint

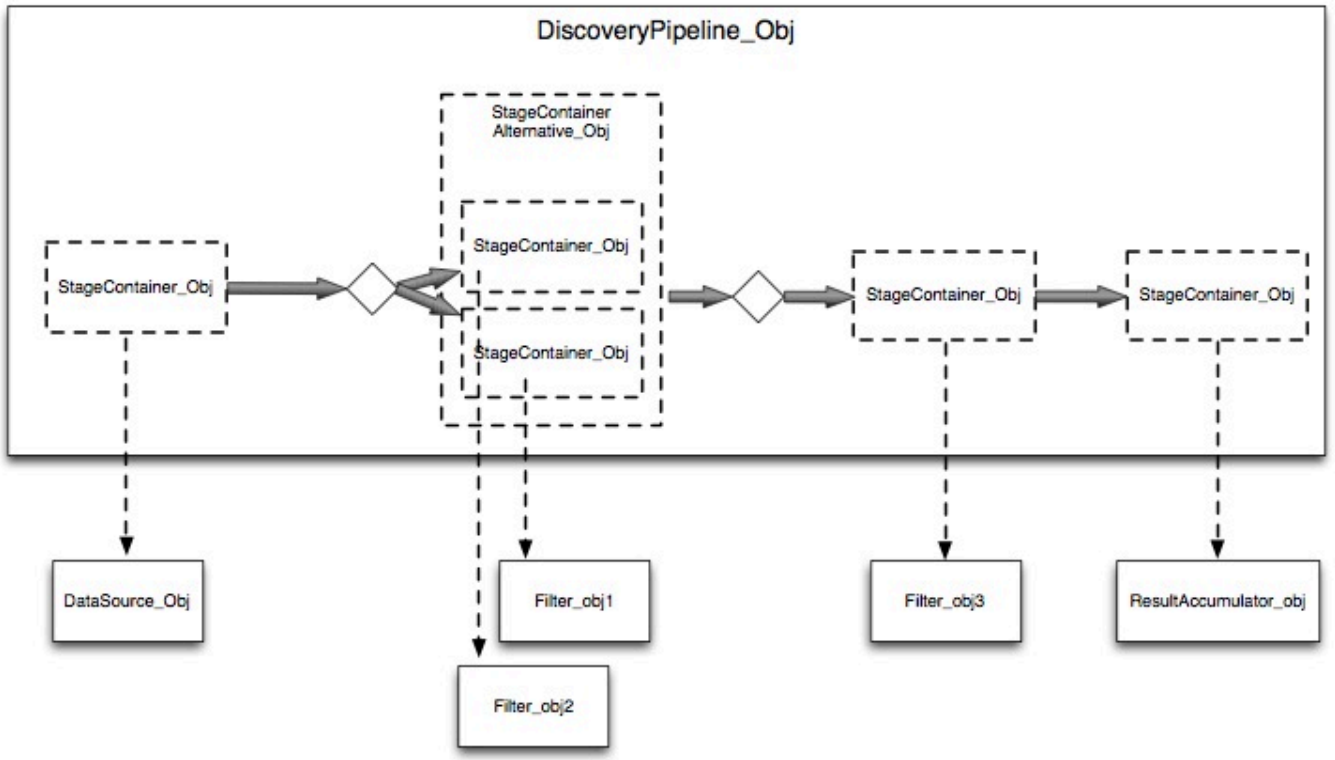




Victor Pankratius – AIST NNX15AG84G

MIT Haystack Observatory
Massachusetts Institute of Technology

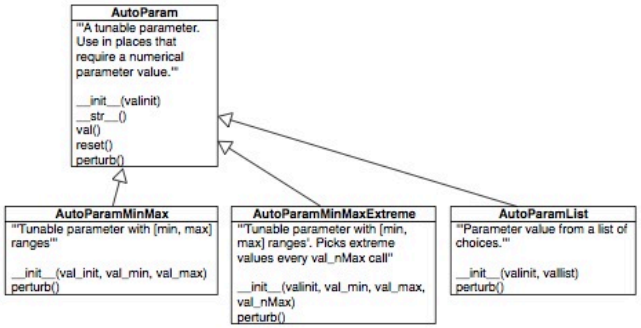
Discovery Framework Architecture Overview



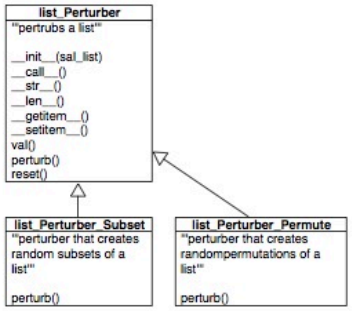
Python Framework for Computer-Aided Discovery - Victor Pankratius

scikit-discovery

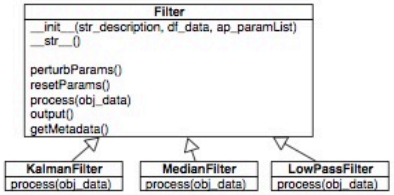
Autoparams



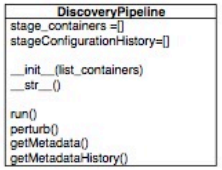
Perturbers



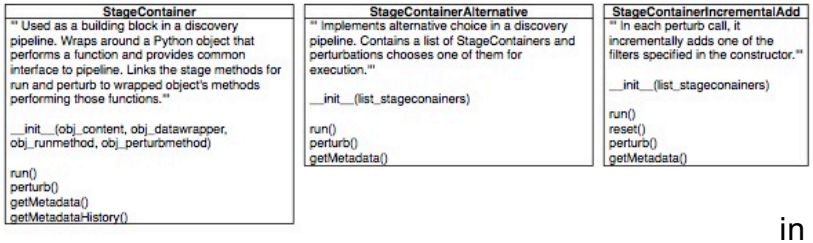
Filters



Discovery Pipeline



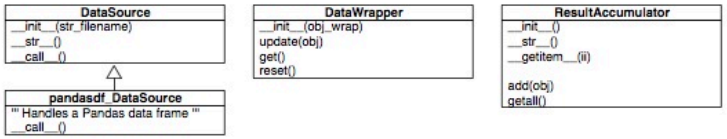
Stage Containers



in addition

- Amazon Offloading
- Plotting
- etc.

scikit-dataaccess : Data Import



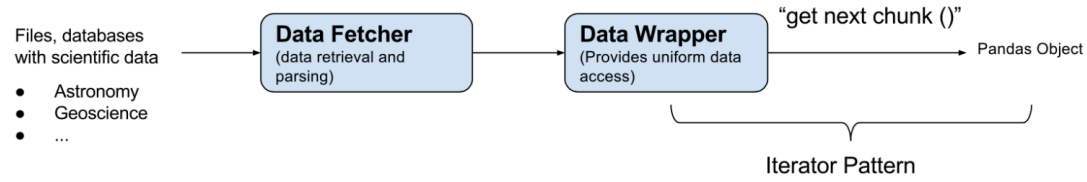
- Data Downloaders
- Data Fetchers
- Data Wrappers

Python Data Access Package Release

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

1 Overview

The Scikit Data Access package simplifies the handling of scientific data sets in Python. It provides a common interface across all data sets, based on a data fetcher and iterator pattern, as illustrated in the Figure below.



This paradigm places the requirements for parsing and interpreting the data inside of the data fetcher, which returns a data wrapper that provides a uniform method for accessing the data. In particular, the data wrapper implements an iterator which returns the next segment of data when requested by another function or by the user.

Advantages of skdacces

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

Install

pip install scikit-dataaccess

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

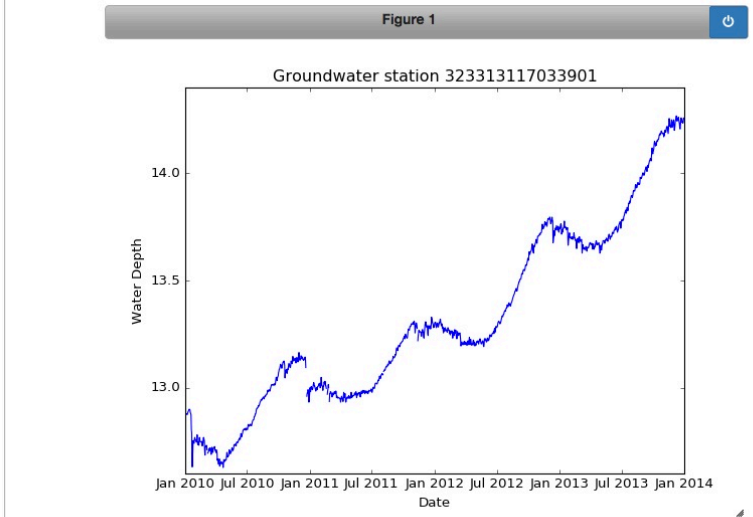
```
In [4]: data_wrapper = groundwater_fetcher.output()
        my_iterator = data_wrapper.getIterator()
```

Each groundwater station can be accessed one at time

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

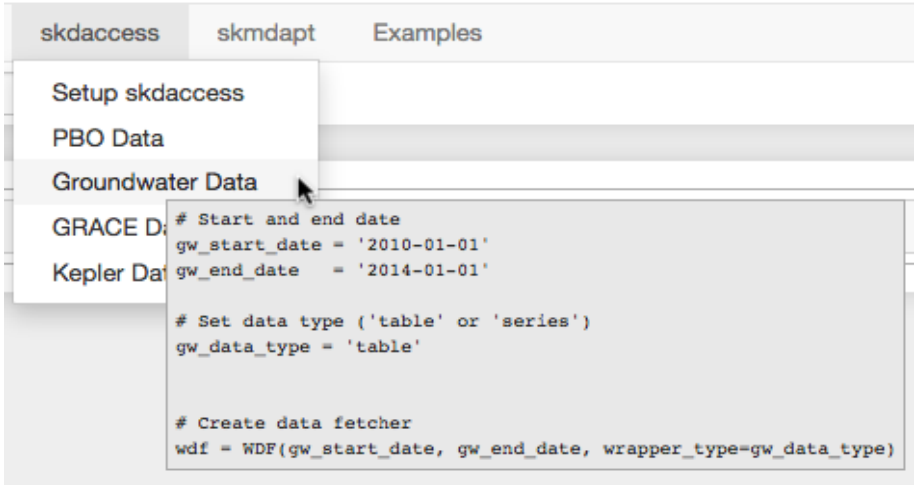
Plot the first station

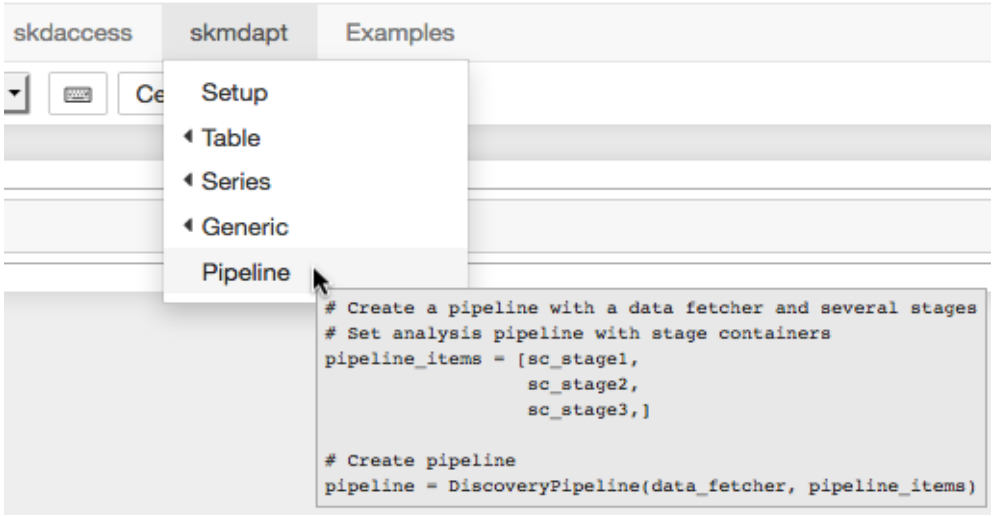
```
In [6]: plt.plot(data);
        plt.ylabel(label);
        plt.xlabel("Date")
        plt.title('Groundwater station ' + data.name);
```



Code Snippets / Workflow Warehouse Examples

- Code snippets aid in pipeline creation
- Data Access Snippets
  - PBO
  - Groundwater
  - GRACE
  - *Kepler*
- Pipeline items
- Basic pipeline with three items





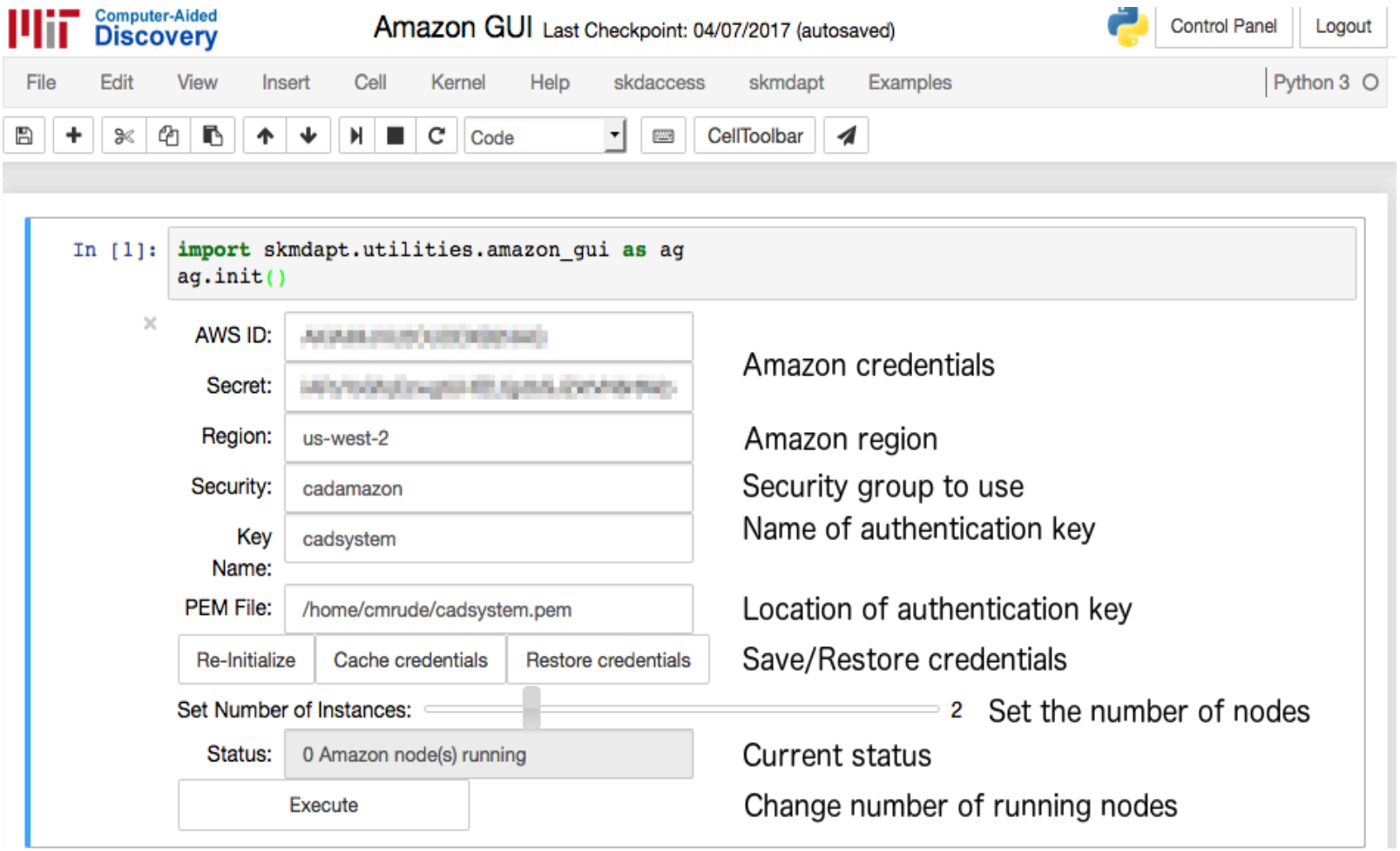
Proof of concept: Transparently offloading a processing pipeline to Amazon

```
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:
```

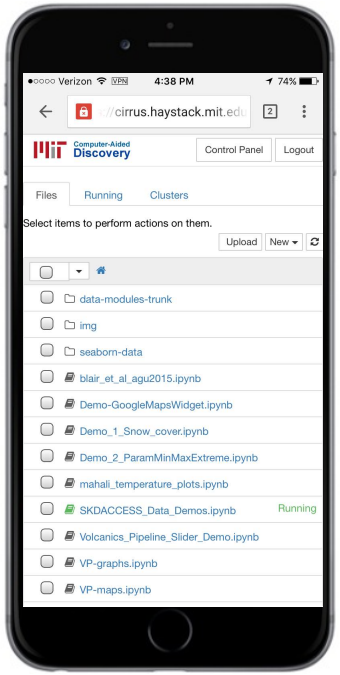
```
In [4]: ap_tau = AF.AutoParamMinMaxExtreme(120,1,500,5)
        ap_sigmaSq = AF.AutoParamMinMaxExtreme(5.5,1,100,5)
        ap_R = AF.AutoParamMinMaxExtreme(1,1,100,5)

        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
```

Controlling Amazon Instances



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



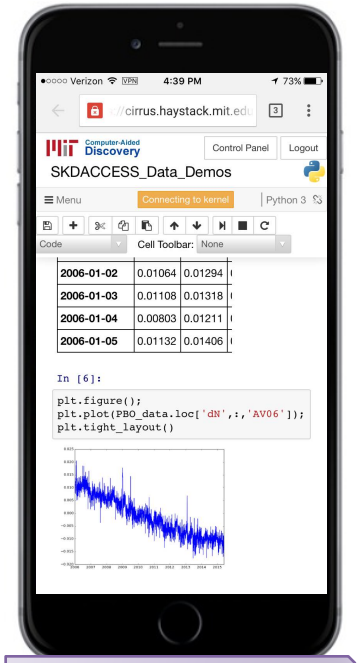
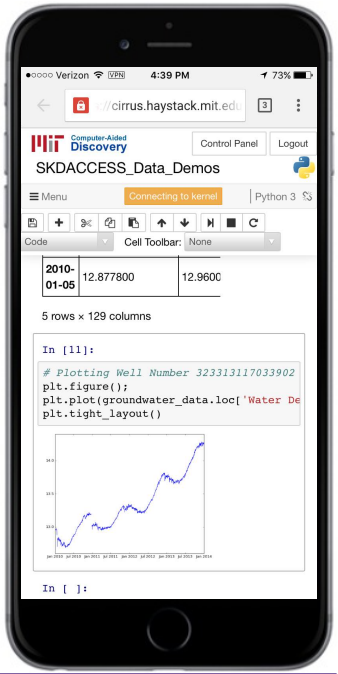
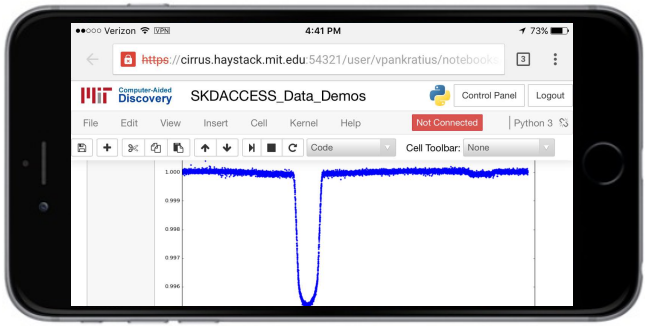


Plate Boundary Observatory Data



USGS Groundwater Data

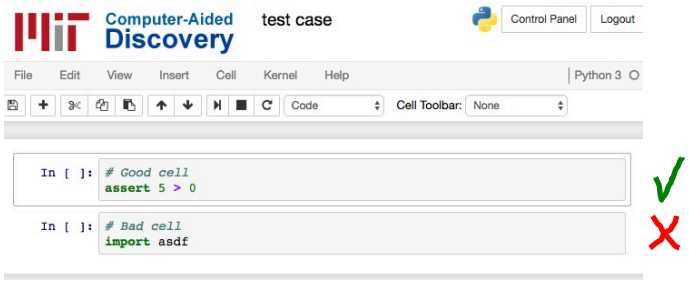


NASA Kepler Data

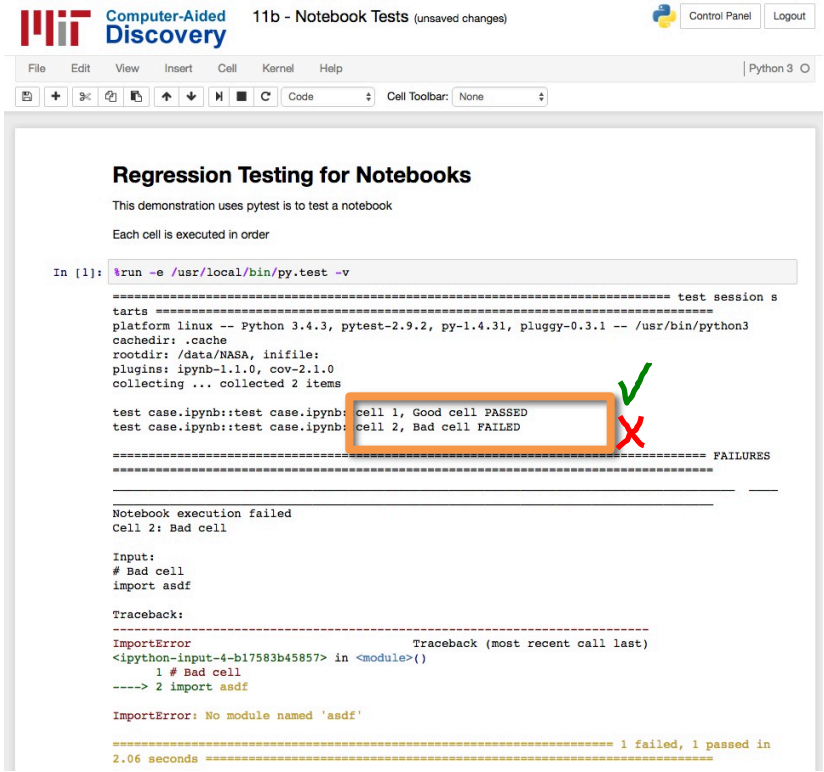
Testing

- 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

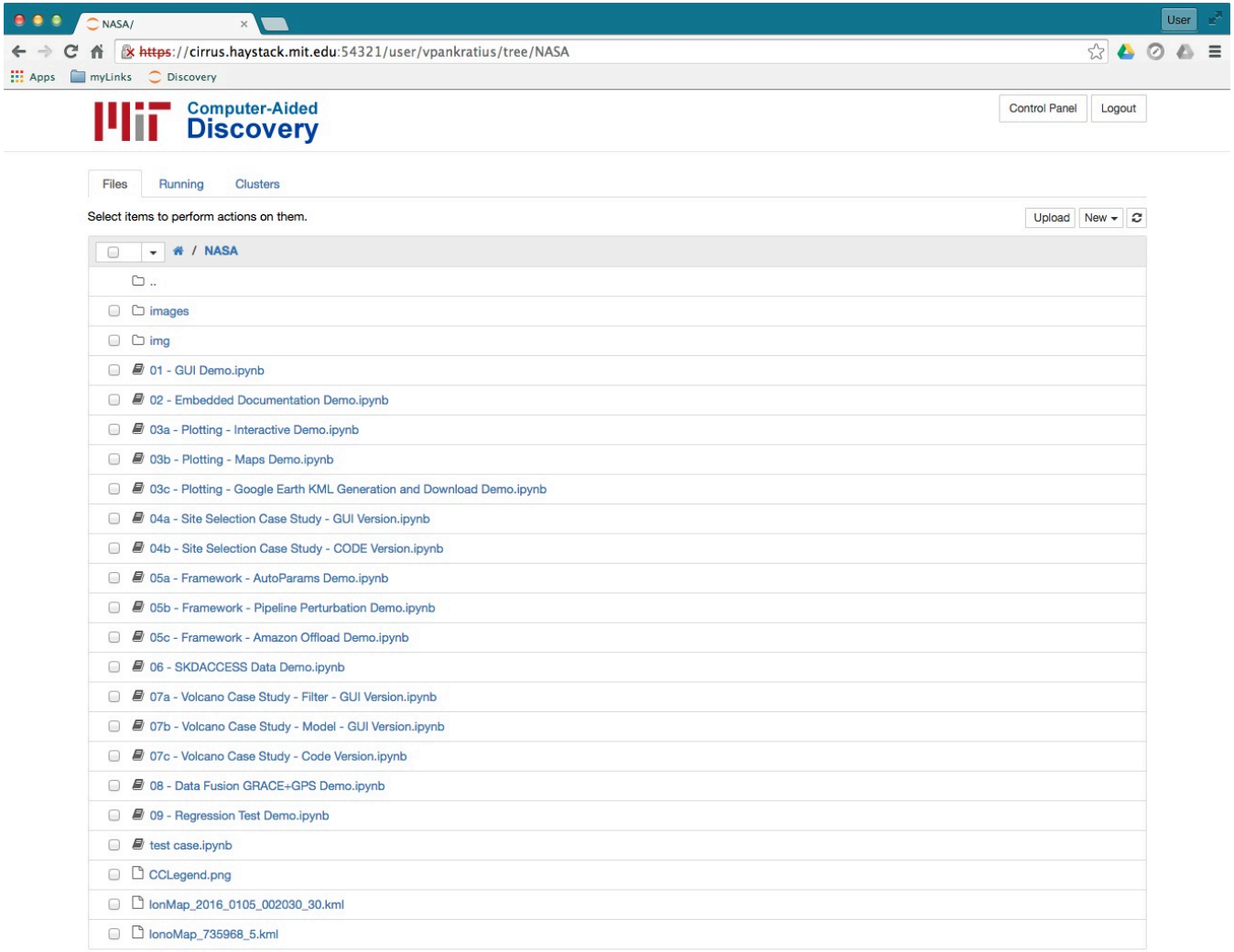


Deployment / Installation Options

Currently Preparing for Community Release

- Cloud: System deployable as instance(s) on Amazon
- Local: VirtualBox virtual machine
- Code: Install your own
  - https://github.com/MITHaystack/scikit-dataaccess
  - https://github.com/MITHaystack/scikit-discovery

Demo Screenshots



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

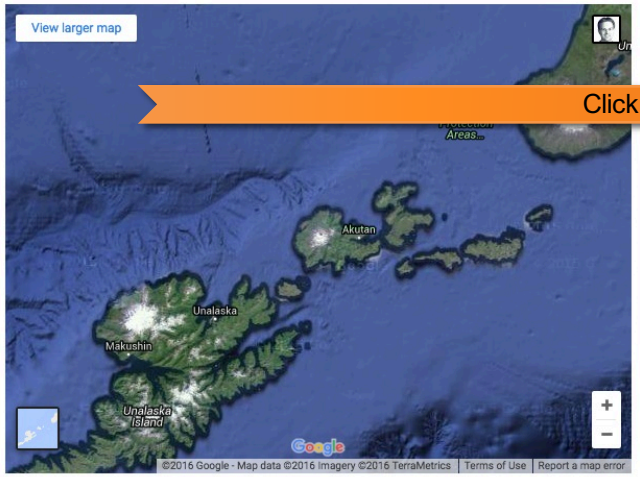
Map Plotting Demonstration

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

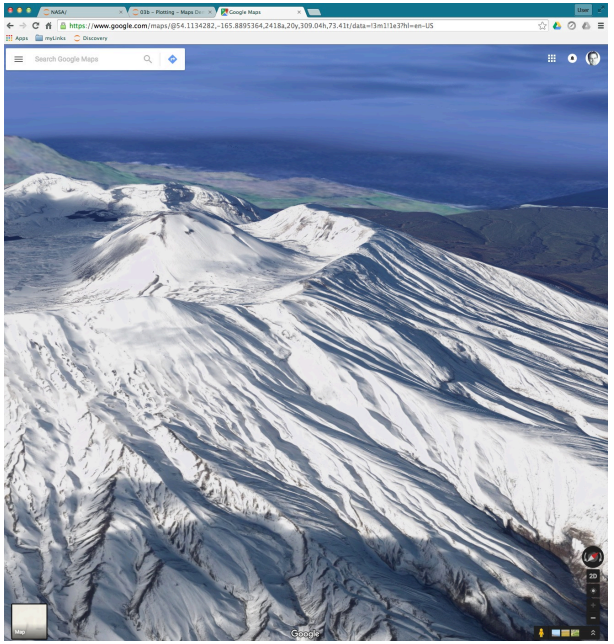
- Example of embedded Google Map display
- Example of Basemap plotting the MA region, with markers for Haystack and MIT
- Example of Basemap plotting, not restricted to the Earth, of a region on the Moon used in the lunar site selection example

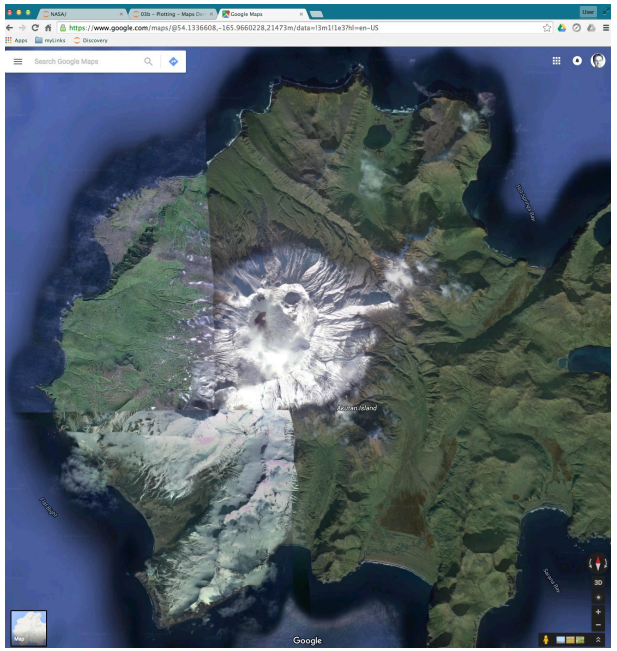
In [1]:
```
# Imports
%matplotlib inline
import matplotlib.pyplot as plt
from IPython.display import HTML, display
from mpl_toolkits.basemap import Basemap
import numpy as np
plt.rcParams['figure.figsize'] = (10.0, 8.0)
plt.rcParams['font.size'] = 12
plt.rcParams['svg.fonttype'] = 'none'
```

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”





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

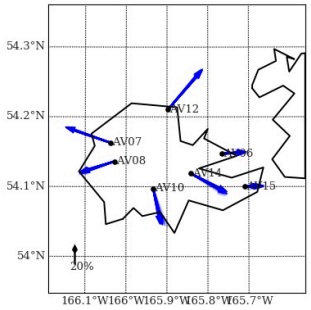
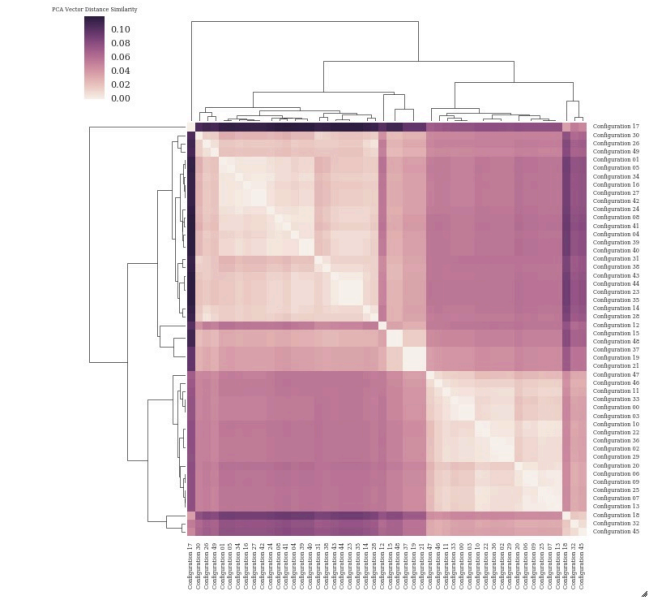


Figure 2



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 — Code — Cell Toolbar: None — Python 3

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

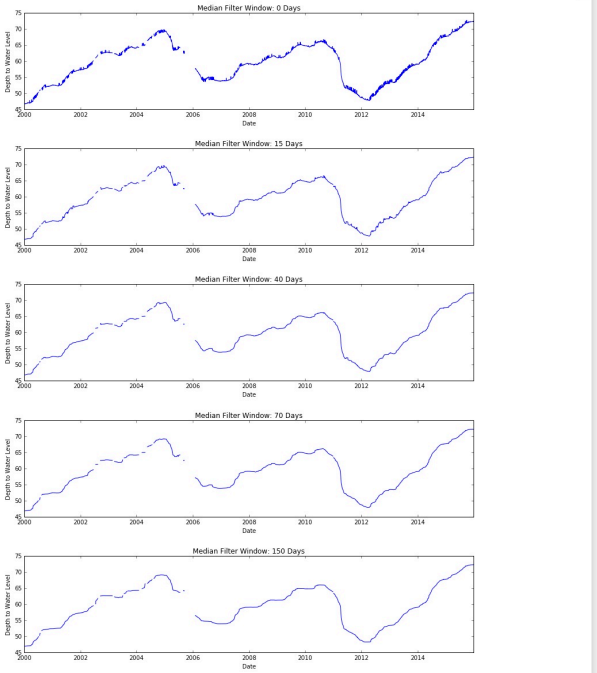
In [8]:
```
# Get the results
results = pipeline.getResults()
metadata = pipeline.getMetadataHistory()
# Loop over each pipeline run
for index,(run,info) in enumerate(zip(results,metadata)):
    # Plot the depth to water level
    plt.plot(run['Data Accumulator'].loc['Water Depth',:,:]);
    # Set xlabel
    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 — Code — Cell Toolbar: None — Python 3



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

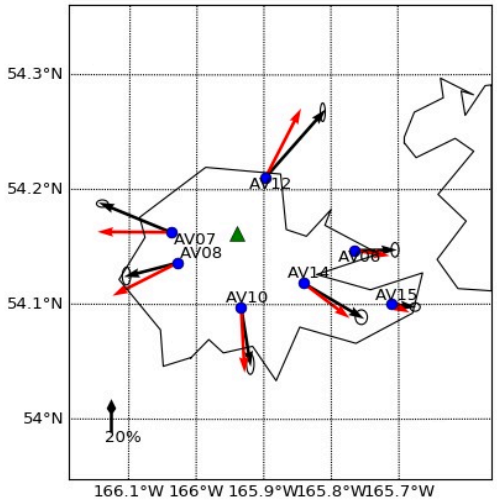
A demonstration of interactive (GUI) filter perturbation of an volcano analysis pipeline.

- 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



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)
```

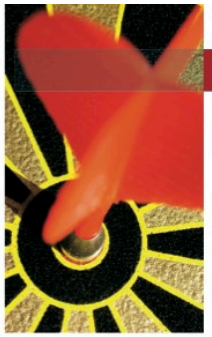
```
perturbRuns = acf.ResultAccumulator()
for ii in tqdm.tqdm(range(50)):
    pipeline_kf1.perturb()
    pipeline_kf1.run()
    pipeline_kf1.dataReset()
    perturbRuns.add(pca_A.getEigenvectors(storeName))
stations = pca_A.results['stations']

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

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

Massachusetts Institute of Technology

Conclusion & Accomplishments

- Computer-Aided Discovery environment
- New discovery in volcanology (inflation events)
- Tested and refined tools around volcanic data sets, groundwater data sets, atmospheric data sets (lee waves), Moon, Mars, integration in Jupyter notebooks
- Transparent offloading of data processing pipelines to cloud environments (e.g., Amazon)
- Open source code (github, MIT license)

  + VMs: Amazon, VirtualBox

Thanks!

@vpankratius

pankrat@mit.edu

AIST NNX15AG84G
PI Pankratius

ACI, AGS INSPIRE
PI Pankratius
ACI1442997, AGS-1343967

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.