Python As A Data Analysis Platform

About me

Name: Deepankar Sharma

Email: dsharma@enthought.com

Email: deepankar.sharma@gmail.com

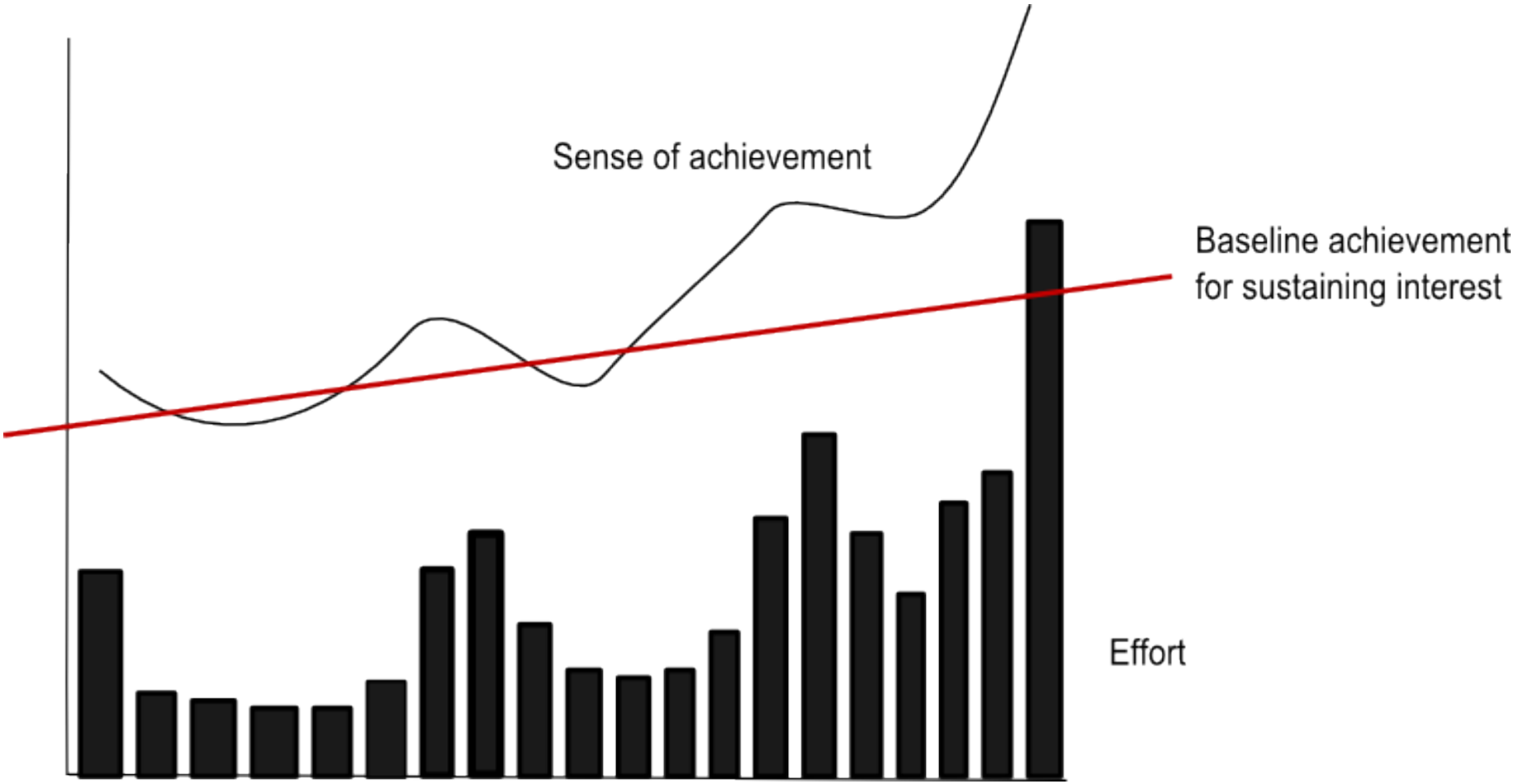
Question about audience

- People who consider themselves programmers
- People who write code on a daily basis
- People who consider Python their primary language
- People who write data driven applications

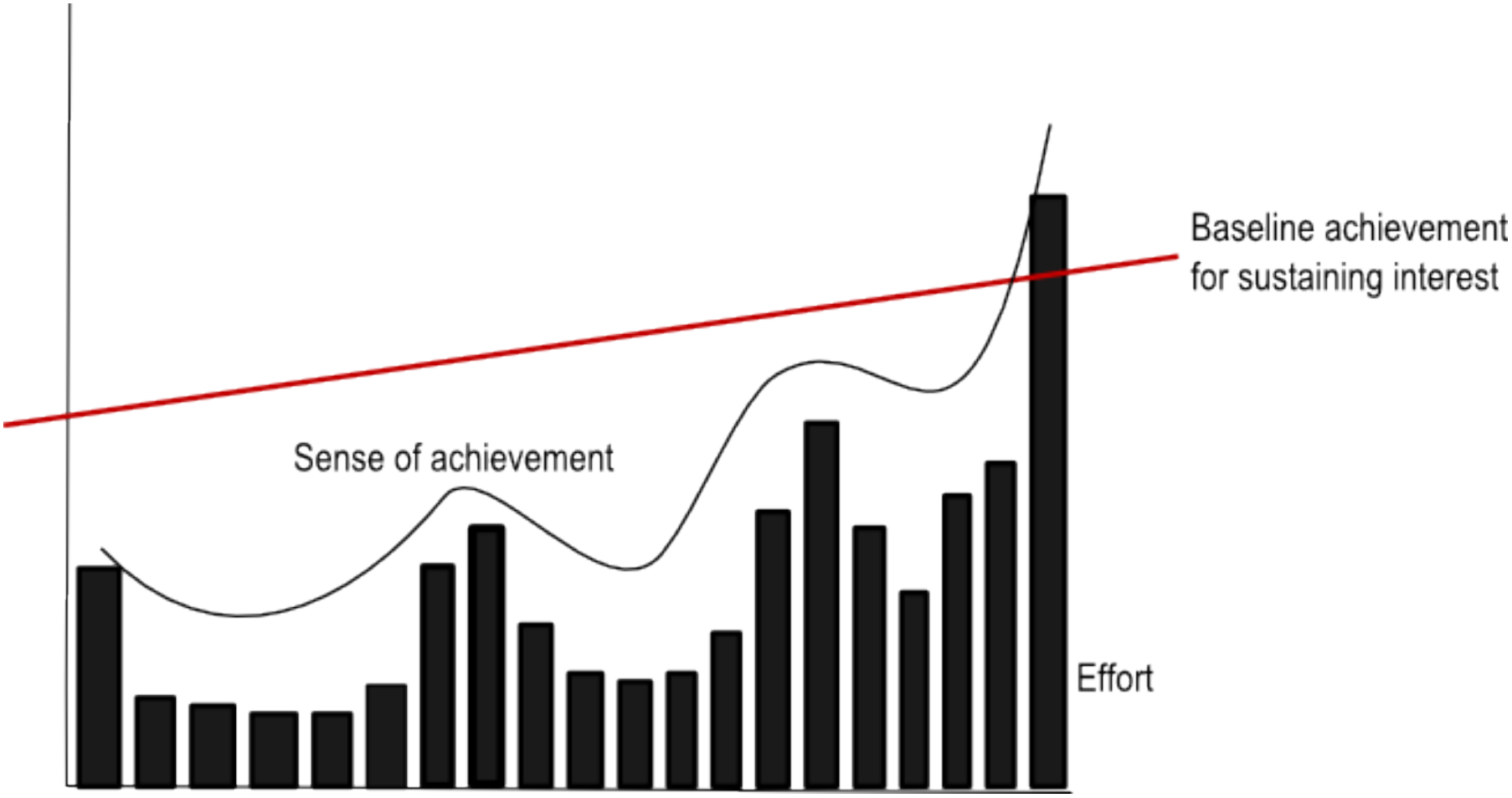
My goals for this talk

- Increase development of data driven applications using Python
- Increase the number of Python based stories on HN front page
- Introduce users to Python libraries for analyzing / visualizing data Python

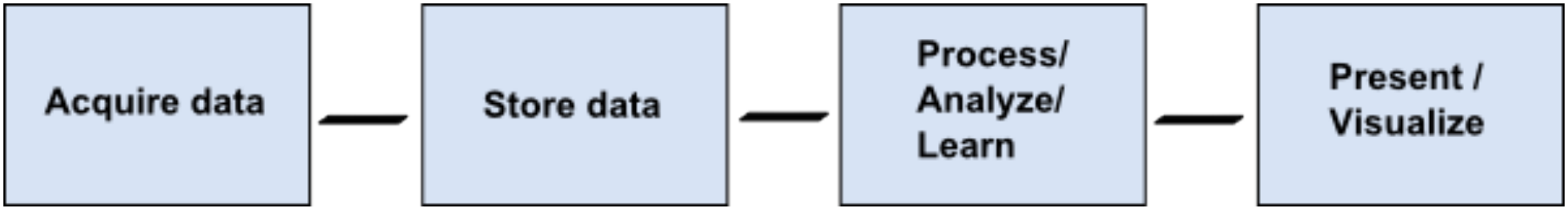
Trajectory of successful project



Trajectory of unsuccessful project



Lets Pick A Problem


Acquire data
Store data
Process/
Analyze/
Learn
Present /
Visualize

Analyzing Weather Data

Source of data

NOAA NATIONAL OCEANIC AND ATMOSPHERIC ADMINISTRATION
UNITED STATES DEPARTMENT OF COMMERCE

http://noaa.gov

	STN---	WBAN	YEARMODA	TEMP		DEWP		SLP		STP		VISIB		WDSP		MXSPD	GUST	MAX	MIN	PRCP	SNDP	FRSHTT
1	722429	99999	20010101	34.5	24	31.9	24	1029.3	19	9999.9	0	7.3	24	6.8	24	9.9	15.9	37.9	33.1	0.47A	999.9	010000
2	722429	99999	20010102	32.9	23	21.6	23	1035.4	21	9999.9	0	10.0	23	8.5	23	12.0	18.1	37.4*	28.4*	0.00I	999.9	000000
3	722429	99999	20010103	32.2	24	21.1	24	1035.9	24	9999.9	0	9.6	24	3.1	24	8.0	999.9	43.0	26.1	0.00I	999.9	000000
4	722429	99999	20010104	42.2	24	31.3	24	1026.4	24	9999.9	0	8.3	24	3.2	21	7.0	999.9	64.9	28.9	0.00I	999.9	000000
5	722429	99999	20010105	48.9	22	37.2	22	1019.9	22	9999.9	0	8.8	22	2.1	20	8.0	999.9	70.0	32.0	0.00I	999.9	000000
6	722429	99999	20010106	49.0	24	43.5	24	1016.5	24	9999.9	0	8.3	24	3.0	24	12.0	999.9	68.0*	33.8*	0.00I	999.9	000000

Data related fields

- Temperature
- Dew point
- Sea level pressure
- Station pressure
- Visibility
- Windspeed
- Max windspeed
- Max wind gust
- Max temp
- Min temp
- Precipitation
- Snow depth

Storing Your Data

Transient Storage

Holding in Numpy arrays

- Numpy -> N-dimensional homogeneous array implemented in C: fast, & memory efficient

```
>>>  a = np.random.randn(1000, 100) ; b = a[::2,:]
```

Numpy arrays are full featured: 60 methods out of the box (max, mean, conjugate, ...) + SCIPY packages add MANY more + Scikits projects (Statsmodel, TimeSeries, ...).

- Structured arrays offer a labeling of fields

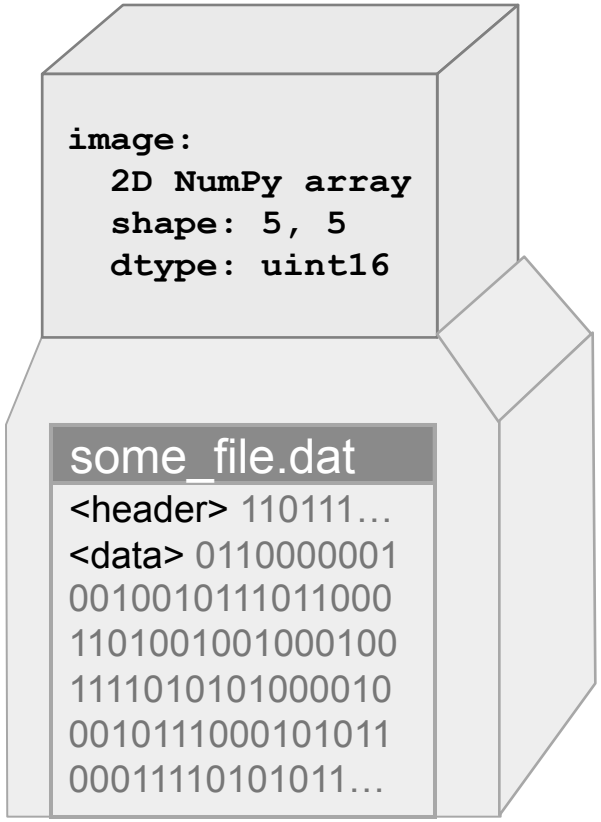
Time	Size	Position				Gain	Samples (2048) ...			
		Az	El	Type	ID					
1172581077060	4108	0.715594	–0.148407	1	4	40	561	1467	997	–30
1172581077091	4108	0.706876	–0.148407	1	4	40	7	591	423	
1172581077123	4108	0.698157	–0.148407	1	4	40	49	–367	–565	–35
1172581077153	4108	0.689423	–0.148407	1	4	40	–55	–953	–1151	–30

```
>>> dt = np.dtype([('Station name', "S10"), ("Elevation", np.float), ("Lat", np.int)])

>>> arr = genfromtxt("station_db.txt", dtype = dt, ...)

>>> print arr["Station name"]
```

Big data: memmap'ed arrays

Memory mapping allows to manipulate arrays of data requiring **more than available RAM**:

```
>>> from numpy import memmap
>>> image = memmap('some_file.dat',
                   dtype=uint16,
                   mode='r+',
                   shape=(5,5),
                   offset=header_size)
```




```
>>> mean_value = image.mean()
>>> scaled_img = image * .5
>>> np.multiply(image,.5,scaled_img)
```

Very efficient thanks to 1. OS caching and 2. the implementation of Numpy arrays (typically 2-3x slower than in memory).

Limitations of memmap

Numpy's memmap module relies on python's mmap which carries OS dependent limitations:

```
>>> from numpy import memmap
>>> a = memmap('some_file.dat',dtype=uint8,
                mode='write', shape=(N,))
```

Responses (python2.7, MacOS with 8GB RAM, 11GB free HD):

	Mac OS (32bit python)	**Win 7 & MacOS (64bit, 3Gb RAM)**	**Linux Ubuntu 11.04 (64bit, 3Gb RAM)**
N = 109**	OK (du = 0.9G)	OK (du = 0.9G)	OK (du = 4K)
N = 3x109**	Overflow error	OK (du = 3G)	OK (du = 4K)
N = 1013**	No space left on device	No space left on device	OK (du = 4K)

Holding data in Pandas I

Pandas (now version 0.7.1) offers thin wrappers around 1,2,3D Numpy arrays.

Author: Wes McKinney, Lambda Foundry, http://pandas.sourceforge.net/

- axis labeling, for example using datetime steps, and nice representation in ipython
- data alignment, data merge (incl. priorities for the various datasets),
- management of missing data
- MANY statistical tools (describe, moving average, covariance, correlation, ...)
- Easy visualization (line, bar chart, boxplot, ...) with Matplotlib

```
>>> from pandas import *
>>> a = [12.3, 15.3, 14.6, np.nan, 17.1, 13.6]
>>> ts = Series(a, index = DateRange('1/1/2000', periods = 6,
offset = datetools.day), name = "Temperature") # 1D
>>> df = DataFrame(ts) # 2D
>>> df['var'] = ts2     # Add another columns
```

Access components: `df.values` (np.ndarray), `df.index` (pandas.Index)

Holding data in Pandas II

Pretty representation:

```
>>> print ts
2000-01-01    12.3
2000-01-02    15.3
2000-01-03    14.6
2000-01-04    NaN
2000-01-05    17.1
Name: Temperature
```

```
>>> print df
            Temperature  var
2000-01-01  12.3         -1.452
2000-01-02  15.3          1.851
2000-01-03  14.6         -0.09037
2000-01-04  NaN          -0.3942
2000-01-05  17.1          1.446
```

Data alignment, data reduction, missing value management

```
ts.align(ts2) ; ts.reindex(ts2.index) ; ts.groupby().apply()
ts.fillna(0.0) ; ts.dropna() ; ts.to_sparse()
```

Loading data from/to files:

```
>>> read_csv, read_table, ts.tofile, ts.to_csv
>>>  HDFStore(), ExcelFile()
```

Persistent Storage

Some Options

Some universal file format (built into the data-structure):

- txt, csv
- binary (watch out!)

Some standard labeled file formats:

- json: json
- HDF: pytables, h5py, pyhdf
- netCDF: netCDF4, (also scipy.io.netcdf, Scientific.io.netcdf)

Some database options

- SQL: sqlalchemy, sqlite3, mysql-python, psycopg…
- No SQL: couchdb, mongodb, cassandra, …

Storing data to HDF5

HDF5 files is the best way to store large datasets during/after processing.

FEATURES

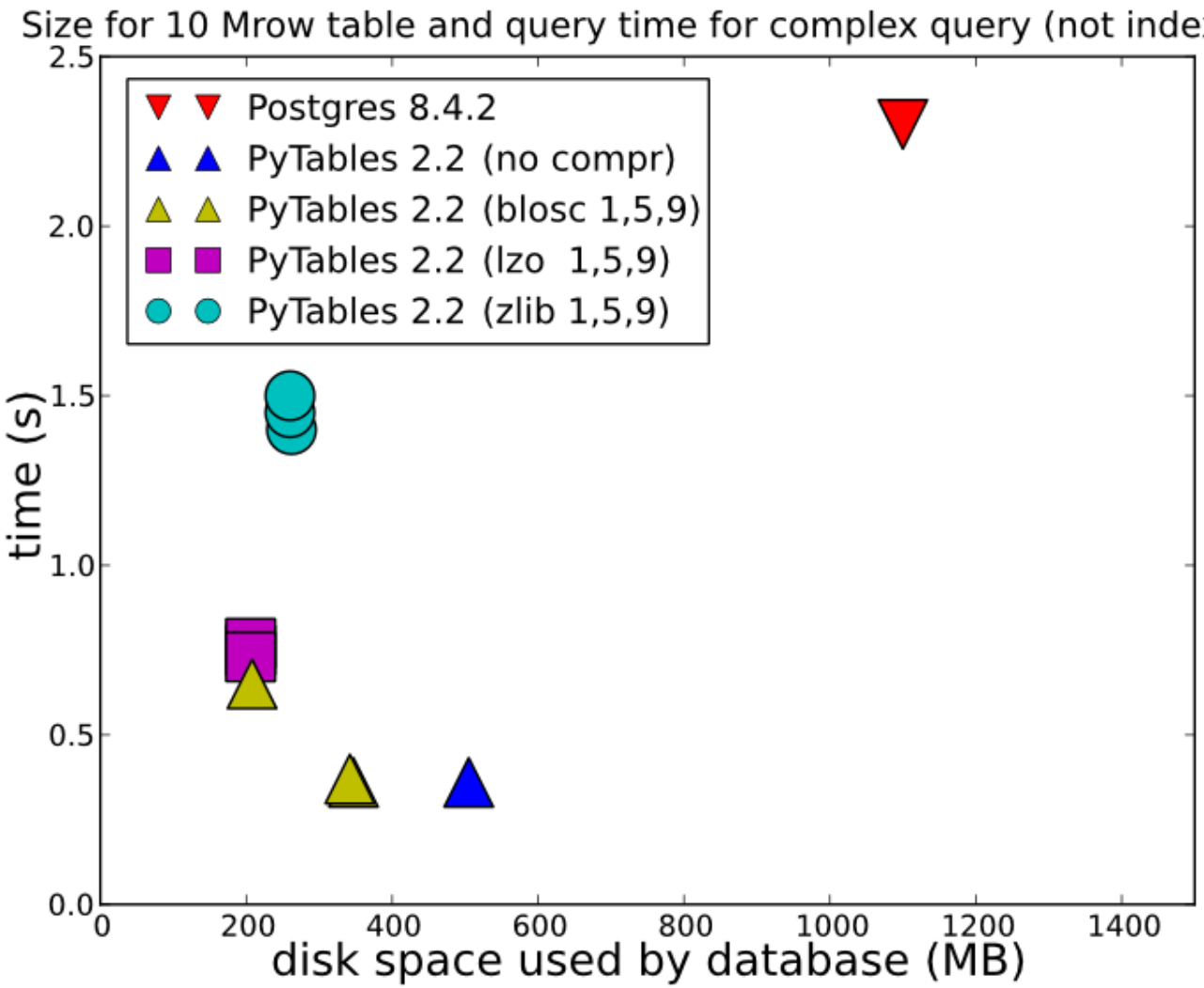
- HDF5 file format is **self-describing**: good for complex data objects
- HDF5 files are **portable**: cross-platform, cross-language (C, C++, Fortran, Java)
- HDF5 is **optimized**: direct access to parts of the file without parsing the entire contents.

See http://www.hdfgroup.org/HDF5

PYTHON LIBRARIES

- **h5py** - "thin wrapper" around the C HDF5 library.
- **PyTables** - Provides some higher level abstractions and efficient tools for retrieval, compression and out-of-core functionalities.

Benchmarking Pytables

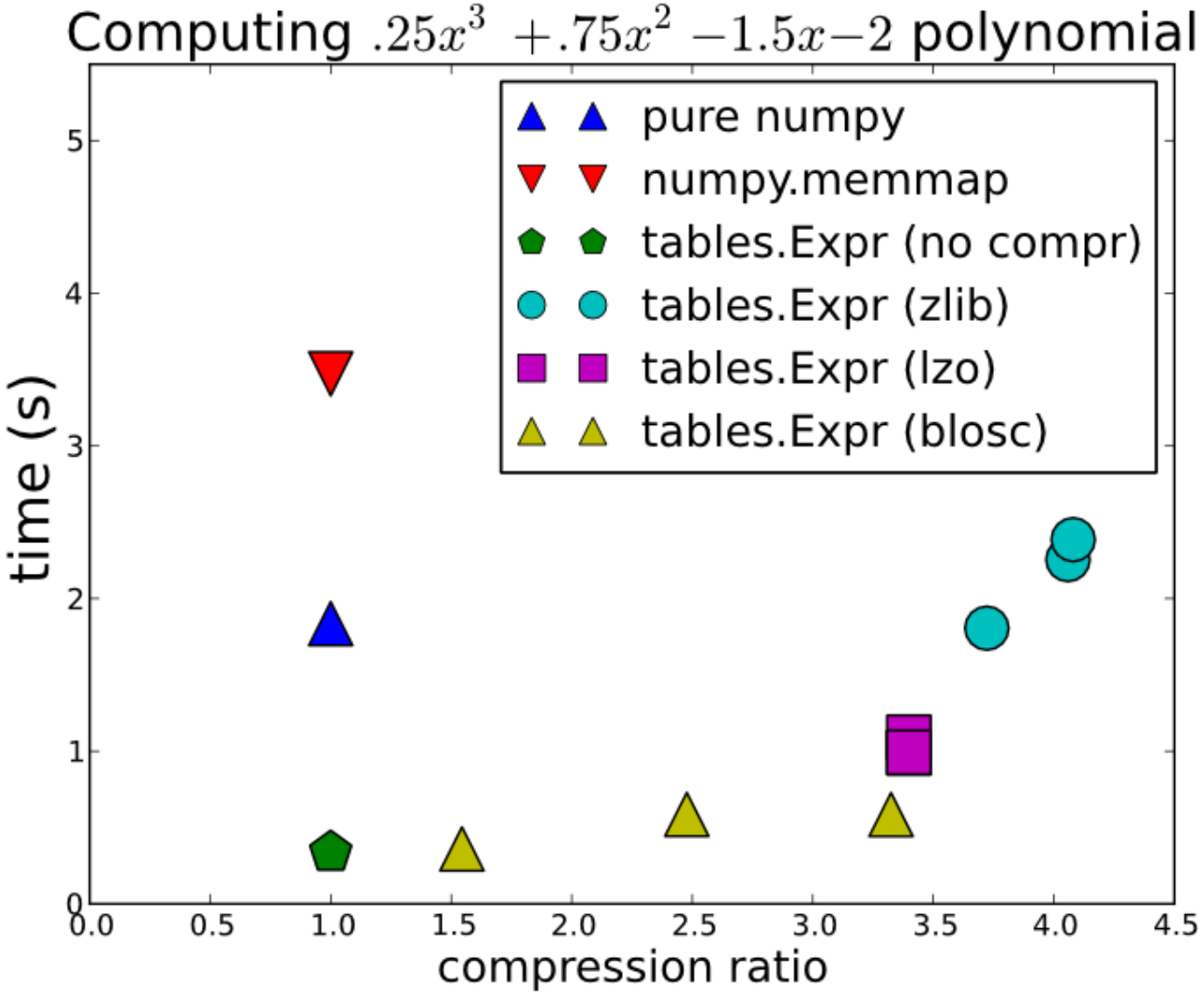


FAST!

EFFICIENT!

Source: http://www.pytables.org/moin/PyTables

Out of core calcs w/ Pytables



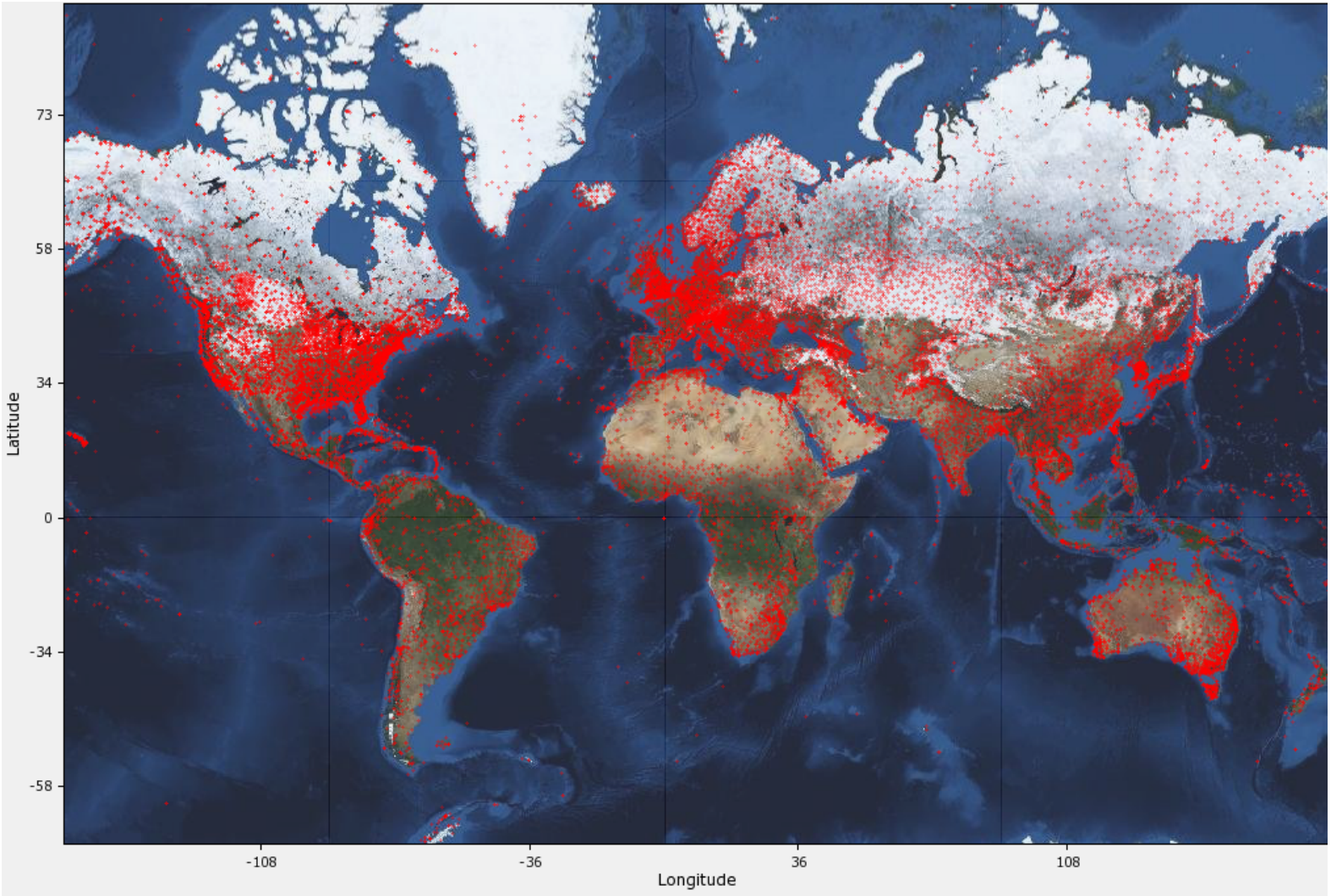
FAST!

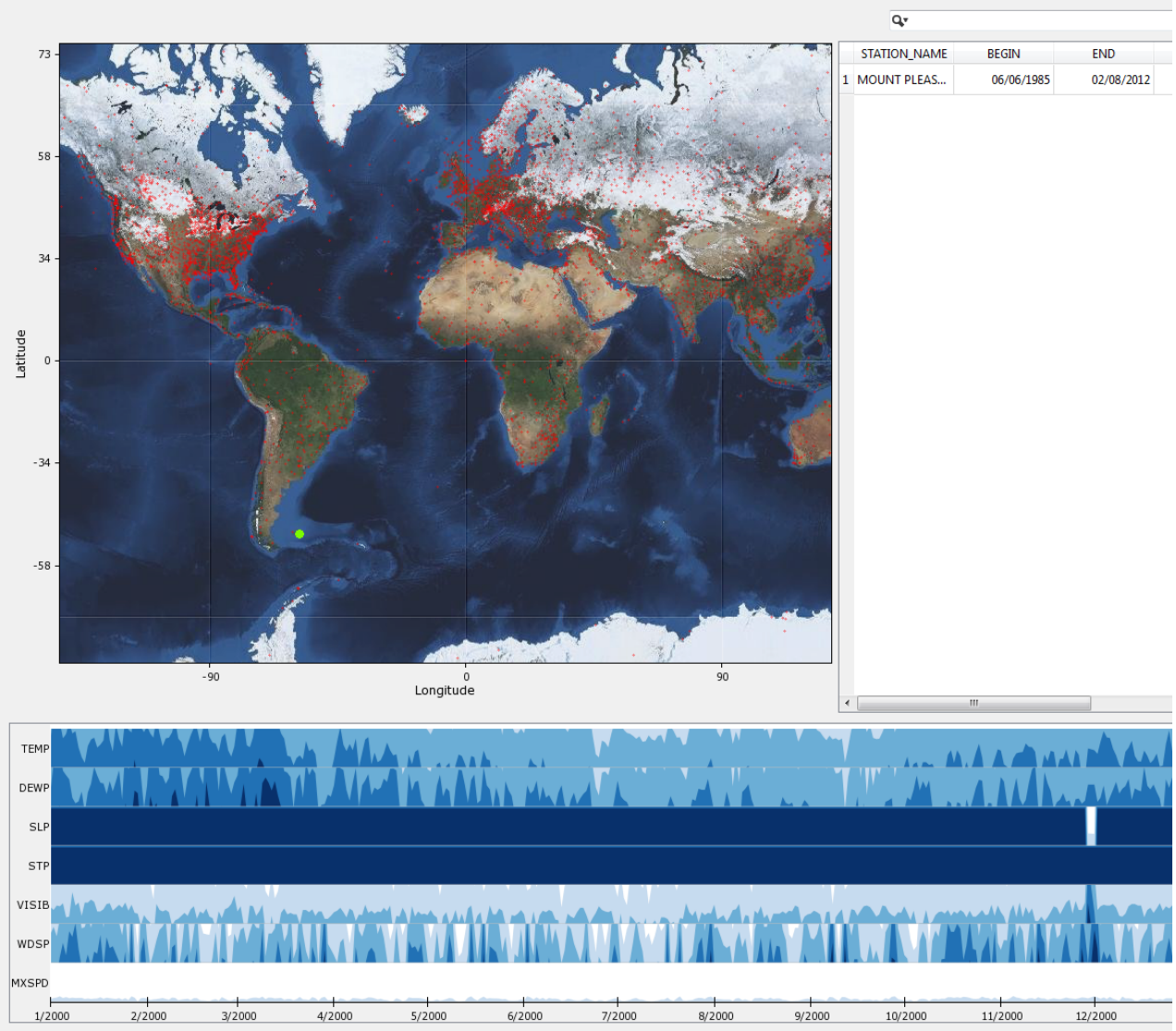
EFFICIENT

Source: http://www.pytables.org/moin/ComputingKernel

Visualizing Data

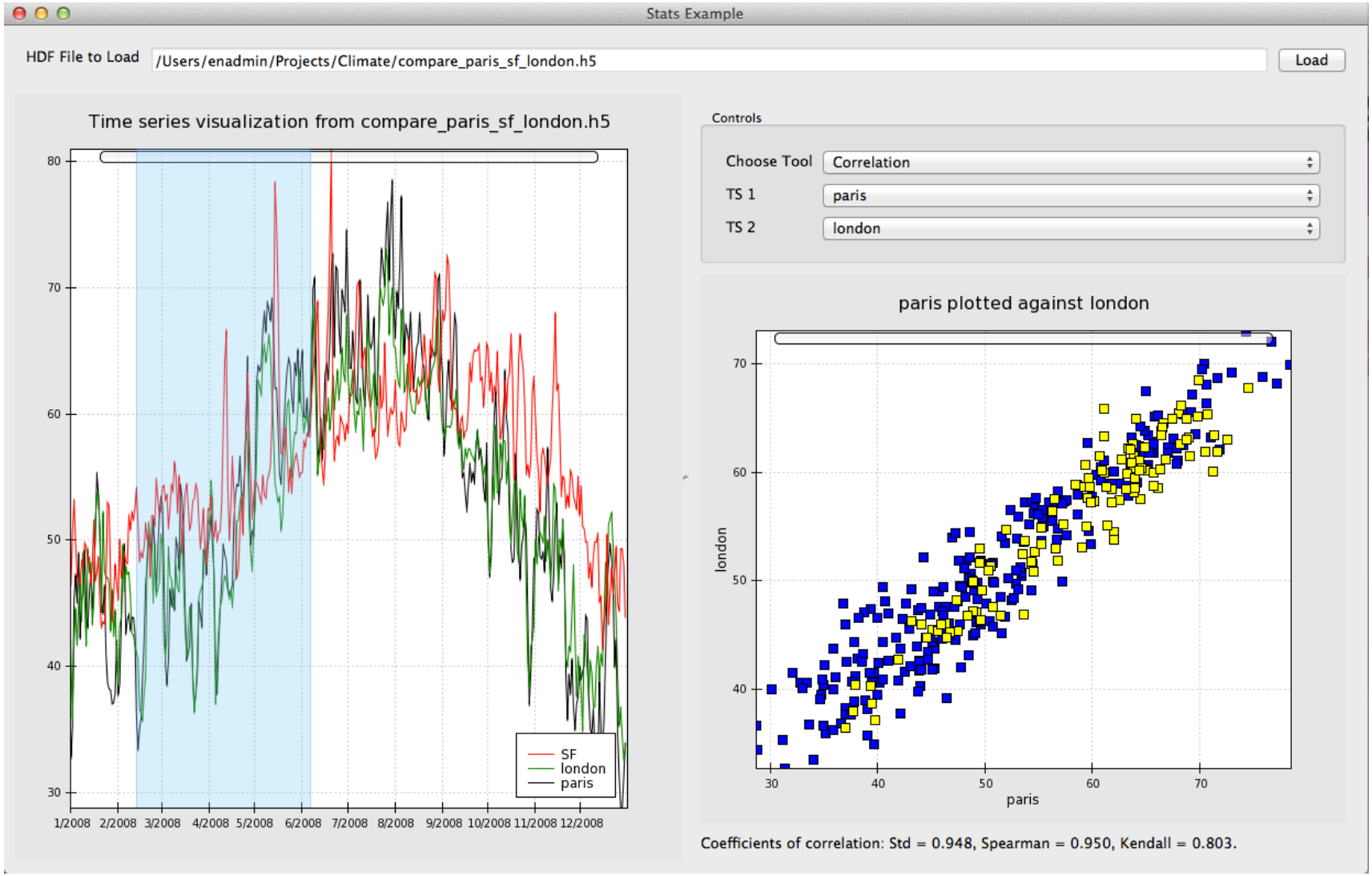
Wonder if there is a way to see those stations on a map.


73
58
34
0
-34
-58
Latitude
-108
-36
36
108
Longitude


73
58
34
0
-34
-58
Latitude
-90
0
90
Longitude
STATION_NAME
BEGIN
END
1
MOUNT PLEAS...
06/06/1985
02/08/2012
TEMP
DEWP
SLP
STP
VISIB
WDSP
MXSPD
1/2000
2/2000
3/2000
4/2000
5/2000
6/2000
7/2000
8/2000
9/2000
10/2000
11/2000
12/2000

Compare Weather From Multiple Cities

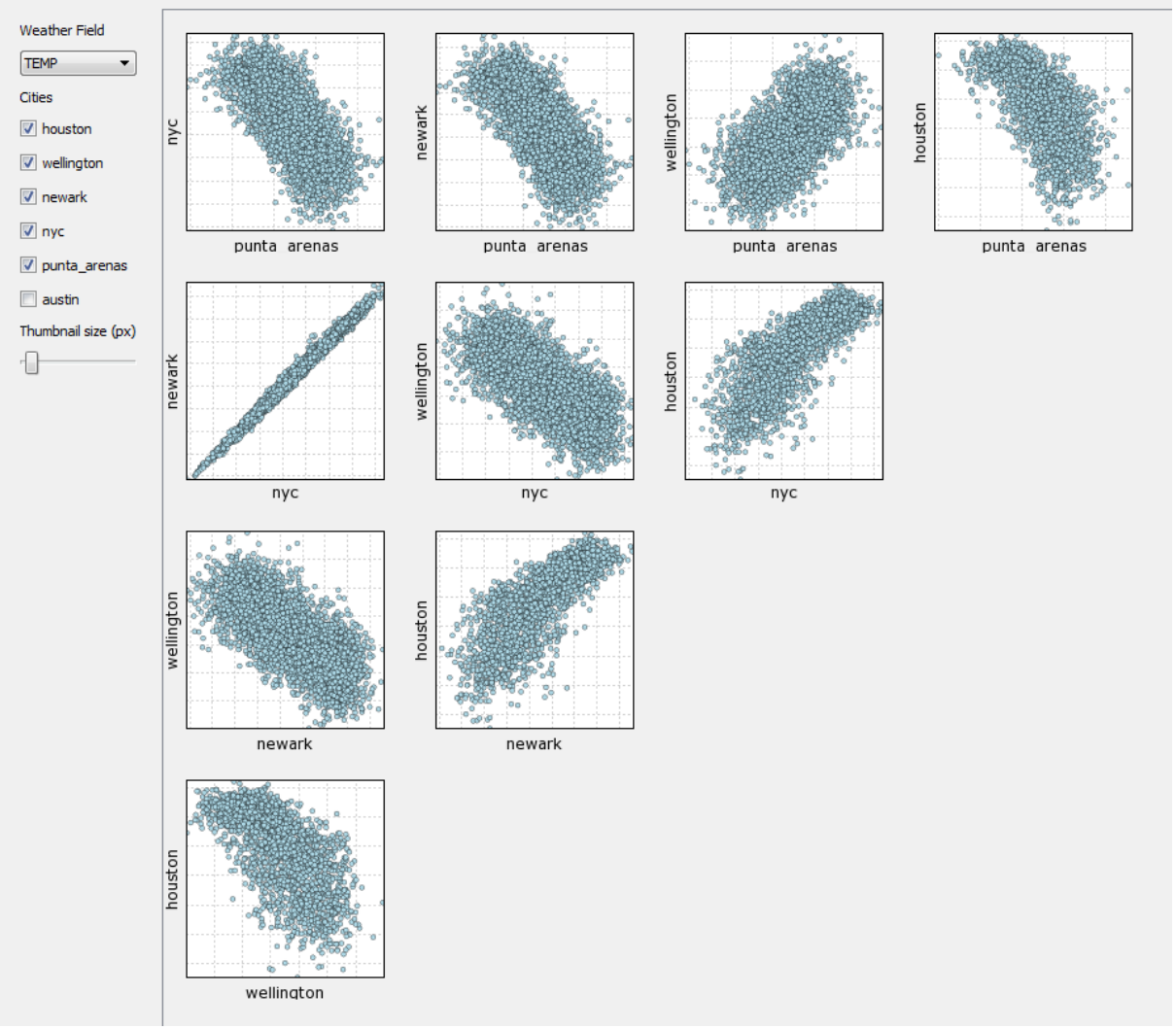
Plot weather data



Source code at http://www.github.com/jonathanrocher/climate_model/

Comparing …
Even More Data

Scatter plot matrix



Filename: scatter_matrix.py

Can I learn something from this data?

Learning from data

- Classify data into categories
- Optimize a function wrt input paremeters
- Create predictive model from data

Support vector machines

Brief Interlude Into Classifiers

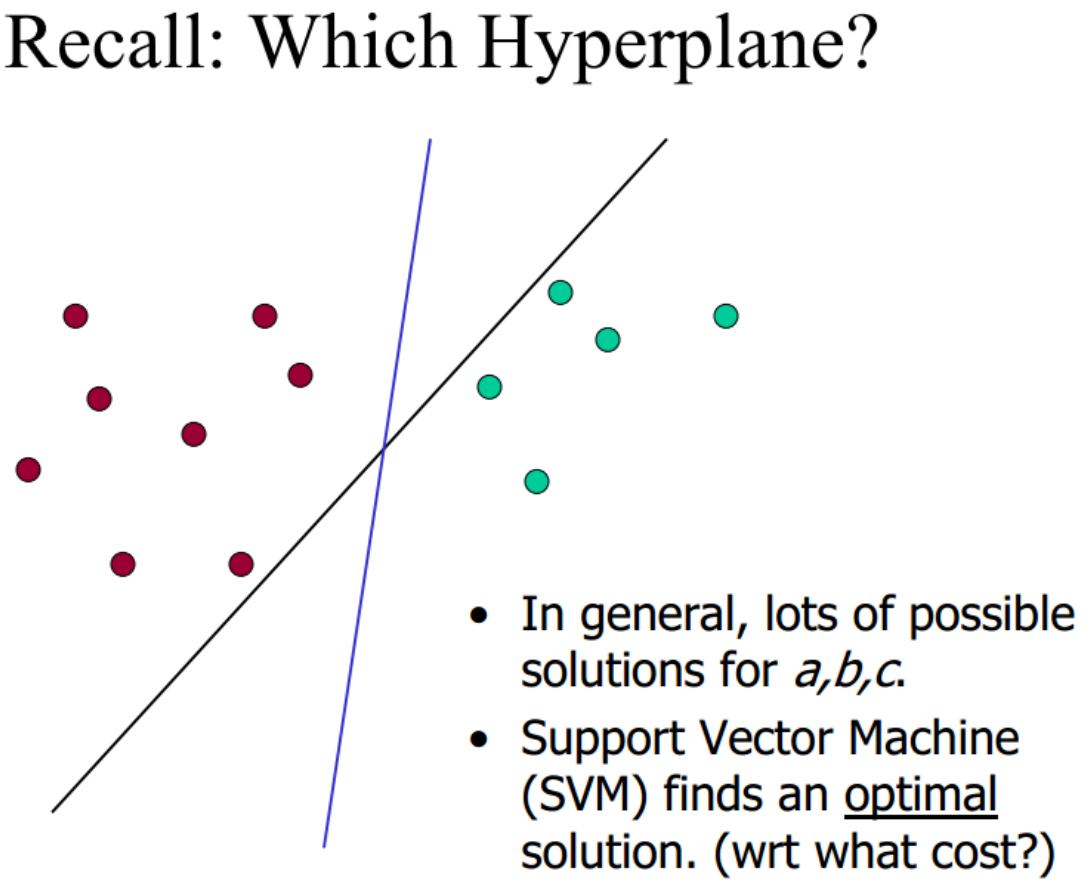
Examples

- Predict if a mail is spam or not
- Sort incoming mail into folders
- Predict if a transaction is fraudulent
- Predict if a patient has a disease

Feature vectors

Mail #	Word1	Word2	Spam?
1	0	1	Y
2	0	1	Y
3	1	0	N
4	1	1	Y
5	1	0	N
6	1	1	N

Classifying data

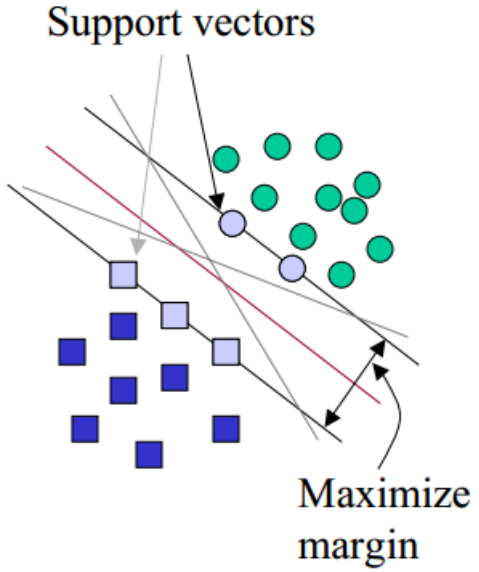


Source: Berwick2003

Support vectors

Support Vector Machine (SVM)

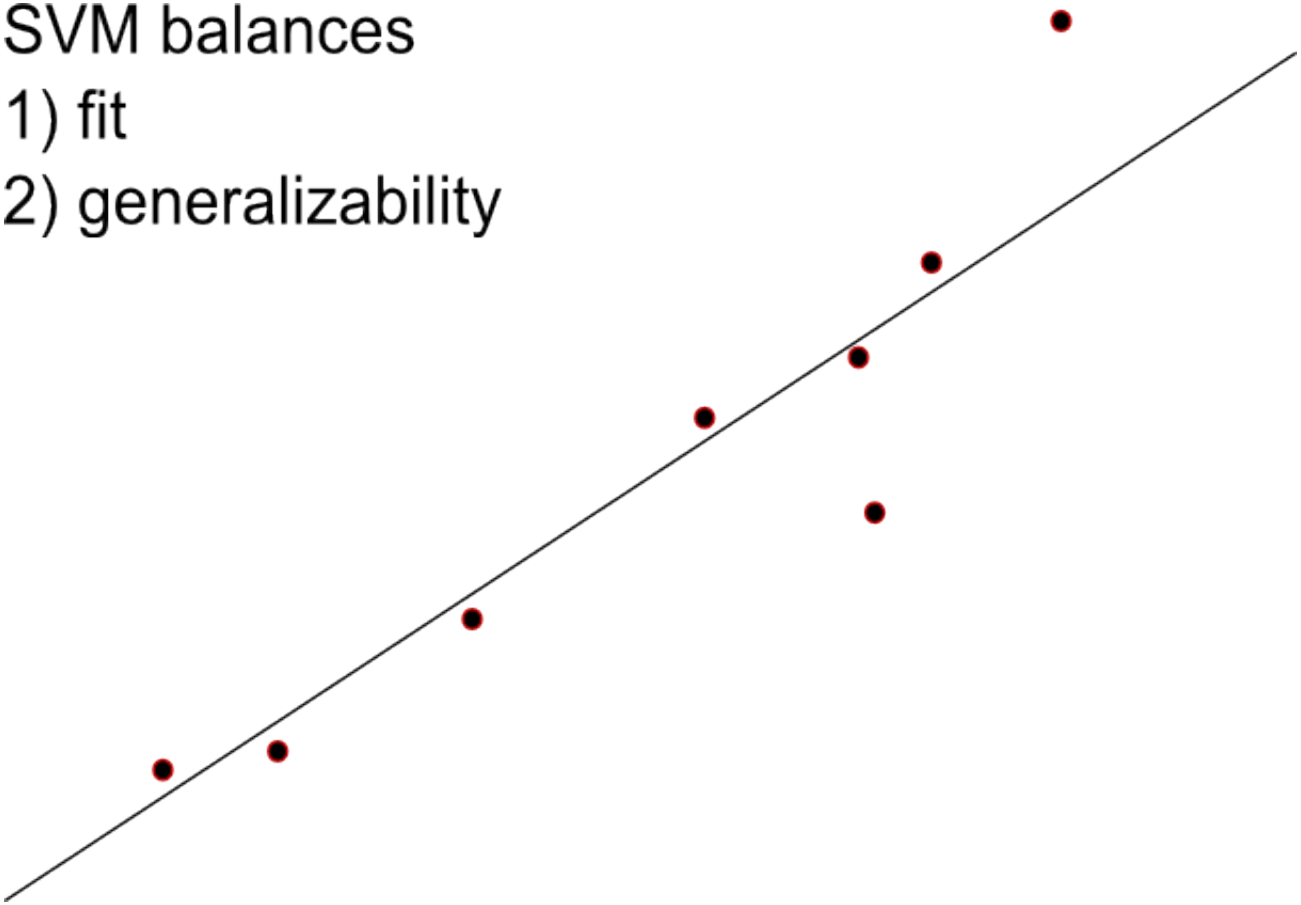
- SVMs maximize the *margin* around the separating hyperplane.
- The decision function is fully specified by a subset of training samples, *the support vectors*.
- *Quadratic programming* problem
- Text classification method du jour



Source: Berwick2003

Support Vector Regression

SVM balances
1) fit
2) generalizability



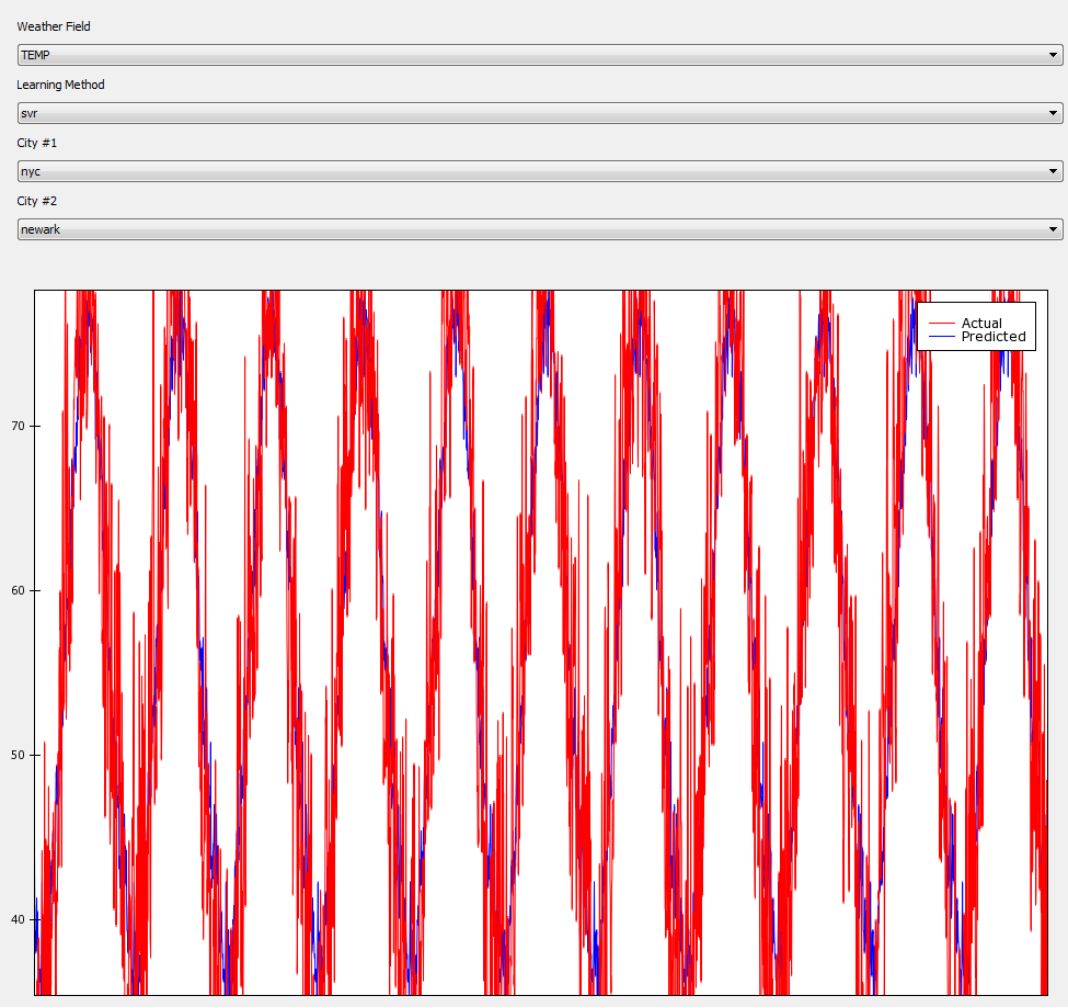
Scikits learn

http://scikit-learn.org

Slide showing predictor

```
from sklearn.svm import SVR
clf = SVR(epsilon=0.2)
clf.fit(X, y)
pred = clf.predict(test)
```

Learn from weather data



Filename: ml_app.py

Applications for this analysis

- Impact of sales campaign
- Effect of hiring star athlete
- Effect of upgrading computer infrastructure
- Predict stock prices ?

Source code repository

https://github.com/jonathanrocher/climate_model/tree/pygotham

Credits for talk

- Jonathan Rocher – This talk builds upon his talk from PyCon
- Naveen Michaud Agrawal – Wrote code for mapping weather stations
- Chris Colbert – Helped debug several issues and and gave Enaml advice
- Sean Ross – Feedback on this talk

Network IO

- Urllib2
- Requests
- Paramiko

Python data structures

- *Numpy*
- *Pandas*
- *Blist*
- *Bitarray*

Python Visualization / Plotting

- Chaco
- Matplotlib
- Networkx