PYTHON DATA TOOLBOX

Hi. I'm Rob Story.

wrobstory.github.io


github.com/wrobstory/pdxpython2015


@oceankidbilly

I work @simple

Great company. Great team.
Interesting Data.

We’re hiring.

Question:

I have data.

It's March 2015.

I want to group things.

or count things.

or average things.

or add things.

What library should I use?

It Depends.

(cop out)

This is why we have a toolbox.

We are lucky to have a PyData ecosystem where there are domain-specific tools for different applications.

Wait: Why should I use Python for Data Analytics anyway?

While Python might not be the fastest language for things like web servers, it is *very* fast for HPC & numerics (because C)

(and maybe Rust in the future?)

Back to choosing a lib:
First we need a dataset.

Diamonds Data:

http://vincentarelbundock.github.io/Rdatasets/datasets.html

(Yep, it's an R website.
Their community is really good at dataset aggregation)

carat	cut	color	clarity	depth	table	price	x	y	z
0.23	Ideal	E	SI2	61.5	55	326	3.95	3.98	2.43
0.21	Premium	E	SI1	59.8	61	326	3.89	3.84	2.31
0.23	Good	E	VS1	56.9	65	327	4.05	4.07	2.31
0.29	Premium	I	VS2	62.4	58	334	4.2	4.23	2.63
0.31	Good	J	SI2	63.3	58	335	4.34	4.35	2.75

My needs are simple.
I don't like dependencies.
I'm on an old version of Python.

Seriously, no dependencies.

stdlib? stdlib.

stdlib works!

But...what if you have 10M rows instead of 50k?

Do you really want to spend your time writing aggregation code?

Are my functions composable? Pure? Lazily evaluated? If I write Python should I care?

What happens when my analysis gets more complicated (or uses time/dates in any way...)?

Is a list of dictionaries the best way to work with tabular data?

I like a functional approach to data analysis (purity, composable, etc)

Map, reduce, filter are my friends.

I think composing data pipelines is awesome, especially if it can handle streaming data.

Toolz!

“Each toolz function consumes just iterables, dictionaries, and functions and each toolz function produces just iterables, dictionaries, and functions.”

This is great. No new data structures to learn!

Toolz!

Toolz is great!

But...I'm still doing things at Python speeds (psst: CyToolz!)

I'm working with tabular data here- can't I have a tabular data interface?

I still have to work with time/dates. I am sad.

I have tabular data. Give me DataFrames.

I want fast/intuitive exploratory analytics.

I want a really, really fast CSV importer.

I want easy interfacing with SQL.

I have timeseries data. Help. Please.

Pandas!

Pandas rocks!

But...I'm working with a lot of N-dimensional data, where I need to do fast numerics on homogeneous arrays.

That being said...I don't want to give up on all of Pandas nice features.

I have N-Dimensional homogeneous arrays.

I want to be able to work with labeled data in a tabular format with these arrays.

I want to serialize to NetCDF (or anything else where homogeneity means fast(er) serialization)

XRAY!

xray is awesome!

But...I want to work with data in a Pandas like way across many data sources.

What if I have some data in SQL, some data in CSVs, some data in HDF5, some data in...

What if I want to do out-of-core computation?

I have BIG DATA. or maybe just Medium Data. How about Bigger-than-I-can-RAM-data.

Why are my analytical/computational expressions tied to my data structure? Can I have expressions that map across data structures *and* storage?

Blaze!

YAY BLAZE!

But...I have Big/Medium/Lots of Data, and Databases aren't fast enough, and I'm running out of disk space.

Isn't there some way to compress my data somehow for these in-memory computations?

I have homogeneous array data (remember xray?)

I want to compress it both in-memory/on-disk, but have that compression be fast enough to perform useful analytics.

bcolz!

bcolz is a big deal!

Being able to perform out-of-core computation on compressed data structures is the next-big-thing for medium data analytics.

But...what about everything else in the Pydata Ecosystem?

What's Left?

(there's more?!)

Numpy!

It would be a disservice to have a Python Data Toolbox presentation and not talk about Numpy.

It is the foundation on which almost *all* of the tools we have talked about today are built.

Pandas? Deeply tied to Numpy.
xray? Directly exposes Numpy.
bcolz? Has it's own array type, still uses Numpy tooling.
Blaze? Most of the internals are working with and leveraging numpy ndarray.

Numpy is still hugely important to PyData

Dask!

“Dask encodes directed acyclic graphs of task-oriented computations using ordinary Python dicts, tuples, and functions.”

Wat?

tl;dr, it allows you to perform out-of-core computation by partitioning arrays/dataframes and performing parallel/scheduled computations on those partitions. Then it puts the pieces back together.

Spark!

"...fast and general-purpose cluster computing system. "

Could do an entire presentation on Spark.

Remember toolz? Think the same type of map/ reduce/flatMap/filter dataflow, except distributed.

Also: Streaming data! Machine Learning! DataFrames! SQL on Spark! Warning, JVM!

THE END!

THANK YOU!