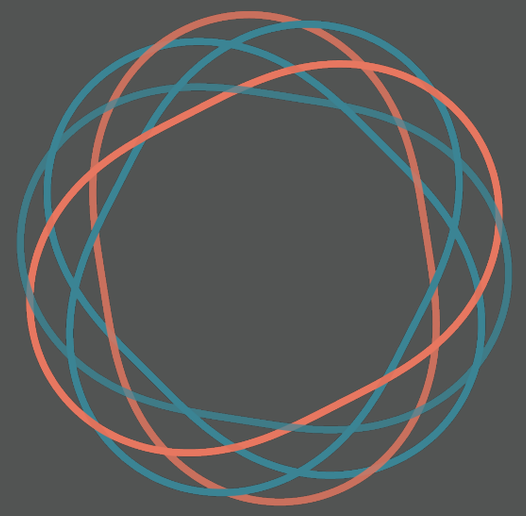
PYTHON DATA BIKESHED

Hi. I'm Rob Story.

github.com/wrobstory/pydataseattle2015

@oceankidbilly

I work @simple



Great company. Great team.
Interesting Data.

We’re hiring.

(A little Python. A lotta JVM)

Question:

I have data.
It's July 2015.
I want to group things.
or count things.
or average things.
or add things.

What library should I use?

It Depends.

(cop out)

Enter:
The Bikeshed

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

Analogy:

Python Data Libs :: Bikes*

Think about tools in terms of ***analysis velocity*** and ***data locality***

*This analogy is going to fall apart extremely quickly but it's kind of fun so lets just run with it

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, old version of Python.

Seriously, **no dependencies.**

stdlib? **stdlib.**

City Bike

Reliable. Familiar. A bit slow.

Velocity: Slower

Locality: Local Memory

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!

Fixie?

“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!

Bike Tools

Velocity: Slow (toolz), Faster (cytoolz)

Locality: Local Memory

Toolz is great!

But…I'm still doing things at CPython speeds (with toolz)

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.**

A brief interlude...

Cython!

"...a superset of the Python language that additionally supports calling C functions and declaring C types"

Numexpr!

"... fast numerical expression evaluator for NumPy"

Numba!

JIT compilation to LLVM with only a few decorators needed


Made in USA 7075
130mmx53t

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 critical PyData infrastructure

/interlude

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!

Geared Commuter

Daily use. Lots of features.

Velocity: Faster

Locality: Local Memory

Pandas rocks!

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

That being said…I don't want to give up on all of Pandas nice **labeling and indexing** features.

I have **N-Dimensional homogeneous arrays**.

I want to be able to easily **aggregate data in multiple dimensions**.

I want to serialize to **NetCDF**.

I might have to deal with **OpenDAP**

XRAY!

BMX Bike

Specialized. Multi-dimensional.

Velocity: Faster

Locality: Local Memory, Multi-Core

xray is awesome!

But…I want to work with data with Pandas-like expressions 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 expressions** tied to my data structure?

Can I have **expressions that map across data structures *and* storage?**

Blaze!

Mountain Bike

Multi-speed, Multi-terrain

Velocity: Varied (computation engine)
Locality: Varied (data source)

YAY BLAZE!

But…I have Big/Medium/Lots of Data, and Databases aren't fast enough, and both **memory and disk space are at a premium**.

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

I have **homogeneous array data**

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

bcolz!

Recumbent

Fast, some real advantages, specific audience

Velocity: Fast

Locality: In-memory/On-disk

bcolz is a big deal!

Being able to perform **aggregation on compressed data on-disk or in-memory** is huge for medium data analytics.

But…what if I *really* want **parallel, out-of-core computing?**

Dask!

"enables **parallel computing** through **task scheduling** and **blocked algorithms**."

Recipe for all-your-cores and out-of-core computation:

1. Partition arrays/iterables/dataframes (blocked algorithms)
2. Perform parallel/scheduled computations on those partitions.
3. Put the pieces back together

Dask!

Arrays: Numpy-Like

DataFrames: DataFrame-like

Sequences: Seq-like

Tandem

Fast, Parallel

Velocity: Fast

Locality: In-memory/On-disk/Distributed

What's Left?

(there's more?!)

Spark!

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

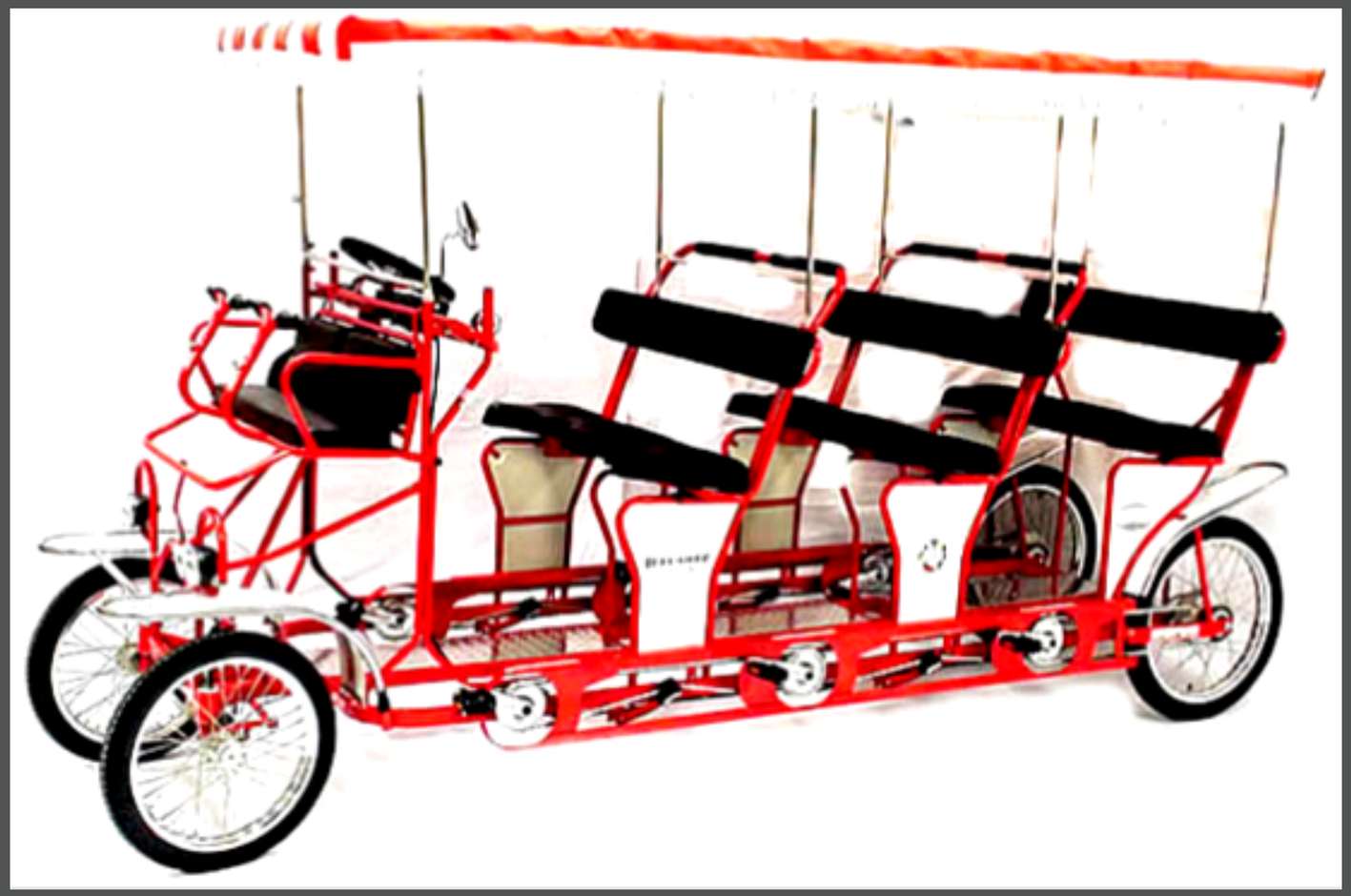
Could do an entire presentation on Spark/PySpark.

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

Also: **DataFrames! Streaming data! Machine Learning!**

Multi-person Surrey

Fast*, Parallel, Distributed



Velocity: Fast

Locality: In-memory/On-disk/Distributed

*This analogy has now completely come off the rails

JVM

Distributed ndarray

- **Bolt:** ndarray backed by Spark (and more?)
- **SArray:** disk/in-memory ndarray
- **SFrame:** disk/in-memory DataFrame
- **DistArray:** Distributed-Array-Protocol
- **Biggus:** Virtual large arrays + lazy eval
- **Spartan:** “Distributed Numpy”


momondo
FIND FLY OG HOTEL

Ibis!

"…pandas-like data expression system"

Built on **Impala** + **Hadoop** (for now)

Some things are **Tedious/Difficult** to do in SQL, much less Map/Reduce.

- **Timeseries queries (window functions)**
- **Correlated subqueries**
- **Self-joins**

Stats + ML

Scikit-learn

IMO, very clearly the benchmark in ML libraries, both in API and documentation

statsmodels

Statistical models in Python built on our PyData toolbox

What should I paint the bikeshed with?

Seaborn + Bokeh

THE END!

THANK YOU!