Python Big Data Analysis and Visualization

using Pandas, Bokeh and Blaze

丁来强

About me

- Splunk Lab @Shanghai
- 10+ years working experiences
- Mainly used C++/Python and JS
- Email: wjo1212 at 163.com
- Wechat: LaiQiangDing

Outline

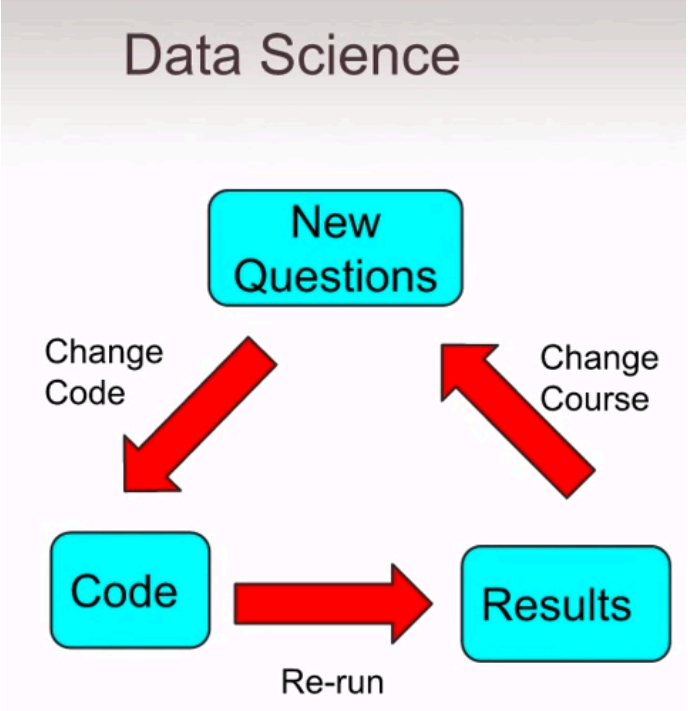
- Data Analysis with Pandas
- Visualization with BokeH
- Big Data Consideration
  - Pandas Chunk
  - Extend to DB/Spark with Blaze
  - Dask

Data Analysis with Pandas

Python Data Overview

Data Science Needs

- Rapid iteration, exploration
- Interactive data analysis
- Statistical modeling tools
- Rich visualization
- Reporting, Excel integration
- High perf computation

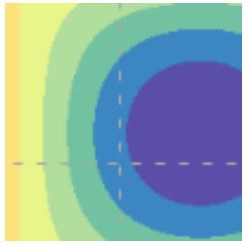


PyData Ecosystem

- Fundamental Libs:
  - numpy
  - scipy
  - matplotlib
- Advanced Libs:
  - pandas
  - sympy
  - SciKit-lean
  - xray
  - Blaze

PyData Ecosystem

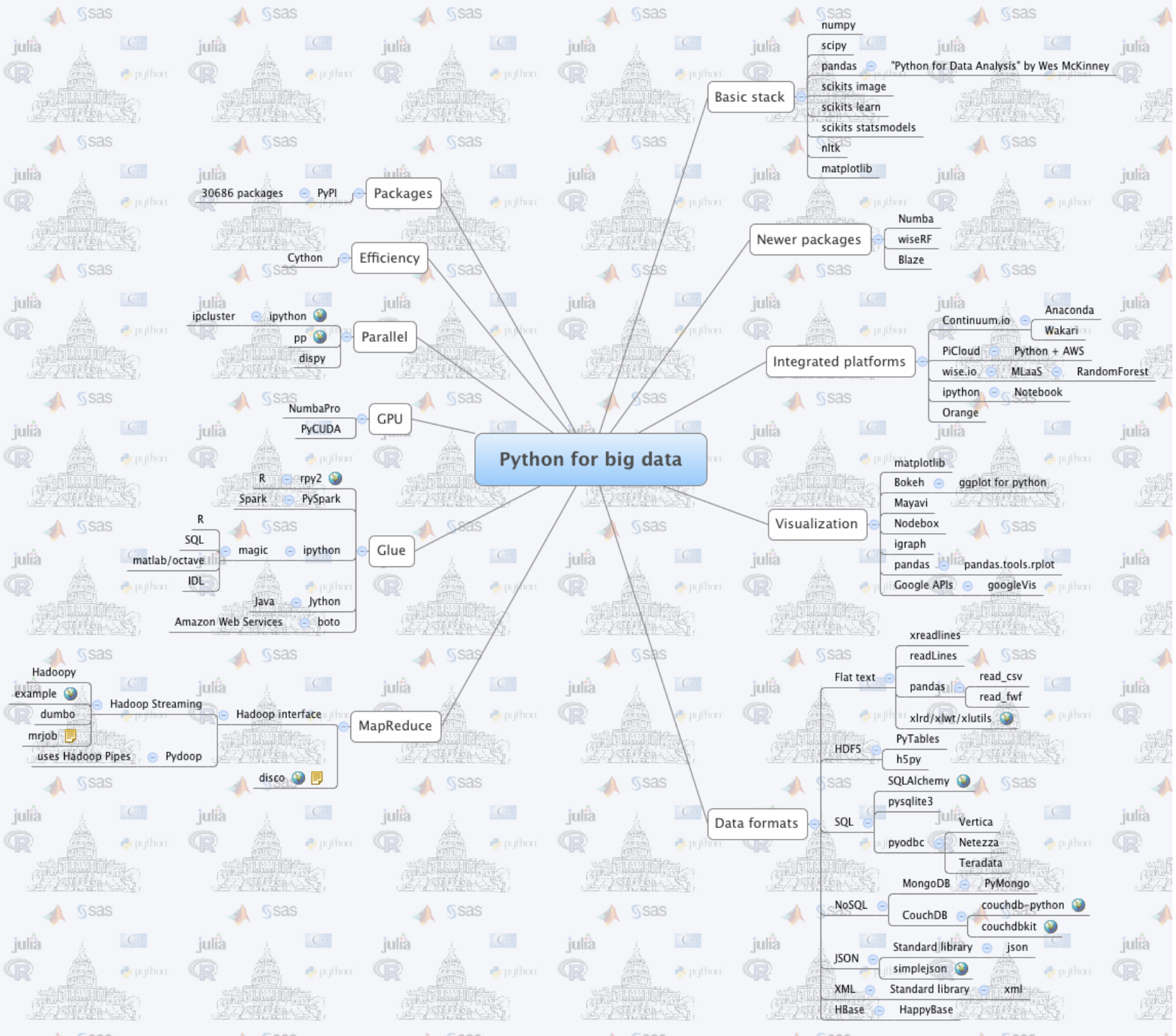
- Visualization
  - bokeh
  - yhat/ggplot
  - searborn
  - vincent, chaco, mayavi…
- IDE, tools:
  - Anaconda: IDE
  - IPython/Notebook



Bokeh

IP[y]:
IPython

ANACONDA®


Python for big data
Basic stack
numpy
scipy
pandas
"Python for Data Analysis" by Wes McKinney
scikits image
scikits learn
scikits statsmodels
nltk
matplotlib
Newer packages
Numba
wiseRF
Blaze
Integrated platforms
Continuum.io
Anaconda
Wakari
PiCloud
Python + AWS
wise.io
MLaaS
RandomForest
ipython
Notebook
Orange
Visualization
matplotlib
Bokeh
ggplot for python
Mayavi
Nodebox
igraph
pandas
pandas.tools.rplot
Google APIs
googleVis
Data formats
Flat text
xreadlines
readLines
pandas
read_csv
read_fwf
xlrd/xlwt/xlutils
HDF5
PyTables
h5py
SQL
SQLAlchemy
pysqlite3
pyodbc
Vertica
Netezza
Teradata
NoSQL
MongoDB
PyMongo
CouchDB
couchdb-python
couchdbkit
JSON
Standard library
json
simplejson
XML
Standard library
xml
HBase
HappyBase
MapReduce
Hadoop interface
Hadoopy
Hadoop Streaming
example
dumbo
mrjob
Pydoop
uses Hadoop Pipes
disco
Glue
R
rpy2
Spark
PySpark
magic
ipython
SQL
matlab/octave
IDL
Java
Jython
Amazon Web Services
boto
GPU
NumbaPro
PyCUDA
Parallel
ipcluster
ipython
pp
dispy
Efficiency
Cython
Packages
30686 packages
PyPI

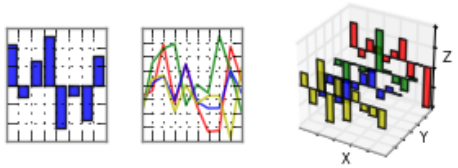
- Fast array lib in C
- ndarray: multi-D array object
- Linear algebra operations
- Random number generation
- Efficient binary IO

IP[y]: IPython
Interactive Computing

- Rich, interactive shell env.
- inline plotting support
- Web Notebook Format and wiki style
- Tab completion and introspection
- % magic commands
- Profile tools

pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



- Powerful data handling tool built on NumPy
- Started from 2008, open source
- Widely used in quant, finance
- Mature, well tested
- Powerful time-series capabilities
- Well supported by other libraries
- Active improvement

Pandas vs. R

- More time series features, better perf.
- Better perf for reshaping and groupby
- Symmetric treatment of row-column operations
- No ggplot2 equivalent, but has certain ecosystem support.
- Python based, more productive for integration

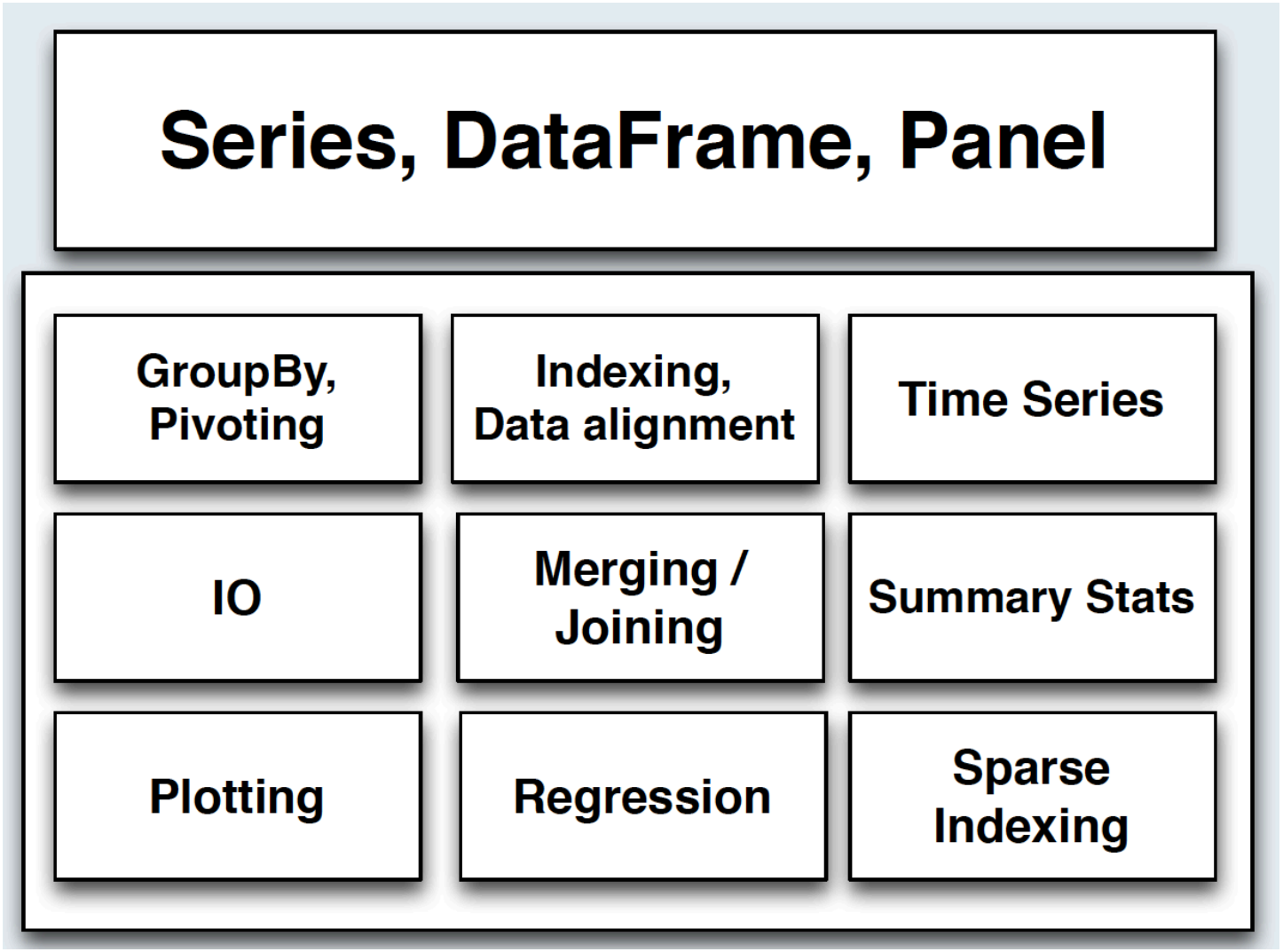
A practice with Pandas

Scenarios

- How to read, contact, merge data sources
- How to handling messy data, normalize and re-shape data
- How to transform and extract data
- How to select, filter and join data
- How to group and analyze data
- How to resample time-series data

Pandas Quick View

Pandas

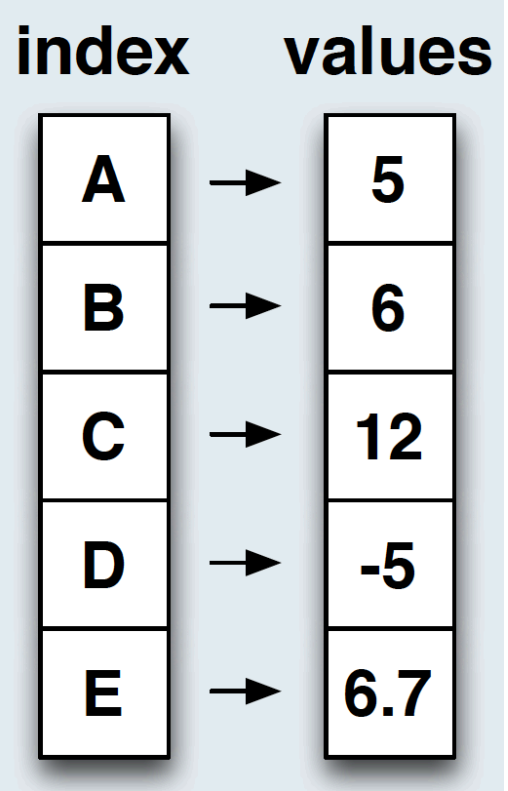

Series, DataFrame, Panel
GroupBy, Pivoting
Indexing, Data alignment
Time Series
IO
Merging / Joining
Summary Stats
Plotting
Regression
Sparse Indexing

Pandas IO

- **Traditional Text:**
  - CSV, excel
- **Managed Storage:**
  - hdf, SQL DBMS
- **Web API:**
  - json, html
- **Others:**
  - stata, clipboard, pickle
  - msgpack, gbq (experimental)

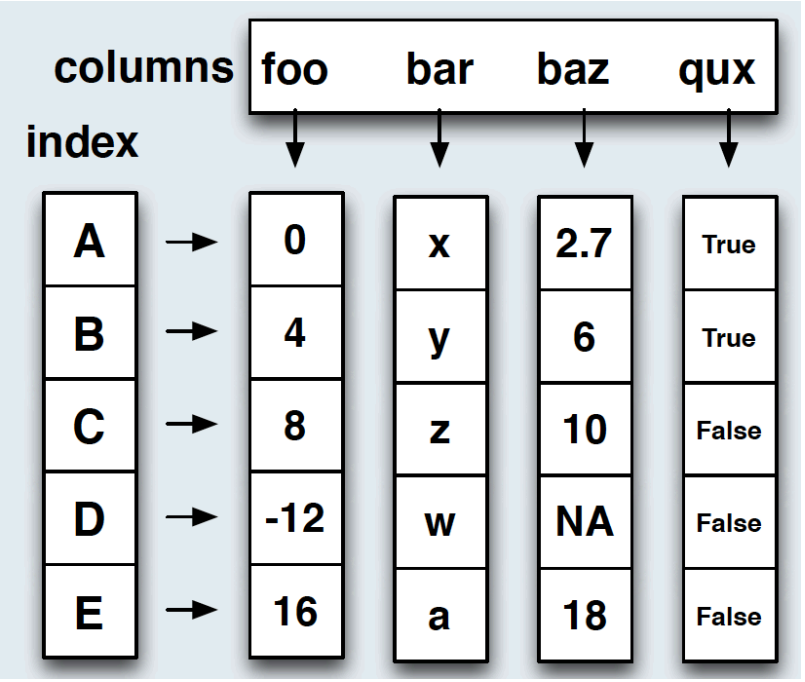
Series

- 1D labeled array for cross-sections, time series
- Array of data, any type
- Array of labels, the ‘index’
- Index could be integer, time, category, period, any others.



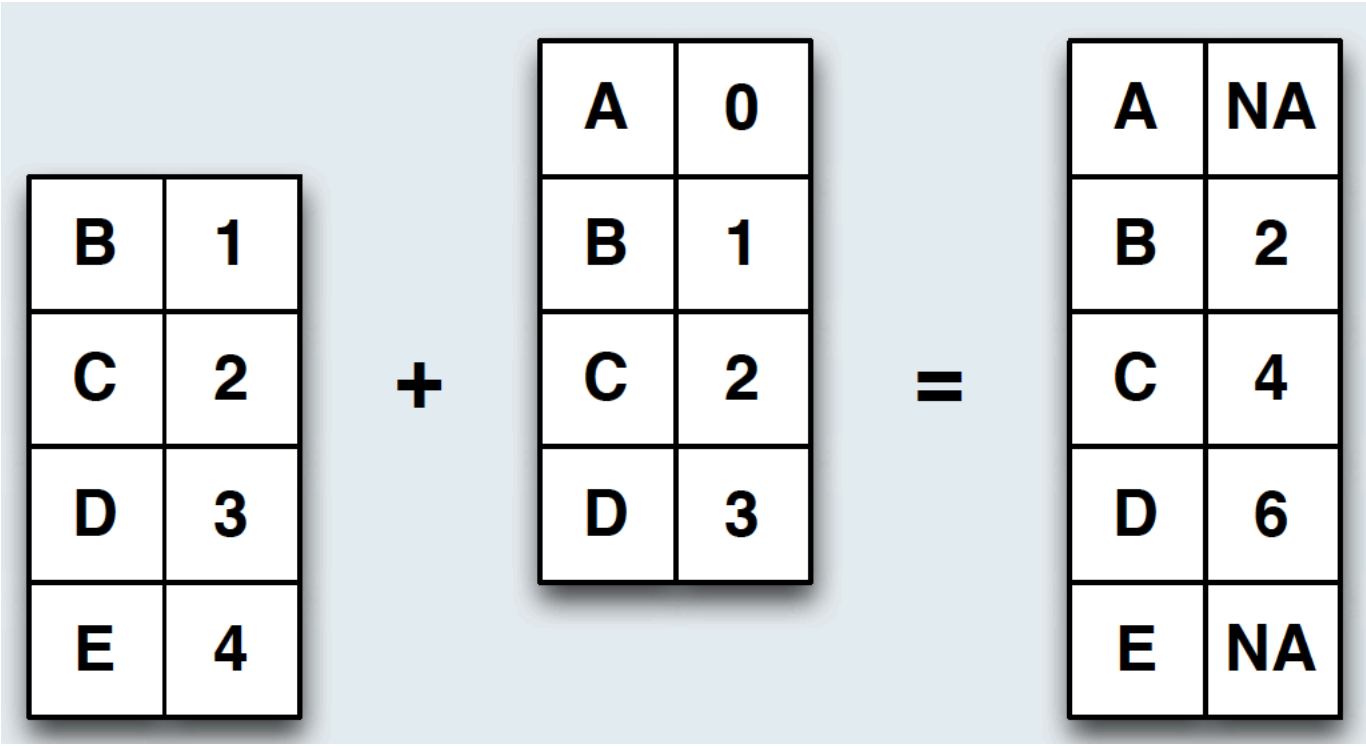
DataFrame

- Table of series objects
- Column could be different types
- Shared row index
- Powerful and handy row-column operations
- Multi-levels indexes are supported



Data Alignment

- Arithmetic auto-aligns data on labels
- DataFrame aligns on row and columns labels



Indexing and Selections

- Select rows/columns by position or labels
- Slices chunks of objects without copying
- Columns operations are easily
- Hierarchy indexing: multiple levels of keys on a single axis
- Many time series conveniences

Time series

- Time series representations
- Fixed frequency and irregular data handling
- Data arithmetic
- Time zone handling
- Easily Resampling
- Interpolating missing values
- Moving windows functions

Date and time types

- Timestamp: specific moment in types
- Period: span of time:
  - 2015Q1, Aug 2015
- Interval: defined by 2 timestamps
- Timedelta or Duration: a length of time
  - 3 days, 20 mins

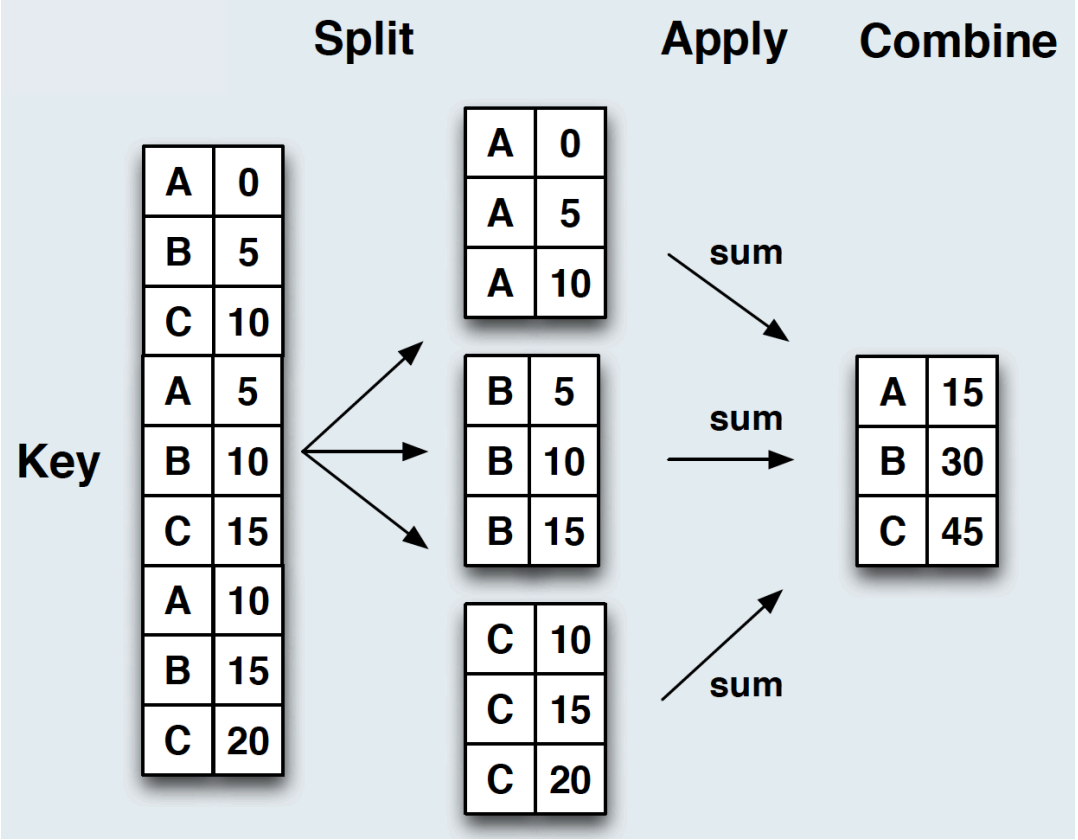
Fixed Frequency Support

- Quit important for finance domain
- Handy tools for time-series handling: shifting, frequency convert and date arithmetic

Name	Description
D	Calendar day
B	Business day
M	Calendar end of month
BM	Business end of month
MS	Calendar start of month
BMS	Calendar start of month
W-{MON, TUE,...}	Weekly on Monday, Tuesday, ...
Q-{JAN, FEB,...}	Quarterly starting on January, February...
A-{JAN, FEB, ...}	Business year end (December)
H	Hour
T	Minute
s	Second
L, ms	Millisecond
U	Microsecond

GroupBy

- Very powerful operations comparing to SQL



Some new features in recent releases

- high level query statement (like SQL)
- pipe and assign with chain operation
- string based operations

```
tips.query('total_bill > 30 and sex == "girl"')

    .assign(bill_person = tips.total_bill / tips.size,
            tip_pct = tips.tip / tips.total_bill)

    .assign(is_order_fish = tips.order.str.contains('fish'),
            is_sat = tips.time.dt.weekday == 5)

    .plot(kind='bar', x='bill_pct', y='tip_pct')
```

BokeH Quick View

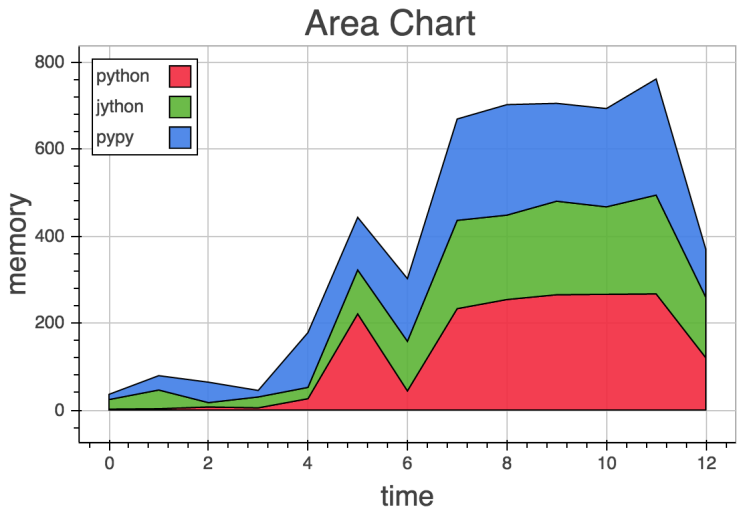
Overview

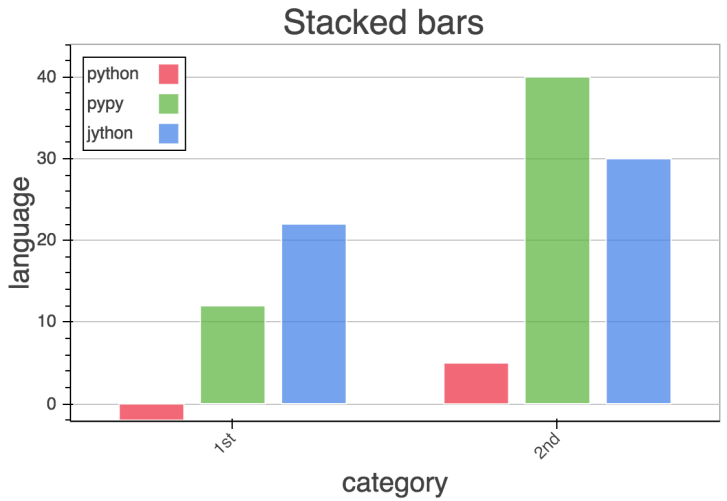
- A Python interactive visualization library
- Targets modern web browsers for presentation.
- Keep style of D3.js
- High-performance interactivity over very large or streaming datasets.
- Simplify interactive plots, dashboards, and data applications

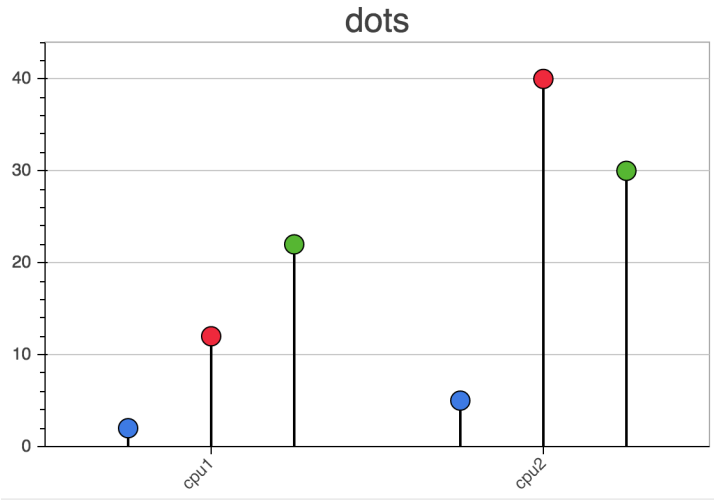
BokeH Features

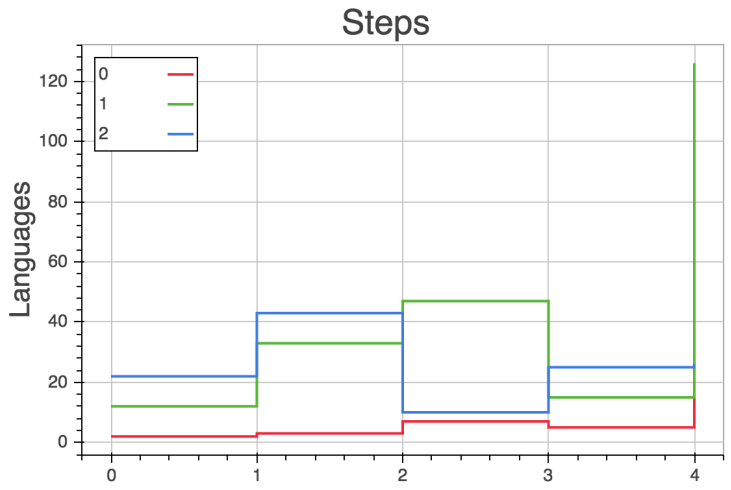
- Full control on plotting
  - From Colors, elements, figures, axis
- High level Charts
- Customization on presentation styles
- Customization on potting tools
- Customization on interactions
- BokeH servers and embedded are provided

High Level Charts

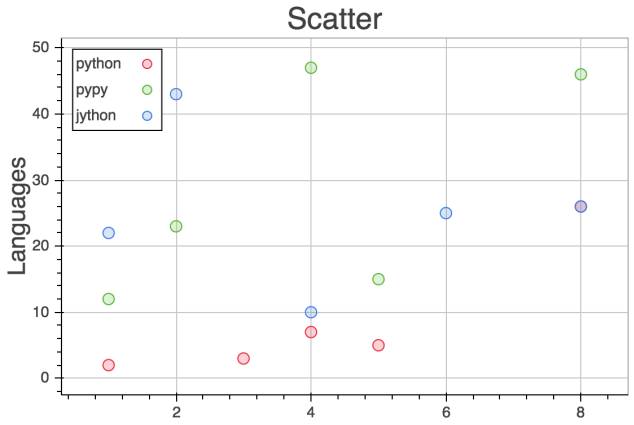


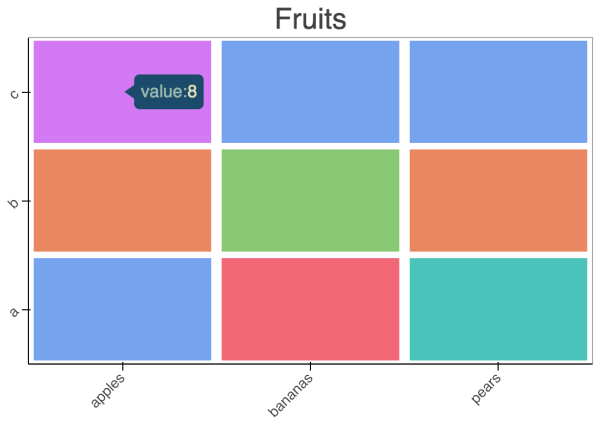


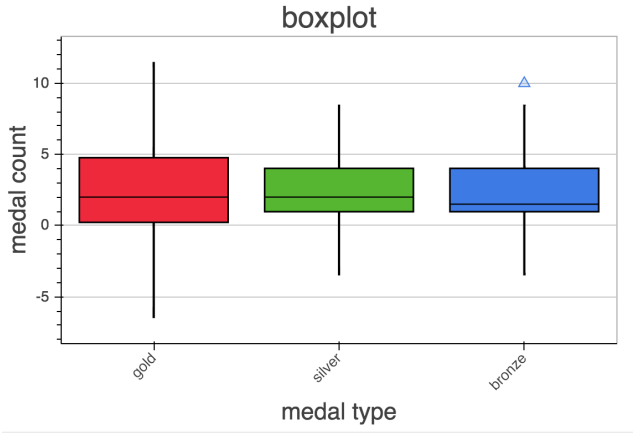


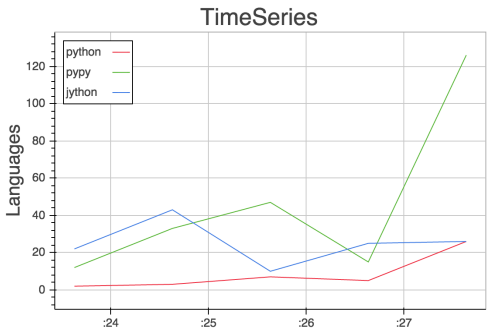


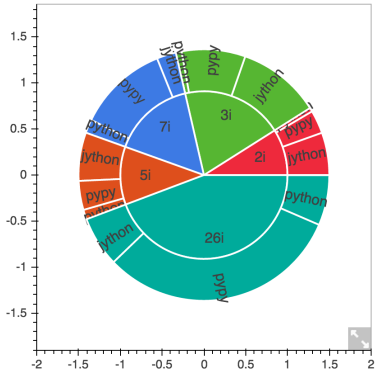
High Level Charts





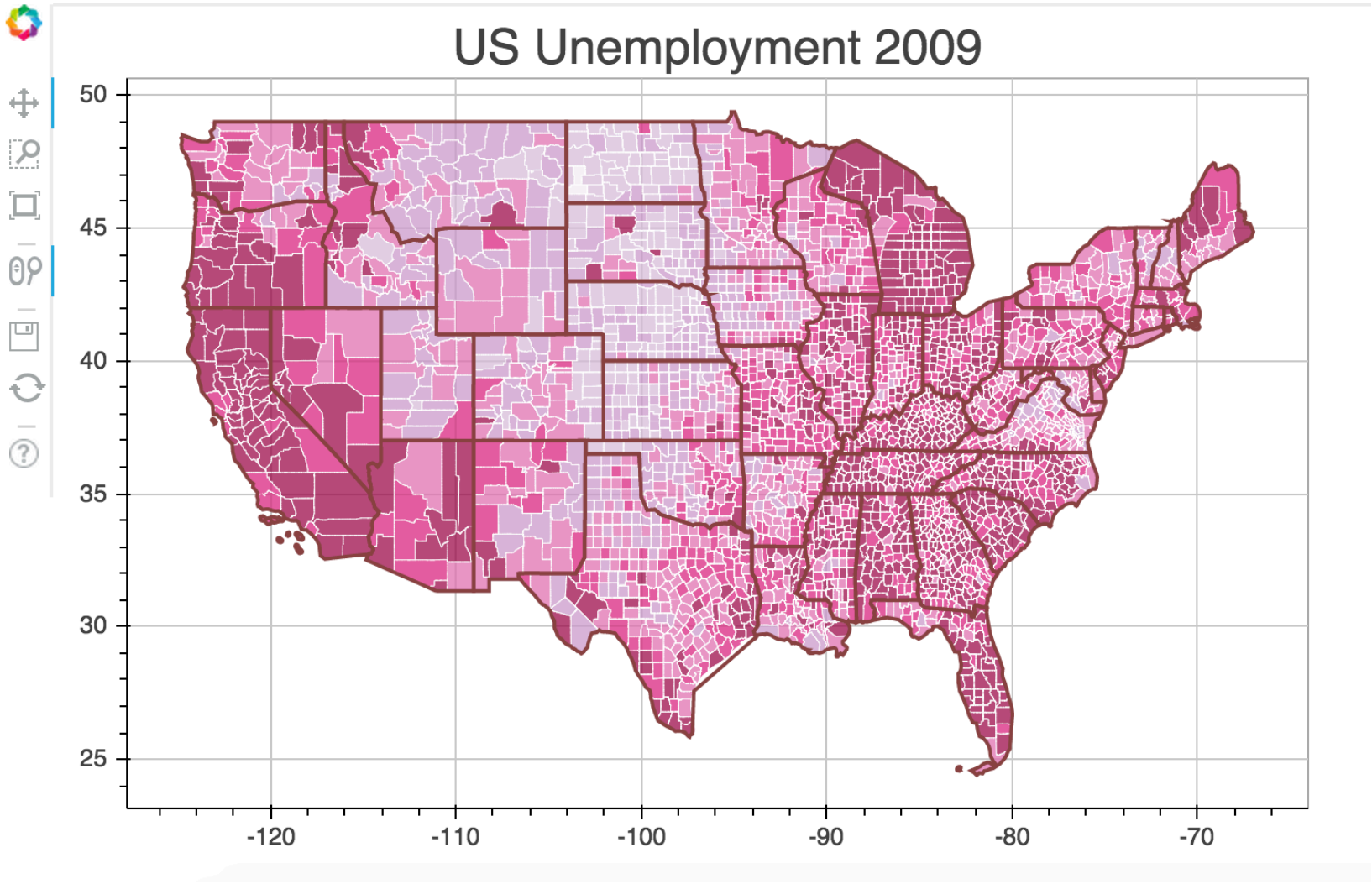




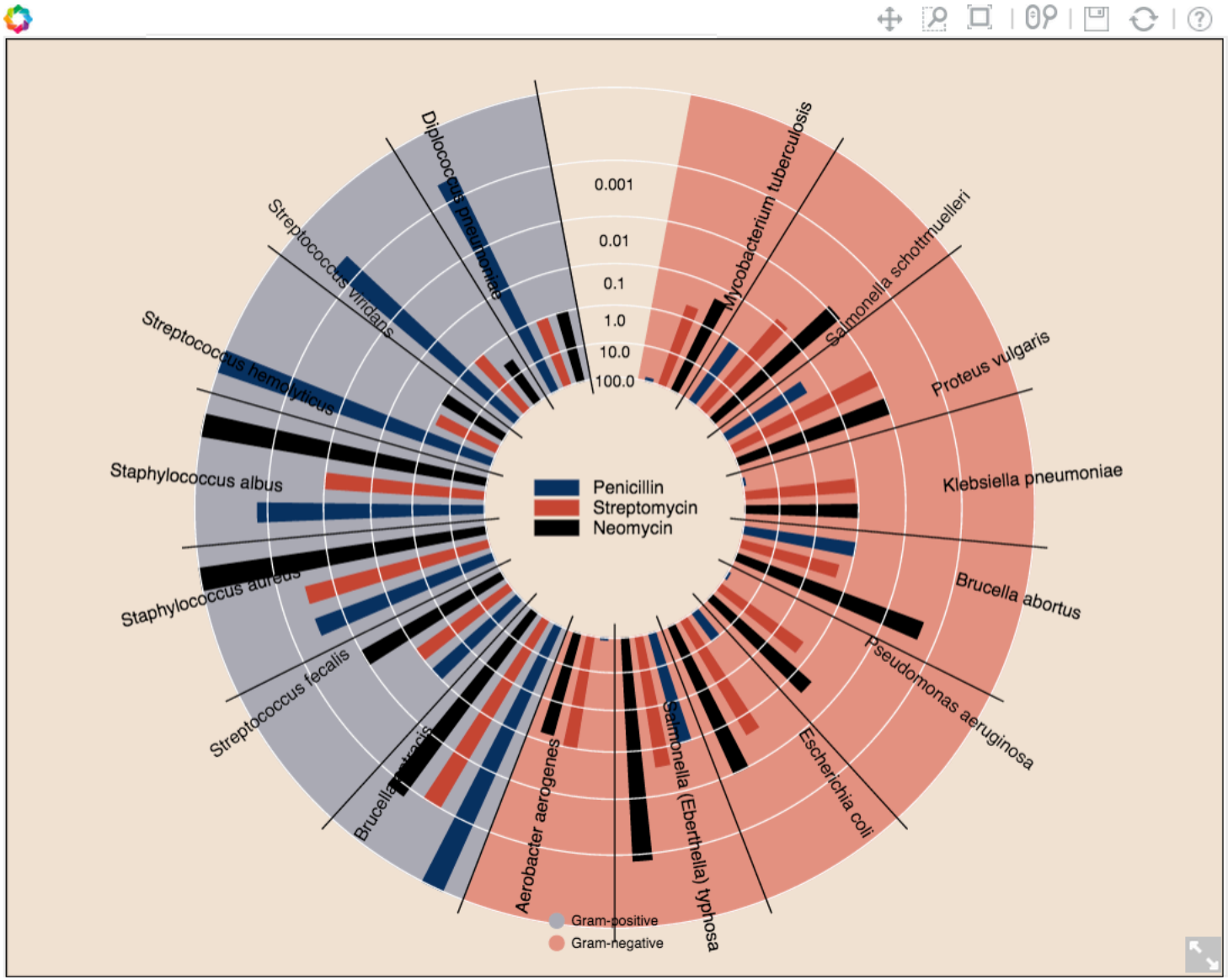


Life's pathetic, go Pythonic! (wjo1212@163.com)

Complex Charts

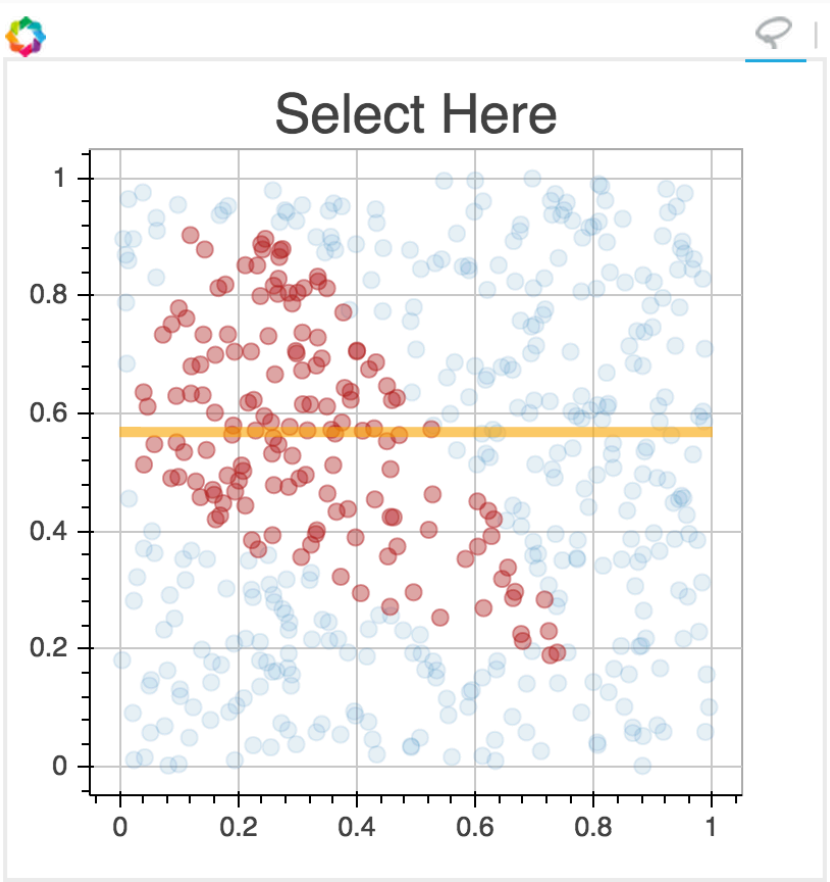


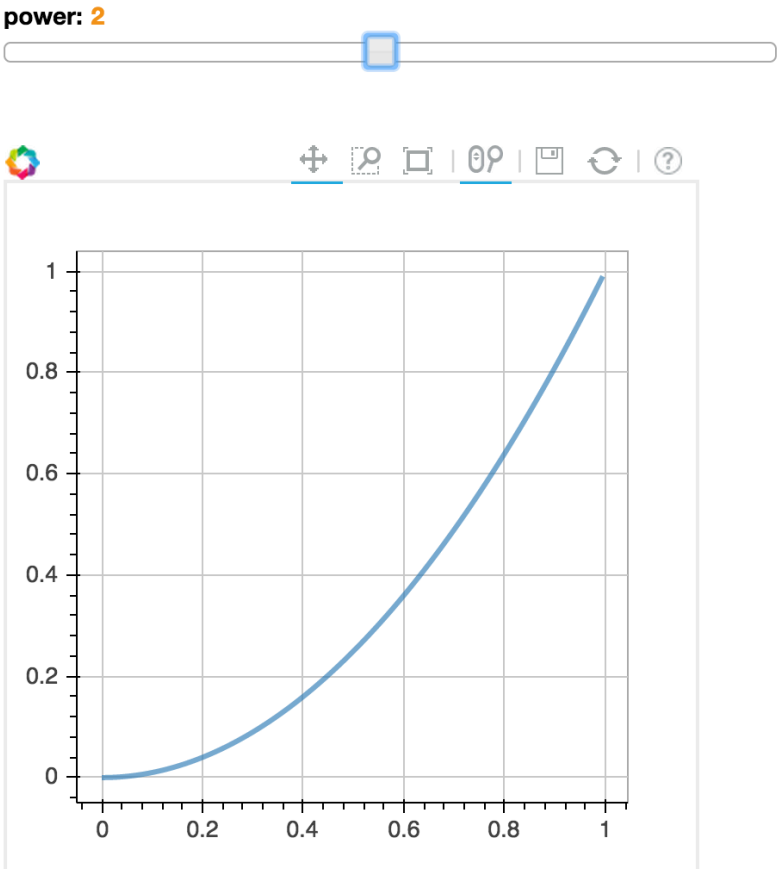
Complex Charts



PyCon 2015

Interaction





A practice with Bokeh

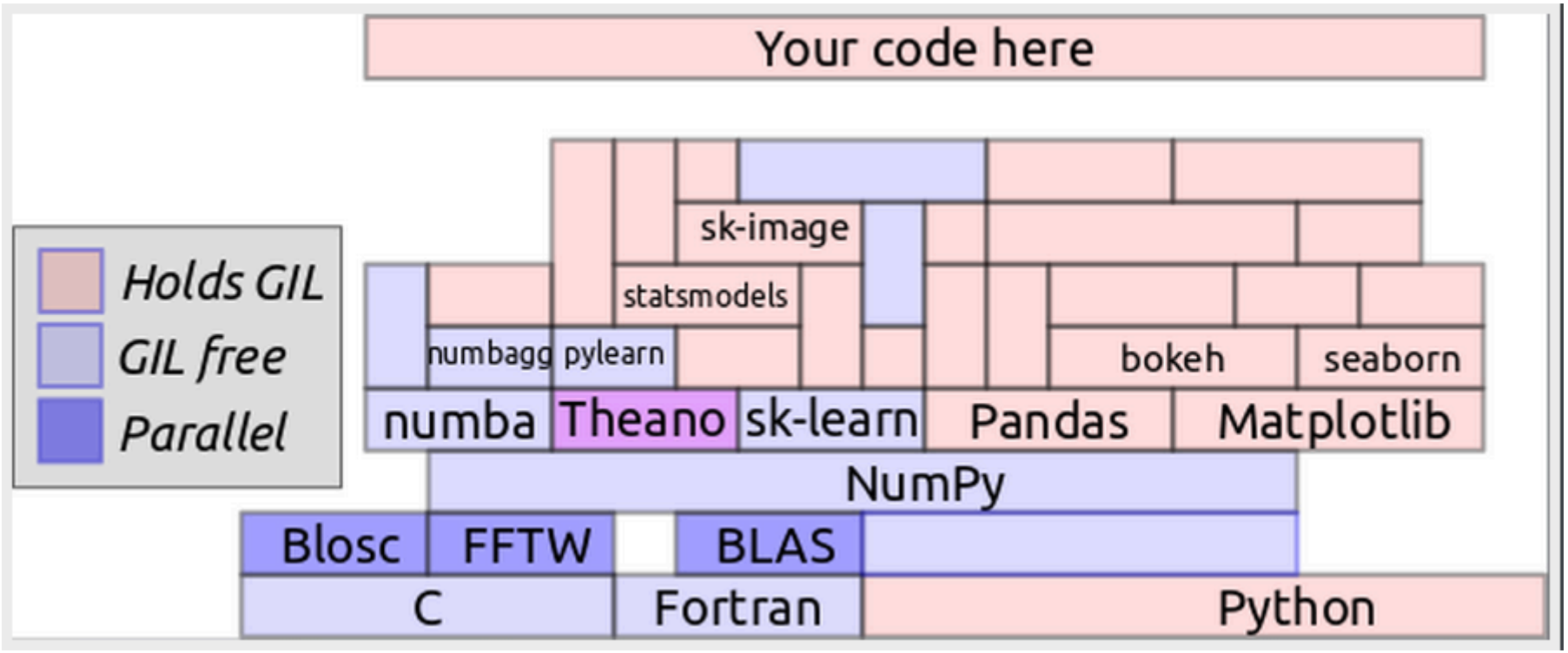
Blaze Quick View

Pandas is a fully in-memory tool..

Pandas Problems with Big data

- Fully in-memory tool
- GIL locked
- Cannot integrate with cluster solutions

GIL within PyData ecosystem



Bigger Data Handling

- Method 1 with Multi-process:

```
import pandas as pd
pd.read_csv(..., chumks=10000000)
```

- Method 2 with Blaze:

```
from blaze import Data
df = Data("sqlite:///larg.db::dta")
```

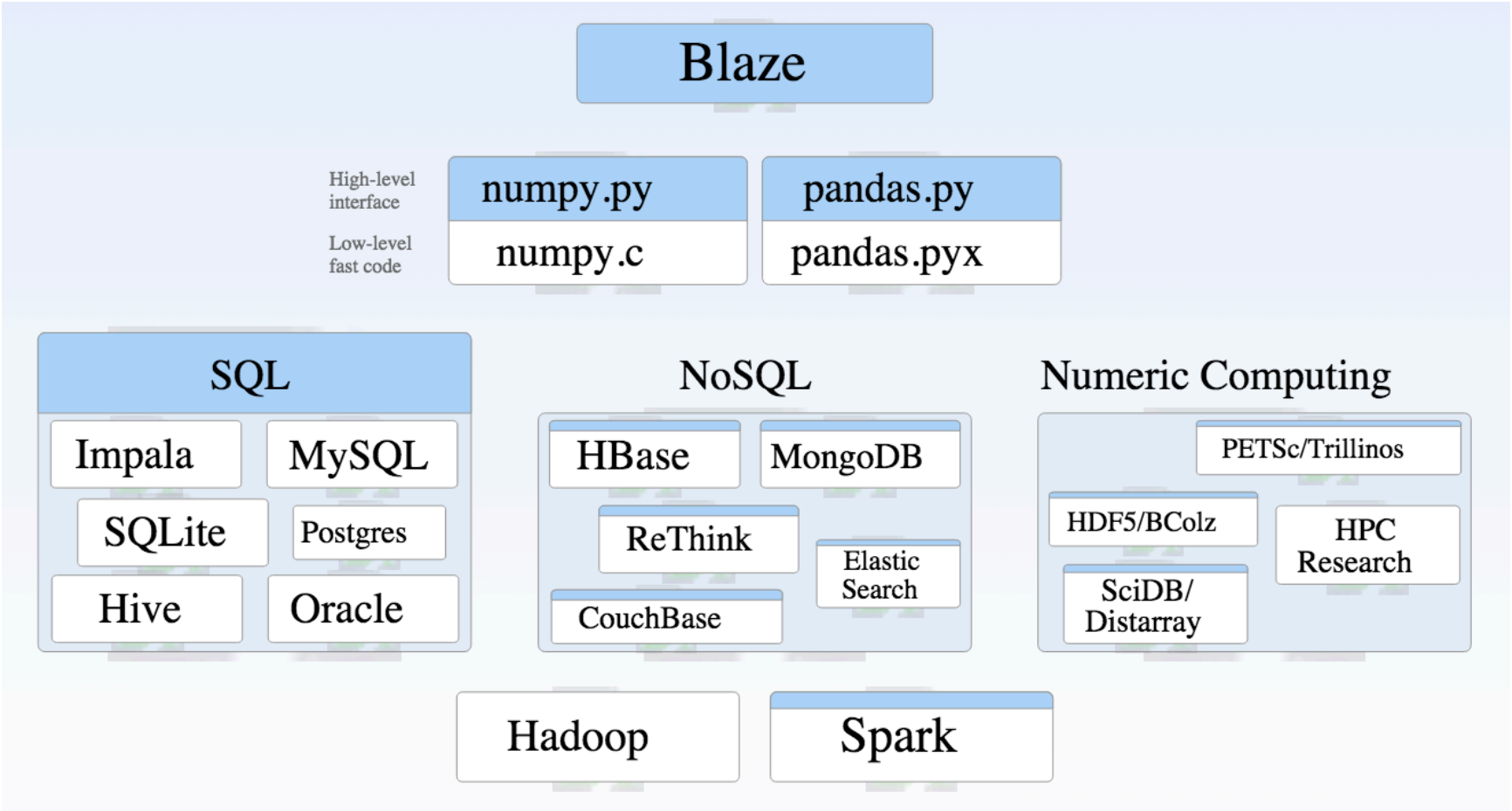
- Method 3 with Dask:

```
import dask.dataframe as dd
df = dd.read_csv('lots-of*.csv')
```

Blaze vs. Pandas

- Blaze is a single interface to query many systems.
- Blaze provides Numpy and Pandas' like syntax for user's operation. But doesn't do actual storage and operations.
- Pandas could be a kind of computing system for Blaze.

Blaze


Blaze
High-level interface
Low-level fast code
numpy.py
numpy.c
pandas.py
pandas.pyx
SQL
Impala
MySQL
SQLite
Postgres
Hive
Oracle
NoSQL
HBase
MongoDB
ReThink
Elastic Search
CouchBase
Numeric Computing
PETSc/Trillinos
HDF5/BColz
HPC Research
SciDB/ Distarray
Hadoop
Spark

Why Blaze?

- Write once, run anywhere
- Scalable development
  - (start with CSV files, end with Impala/Spark)
- Rapid prototyping
  - (try Postgres, MongoDB, Spark, see what suits you best)
- Cross-backend query optimization

Why not Blaze?

- Interfaces are quite limited
- Not as powerful and flexible as Pandas.
- Functions are limited, not for complete data normalization and reshaping purpose.
- Performance is not so good in some scenarios
- Immature and no so well tested.
- Still in quick development, interfaces changes frequently

Blaze computing logic

```
from blaze.compute import compute
from blaze.expr import Symbol
```

```
bank = Symbol('bank',
    'var * {id:int, name:string, balance:int}')

deadbeats = bank[bank.balance < 0].name
```

Blaze with Python

```
L = [[1, 'Alice',   100],
     [2, 'Bob',    -200],
     [3, 'Charlie', 300],
     [4, 'Dennis',  400],
     [5, 'Edith',  -500]]


list(compute(deadbeats, L)
['Bob', 'Edith']
```

Blaze with Pandas

- Re-use the computing logic

```
df = DataFrame([[1, 'Alice',   100],
                [2, 'Bob',    -200],
                [3, 'Charlie', 300],
                [4, 'Dennis',  400],
                [5, 'Edith',  -500]],
               columns=['id', 'name', 'balance'])

list(compute(deadbeats, df)
1       Bob
4     Edith
```

Blaze with Spark

- Re-use the computing logic

```
import pyspark

sc = pyspark.SparkContext('local', 'Blaze-demo')
rdd = odo(sc, L)

compute(deadbeats, rdd).collect()
['Bob', 'Edith']
```

About Dask

- Another version of Pandas provides similar interfaces but handled on blocked algorithms in parallel on a single machine.
- Interfaces are limited comparing to Pandas.
- Immature and no so well tested.

Demo with Bokeh (and/or Dask)

Thank you!

Reference

- **Tools used:**
  - IPython
  - Pandas / NumPy
  - Bokeh
  - Blaze / odo
  - Dask
- **Some relative presentaions**
  - http://slides.com/phillipcloud/toys-in-blaze-land
  - http://blaze.pydata.org/en/latest/_static/presentations/blaze.html
  - http://qwafafew.org/images/uploads/PANDAS_20120423.pdf

人生苦短 python当歌

Life's pathetic, go Pythonic! (wjo1212@163.com)