Interactive HPC

with Jupyter, IPython, and Dask

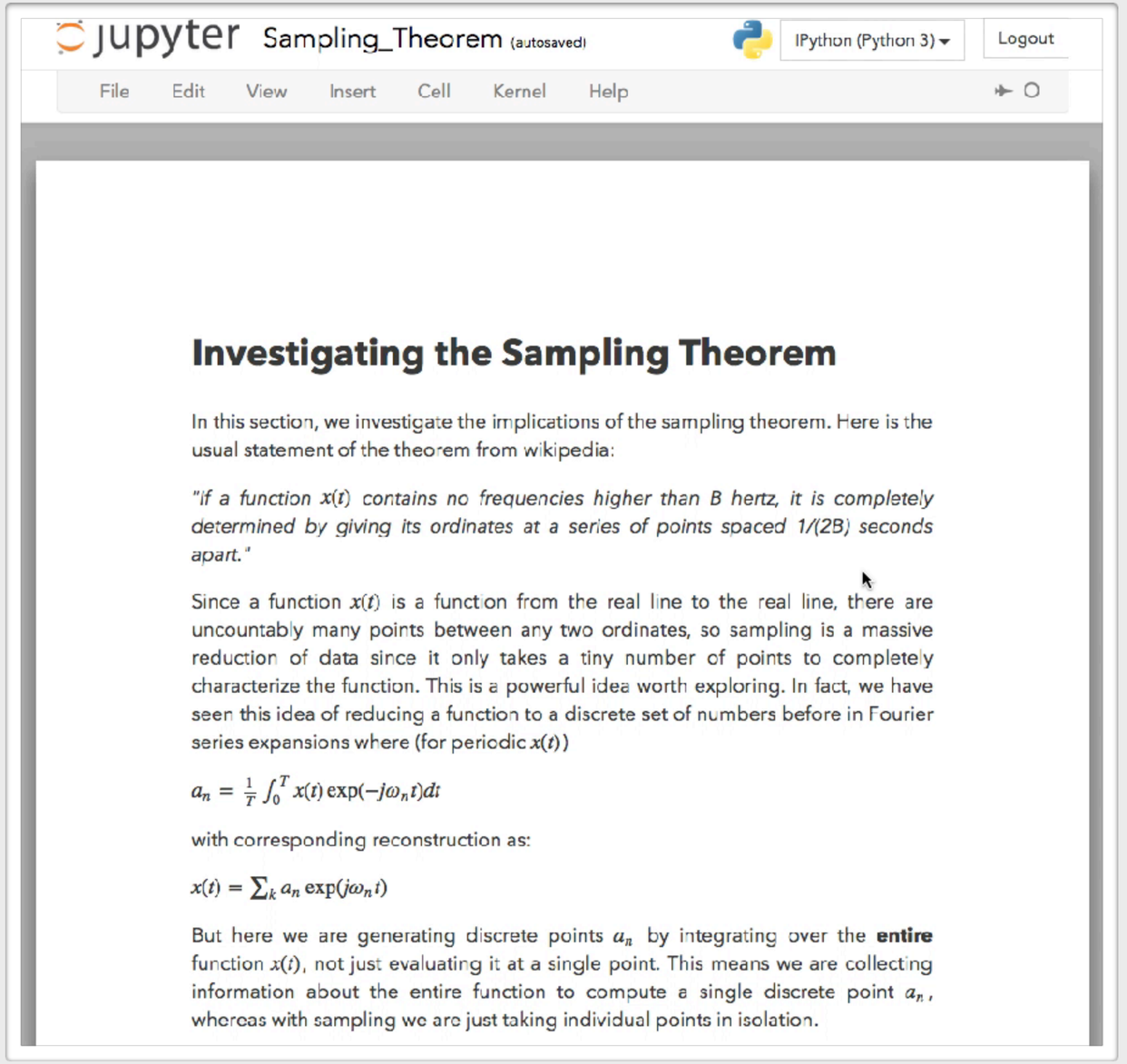
SOS 21

Min Ragan-Kelley
Simula Research Lab

Jupyter Notebooks

- Hugely popular in Data Science
- Part of reproducible research workflows
- Included in open science publications
- Used widely in education, including data science and computational sciences
- Interface to many existing Computational Environments (Google, Microsoft, IBM, Domino, more)

Jupyter Notebooks



https://github.com/unpingco/Python-for-Signal-Processing

Jupter Notebooks

JSON document format

Notebook = sequence of cells

Markdown cells with LaTeX math

Code cells with input and output

Convertible to HTML, TeX, PDF, etc.

We have already computed $P(X|A)$ above. On the other hand, $P(X|\sim A)$ is subjective: our code can pass tests but still have a bug in it, though the probability there is a bug present is reduced. Note this is dependent on the number of tests performed, the degree of complication in the tests, etc. Let's be conservative and assign $P(X|\sim A) = 0.5$. Then

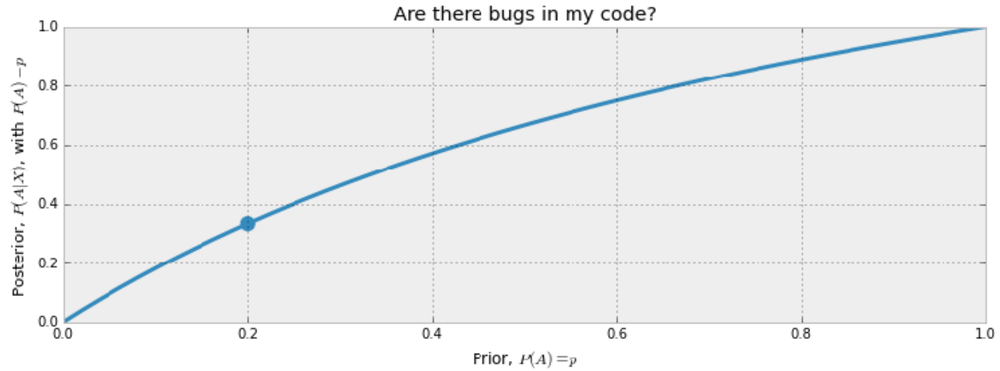
$$P(A|X) = \frac{1 \cdot p}{1 \cdot p + 0.5(1-p)}$$

$$= \frac{2p}{1+p}$$

This is the posterior probability. What does it look like as a function of our prior, $p \in [0, 1]$?

```
figsize(12.5, 4)
p = np.linspace(0, 1, 50)
plt.plot(p, 2 * p / (1 + p), color="#348ABD", lw=3)
# plt.fill_between(p, 2*p/(1+p), alpha=.5, facecolor=["#A60628"])
plt.scatter(0.2, 2 * (0.2) / 1.2, s=140, c="#348ABD")
plt.xlim(0, 1)
plt.ylim(0, 1)
plt.xlabel("Prior, $P(A) = p$")
plt.ylabel("Posterior, $P(A|X)$, with $P(A) = p$")
plt.title("Are there bugs in my coce?")
```

```
<matplotlib.text.Text at 0x1051de650>
```



https://github.com/CamDavidsonPilon/Probabilistic-Programming-and-Bayesian-Methods-for-Hackers

Jupyter Protocol: REP*L

any mime-type output

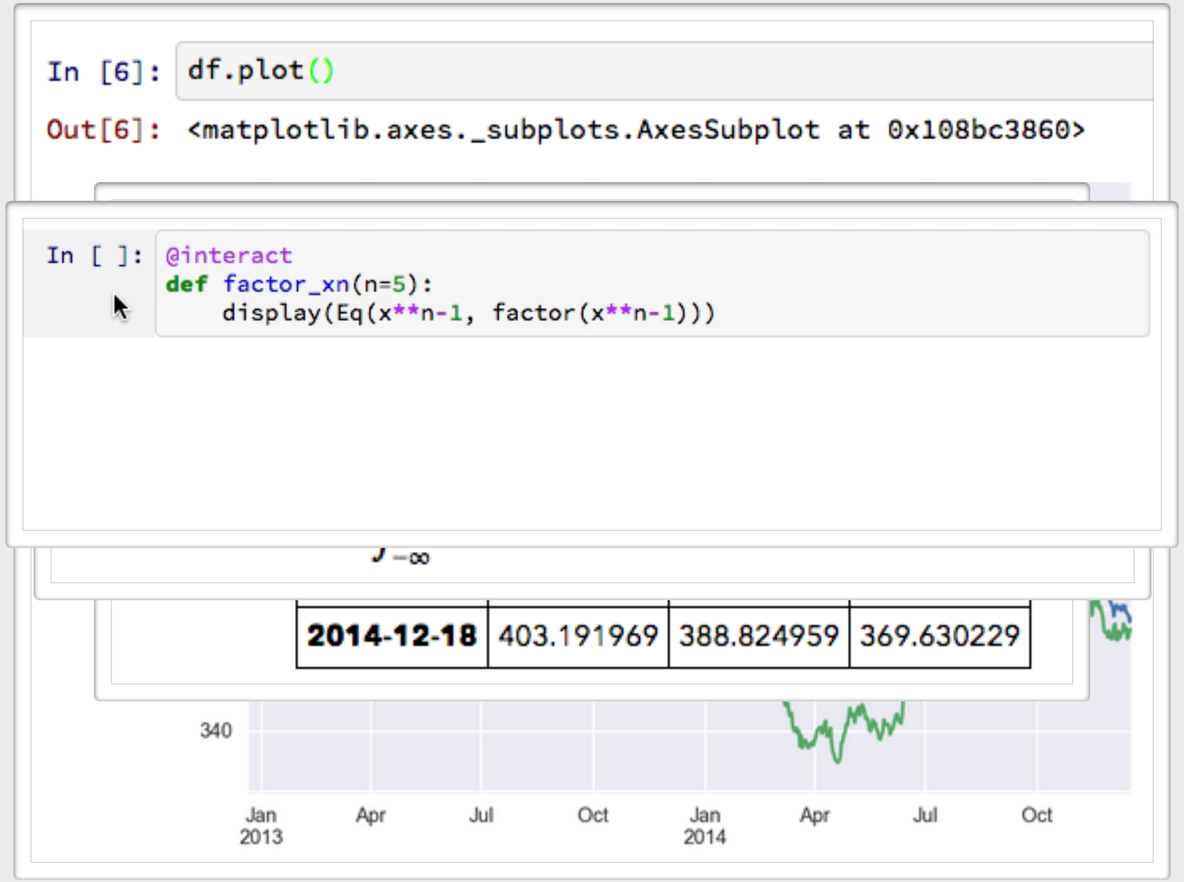
text

svg, png, jpeg

latex, pdf

html, javascript

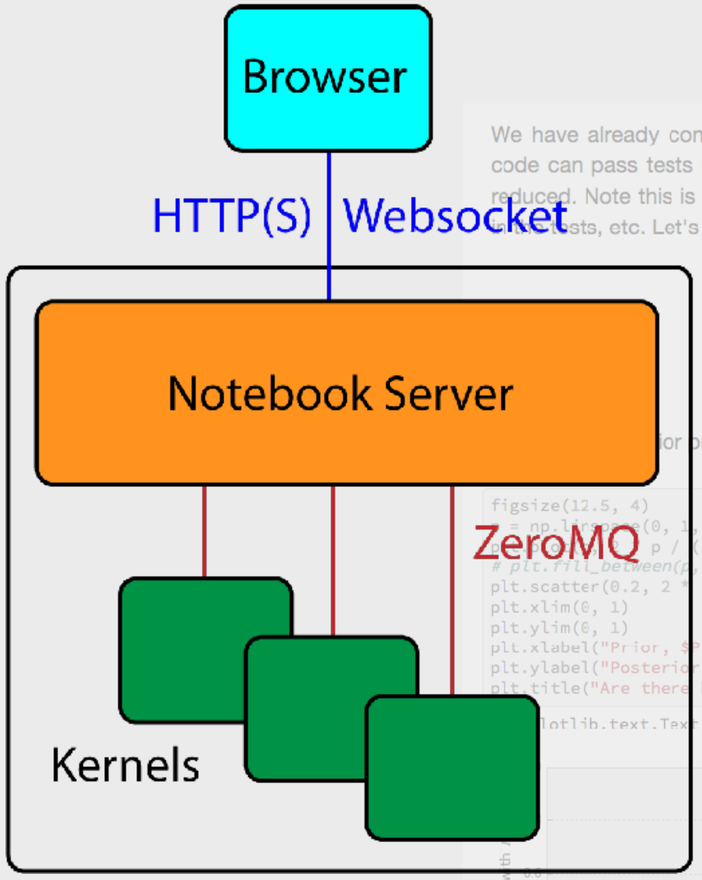
interactive widgets

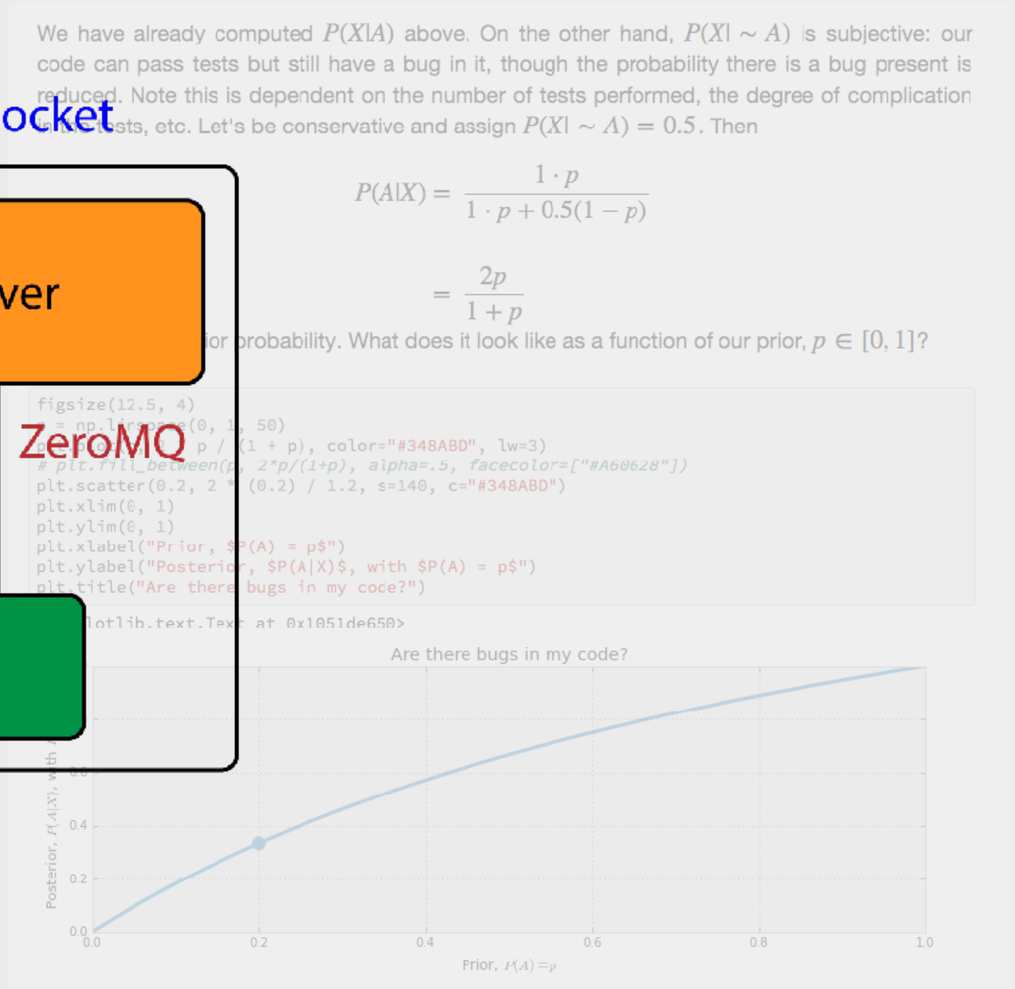


Jupyter Protocol: Language Agnostic

Jupyter Notebooks

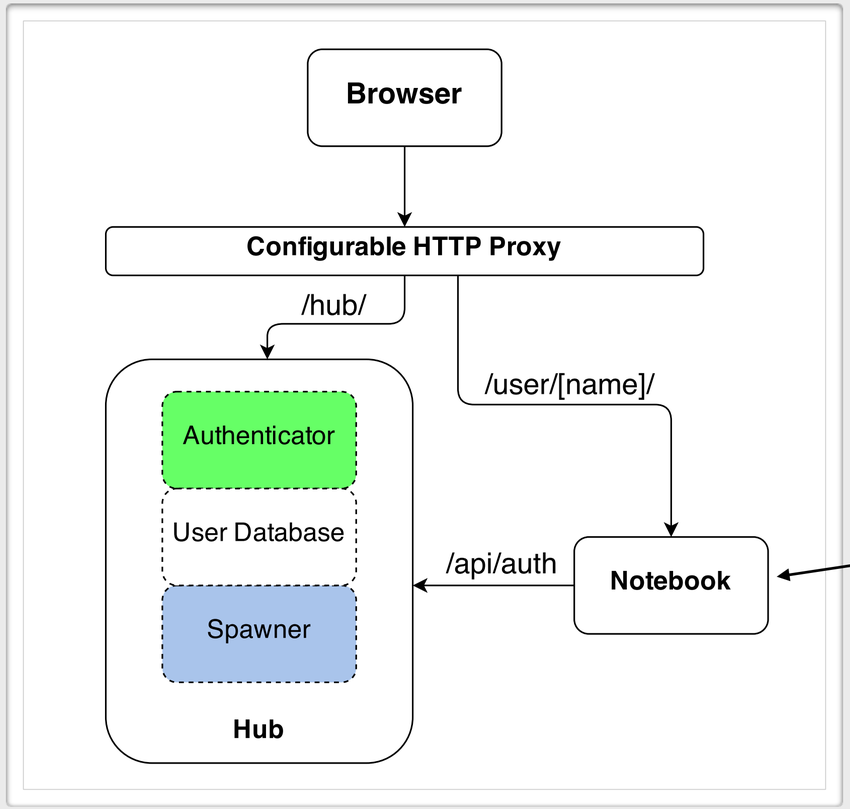
Web Application





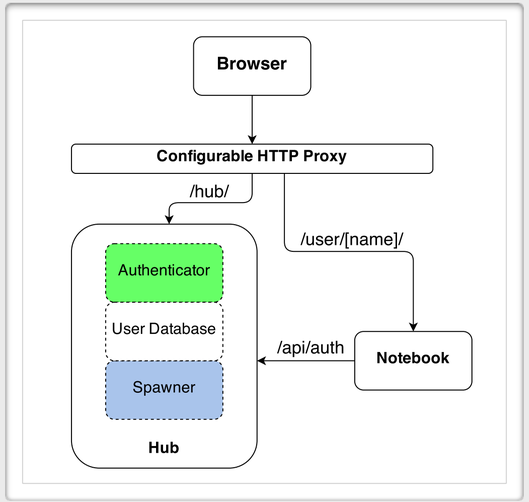
JupyterHub

Authenticated Notebook Service



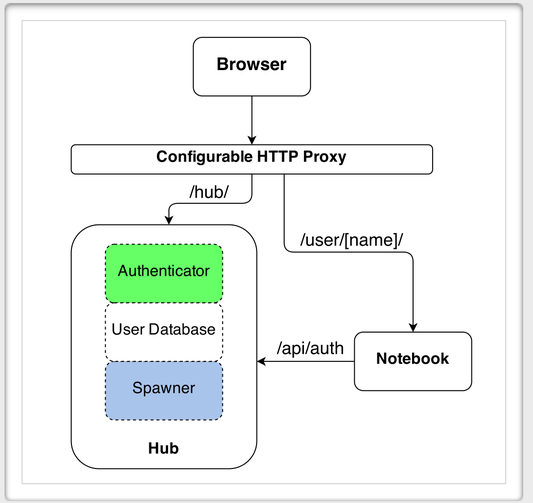
JupyterHub

- **Proxy** in front (configurable-http-proxy)
- **Authenticator** handles authentication
- **Spawner** launches single-user servers
- **Services** for integrating with the Hub
- **Hub** connects everything



JupyterHub

- Allows Jupyter notebooks to be the access point for your cluster
- Integrates with existing authentication
- Can spawn notebook servers via PBS/ SLURM/Kubernetes/etc.
- in use at institutions:
  UC Berkeley, NERSC, XSEDE, Compute Canada,
  San Diego Supercomputing Center, Bryn Mawr, more



IPython and Dask

Two libraries for interactive Data Science

IPython Parallel

- **Direct** multiplexing API to remote namespaces
- Interactive **MPI** Simulations
- Facilitated debugging or steering traditional MPI
- Supports simple task-farming as well
- Focused on **interactivity**, **debugging**
- concurrent.futures-based **Executor** API

Dask Distributed

- **Graph** evaluation engine
- High-level APIs for **collections, data frames, arrays**
- Lazy graph evaluation - only compute what you need
- Detailed profiling information
- Data-source support for S3, HDFS, files, HDF5, memory, etc.
- concurrent.futures-based **Executor** API

Demo

Takeaways

- **Jupyter** is a web-based application for interactive computing, widely used by data scientists
- **JupyterHub** has the building blocks for institutional deployments, enabling shell-free access to compute
- **IPython Parallel** enables interactive steering, development, and debugging of traditional parallel tasks, including **MPI** jobs
- **Dask Distributed** provides high-level APIs for distributed algorithms and data structures, with analogies to Apache Spark

Interactive HPC

with Jupyter, IPython, and Dask

SOS 21

Min Ragan-Kelley
Simula Research Lab