Computational Libraries: How to Find Them, Use Them, Predict Their Demise

Illustrated by “hacking Spark for research work”

Nathan M. Palmer

Johns Hopkins University, April 18, 2017

Introduction / Overview

- Computational Libraries
  - What they are, how to find them, how to tell if they are going to disappear
- Specific example to illustrate: Spark
  - Ideally going to load, run an example...
  - ...which is HARK-able (but not in HARK)
- If time, free dissertation topic (k-fold cross-validation of structural models of approximate optimal behavior)
- Anecdotes along the way

Introduction / Overview

- Computational Libraries
  - What they are, how to find them, how to tell if they are going to disappear
- Specific example to illustrate: Spark
  - Ideally going to load, run an example...
  - ...which is HARK-able (but not in HARK)
- If time, free dissertation topic (k-fold cross-validation of structural models of approximate optimal behavior)
- Anecdotes along the way

Introduction / Overview

- Computational Libraries
  - What they are, how to find them, how to tell if they are going to disappear
- Specific example to illustrate: Spark
  - Ideally going to load, run an example...
  - ...which is HARK-able (but not in HARK)
- If time, free dissertation topic (k-fold cross-validation of structural models of approximate optimal behavior)
- Anecdotes along the way

Introduction / Overview

- Computational Libraries
    - What they are, how to find them, how to tell if they are going to disappear
- Specific example to illustrate: Spark
    - Ideally going to load, run an example...
    - ...which is HARK-able (but not in HARK)
- If time, free dissertation topic (k-fold cross-validation of structural models of approximate optimal behavior)
- Anecdotes along the way

- Computational Libraries
  - What they are, how to find them, how to tell if they are going to disappear
- Specific example to illustrate: Spark
  - Ideally going to load, run an example...
  - ...which is HARK-able (but not in HARK)
- If time, free dissertation topic (k-fold cross-validation of structural models of approximate optimal behavior)
- Anecdotes along the way

Background: A Little About Myself

- Minimal history:
    - CS background, macro interests, current work

Background: A Little About Myself

- Things I work on now
  - ABMs of complicated systems (eg. macreconomy, DC MSA housing market, limit order book)
  - HARK
  - Learning-to-optimize (with no priors) – approx. DP, RL, social learning
    - Aside: "rules of thumb vs optimization: why not both?"
    - Some lit: Gabaix QJE (2014), Howitt & Ozak JEDC (2014), Evans & McGough (2015), Lettau and Uhlig (1999), Allen and Carroll (2001) [and cottage literature], Arifovic (many), ...

Background: A Little About Myself

- Things I work on now
  - ABMs of complicated systems (eg. macreconomy, DC MSA housing market, limit order book)
  - HARK
  - Learning-to-optimize (with no priors) – approx. DP, RL, social learning
    - Aside: "rules of thumb vs optimization: why not both?"
    - Some lit: Gabaix QJE (2014), Howitt & Ozak JEDC (2014), Evans & McGough (2015), Lettau and Uhlig (1999), Allen and Carroll (2001) [and cottage literature], Arifovic (many), ...

Background: A Little About Myself

- Things I work on now
    - ABMs of complicated systems (eg. macreconomy, DC MSA housing market, limit order book)
    - HARK
    - Learning-to-optimize (with no priors) – approx. DP, RL, social learning
        - Aside: "rules of thumb vs optimization: why not both?"
        - Some lit: Gabaix QJE (2014), Howitt & Ozak JEDC (2014), Evans & McGough (2015), Lettau and Uhlig (1999), Allen and Carroll (2001) [and cottage literature], Arifovic (many), ...

Background: A Little About Myself

- Things I work on now
  - ABMs of complicated systems (eg. macreconomy, DC MSA housing market, limit order book)
  - HARK
  - Learning-to-optimize (with no priors) – approx. DP, RL, social learning
    - Aside: "rules of thumb vs optimization: why not both?"
    - Some lit: Gabaix QJE (2014), Howitt & Ozak JEDC (2014), Evans & McGough (2015), Lettau and Uhlig (1999), Allen and Carroll (2001) [and cottage literature], Arifovic (many), ...

Outline

Outline

Step back for a moment

- A word on "academia as the final guilds"
    - Show things that have worked, give background
    - Not exhaustive, not necessarily the absolute best (although the longer I work on stochastic approx techniques...)
    - We in research are solving a very coarse, high-dim problem – soft info suprisingly important

Step back for a moment

- A word on "academia as the final guilds"
  - Show things that have worked, give background
  - Not exhaustive, not necessarily the absolute best (although the longer I work on stochastic approx techniques...)
  - We in research are solving a very coarse, high-dim problem – soft info suprisingly important

Step back for a moment

- A word on “academia as the final guilds”
    - Show things that have worked, give background
    - Not exhaustive, not necessarily the absolute best (although the longer I work on stochastic approx techniques...)
    - We in research are solving a very coarse, high-dim problem – soft info suprisingly important

Step back for a moment

- A word on “academia as the final guilds”
  - Show things that have worked, give background
  - Not exhaustive, not necessarily the absolute best (although the longer I work on stochastic approx techniques...)
  - We in research are solving a very coarse, high-dim problem – soft info suprisingly important

What Are Libraries, Why Important?

- Relevant from background: C++ vs Java and "standard library"
- Java (and Python) strength: massive libraries

What Are Libraries, Why Important?

- Relevant from background: C++ vs Java and "standard library"
- Java (and Python) strength: massive libraries

Organizations and Libraries

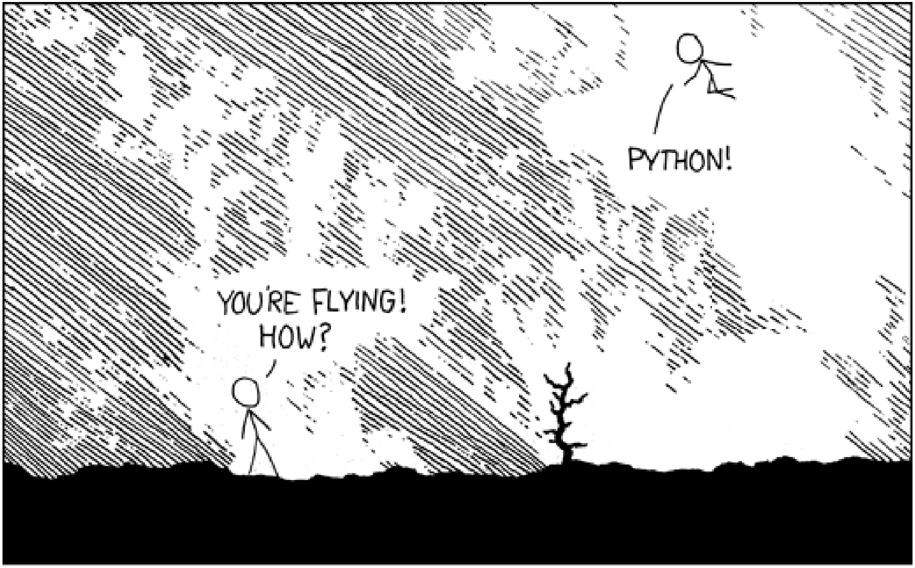
Organizations supporting:

- NASA, Bloomberg, IBM, Los Alamos, Lawrence Livermore, many more

Libraries:

- Anaconda (package system)
- NumPy, SciPy, Pandas, Statsmodels, Scikit-learn, Numba, Spark
- Quant-Econ, NetworkX, AstroPy, PyMC3
- Too many to list:
    - compilers, grid and multiprocessing computing, GPU support, web scraping, NLP
- nflgame . . .

import antigravity



Some Anecdotes

"Why Python is steadily eating other languages' lunch" goo.gl/OUZet9

- Computational neuro: "3 years ago, five languages; now Python"

Some Anecdotes

"Why Python is steadily eating other languages' lunch" goo.gl/OUZet9

- Computational neuro: "3 years ago, five languages; now Python"

John Cochrane, "Eight young stars:"

> *If you're going in to economics these days, learn python, R, stata, html, java; know how to scrape data from the web, run a large programming task in a disciplined style, manipulate and clean large data sets. That's the key intellectual arbitrage behind the young stars' work today, and way more important than measure theory and real analysis!*

Early Problems with Python (and R)

- Python was slow
- Python was single-threaded
    - The dreaded "GIL"

Early Problems with Python (and R)

- Python was slow
- Python was single-threaded
  - The dreaded “GIL”

Early Problems Solved

- However, major strength: very large community converged
  - NumPy / SciPy, Numba/PyPy/Cython
  - many parallel solutions
- Solution took the form of decentralized libraries
  - caveat: "ground littered with failed projects," will talk a little about judging what works...

Early Problems Solved

- However, major strength: very large community converged
  - NumPy / SciPy, Numba/PyPy/Cython
  - many parallel solutions
- Solution took the form of decentralized libraries
  - caveat: "ground littered with failed projects," will talk a little about judging what works...

Early Problems Solved

- However, major strength: very large community converged
  - NumPy / SciPy, Numba/PyPy/Cython
  - many parallel solutions
- Solution took the form of decentralized libraries
  - caveat: "ground littered with failed projects," will talk a little about judging what works...

Early Problems Solved

- However, major strength: very large community converged
  - NumPy / SciPy, Numba/PyPy/Cython
  - many parallel solutions
- Solution took the form of decentralized libraries
  - caveat: "ground littered with failed projects," will talk a little about judging what works...

Aside on R Libraries

- Early on, R “hacked” academic incentives: Journal of Statistical Software
- Many, many high-quality statistical libraries now
- We need to learn from them!

Aside on R Libraries

- Early on, R “hacked” academic incentives: Journal of Statistical Software
- Many, many high-quality statistical libraries now
- We need to learn from them!

Aside on Julia, Python 3 and libraries

- Julia: stability of language, library system, extent of community: feels like scientific Python about 8 years ago
- Python 3: previously libraries have been the main reason for holding back
    - however p(find library | py3) is hitting acceptable level

Aside on Julia, Python 3 and libraries

- Julia: stability of language, library system, extent of community: feels like scientific Python about 8 years ago
- Python 3: previously libraries have been the main reason for holding back
  - however p(find library | py3) is hitting acceptable level

Aside on Julia, Python 3 and libraries

- Julia: stability of language, library system, extent of community: feels like scientific Python about 8 years ago
- Python 3: previously libraries have been the main reason for holding back
    - however p(find library | py3) is hitting acceptable level

Outline

Finding Libraries

- Let's say you want to speed up a simulation. Options:
  - Vectorize where can
  - Compile where can
  - Parallelize where can
  - Auto-diff where can [conda install autograd]
- Solution may be in a library somewhere

Finding Libraries

- Let's say you want to speed up a simulation. Options:
    - Vectorize where can
    - Compile where can
    - Parallelize where can
    - Auto-diff where can [conda install autograd]
- Solution may be in a library somewhere

Finding Libraries

- Let's say you want to speed up a simulation. Options:
  - Vectorize where can
  - Compile where can
  - Parallelize where can
  - Auto-diff where can [conda install autograd]
- Solution may be in a library somewhere

Finding Libraries

- Let's say you want to speed up a simulation. Options:
  - Vectorize where can
  - Compile where can
  - Parallelize where can
  - Auto-diff where can [conda install autograd]
- Solution may be in a library somewhere

Finding Libraries

- Let’s say you want to speed up a simulation. Options:
  - Vectorize where can
  - Compile where can
  - Parallelize where can
  - Auto-diff where can [conda install autograd]
- Solution may be in a library somewhere

Finding Libraries: Judging Quality

- Found a library – how to judge quality?
  - papers published using it? (gold standard)
  - do they have a stable version? ($0.1.0 < 1.2.x$)
  - lots of activity on development page? (easier to 'watch' now with github...)
    - Eg. I am 'watching' PyFlux right now, TS metrics library for Python
  - At least 2, ideally 3-4 main developers? Many contributors? Over time?
- If lots of hype, has it survived 2-3 years?
- Cheating, in Python: "conda install ______ works?"

Finding Libraries: Judging Quality

- Found a library – how to judge quality?
  - papers published using it? (gold standard)
  - do they have a stable version? (0.1.0 $<$ 1.2.x)
  - lots of activity on development page? (easier to 'watch' now with github...)
    - Eg. I am 'watching' PyFlux right now, TS metrics library for Python
  - At least 2, ideally 3-4 main developers? Many contributors? Over time?
- If lots of hype, has it survived 2-3 years?
- Cheating, in Python: "conda install ______ works?"

Finding Libraries: Judging Quality

- Found a library – how to judge quality?
  - papers published using it? (gold standard)
  - do they have a stable version? ($0.1.0 < 1.2.x$)
  - lots of activity on development page? (easier to 'watch' now with github...)
    - Eg. I am 'watching' PyFlux right now, TS metrics library for Python
  - At least 2, ideally 3-4 main developers? Many contributors? Over time?
- If lots of hype, has it survived 2-3 years?
- Cheating, in Python: "conda install ______ works?"

Finding Libraries: Judging Quality

- Found a library – how to judge quality?
    - papers published using it? (gold standard)
    - do they have a stable version? ($0.1.0 < 1.2.x$)
    - lots of activity on development page? (easier to 'watch' now with github...)
        - Eg. I am 'watching' PyFlux right now, TS metrics library for Python
    - At least 2, ideally 3-4 main developers? Many contributors? Over time?
- If lots of hype, has it survived 2-3 years?
- Cheating, in Python: "conda install ______ works?"

Finding Libraries: Judging Quality

- Found a library – how to judge quality?
  - papers published using it? (gold standard)
  - do they have a stable version? ($0.1.0 < 1.2.x$)
  - lots of activity on development page? (easier to 'watch' now with github...)
    - Eg. I am 'watching' PyFlux right now, TS metrics library for Python
  - At least 2, ideally 3-4 main developers? Many contributors? Over time?
- If lots of hype, has it survived 2-3 years?
- Cheating, in Python: "conda install ______ works?"

Finding Libraries: Judging Quality

- Found a library – how to judge quality?
    - papers published using it? (gold standard)
    - do they have a stable version? ($0.1.0 < 1.2.x$)
    - lots of activity on development page? (easier to 'watch' now with github...)
        - Eg. I am 'watching' PyFlux right now, TS metrics library for Python
    - At least 2, ideally 3-4 main developers? Many contributors? Over time?
- If lots of hype, has it survived 2-3 years?
- Cheating, in Python: "conda install ______ works?"

Fixing the Inevitable Betrayal: Update Breaks Everything

- If not a glutton for punishment, use stable version
- However even stable versions change (1.x -> 2.x)
- How to fix when changes?
- If pure Python solution, "environments" can fix 90%
    - I know many people who use these

Fixing the Inevitable Betrayal: Update Breaks Everything

- If not a glutton for punishment, use stable version
- However even stable versions change (1.x -> 2.x)
- How to fix when changes?
- If pure Python solution, “environments” can fix 90%
  - I know many people who use these

Fixing the Inevitable Betrayal: Update Breaks Everything

- If not a glutton for punishment, use stable version
- However even stable versions change (1.x -> 2.x)
- How to fix when changes?
- If pure Python solution, “environments” can fix 90%
    - I know many people who use these

Fixing the Inevitable Betrayal: Update Breaks Everything

- However if libraries are 'outside' Python enough may still have problems
  - Carefully track versions of tools used
  - Use repo system!
  - Just repair everything...
- Alternative: keep everything up-to-date, fix breaks as they occur
  - Works best when mature versioning system like Anaconda

Fixing the Inevitable Betrayal: Update Breaks Everything

- However if libraries are 'outside' Python enough may still have problems
  - Carefully track versions of tools used
  - Use repo system!
  - Just repair everything...
- Alternative: keep everything up-to-date, fix breaks as they occur
  - Works best when mature versioning system like Anaconda

Fixing the Inevitable Betrayal: Update Breaks Everything

- However if libraries are 'outside' Python enough may still have problems
  - Carefully track versions of tools used
  - Use repo system!
  - Just repair everything...
- Alternative: keep everything up-to-date, fix breaks as they occur
  - Works best when mature versioning system like Anaconda

Outline

Parallel Computing in Python

- Recall the dreaded GIL
  - From stackoverflow: "Instead of discussing the GIL, how about something simpler, like the Middle East?"
- **Lightning**, low-quality discussion of GIL

Parallel Computing in Python

- Recall the dreaded GIL
    - From stackoverflow: "Instead of discussing the GIL, how about something simpler, like the Middle East?"
- **Lightning**, low-quality discussion of GIL

I've Never Been Worried about GIL

- ...for a number of reasons:
    - it is implementation-specific (has to do with thread-safeness of 'virtual machine' interface)
    - **many** interested parties in creating solutions
    - many ways to tackle parallelization and research Qs are flexible

I've Never Been Worried about GIL

- ...for a number of reasons:
    - it is implementation-specific (has to do with thread-safeness of 'virtual machine' interface)
    - **many** interested parties in creating solutions
    - many ways to tackle parallelization and research Qs are flexible

I've Never Been Worried about GIL

- ...for a number of reasons:
  - it is implementation-specific (has to do with thread-safeness of 'virtual machine' interface)
  - **many** interested parties in creating solutions
  - many ways to tackle parallelization and research Qs are flexible

Many Parallel Solution Libraries

- A number of solutions over the years:
    - numpy / scipy parallelization
    - multiprocessing, joblib, dill
    - Cython's prange, Numba's prange
    - Pypy "stm"
    - Jython (but don't)
    - Grids: Slurm, PiCloud, Hadoop/Spark

Many Parallel Solution Libraries

- A number of solutions over the years:
  - numpy / scipy parallelization
  - multiprocessing, joblib, dill
  - Cython's prange, Numba's prange
  - Pypy "stm"
  - Jython (but don't)
  - Grids: Slurm, PiCloud, Hadoop/Spark

Many Parallel Solution Libraries

- A number of solutions over the years:
    - numpy / scipy parallelization
    - multiprocessing, joblib, dill
    - Cython's prange, Numba's prange
    - Pypy "stm"
    - Jython (but don't)
    - Grids: Slurm, PiCloud, Hadoop/Spark

Many Parallel Solution Libraries

- A number of solutions over the years:
    - numpy / scipy parallelization
    - multiprocessing, joblib, dill
    - Cython's prange, Numba's prange
    - Pypy "stm"
    - Jython (but don't)
    - Grids: Slurm, PiCloud, Hadoop/Spark

Beware the Failed Solutions

- However note also that "trail littered with the remains" of failed projects:
  - PiCloud
  - NumbaPro
  - Nuitka, parakeet, copperhead (NVIDIA-funded), pythran, pyston

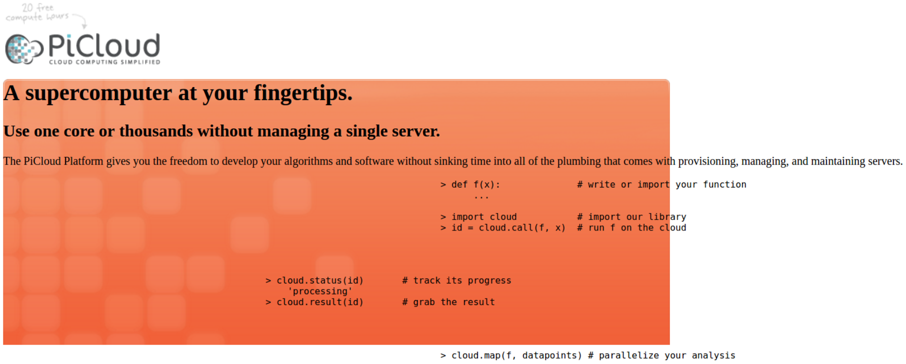
Beware the Failed Solutions

- However note also that "trail littered with the remains" of failed projects:
  - PiCloud
  - NumbaPro
  - Nuitka, parakeet, copperhead (NVIDIA-funded), pythran, pyston

Beware the Failed Solutions

- However note also that “trail littered with the remains” of failed projects:
  - PiCloud
  - NumbaPro
  - Nuitka, parakeet, copperhead (NVIDIA-funded), pythran, pyston

Example: PiCloud



A supercomputer at your fingertips.

Use one core or thousands without managing a single server.

The PiCloud Platform gives you the freedom to develop your algorithms and software without sinking time into all of the plumbing that comes with provisioning, managing, and maintaining servers.

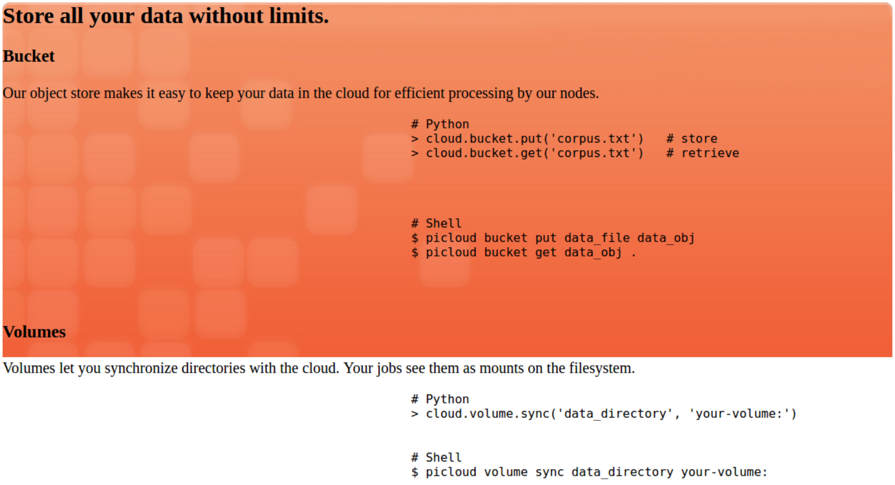
```
> def f(x):              # write or import your function
      ...

> import cloud           # import our library
> id = cloud.call(f, x)  # run f on the cloud
```

```
> cloud.status(id)       # track its progress
    'processing'
> cloud.result(id)       # grab the result
```

```
> cloud.map(f, datapoints) # parallelize your analysis
```

Example: PiCloud



Store all your data without limits.

Bucket

Our object store makes it easy to keep your data in the cloud for efficient processing by our nodes.

```
# Python
> cloud.bucket.put('corpus.txt')   # store
> cloud.bucket.get('corpus.txt')   # retrieve
```

```
# Shell
$ picloud bucket put data_file data_obj
$ picloud bucket get data_obj .
```

Volumes

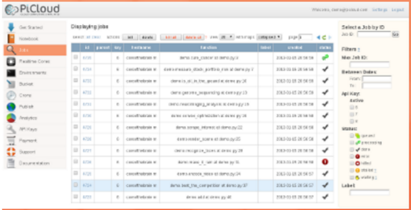
Volumes let you synchronize directories with the cloud. Your jobs see them as mounts on the filesystem.

```
# Python
> cloud.volume.sync('data_directory', 'your-volume:')
```

```
# Shell
$ picloud volume sync data_directory your-volume:
```

Example: PiCloud

Drill down into your Computational History

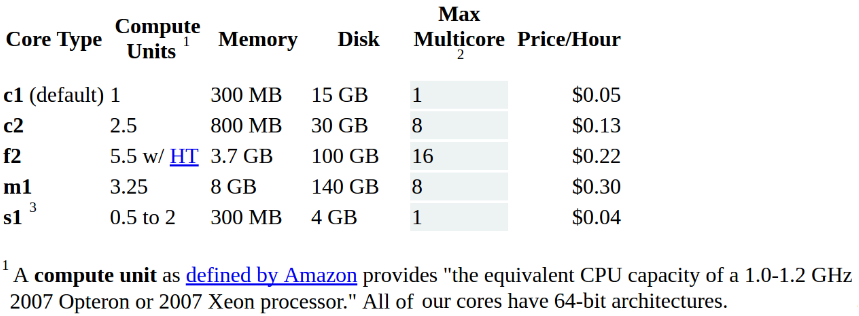


The days of sifting through server logs are over.

We consolidate all of your history into a single interface. You can see a task's:

- Standard Output & Error (Realtime)
- Memory & Swap Usage (Realtime)
- User & System CPU Time (Realtime)
- Exception & Traceback
- Execution Profile
- Status & Runtime
- Compute 10,000+ Cores

Example: PiCloud



Core Type	Compute Units[1]	Memory	Disk	Max Multicore[2]	Price/Hour
c1 (default)	1	300 MB	15 GB	1	$0.05
c2	2.5	800 MB	30 GB	8	$0.13
f2	5.5 w/ HT	3.7 GB	100 GB	16	$0.22
m1	3.25	8 GB	140 GB	8	$0.30
s1[3]	0.5 to 2	300 MB	4 GB	1	$0.04

[1] A **compute unit** as defined by Amazon provides "the equivalent CPU capacity of a 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor." All of our cores have 64-bit architectures.

Goal: Spark + Hadoop as PiCloud Replacement

- Setup will likely be more work, but not nearly as much work as setting up Hadoop for this purpose

Outline

What is Spark

- Spark is a layer on top of Hadoop
    - Lightning aside on Hadoop, and why useful
- Hadoop a pain to set up and run directly
    - Connecting all the wires, maintaining is a pain
    - Map/Reduce a pain
    - Spark's RDD makes creating flexible parallelizable code easy
- Spark claims much more spead for raw computation

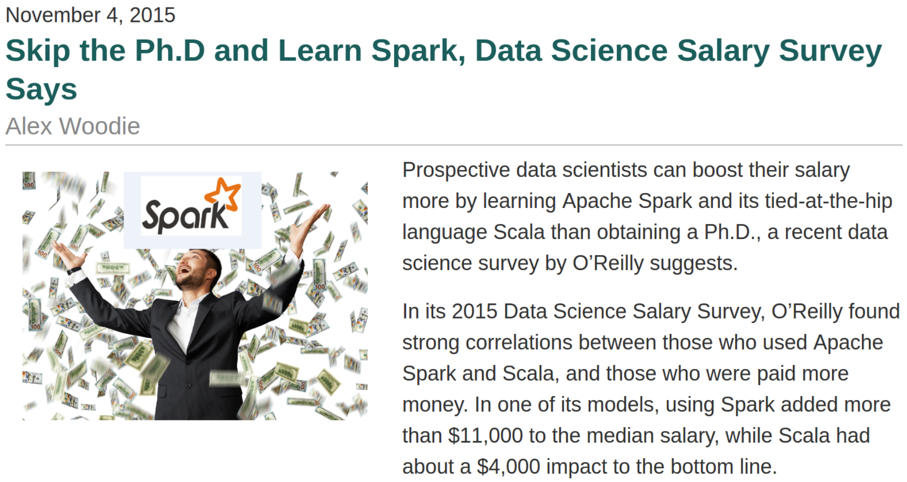
What is Spark

- Spark is a layer on top of Hadoop
    - Lightning aside on Hadoop, and why useful
- Hadoop a pain to set up and run directly
    - Connecting all the wires, maintaining is a pain
    - Map/Reduce a pain
    - Spark's RDD makes creating flexible parallelizable code easy
- Spark claims much more spead for raw computation

What is Spark

- Spark is a layer on top of Hadoop
    - Lightning aside on Hadoop, and why useful
- Hadoop a pain to set up and run directly
    - Connecting all the wires, maintaining is a pain
    - Map/Reduce a pain
    - Spark's RDD makes creating flexible parallelizable code easy
- Spark claims much more spead for raw computation

Noah Smith's "If you get a PhD..."



November 4, 2015

Skip the Ph.D and Learn Spark, Data Science Salary Survey Says

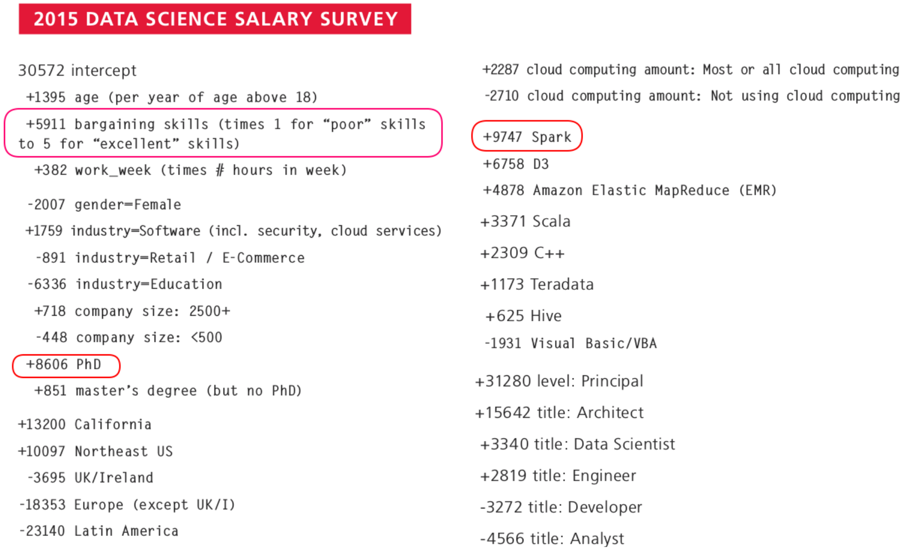
Alex Woodie

Prospective data scientists can boost their salary more by learning Apache Spark and its tied-at-the-hip language Scala than obtaining a Ph.D., a recent data science survey by O'Reilly suggests.

In its 2015 Data Science Salary Survey, O'Reilly found strong correlations between those who used Apache Spark and Scala, and those who were paid more money. In one of its models, using Spark added more than $11,000 to the median salary, while Scala had about a $4,000 impact to the bottom line.

From O'Reilly's "Data Science Survey," 2015 and 2016

Aside on Communicating To Technical Public...



2015 DATA SCIENCE SALARY SURVEY

30572	intercept
+1395	age (per year of age above 18)
+5911	bargaining skills (times 1 for “poor” skills to 5 for “excellent” skills)
+382	work_week (times # hours in week)
-2007	gender=Female
+1759	industry=Software (incl. security, cloud services)
-891	industry=Retail / E-Commerce
-6336	industry=Education
+718	company size: 2500+
-448	company size: <500
+8606	PhD
+851	master’s degree (but no PhD)
+13200	California
+10097	Northeast US
-3695	UK/Ireland
-18353	Europe (except UK/I)
-23140	Latin America
+2287	cloud computing amount: Most or all cloud computing
-2710	cloud computing amount: Not using cloud computing
+9747	Spark
+6758	D3
+4878	Amazon Elastic MapReduce (EMR)
+3371	Scala
+2309	C++
+1173	Teradata
+625	Hive
-1931	Visual Basic/VBA
+31280	level: Principal
+15642	title: Architect
+3340	title: Data Scientist
+2819	title: Engineer
-3272	title: Developer
-4566	title: Analyst

Outline

Let's Install Spark...

- See README here:
  https://github.com/compumetrika/fun-with-spark/

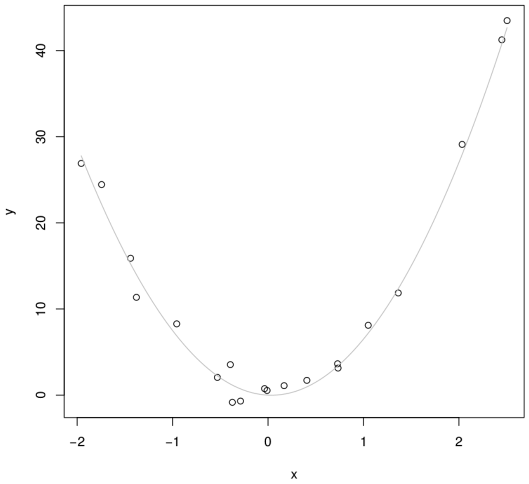
Let's Run an Example Notebook...

- See the example here: https://github.com/compumetrika/fun-with-spark/blob/master/Spark-Regression-Example.ipynb
- Also accessible from README.md

Outline

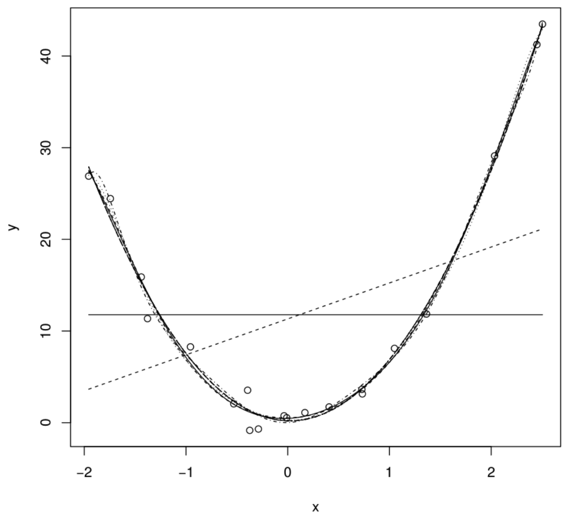
Selection via Expected Loss

Consider the following artificial data:[1]

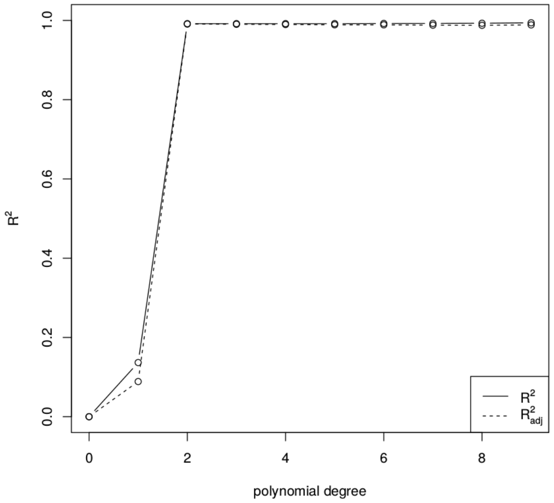


[1]Discussion from Cosma Shalizi's excellent "Advanced Data Analysis from an Elementary Point of View"

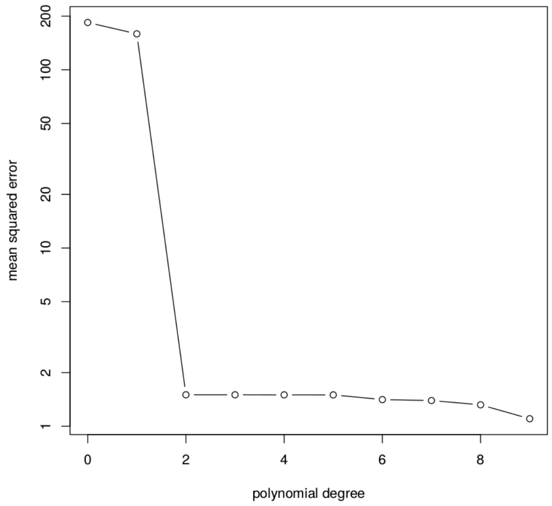
Polynomial Overfitting



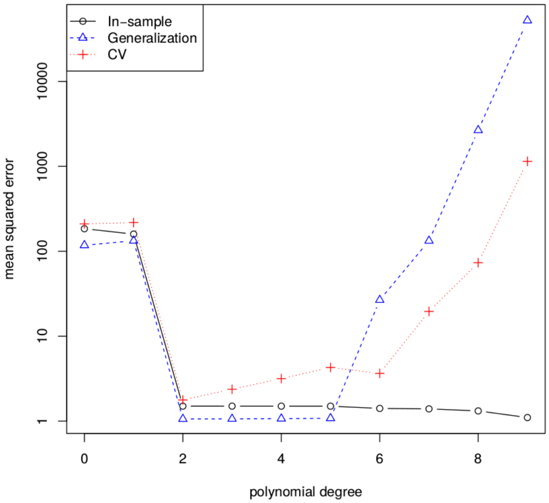
R-squared Looks Great



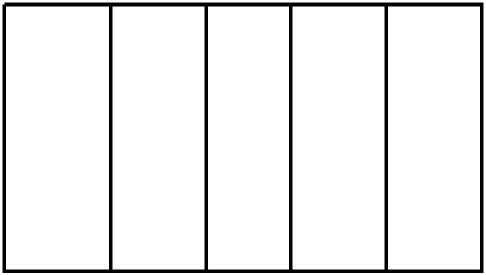
Loss Function (SSE) Looks Great



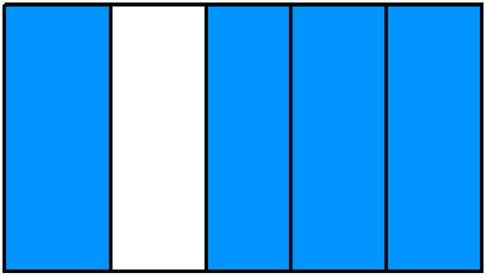
However, Very Poor Out of Sample Fit



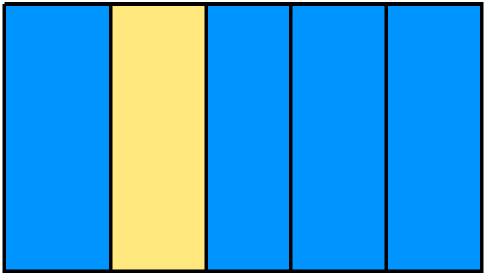
Data Selection



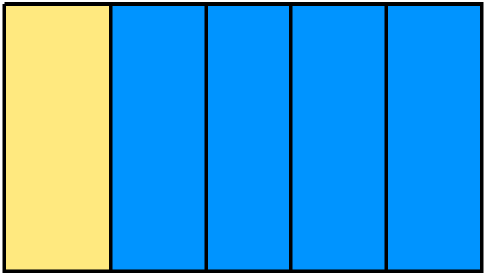
Data Selection



Data Selection



Data Selection



Cross-Validation is Embarrassing Parallel

- k-fold cross-validation is embarrassingly parallel
- Finding variance on k-fold cross-validation, eg. via bootstrap, also embarrassingly parallel

Summary

- We've talked about finding libraries, installing them, and running them
- ...using a specific example, Spark
- Any Qs?

Summary

- We've talked about finding libraries, installing them, and running them
- ...using a specific example, Spark
- Any Qs?